

**Технология комплексной поддержки
жизненного цикла семантически совместимых
интеллектуальных компьютерных систем
нового поколения**

– Монография –

22 февраля 2023 г.

Оглавление

Предисловие	13
Авторское предисловие	17
Часть 1. Введение в интеллектуальные компьютерные системы нового поколения и технологию комплексной поддержки их жизненного цикла	23
Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем	25
<i>Загорский А.С.</i>	
<i>Голенков В.В.</i>	
<i>Шункевич Д.В.</i>	
§ 1.1.1. Понятие интеллектуальной кибернетической системы	25
Пункт 1.1.1.1. Типология кибернетических систем	27
Пункт 1.1.1.2. Структура кибернетической системы	28
Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы	31
Пункт 1.1.1.4. Комплекс свойств, определяющих качество физической оболочки кибернетической системы	31
Пункт 1.1.1.5. Комплекс свойств, определяющих качество информации, хранимой в памяти кибернетической системы	33
Пункт 1.1.1.6. Комплекс свойств, определяющих качество решателя задач кибернетической системы	34
Пункт 1.1.1.7. Комплекс свойств, определяющих уровень обучаемости кибернетической системы	36
Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы	37
§ 1.1.2. Понятие интеллектуальной многоагентной системы	38
§ 1.1.3. Эволюция традиционных и интеллектуальных компьютерных систем	42
Глава 1.2. Интеллектуальные компьютерные системы нового поколения	45
<i>Голенков В.В.</i>	
<i>Шункевич Д.В.</i>	
<i>Ковалёв М.В.</i>	
<i>Садовский М.Е.</i>	
§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения	46
§ 1.2.2. Принципы, лежащие в основе смыслового представления информации	53
§ 1.2.3. Принципы, лежащие в основе многоагентных моделей решателей задач интеллектуальных компьютерных систем нового поколения	53
§ 1.2.4. Принципы, лежащие в основе онтологических моделей мультимодальных интерфейсов интеллектуальных компьютерных систем нового поколения	55
§ 1.2.5. Достоинства предлагаемых принципов, лежащих в основе интеллектуальных компьютерных систем нового поколения	57
Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения	61
<i>Голенков В.В.</i>	
<i>Шункевич Д.В.</i>	
<i>Ковалёв М.В.</i>	
<i>Садовский М.Е.</i>	
§ 1.3.1. Технология OSTIS (Open Semantic Technology for Intelligent Systems)	61
§ 1.3.2. Семантически совместимые ostis-системы	64

Часть 2. Смыслоное представление и онтологическая систематизация знаний в интеллектуальных компьютерных системах нового поколения	67
Глава 2.1. Информационные конструкции и языки	69
<i>Никифоров С.А.</i>	
<i>Гойло А.А.</i>	
§ 2.1.1. Формализация понятия информационной конструкции	69
§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем	78
§ 2.1.3. Файлы ostis-систем	79
Глава 2.2. Смыслоное представление знаний в памяти ostis-систем	81
<i>Иващенко В.П.</i>	
<i>Голенков В.В.</i>	
§ 2.2.1. SC-код (Semantic Computer Code)	82
§ 2.2.2. Базовая денотационная семантика SC-кода	83
Пункт 2.2.2.1. Семантическая классификация sc-элементов по базовым признакам	84
Пункт 2.2.2.2. Уточнение смысла выделенных классов sc-элементов и связей между этими классами	88
Пункт 2.2.2.3. Структура базовой семантической спецификации sc-элемента	99
Пункт 2.2.2.4. Онтологическая формализация Базовой денотационной семантики SC-кода	104
§ 2.2.3. Синтаксис SC-кода	108
Пункт 2.2.3.1. Синтаксические особенности SC-кода	109
Пункт 2.2.3.2. Синтаксическое Ядро SC-кода	110
Пункт 2.2.3.3. Уточнение понятия синтаксически корректной sc-конструкции	114
Пункт 2.2.3.4. Синтаксические расширения Ядра SC-кода	115
§ 2.2.4. Файлы ostis-системы	116
§ 2.2.5. Смыслоное пространство ostis-систем	116
Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний	125
<i>Садовский М.Е.</i>	
<i>Голенков В.В.</i>	
<i>Жмырко А.В.</i>	
§ 2.3.1. Внешние идентификаторы sc-элементов – знаков, входящих в конструкции SC-кода	125
Пункт 2.3.1.1. Понятие простого идентификатора sc-элемента	129
Пункт 2.3.1.2. Понятие сложного идентификатора sc-элемента	131
§ 2.3.2. Язык внешнего графического представления конструкций SC-кода – SCg-код (Semantic Code graphical)	133
Пункт 2.3.2.1. Синтаксис SCg-кода	133
Пункт 2.3.2.2. Денотационная семантика SCg-кода	136
Пункт 2.3.2.3. Иерархическое семейство подъязыков, семантически эквивалентных SCg-коду	137
§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода – SCs-код (Semantic Code string)	140
Пункт 2.3.3.1. Синтаксис SCs-кода	141
Пункт 2.3.3.2. Денотационная семантика SCs-кода	148
Пункт 2.3.3.3. Иерархическое семейство подъязыков, семантически эквивалентных SCs-коду	155
§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода – SCn-код (Semantic Code natural)	157
Пункт 2.3.4.1. Синтаксис SCn-кода	157
Пункт 2.3.4.2. Денотационная семантика языка внешнего форматированного представления информационных конструкций внутреннего языка ostis-систем	159
Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах	161
<i>Бутрин С.В.</i>	
<i>Шункевич Д.В.</i>	
§ 2.4.1. Формальная онтология множеств	162
§ 2.4.2. Формальная онтология связок и отношений	165
§ 2.4.3. Формальная онтология параметров, величин и шкал	173
§ 2.4.4. Формальная онтология чисел и числовых структур	177
§ 2.4.5. Формальная онтология темпоральных сущностей	179
§ 2.4.6. Формальная онтология ситуаций и событий, описывающих динамику баз знаний ostis-систем	187

Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий

191

*Голенков В.В.**Банцевич К.А.*

§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем	192
§ 2.5.2. Формализация понятия структуры	194
§ 2.5.3. Формализация понятия семантической окрестности	199
§ 2.5.4. Формализация понятия предметной области	202
§ 2.5.5. Формализация понятия онтологии	205

Глава 2.6. Смыслоное представление логических формул и высказываний в различного вида логиках 211*Шункевич Д.В.**Василевская А.П.**Орлов М.К.**Иващенко В.П.*

§ 2.6.1. Смыслоное представление логических формул и высказываний в прикладных логиках	225
§ 2.6.2. Смыслоное представление логических формул и высказываний в неклассических логиках	225

Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах

227

*Никифоров С.А.**Гойло А.А.*

§ 2.7.1. Формализация естественных языков	229
§ 2.7.2. Формализация синтаксиса естественных языков	230
§ 2.7.3. Формализация денотационной семантики естественных языков	235

Часть 3. Многоагентные модели решателей задач интеллектуальных компьютерных систем нового поколения

241

Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии

243

*Шункевич Д.В.**Ковалёв М.В.**Никифоров С.А.*

§ 3.1.1. Глобальная предметная область и онтология действий, действий, методов, средств и технологий	244
Пункт 3.1.1.1. Понятие действия и классификация действий	244
Пункт 3.1.1.2. Понятие задачи и классификация задач	246
Пункт 3.1.1.3. Понятие класса действий и класса задач	248
Пункт 3.1.1.4. Понятие метода	250
Пункт 3.1.1.5. Спецификация методов и понятие навыка	252
Пункт 3.1.1.6. Понятие класса методов и языка представления методов	253
Пункт 3.1.1.7. Общая классификация языков представления методов	255
Пункт 3.1.1.8. Понятие модели решения задач	256
§ 3.1.2. Локальные предметные области и онтологии действий	257

Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

259

Шункевич Д.В.

§ 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению	261
Пункт 3.2.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач	261
Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах	264
§ 3.2.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой	269
§ 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты	272
§ 3.2.4. Принципы синхронизации деятельности sc-агентов	278
§ 3.2.5. Базовый язык программирования ostis-систем	287
Пункт 3.2.5.1. Денотационная семантика Базового языка программирования ostis-систем	297
Пункт 3.2.5.2. Операционная семантика Базового языка программирования ostis-систем	303
§ 3.2.6. Решатели задач ostis-систем	304

§ 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач	307
Глава 3.3. Семантическая теория программ для ostis-систем	311
<i>Зотов Н.В.</i>	
<i>Шункевич Д.В.</i>	
§ 3.3.1. Проблемы текущего состояния в области разработки и применения языков программирования	312
§ 3.3.2. Существующие онтологии языков программирования	313
§ 3.3.3. Предлагаемый подход к разработке технологий программирования для ostis-систем	314
§ 3.3.4. Синтаксис и семантика программ в ostis-системах	315
Пункт 3.3.4.1. Синтаксис программ в ostis-системах	315
Пункт 3.3.4.2. Денотационная семантика программ в ostis-системах	316
Пункт 3.3.4.3. Операционная семантика программ в ostis-системах	320
§ 3.3.5. Синтаксис и семантика языков программирования в ostis-системах	322
§ 3.3.6. Help-система поддержки проектирования и разработки программ в ostis-системах	322
§ 3.3.7. Критерии эффективности (качества) программ в ostis-системах	323
Глава 3.4. Язык вопросов для ostis-систем	327
<i>Самодумкин С.А.</i>	
<i>Шункевич Д.В.</i>	
§ 3.4.1. Синтаксис языка вопросов для ostis-систем	327
§ 3.4.2. Денотационная семантика языка вопросов для ostis-систем	327
§ 3.4.3. Операционная семантика языка вопросов для ostis-систем	327
Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах	329
<i>Василевская А.П.</i>	
<i>Орлов М.К.</i>	
<i>Шункевич Д.В.</i>	
§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами	329
§ 3.5.2. Языки продукционного программирования, используемые ostis-системами	338
Пункт 3.5.2.1. Синтаксис языков продукционного программирования, используемых ostis-системами .	338
Пункт 3.5.2.2. Денотационная семантика языков продукционного программирования, используе- мых ostis-системами	338
Пункт 3.5.2.3. Операционная семантика языков продукционного программирования, используе- мых ostis-системами	338
Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах	339
<i>Ковалёв М.В.</i>	
<i>Крощенко А.А.</i>	
<i>Головко В.А.</i>	
§ 3.6.1. Модели искусственных нейронных сетей, используемых в ostis-системах	340
Пункт 3.6.1.1. Денотационная семантика моделей искусственных нейронных сетей, используе- мых в ostis-системах	342
Пункт 3.6.1.2. Операционная семантика моделей искусственных нейронных сетей, используемых в ostis-системах	349
§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных ней- ронных сетей, семантически совместимых с базами знаний ostis-систем	353
Часть 4. Онтологические модели интерфейсов интеллектуальных компьютерных си- стем нового поколения	367
Глава 4.1. Общие принципы организации интерфейсов ostis-систем	369
<i>Садовский М.Е.</i>	
§ 4.1.1. Структура интерфейсов ostis-систем	369
§ 4.1.2. Интерфейсные действия пользователей ostis-систем	372
§ 4.1.3. Сообщения, входящие в ostis-систему и выходящие из неё	373
§ 4.1.4. Действия и внутренние агенты пользовательского интерфейса ostis-системы	374
Глава 4.2. Естественно-языковые интерфейсы ostis-систем	375

*Никифоров С.А.**Гойло А.А.**Крощенко А.А.**Захарьев В.А.**Цянь Л.*

§ 4.2.1. Предметная область и онтология естественно-языковых интерфейсов ostis-систем	376
§ 4.2.2. Предметная область и онтология лексического анализа естественно-языковых сообщений, входящих в ostis-систему	377
§ 4.2.3. Предметная область и онтология синтаксического анализа естественно-языковых сообщений, входящих в ostis-систему	379
§ 4.2.4. Предметная область и онтология понимания естественно-языковых сообщений, входящих в ostis-систему	379
§ 4.2.5. Предметная область и онтология синтеза естественно-языковых сообщений ostis-системы	384
§ 4.2.6. Модели, методы и средства адаптации пользовательских интерфейсов к носителям китайского языка	384

Глава 4.3. Аудиоинтерфейс ostis-систем

385

*Азаров И.С.**Вашкевич М.И.**Захарьев В.А.**Лихачев Д.С.**Петровский Н.А.*

§ 4.3.1. Анализ существующих подходов к разработке аудиоинтерфейсов интеллектуальных компьютерных систем	385
§ 4.3.2. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов	386
§ 4.3.3. Предметная область и онтология задач аудиоинтерфейса ostis-систем	388
§ 4.3.4. Предметная область и онтология моделей параметрического представления сигнала	393

Глава 4.4. 3D-модель внешней среды в ostis-системах

397

*Головатая Е.А.**Головатый А.И.*

§ 4.4.1. Прикладные задачи трехмерной реконструкции	397
§ 4.4.2. Анализ существующих подходов к трехмерной реконструкции	398
§ 4.4.3. Предлагаемый подход к трехмерной реконструкции с использованием базы знаний ostis-системы	399
§ 4.4.4. Семантическое представление объектов и сцены	400
§ 4.4.5. Трехмерное представление объектов в сцене	402
§ 4.4.6. Трехмерная реконструкция объектов окружающего мира	403
§ 4.4.7. Системы локального позиционирования, использующиеся в задачах трехмерной реконструкции	405
§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции	407
Пункт 4.4.8.1. Оптические системы компьютерного зрения	407
Пункт 4.4.8.2. Локальные признаки изображений	408
§ 4.4.9. Онтология действий, выполняемых в предметной области трехмерной реконструкции	409
Пункт 4.4.9.1. Действия по формированию промежуточного трехмерного представления	410
Пункт 4.4.9.2. Действия по формированию итогового трехмерного представления	412
Пункт 4.4.9.3. Действия по подбору и генерации алгоритма	413

Часть 5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения

415

Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

417

*Орлов М.К.**Шункевич Д.В.**Ковалёв М.В.**Садовский М.Е.**Загорский А.Г.**Банцевич К.А.*

§ 5.1.1. Анализ современных библиотек многократно используемых компонентов	419
--	-----

§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем	422
§ 5.1.3. Менеджер многократно используемых компонентов ostis-систем	433
§ 5.1.4. Многократно используемые встраиваемые ostis-системы	434
Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем	435
<i>Бутрин С.В.</i>	
<i>Шункевич Д.В.</i>	
<i>Банцевич К.А.</i>	
§ 5.2.1. Действия и методики проектирования баз знаний ostis-систем	435
§ 5.2.2. Индивидуальный аспект проектирования и разработки баз знаний ostis-систем	436
§ 5.2.3. Коллективный аспект проектирования и разработки баз знаний ostis-систем	436
§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы	436
§ 5.2.5. Логико-семантическая модель ostis-системы обнаружения и анализа информационных дыр в базе знаний ostis-системы	439
§ 5.2.6. Логико-семантическая модель ostis-системы автоматизации управления взаимодействием разработчиков различных категорий в процессе проектирования базы знаний ostis-системы	439
§ 5.2.7. Многократно используемые компоненты баз знаний ostis-систем	440
Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем	441
<i>Шункевич Д.В.</i>	
<i>Марковец В.С.</i>	
§ 5.3.1. Действия и методики проектирования решателей задач ostis-систем	441
§ 5.3.2. Логико-семантическая модель комплекса ostis-систем автоматизации проектирования решателей задач ostis-систем	444
Пункт 5.3.2.1. Семантическая модель базы знаний системы автоматизации проектирования решателей задач ostis-систем	445
Пункт 5.3.2.2. Семантическая модель решателя задач системы автоматизации проектирования решателей задач ostis-систем	448
Пункт 5.3.2.3. Семантическая модель пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем	450
§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем	451
§ 5.3.4. Многократно используемые внутренние агенты ostis-систем	454
Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем	455
<i>Садовский М.Е.</i>	
<i>Жмырко А.В.</i>	
§ 5.4.1. Действия и методики проектирования интерфейсов ostis-систем	455
§ 5.4.2. Логико-семантическая модель ostis-системы автоматизации проектирования интерфейсов ostis-систем	457
§ 5.4.3. Многократно используемые компоненты интерфейсов ostis-систем	457
Часть 6. Платформы реализации интеллектуальных компьютерных систем нового поколения	459
Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем	461
<i>Шункевич Д.В.</i>	
§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению	461
§ 6.1.2. Методы и средства реализации ostis-систем	464
§ 6.1.3. Уточнение понятия ostis-платформы	468
Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем	473
<i>Голенков В.В.</i>	
<i>Шункевич Д.В.</i>	
<i>Гулякина Н.А.</i>	
<i>Захарьев В.А.</i>	
§ 6.2.1. Современное состояние в области разработки компьютеров для интеллектуальных систем	473

§ 6.2.2. Общие принципы, лежащие в основе ассоциативных семантических компьютеров для <i>ostis</i> -систем	474
§ 6.2.3. Архитектура ассоциативных семантических компьютеров для <i>ostis</i> -систем	476
Глава 6.3. Программная платформа <i>ostis</i>-систем	479
<i>Зотов Н.В.</i>	
<i>Шункевич Д.В.</i>	
§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных систем	480
§ 6.3.2. Принципы, лежащие в основе, и аналоги предлагаемого Программного варианта реализации <i>ostis</i> -платформы	483
§ 6.3.3. Реализация sc-памяти в Программной платформе <i>ostis</i> -систем	484
Пункт 6.3.3.1. Формальная спецификация sc-языка внутреннего представления sc-конструкций в sc-памяти <i>ostis</i> -платформы	486
Пункт 6.3.3.2. Синтаксис SCin-кода	486
Пункт 6.3.3.3. Денотационная семантика SCin-кода	489
Пункт 6.3.3.4. Описание процесса и пример трансляции sc-текста в sc-память <i>ostis</i> -платформы	491
Пункт 6.3.3.5. Достоинства и недостатки текущего варианта Реализации sc-памяти и языка внутреннего представления sc-конструкций в sc-памяти <i>ostis</i> -платформы	492
§ 6.3.4. Реализация файловой памяти в Программной платформе <i>ostis</i> -систем	493
Пункт 6.3.4.1. Формальная спецификация sc-языка внутреннего представления конструкций, не принадлежащих SC-коду, в файловой памяти <i>ostis</i> -платформы	494
Пункт 6.3.4.2. Синтаксис SCfin-кода	494
Пункт 6.3.4.3. Денотационная семантика SCfin-кода	495
Пункт 6.3.4.4. Описание процесса и пример трансляции внешних информационных конструкций в файловую память <i>ostis</i> -платформы	495
Пункт 6.3.4.5. Достоинства и недостатки Реализации файловой памяти <i>ostis</i> -платформы и sc-языка внутреннего представления внешних информационных конструкций в файловой памяти <i>ostis</i> -платформы	497
§ 6.3.5. Реализация подсистемы взаимодействия с sc-памятью на основе языка JSON	497
Пункт 6.3.5.1. Формальная спецификация sc-языка представления сообщений между подсистемами Программного варианта реализации <i>ostis</i> -платформы. SC-JSON-код	499
Пункт 6.3.5.2. Синтаксис SC-JSON-кода	499
Пункт 6.3.5.3. Грамматика SC-JSON-кода	500
Пункт 6.3.5.4. Серверная система на основе <i>Websocket</i> и JSON, обеспечивающая сетевой доступ к sc-памяти	502
§ 6.3.6. Реализация интерпретатора sc-моделей пользовательских интерфейсов	505
Пункт 6.3.6.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов	505
Пункт 6.3.6.2. Достоинства и недостатки текущего варианта Реализации интерпретатора sc-моделей пользовательских интерфейсов	506
Часть 7. Экосистема OSTIS	509
Глава 7.1. Структура и проблемы организации человеческой деятельности	511
<i>Голенков В.В.</i>	
<i>Гулякина Н.А.</i>	
§ 7.1.1. Основные понятия, лежащие в основе формального описания структуры человеческой деятельности	511
§ 7.1.2. Структура и проблемы организации человеческой деятельности в области Искусственного интеллекта	515
Пункт 7.1.2.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта	515
Пункт 7.1.2.2. Структура деятельности в области Искусственного интеллекта	518
Пункт 7.1.2.3. Текущее состояние и современные проблемы Искусственного интеллекта	522
Пункт 7.1.2.4. Ключевые задачи и методологические проблемы современного этапа развития Искусственного интеллекта	525
Пункт 7.1.2.5. Комплексная автоматизация человеческой деятельности в области Искусственного интеллекта с помощью интеллектуальных компьютерных систем нового поколения	529
§ 7.1.3. Ключевые виды и области человеческой деятельности	530
§ 7.1.4. Проблемы и перспективы комплексной автоматизации всевозможных видов и областей человеческой деятельности с помощью интеллектуальных компьютерных систем нового поколения	530

Пункт 7.1.4.1. Общие принципы систематизации человеческой деятельности и ее комплексной автоматизации с помощью интеллектуальных компьютерных систем нового поколения	530
Пункт 7.1.4.2. Многообразие видов человеческой деятельности и связей между ними	531
Глава 7.2. Метасистема OSTIS	537
<i>Голенков В.В.</i>	
<i>Шункевич Д.В.</i>	
<i>Банцевич К.А.</i>	
<i>Загорский А.Г.</i>	
§ 7.2.1. Структура, назначение, особенности и достоинства Метасистемы OSTIS	538
Пункт 7.2.1.1. Структура Метасистемы OSTIS	538
Пункт 7.2.1.2. Назначение Метасистемы OSTIS	538
Пункт 7.2.1.3. Особенности Метасистемы OSTIS	539
Пункт 7.2.1.4. Достоинства Метасистемы OSTIS	539
§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS	539
Пункт 7.2.2.1. Оглавление Стандарта OSTIS	539
Пункт 7.2.2.2. Общая структура Стандарта OSTIS	540
Пункт 7.2.2.3. Ключевые знаки Стандарта OSTIS	541
Пункт 7.2.2.4. Назначение Стандарта OSTIS	541
Пункт 7.2.2.5. Аналоги Стандарта OSTIS	542
Пункт 7.2.2.6. Особенности Стандарта OSTIS	542
Пункт 7.2.2.7. Пользователь Стандарта OSTIS	543
Пункт 7.2.2.8. Авторский коллектив Стандарта OSTIS	543
Пункт 7.2.2.9. Редакционная коллегия Стандарта OSTIS	544
Пункт 7.2.2.10. Требования, предъявляемые к Стандарту OSTIS	545
Пункт 7.2.2.11. Правила построения Стандарта OSTIS	546
Пункт 7.2.2.12. Направления развития Стандарта OSTIS	547
Пункт 7.2.2.13. Достоинства Стандарта OSTIS	548
Пункт 7.2.2.14. Стандарт OSTIS как основная часть базы знаний Метасистемы OSTIS	548
Глава 7.3. Структура Экосистемы OSTIS	553
<i>Загорский А.Г.</i>	
<i>Голенков В.В.</i>	
<i>Шункевич Д.В.</i>	
§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ	554
Пункт 7.3.1.1. Формальная модель Экосистемы интеллектуальных компьютерных систем нового поколения	554
Пункт 7.3.1.2. Поддержка совместимости между ostis-системами, входящими в состав Экосистемы OSTIS	555
Пункт 7.3.1.3. Описание структуры Экосистемы OSTIS	556
§ 7.3.2. Автоматизация человеческой деятельности в области Искусственного интеллекта, осуществляемая в рамках Экосистемы OSTIS	558
§ 7.3.3. Принципы автоматизации различных видов и областей человеческой деятельности в рамках Экосистемы OSTIS	558
§ 7.3.4. Семантически совместимые интеллектуальные ostis-порталы научных знаний	558
§ 7.3.5. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения	559
§ 7.3.6. Персональные ostis-ассистенты пользователей	560
Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами	563
<i>Загорский А.Г.</i>	
<i>Коршунов Р.А.</i>	
<i>Таранчук В.Б.</i>	
<i>Шункевич Д.В.</i>	
§ 7.4.1. Общие принципы интеграции Экосистемы OSTIS с современными сервисами и информационными ресурсами	563
§ 7.4.2. Принципы интеграции Экосистемы OSTIS с разнородными сервисами	564
§ 7.4.3. Принципы интеграции Экосистемы OSTIS со структурированными информационными ресурсами	565
§ 7.4.4. Интеграция инструментов компьютерной алгебры в приложения OSTIS	565

Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS	567
<i>Гулякина Н.А.</i>	
<i>Козлова Е.И.</i>	
<i>Гракова Н.В.</i>	
<i>Осъкин А.Ф.</i>	
<i>Головатый А.И.</i>	
<i>Самодумкин С.А.</i>	
<i>Ли В.</i>	
§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем	567
§ 7.5.2. Автоматизация среднего образования с помощью ostis-систем	569
§ 7.5.3. Автоматизация высшего образования с помощью ostis-систем	575
§ 7.5.4. Семантические модели и средства контроля знаний пользователей в ostis-системах	576
§ 7.5.5. Дидактические знания	576
Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов	577
<i>Андрющевич А.А.</i>	
<i>Войтешенко И.С.</i>	
§ 7.6.1. Анализ существующих подходов к созданию умных домов	577
§ 7.6.2. Предлагаемый подход к созданию умных домов	578
§ 7.6.3. Многокомпонентная модель умных домов	579
§ 7.6.4. Подсистемы умного дома	580
Пункт 7.6.4.1. Подсистема доступа в жилое помещение	581
Пункт 7.6.4.2. Подсистема наблюдения за одинокими пожилыми людьми	582
Пункт 7.6.4.3. Подсистема для управления освещенностью жилища	582
Пункт 7.6.4.4. Подсистема управления энергопотреблением и энергоэффективностью	582
§ 7.6.5. Элементы технической реализации умных домов	583
Пункт 7.6.5.1. Запуск Node-RED на виртуальной машине в Yandex Cloud	584
Пункт 7.6.5.2. Создание объектов сервиса Yandex IoT Core. Интеграция с Node-RED	584
Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS	585
<i>Иванюк Д.С.</i>	
<i>Таберко В.В.</i>	
<i>Прохоренко В.А.</i>	
<i>Смородин В.С.</i>	
§ 7.7.1. Адаптивное управление технологическим циклом производства на основе Технологии OSTIS	585
Пункт 7.7.1.1. Онтология предметной области “технологические процессы с вероятностными характеристиками”	585
Пункт 7.7.1.2. Решатели задач ostis-системы адаптивного управления вероятностным технологическим процессом производства	595
§ 7.7.2. Построение умных предприятий рецептурного производства с помощью ostis-систем	601
Пункт 7.7.2.1. Проблемы проектирования умных предприятий	601
Пункт 7.7.2.2. Подходы к проектированию и стандартизации умных предприятий рецептурного производства с помощью ostis-систем	603
Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения	607
<i>Самодумкин С.А.</i>	
§ 7.8.1. Требования, предъявляемые к интеллектуальным геоинформационным системам нового поколения	607
§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами	607
§ 7.8.3. Основные компоненты формальных онтологий, используемых в геоинформационных системах	607
Пункт 7.8.3.1. Геосемантические элементы: местоположение, топология, близость, ориентация, динамика	607
Пункт 7.8.3.2. Формализация пространственных отношений в геоинформационных системах	607
§ 7.8.4. Стратифицированная модель информационного пространства объектов местности	607
§ 7.8.5. Формальная денотационная семантика языка карт	607
§ 7.8.6. Формальная онтология объектов местности	607
§ 7.8.7. Средства автоматизации проектирования интеллектуальных геоинформационных систем	607
Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS	609

*Чертов В.М.**Захаров В.В.**Крищенович В.А.*

§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения	609
§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем	611

Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии**615***Голенков В.В.**Гулякина Н.А.***Используемые сокращения и предметный указатель OSTIS** **619****Библиография OSTIS** **627**

Предисловие

Я уже не раз на площадке Гармиша упоминал о проблемах построения суверенитета в небольших молодых странах. Сегодня я хотел бы коснуться одной общей проблемы обеспечения суверенитета, верной практически для любого государства.

Всеобщая цифровизация потребовала массовой разработки софта и в "программисты" массово хлынула молодежь, начиная уже со школьной скамьи.

Действительно, зачем толковым школьникам идти на инженерные, естественнонаучные, конструкторские, сложносистемные специальности, если можно быстро вписаться в "технологический конвейер" разработки софта и гарантированно получать зарплату, которая намного выше зарплаты инженеров, конструкторов, проектировщиков сложных систем? И этот технологический конвейер быстро и массово превращает, среднестатистического программиста в ремесленника, даже, наверное, более точно в ИТ-пролетариат. Причем это не локальный ИТ-пролетариат, а глобальный ИТ-пролетариат (можно сказать даже глобально управляемый), формируемый по клише "граждан мира". Хотя, с резким торможением глобализации перспективы гражданства мира стали не такими уж и очевидными и ясными.

Известный английский историк А.Дж.Тайнби в свое время ввел понятие "опасные классы". Он использовал его при описании процессов конца 18-начала 19 веков, когда вырванное из сельской местности население превратилось в "опасные классы". За десятилетия удалось их превратить в более-менее устойчивый пролетариат. Возможно сейчас процессы с опасными классами повторяются, в частности, с айтишниками, которые оказываются мало приспособленными к реалиям современной жизни. Но глобализация с цифровизацией превратила их в "граждан мира" (как им очень хотелось), но в виде недоформированного ИТ-пролетариата (что их жестко и неприятно приземлило).

Сейчас затруднительно адекватно использовать термины айтишник, программист. Всеобщая цифровизация практически быстро размывает эти понятия. На постсоветском пространстве понятие айтишник, а иногда даже и программист часто рассматривается как нечто единое, объединяющее и тестировщиков, и кодировщиков, и специалистов по профессиональному интеллигенту, анализу больших данных, и системных архитекторов и т.д. В современной мировой практике этой цельности нет – все это совершенно разные категории специалистов. Поэтому, когда иногда говорят о новом классе – айтишников, то такого класса строго говоря и нет. Есть, наверное, достаточно массовый айтишный пролетариат (ИТ-пролетариат), которому очень хотелось бы идентифицировать себя с хайтеком. Но, если специалисты в области искусственного интеллекта, анализа больших данных, системные архитекторы, бизнес-аналитики, получившие серьезное образование, будут востребованы, то профессии тестировщика и кодировщика и близкие к ним, но с модными названиями могут быстро исчезнуть. В частности, искусственный интеллект способен этому сильно помочь. А своего рода миф о том, что нужно так много программистов отчасти поддерживается тем, что просто создается очень много некачественного кода (софта).

Мы всегда должны помнить, что в эпоху всеобщей цифровизации роль технологического суверенитета резко возрастает. Но без молодежи невозможно развивать технологический суверенитет. А молодежь мы упускаем. То есть молодежь вроде бы и стремится к новым технологиям, мы вроде бы и поощряем такое стремление, но этот процесс какой-то однобокий. Возьмем опять ИТ-сферу на постсоветском пространстве.

В университетах, где готовят ИТ- специалистов, много учебных центров и классов по Microsoft, Oracle, Cisco, SAP и т. п. Но это не научно-исследовательские центры, а банальные центры навыков, компетенций по простому кодированию, тестированию, внедрению и эксплуатации. Конечно, это неплохо, если бы в итоге не подменялись основные университетские курсы. Студенты видели, что в таких центрах разработки не нужны серьезная математика, физика. Терялась мотивация. Наиболее способные молодые специалисты плавно перетекали из центров разработки в центры за пределами страны, где этапы жизненных циклов разработки более интересны. Как правило, они не возвращались. Например, для нашей небольшой страны это существенные потери. Как выясняется, нередко в подобной эмиграции главное даже не заработка, а возможность творчества вместо ремесла и промышленного

программирования. По сути, такая модель ускорила разрушение инженерного, конструкторского, общесистемного образования. Утрачивалась мотивация к изучению математических, физических, сложносистемных дисциплин. Действительно, студенту незачем тратить время на их глубокое освоение, если достаточно получить навыки по кодированию и тестированию софта на основе несложных технологических платформ и легко найти работу в одной из конвейерных компаний по разработке софта. По сути образование подстраивалось под такую модель и планомерно разрушалось при активном участии тех же западных конвейеров.

Вспомним, что традиционные западные обыватели мало интересуются глобальными вещами. Их интерес в основном локален. Например, решения муниципальных властей для них интереснее решений властей штата или земли, не говоря уже о решениях федеральных властей. Конечно могут быть исключения, но они лишь, как обычно, подтверждают общность. Точно таким же формируется у нас и молодое поколение, особенно айтишное. Действительно система образования и воспитания не воспроизводит сейчас системность во взглядах и оценках. Хорошая системность предполагает умение анализировать и синтезировать информацию, в том числе и по-глобальному. Советская система образования и воспитания, пусть и несколько односторонне, все-таки этому способствовала. Теперь же, в связи со старением и уходом поколений-носителей такой концепции, такая система образования и воспитания полностью исчезает. А новая не создана, вернее лихорадочно, бессистемно создается как ухудшенная калька с быстроразрушающейся западной системы, причем, зачастую, даже не с исконно западной, а как калька с восточно-европейской кальки. И у молодежи формируется основная парадигма, зачем думать о глобальном, системно, главное, чтобы в локальном мире было все хорошо и как следствие нынешние молодежные установки: "жизнь в моменте", "хорошо в моменте", "хорошо здесь и сейчас" и тому подобное.

И именно айтишный пролетариат являются основным носителем таких установок. И понятно почему. Они работают практически полностью в западном (американском) конвейере, причем выполняя в основном самые примитивные операции, то есть работают как современный ИТ-пролетариат, даже не как инженера, технологии, не говоря уже о конструкторах или архитекторах. То есть совершенно не элита.

По такой модели мы получили лишь ускорение потери суверенитета технологического, и как следствие разрушение странового суверенитета. Хотя последние несколько лет наконец-то многие в мире спохватились, что нужен суверенитет для стран. Глобалистский мир оказался не столь уж безопасен и привлекателен, как был распиарен за последние лет тридцать, граждане мира, как мировые кочевники, не столь уж и конструктивны и креативны – собственно, все как всегда: хороша только виртуальная картинка, в существенно иных условиях идеи глобализма могли бы быть и продуктивны, но народ не тот, элиты не те и тому подобное. Модно стало говорить о цифровом суверенитете, суверенитете в области высоких технологий, суверенитете в области ИКТ и так далее. Соответственно, откликаясь на моду, хайп (а сейчас многое пытаются делать на волнах хайпа) стали писать концепции и стратегии для разного вида суверенитета. Хотя после стольких лет построения глобалистского мира (как видим теперь не очень-то удачного), строить суверенитеты многим странам крайне затруднительно.

Для небольших стран хотелось бы отметить следующее.

В условиях достаточно жесткого противостояния и противоборства нет смысла стремится к топовым с точки зрения глобальных экспертов, существующих, как обычно, на средства ТНК, ИКТ-решениям. Те же топовые чипы 2-3 нано зачем нужны на данном этапе для оборонки, государства, индустрии, социальной сферы и т.п. Можно и 90 нано обойтись. Не нужно путать требования потребительского рынка и требования для решения государственных, странных задач. Тот же Apple, или Microsoft, или Samsung, создавая гаджеты и сервисы для конечного потребителя решают совершенно иные задачи. Им нужно удержать лидерство на рынке, чтобы сохранить миллиардные прибыли, Отсюда и 2-3 нано в чипах, и все новые и новые операционные системы, и все новые и новые приложения. Бег по кругу, новый системный и прикладной софт требует нового железа, новое железо требует нового софта – и за все расплачивается конечный потребитель. А решение задачи сохранения суверенитета страны в текущей ситуации такой безудержной гонки совершенно и не требует. И чипы нужны попроще, но понадежнее, и операционные системы нужны не навороченные, и приложения нужны не такие уж многофункциональные. Классический пример с MS Word. Обычный пользователь использует не более 5% возможностей редактора. А, например, госслужащим и им подобным и 5% много. Точно также и с большинством софта. Нужны гораздо более простые, но стабильные и надежные системы. И тогда, чтобы создать такие системы нужны не сотни тысяч ИТ-пролетариата, а сотни, ну может быть тысячи высококвалифицированных разработчиков и относительно небольшое количество ИТ-пролетариев. Еще раз подчеркну, страна для своего суверенитета и транснациональный ИТ-бизнес решают совершенно разные задачи и преследуют совершенно разные цели. Можно еще напомнить, что массы ИТ-пролетариата бизнесу нужны и для того, чтобы раздувать значимость фирмы, создавать иллюзию гигантизма, особенно, если твоя фирма на IPO в Нью-Йорке или Лондоне, или хотя бы в Гонконге. Понятно, что массовый инвестор больше поверит в фирму с десятками тысяч работников, чем в небольшую команду.

Но коль уж мы от цифровизации никуда не денемся, то нужна цифровизация хорошо продуманная, просчитанная, с глубоким прогнозированием последствий, а не безудержная, с агрессивной рекламой, что создает у людей совершенно ложные ориентиры. В подавляющем случае это делается опять же непрофессионально айтишным пролетариатом, втянутым в разработку на волне хайпа о неограниченной цифровизации всего и вся. А для того чтобы это понимать, молодежи нужно соответствующее образование. И выбора нет – либо мы такое образование очень

оперативно сформируем, либо окончательно потеряем думающую молодежь, и, соответственно, окончательно потеряем и суверенитет в условиях перехода информационных технологий на принципиально новый уровень, обеспечивающий создание эффективно взаимодействующих (интероперабельных) интеллектуальных компьютерных систем.

Минск,
** 2023

A.H. Курбацкий

Авторское предисловие

Ключевой особенностью **компьютерных систем нового поколения** является их **интероперабельность**, т.е. способность к эффективному осознанному взаимодействию, необходимым условием которой является способность к взаимопониманию, т.е. **семантическая совместимость**. Таким образом, каждая **компьютерная система нового поколения** должна:

- знать свои обязанности и возможности;
- уметь координировать свои действия с другими компьютерными системами нового поколения
 - в ходе коллективного решения сложных задач;
 - в заранее не предусмотренных (нештатных) обстоятельствах.

Переход от современных компьютерных систем (в том числе, и интеллектуальных компьютерных систем) к компьютерным системам нового поколения, которые, очевидно, должны обладать достаточно высоким уровнем интеллекта, означает переход к принципиально новому технологическому укладу в области автоматизации различных видов человеческой деятельности и предполагает переосмысление и использование всего опыта, накопленного при разработке и эксплуатации самого различного вида **компьютерных систем**. Поэтому участие в проекте создания **Технологии комплексной поддержки жизненного цикла компьютерных систем нового поколения** не требует от специалистов радикального изменения области их научных интересов. Требуется просто учет дополнительных требований к формализации своих результатов. Основным из этих требований является **семантическая совместимость** с другими (смежными) результатами.

К настоящему моменту завершен первый этап разработки **Технологии поддержки жизненного цикла компьютерных систем нового поколения**, которая названа нами **Технологией OSTIS** (Open Semantic Technology for Intelligent Systems). Началом указанного первого этапа является первая конференция OSTIS (Минск, 10 – 12 февраля 2011 года).

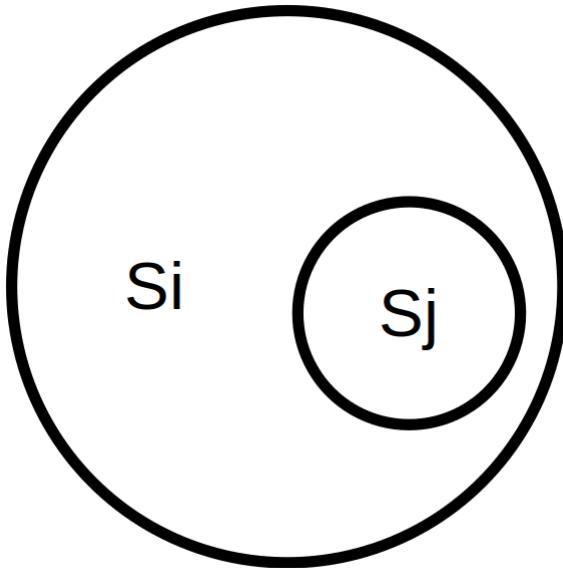
К текущему моменту сформировался работоспособный стартовый авторский коллектив в рамках созданного Учебно-научного объединения по Искусственному интеллекту, в состав которого входят ведущие университеты РБ (в четырех из которых открыта специальность “Искусственный интеллект”), и ряд других организаций.

Результаты первого этапа разработки **Технологии OSTIS** отражены в материалах проведенных **конференций OSTIS**, в первой версии формализованного **Стандарта Технологии OSTIS** и в первой версии **Метасистемы OSTIS**, которая непосредственно и осуществляет автоматизацию проектирования, реализации, реинжиниринга и эксплуатации **интеллектуальных компьютерных систем нового поколения** и гарантирует поддержку их **семантической совместимости** и интероперабельности не только на этапе проектирования, но и в ходе эксплуатации.

На следующем этапе разработки **Технологии поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения** требуется существенное расширение фронта работ и соответствующего авторского коллектива. Этому, в частности, была посвящена конференция OSTIS-2022 (24-26 ноября 2022 года), которая внесла важный вклад в развитие Открытого проекта развития технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения (Проекта OSTIS).

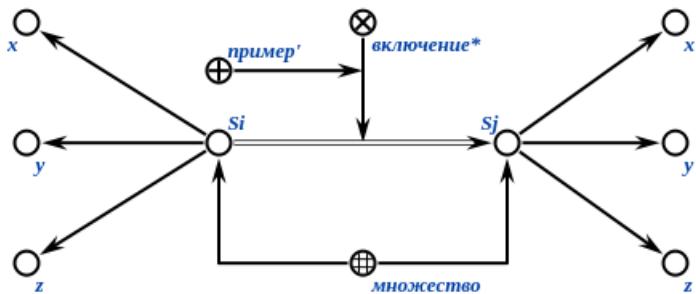
Примеры рисунков

Рисунок [Пример обычного рисунка](#) показывает, как надо оформлять рисунки. Размер рисунка можно задавать при помощи параметра scale.



= Пример обычного рисунка

Рисунок [Пример рисунка в SCg-коде](#) показывает, как надо оформлять рисунки в SCg-коде. Параметр scale должен быть выставлен равным 0.8, для того чтобы все рисунки в SCg-коде имели одинаковый масштаб и размер идентификаторов примерно соответствовал размеру шрифта основного текста.



= Пример рисунка в SCg-коде

Правила оформления scn-текста

Для оформления списков внутри естественно-языкового файла, находящегося в рамках формального sc.n-текста, необходимо использовать окружение `scnitemize`:

```
\begin{scnitemize}
    \item text
\end{scnitemize}
```

Для оформления списков в рамках естественного-языкового текста (не SCn), следует использовать окружение `textitemize`:

```
\begin{textitemize}
    \item text
\end{textitemize}
```

Использование шрифтов

```
\underline{text} or \uline{text}
\textit{text}
\emph{text}
\textbf{text}
\textup{text}
```

Результат компиляции:

подчеркивание или подчеркивание

курсив

выделенный шрифт

полужирный

прямой шрифт

Использование символов

```
text\scnrolesign
text\textasciicircum
```

Результат компиляции:

текст'

текст^

Рассмотрим примерную структуру описания scn-конструкции. Пример исходного кода:

```
\begin{SCn}
    \scnheader{header}
    \scnidtf{frequently used sc-identifier}{text}
    \scnidtf{text}
    \scniselement{text}
    \scnhaselement{text}
    \scnsuperset{text}
    \scnsubset{text}
    \scnidtfdef{text}
    \scndefinition{text}
    \scnidtfexp{text}
    \scnexplanation{text}
    \scntext{text}
    {Text
        \begin{textitemize}
            \item text;
            \item text
        \end{textitemize}
    }
    \scnnote{text}
    \scncomment{text}
    %dividing
    \begin{scnsubdividing}
        \scnitem{text}
        \scnitem{text}
        %nesting change
        \begin{scnidt}
            \scnidtf{text}
            \scneq{\textup{() text \textup{()}}}
            \begin{scneqset}
                \scnitem{text}
                \scnitem{text}
            \end{scneqset}
            \end{scnidt}
            \end{scnsubdividing}
            \scnrelfrom{dividing}{attribute}
            \begin{scnidt}
                \begin{scneqset}
```

```

        \scnitem{text}
        \scnitem{text}
\end{scnneqtoset}
\begin{scnindent}
\begin{scnrelfromset}{relation}
        \scnitem{text}
        \scnitem{text}
\end{scnrelfromset}
\begin{scnrelfromset}{relation}
        \scnfileitem{text}
        \scnfileitem{text}
\end{scnrelfromset}
\begin{scnrelfromlist}{note}
        \scnfileitem{text}
        \scnfileitem{text}
\end{scnrelfromlist}
\scnrelboth{analog}{dividing*}
\begin{scnrelfromlistcustom}{principles underlying}
        \scnitemcustom{text}
\end{scnrelfromlistcustom}
\scnrelfrom{second domain}{text}
\end{SCn}

```

Результат компиляции:

ключевой sc-элемент

:= часто используемый *sc-идентификатор**:
 [text]

:= [сионимичный идентификатор]

∈ принадлежность

∃ принадлежность

▷ строгое надмножество — «включает в себя»

⊍ строгое подмножество — «включено в»

:= определение*:
 [определение]

⇒ определение*:
 [определение]

:= пояснение*:
 [пояснение]

⇒ пояснение*:
 [пояснение]

⇒ пояснение*:
 [Текст
 • текст;
 • текст.
]

⇒ примечание*:
 [примечание]

⇒ комментарий*:
 [комментарий]

⇒ разбиение*:
 {• текст
 • текст
 := [текст]
 = (текст)
 = {• текст
 • текст
 }

}

⇒ разбиение*:
 Признак
 = {• текст
 • текст
 }

⇒ отношение*:
 {• текст
 • текст
 }

- ⇒ *отношение**:
 - {• [текст]
 - [текст]
- ⇒ *примечание**:
 - [текст]
 - [текст]
- ↔ *аналог**:
 - разбиение**
- ⇒ *принципы, лежащие в основе**:
 - текст
 - текст
- ⇒ *второй домен**:
 - текст

Пример исходного кода:

```
\begin{SCn}
  \scnheader{relation*}
  \scnrelto{second domain}{text}
\end{SCn}
```

Результат компиляции:

```
отношение*
⇐ второй домен*:
текст
```

Пример исходного кода (следует отличать):

```
\begin{SCn}
  \scnheader{should be distinguished*}
  \begin{scnhaselementset}
    \scnitem{text}
    \scnitem{text}
  \end{scnhaselementset}
\end{SCn}
```

Результат компиляции:

```
следует отличать*
∃ {• текст
  • текст
}
```

Часть 1.

Введение в интеллектуальные компьютерные системы нового поколения и технологии комплексной поддержки их жизненного цикла

⇒ *аннотация**:

[Уточнение понятия интеллектуальной системы. Требования предъявляемые в интеллектуальным компьютерным системам следующего (нового) поколения и принципы, лежащие в их основе. Структура и особенности жизненного цикла интеллектуальных компьютерных систем нового поколения. Технология комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения (Технология OSTIS — Open Semantic Technology for Intelligent Systems)]

⇒ *подраздел**:

- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

Глава 1.1.

Факторы, определяющие уровень интеллекта кибернетических систем

⇒ *автор**:

- Загорский А.С.
- Голенков В.В.
- Шункевич Д.В.

⇒ *аннотация**:

[Рассмотрена иерархическая система свойств (в т.ч. способностей) кибернетических систем, определяющих их качество и позволяющих сформулировать требования, которым должна удовлетворять высокоинтеллектуальная система (кибернетическая система с сильным интеллектом). Уровень качества кибернетических систем определяется достаточно большим набором свойств (параметров, характеристик) кибернетических систем, каждое из которых определяет уровень качества кибернетической системы в соответствующем аспекте (рассмотренном), указывая (задавая) уровень развития конкретных способностей и возможностей кибернетической системы. При этом важно подчеркнуть следующее:

- существенное значение имеет не столько сам набор свойств, а иерархия этих свойств, позволяющая уточнять направления проявления каждого свойства;
- существенное значение также имеет баланс уровней развития различных свойств — вклад разных свойств, обеспечивающих значение одного и того же свойства более высокого уровня иерархии, а значение этого свойства более высокого уровня может быть разным. Из этого следует, что не всегда следует акцентировать внимание на развитие некоторых свойств (характеристик). Нужен целостный, коллективный подход;
- рассмотренная иерархия свойств кибернетических систем является общей как для естественных, так и для искусственных кибернетических систем;
- приведенная иерархическая детализация свойств кибернетических систем (с помощью отношения “частное свойство*” и отношения “свойство-предпосылка*”), определяющих качество таких систем, (1) дает возможность четко определить направления совершенствования (развития) кибернетических систем и (2) дает ориентир (систему критериев) для обоснования конкретных предложений по совершенствованию компьютерных систем, а также для сравнения различных альтернативных предложений;
- особое значение для развития кибернетических систем имеют такие их свойства, как стратифицированность, рефлексивность и социализация;
- важное значение имеет не только совершенствование кибернетических систем в соответствии с иерархической системой их свойств, но и совершенствование (в том числе, детализация) самой этой иерархической системы свойств.

Свойства (способности), которым должны удовлетворять *интеллектуальные системы*, рассматриваются в целом ряде публикаций. Тем не менее, для *практической реализации компьютерных систем*, обладающих указанными свойствами (способностями), то есть *интеллектуальных компьютерных систем*, необходимо детализировать (уточнить) эти *свойства*, пытаясь свести их к более конструктивным, прозрачным и понятным для реализации свойствам.]

⇒ *подраздел**:

- § 1.1.1. Понятие интеллектуальной кибернетической системы
- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 1.1.3. Эволюция традиционных и интеллектуальных компьютерных систем

§ 1.1.1. Понятие интеллектуальной кибернетической системы

⇒ *ключевое понятие**:

- кибернетическая система
- качество кибернетических систем^λ
- уровень интеллекта кибернетических систем^λ
- интеллектуальная кибернетическая система

⇒ *ключевое знание**:

- Классификация кибернетических систем
- Обобщенная архитектура кибернетических систем
- Факторы, определяющие качество кибернетических систем
- Факторы, определяющие уровень интеллекта кибернетических систем

⇒ библиографическая ссылка*:

- Глушков В.М.Кибернетика-1979ст
- Варшавский В.А..ОркесИБД-1984кн
- Finn2021
- Nilsson N.J.HumanLAIBS-2005art
- Kerr C..aConceMftI-2006art
- Анцыферов С.С.ОценкКИС-2013art
- Fry R.L.tEngin oCS-2002art
- Melekhova O..aDecenSfCD-2018art
- Ouksel A.M..SemantGIS-1999art
- Lopes L.S..SemantIoT-2022art
- Hamilton Inter aKEftG-2006art
- Neiva F.W..TowarPItSC-2016art

⇒ подраздел*:

- Пункт 1.1.1.1. Типология кибернетических систем
- Пункт 1.1.1.2. Структура кибернетической системы
- Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы
- Пункт 1.1.1.4. Комплекс свойств, определяющих качество физической оболочки кибернетической системы
- Пункт 1.1.1.5. Комплекс свойств, определяющих качество информации, хранимой в памяти кибернетической системы
- Пункт 1.1.1.6. Комплекс свойств, определяющих качество решателя задач кибернетической системы
- Пункт 1.1.1.7. Комплекс свойств, определяющих уровень обучаемости кибернетической системы
- Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы

Кибернетическая система есть система, которая способна управлять своими действиями, адаптируясь к изменениям состояния внешней среды (среды своего "обитания") в целях самосохранения (сохранения своей целостности и "комфортности" существования путем поддержания своих "жизненно" важных параметров в определенных рамках "комфортности") и в целях формирования определенных реакций (воздействий на внешнюю среду) в ответ на определенные стимулы (на определенные ситуации или события во внешней среде), а также которая способна (при соответствующем уровне развития) эволюционировать в направлении:

- изучения своей внешней среды как минимум для предсказания последствий своих воздействий на внешнюю среду, а также для предсказания изменений внешней среды, которые не зависят от собственных воздействий;
- изучения самой себя и, в частности, своего взаимодействия с внешней средой;
- создания технологий (методов и средств), обеспечивающих изменение своей внешней среды (условий своего существования) в собственных интересах.

В работе *Глушков В.М.Кибернетика-1979ст* сформировано определение кибернетической системы.

кибернетическая система

- := [адаптивная система]
- := [целенаправленная система]
- := [активный субъект самостоятельной деятельности]
- := [материальная сущность, способная целенаправленно (в своих интересах) воздействовать на среду своего обитания как минимум для сохранения своей целостности, жизнеспособности, безопасности]
- ⇒ **примечание*:**
 - [Уровень (степень) адаптивности, целенаправленности, активности у систем, основанных на обработке информации может быть самым различным.]
- := [система, организация функционирования которой основано на обработке информации о той среде, в которой существует эта система]
- := [материальная сущность, способная к активной целенаправленной деятельности, которая на определенном уровне развития указанной сущности становится "осмысленной планируемой, преднамеренной деятельностью"]
- := [субъект, способный на самостоятельное выполнение некоторых "внутренних" и "внешних" действий либо порученных извне, либо инициированных самим субъектом]
- := [сущность, способная выполнять роль субъекта деятельности]

- := [естественная или искусственно созданная система, способная мониторить и анализировать свое состояние и состояние окружающей среды, а также способная достаточно активно воздействовать на собственное на собственное состояние и на состояние окружающей среды]
- := [система, способная в достаточной степени самостоятельно взаимодействовать со своей средой, решая различные задачи]
- := [система, основанная на обработке информации]

Термин "кибернетическая система" имеет четкое количественное определение. Это система, которая динамически сопоставляет полученную информацию с выбранными действиями, относящимися к вычислительной задаче, которая определяет основную цель системы или машины *Fry R.L.Engin oCS-2002art.*

Пункт 1.1.1.1. Типология кибернетических систем

кибернетическая система

⇒ разбиение*:

Признак естественности или искусственности кибернетических систем

- = {• естественная кибернетическая система
 - := [кибернетическая система естественного происхождения]
 - ▷ человек
 - компьютерная система
 - := [искусственная кибернетическая система]
 - := [кибернетическая система искусственного происхождения]
 - := [технически реализованная кибернетическая система]
 - симбиоз естественных и искусственных кибернетических систем
 - := [кибернетическая система, в состав которой входят компоненты как естественного, так и искусственного происхождения]
 - ▷ сообщество компьютерных систем и людей
- }

Особенностью компьютерных систем является то, что они могут выполнять "роль" не только продуктов соответствующих действий по реализации этих систем, но и сами являются *субъектами**, способными выполнять (автоматизировать) широкий спектр действий. При этом интеллектуализация этих систем существенно расширяет этот спектр.

кибернетическая система

⇒ разбиение*:

Структурная классификация кибернетических систем

- = {• простая кибернетическая система
 - индивидуальная кибернетическая система
 - многоагентная система
}

⇒ разбиение*:

Классификация кибернетических систем по признаку наличия надсистемы и роли в рамках этой надсистемы

- = {• кибернетическая система, не являющаяся частью никакой другой кибернетической системы
 - := [кибернетическая система, не имеющая надсистем]
 - кибернетическая система, встроенная в индивидуальную кибернетическую систему
 - агент многоагентной системы
 - := [кибернетическая система, являющаяся агентом одной или нескольких многоагентных систем]
}

Простая кибернетическая система есть *кибернетическая система*, уровень развития которой находится ниже уровня *индивидуальных кибернетических систем* и которая является специализированным специализированным решателем задач, реализующим (интерпретирующими) чаще всего один *метод* решения задач и, соответственно, решающим только *задачи* заданного *класса задач*. Простая кибернетическая система может быть *компонентом**, встроенным в *индивидуальную кибернетическую систему*, а также может быть *агентом** *многоагентной системы*, являющейся коллективом из простых кибернетических систем.

Индивидуальная кибернетическая система — условно выделенный уровень развития *кибернетических систем*, в основе которого лежит переход от *специализированного решателя задач* к *индивидуальному решателю задач*,

обеспечивающему интерпретацию произвольного (нефиксированного) набора *методов* (программ) решения задач при условии, если эти *методы* введены (загружены, записаны) в память кибернетической системы. Признаками индивидуальных кибернетических систем являются:

- наличие *памяти*, предназначенной для хранения как минимум интерпретируемых *методов* (программ) и обеспечивающей корректировку (редактирование) хранимых *методов*, а также их удаление из *памяти* и ввод (запись) в память новых *методов*;
- легкая возможность "перепрограммировать" кибернетическую систему на решение других задач, что обеспечивается наличием *универсальной модели решения задач* и, соответственно, *универсальным интерпретатором любых моделей*, представленных (записанных) на соответствующем языке;
- наличие пусть даже простых средств коммуникации (обмена информацией) с другими кибернетическими системами (например, с людьми);
- способность входить в различные коллективы кибернетических систем.

Класс индивидуальных кибернетических систем — это определенный этап эволюции кибернетических систем, означающий переход к кибернетическим системам, которые способны самостоятельно "выживать". Индивидуальная кибернетическая система может быть агентом (членом) многоагентной системы (членом коллектива индивидуальных кибернетических систем), но некоторые многоагентные системы могут состоять из агентов, не являющихся индивидуальными кибернетическими системами, представляющих собой простые специализированные кибернетические системы, выполняющие достаточно простые действия ([Степанюк. В.Л. ЛокалОргИС-2004кн , Д.фон. Нейман. ТеорСамовосАвтомат-1971кн](#)).

Примерами кибернетической системы, встроенной в индивидуальную кибернетическую систему, являются sc-агент *ostis*-системы, решатель задач *ostis*-системы.

Многоагентная система представляет собой коллектив взаимодействующих автономных кибернетических систем, имеющих общую среду обитания (жизнедеятельности).

многоагентная система

⇒ разбиение*:

- {• коллекти夫 из простых кибернетических систем
- коллекти夫 из индивидуальных кибернетических систем
- коллекти夫 из индивидуальных и простых кибернетических систем

}

⇒ разбиение*:

- {• одноуровневый коллектив кибернетических систем
 - := [многоагентная система, агентами которой не могут быть многоагентные системы]
- иерархический коллектив кибернетических систем
 - := [многоагентная система, по крайней мере одним агентом которой является многоагентная система]

}

Коллектив индивидуальных кибернетических систем — многоагентная система, агентами (членами) которой являются индивидуальные (!) кибернетические системы. Примерами коллектива индивидуальных кибернетических систем могут быть как коллективы людей, так и коллективы компьютерных систем и людей.

Одноуровневый коллектив кибернетических систем определён как специализированное средство решения задач, реализующее либо одну модель параллельного (распределенного) решения задач соответствующего класса, либо комбинацию фиксированного числа разных и параллельно реализованных моделей решения задач.

Агентами иерархического коллектива кибернетических систем могут быть:

- индивидуальные кибернетические системы;
- коллективы индивидуальных кибернетических систем;
- коллективы, состоящие из индивидуальных кибернетических систем и коллективов индивидуальных кибернетических систем и т.д.

Пункт 1.1.1.2. Структура кибернетической системы

Рассмотрим структуру кибернетической системы.

кибернетическая система

⇒ обобщенная декомпозиция*:

- {• информация, хранимая в памяти кибернетической системы
- абстрактная память кибернетической системы
- решатель задач кибернетической системы

- физическая оболочка кибернетической системы
- }

Информация, хранимая в памяти *кибернетической системы*, представляет собой информационную модель среды, в которой действует (существует, функционирует) эта *кибернетическая система*. Текущее состояние памяти кибернетической системы.

Абстрактная память кибернетической системы есть внутренняя абстрактная информационная среда кибернетической системы, представляющая собой динамическую информационную конструкцию, каждое состояние которой есть не что иное, как информация, хранимая в памяти кибернетической системы в соответствующий момент времени. Является подмножеством динамической информационной конструкции.

Решателем задач кибернетической системы называют совокупность всех навыков (умений), приобретенных кибернетической системой к рассматриваемому моменту. Встроенный в кибернетическую систему субъект, способный выполнять целенаправленные ("осознанные") действия во внешней среде этой кибернетической системы, а также в её внутренней среде (в абстрактной памяти).

Рассмотрим подробнее понятие действия кибернетической системы.

действие кибернетической системы

- С *действие*
 - := [целенаправленное действие, выполняемое кибернетической системой, а точнее, её решателем задач]
 - ⇒ разбиение*:
 - {• *внешнее действие кибернетической системы*
 - := [действие, выполняемое кибернетической системой в её внешней среде]
 - := [поведенческое действие]
 - *действие кибернетической системы, выполняемое в собственной физической оболочке*
 - *действие кибернетической системы, выполняемое в собственной абстрактной памяти*
- }

Говоря о действиях кибернетической системы, выполняемых в собственной абстрактной памяти, подразумеваются действия, направленные на преобразование информации, хранимой в памяти, но никак не на преобразование физической памяти (физической оболочки абстрактной памяти).

Каждое сложное действие, выполняемое кибернетической системой вне собственного абстрактной памяти, включает в себя поддействия, выполняемые в указанной абстрактной памяти. Это означает, что все внешние действия кибернетической системы управляются внутренними её действиями (действиями в абстрактной памяти).

Интерфейс кибернетической системы — условно выделяемый компонент *решателя задач кибернетической системы*, обеспечивающий решение *интерфейсных задач*, направленных на непосредственную реализацию взаимодействия *кибернетической системы* с её внешней средой. Решатель интерфейсных задач кибернетической системы. Стоит отличать понятие интерфейса кибернетической системы и понятие физического обеспечения интерфейса кибернетической системы.

Физическая оболочка кибернетической системы — часть кибернетической системы, являющаяся "посредником" между её внутренней средой (памятью, в которой хранится и обрабатывается информация кибернетической системы) и её внешней средой.

кибернетическая система

- ⇒ обобщенная декомпозиция*:
 - {• память кибернетической системы
 - процессор кибернетической системы
 - физическое обеспечение интерфейса кибернетической системы
 - корпус кибернетической системы
- }

физическое обеспечение интерфейса кибернетической системы

- ⇒ обобщенная декомпозиция*:
 - {• сенсорная подсистема физической оболочки кибернетической системы
 - эффекторная подсистема физической оболочки кибернетической системы
- }

Память кибернетической системы — физическая оболочка (реализация) абстрактной памяти *кибернетической системы*, внутренней среды *кибернетической системы*, в рамках которой *кибернетическая система* формирует и использует (обрабатывает) информационную модель своей внешней среды.

Не каждая *кибернетическая система* имеет *память*. В *кибернетических системах*, которые не имеют *памяти*, обработка информации сводится к обмену сигналами между компонентами этих систем. Появление в *кибернетических системах* памяти как среды для "централизованного" хранения и обработки *информации* является важнейшим этапом их эволюции. Дальнейшая эволюция *кибернетических систем* во многом определяется:

- качеством памяти* как среды для хранения и обработки информации;
- качеством информации (информационной модели), хранимой в памяти кибернетической системы;

Компонент кибернетической системы, в рамках которого *кибернетическая система* осуществляет отображение (формирование информационной модели) среды своего существования, а также использование этой информационной модели для управления собственным поведением в указанной среде

Сам факт появления в кибернетической системе памяти, которая (1) обеспечивает представление различного вида информации о среде, в рамках которой кибернетическая система решает различные задачи (выполняет различные действия), (2) обеспечивает хранение достаточно полной информационной модели указанной среды (достаточно полной для реализации своей деятельности), (3) обеспечивает высокую степень гибкости указанной хранимой в памяти информационной модели среды жизнедеятельности (т.е. лёгкость внесения изменений в эту информационную модель), существенно повышает уровень адаптивности кибернетической системы к различным изменениям своей среды. Появление *памяти* в кибернетических системах является основным признаком перехода от "простых" автоматов к компьютерным системам, от роботов 1-го поколения к роботам следующих поколений.

Принципы организации памяти кибернетической системы могут быть разными (ассоциативная, адресная, структурно фиксированная/структурно перестраиваемая, нелинейная/линейная). От организации памяти во многом зависит её качество.

Процессором кибернетической системы называют физически реализованный интерпретатор хранимых в памяти кибернетической системы программ, соответствующих базовой модели решения задач для данной кибернетической системы. Такая модель решения задач для данной кибернетической системы является моделью решения задач самого нижнего уровня и не может быть интерпретирована с помощью другой модели решения задач, используемой этой же кибернетической системой. Она может быть проинтерпретирована либо путем аппаратной реализации такого интерпретатора, либо путём его программной реализации, например, на современных компьютерах. В последнем случае, кроме собственного интерпретатора, необходимо также построить модель памяти реализуемой кибернетической системы.

Развитие рынка интеллектуальных компьютерных систем существенно сдерживается неприспособленностью современного поколения компьютеров к реализации на их основе интеллектуальных компьютерных систем. Попытки создания компьютеров, приспособленных к реализации интеллектуальных компьютерных систем, не привели к успеху, т.к. эти проекты были направлены на выполнение частных требований, предъявляемых к аппаратному уровню интеллектуальных систем, что неминуемо приводило к приспособленности создаваемых компьютеров к реализации не всего многообразия интеллектуальных компьютерных систем, а только некоторых подмножеств таких систем. Указанные подмножества интеллектуальных компьютерных систем в основном определялись ориентацией на конкретные используемые модели решения интеллектуальных задач, тогда как важнейшим фактором, определяющим уровень интеллекта кибернетических систем, является их универсальность в плане многообразия используемых моделей решения задач. Следовательно, компьютер для интеллектуальных компьютерных систем должен быть эффективным аппаратным интерпретатором любых моделей решения задач, как интеллектуальных, так и достаточно простых.

Таким образом, выделено следующее семейство отношений, заданных на множестве кибернетических систем.

отношение, заданное на множестве кибернетических систем

- Э *память кибернетической системы**
- Э *процессор кибернетической системы**
- Э *член коллектива**
- Э *внешняя среда кибернетической системы**
- Э *сенсор кибернетической системы**
- Э *эффектор кибернетической системы**
- Э *физическая оболочка кибернетической системы**
- Э *информация, хранимая в памяти кибернетической системы**
- Э *абстрактная память кибернетической системы**
- Э *часть**
- С *встроенная кибернетическая система**

Понятие *внешней среды кибернетической системы** является понятием относительным, т.к. (1) разные кибернетические системы могут иметь разную внешнюю среду и (2) одна кибернетическая система может входить в состав внешней среды другой кибернетической системы. В общем случае среда жизнедеятельности *кибернетиче-*

ской системы включает в себя (1) *внешнюю среду** этой системы, (2) *физическую оболочку** этой системы и (3) её *абстрактную память*, т.е. *внутреннюю среду**, которая является хранилищем информационной модели всей среды

Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы

Качество кибернетической системы есть интегральная, комплексная оценка уровня развития кибернетической системы. Это свойство, характеристика кибернетических систем, признак их классификации, который позволяет разместить эти системы по "ступенькам" некоторой условной "эволюционной лестницы". На каждую такую "ступеньку" попадают кибернетические системы, имеющие одинаковый уровень развития, каждому из которых соответствует свой набор значений дополнительных свойств кибернетических систем, которые уточняют соответствующий уровень развития кибернетических систем. Такой эволюционный подход к рассмотрению кибернетических систем даёт возможность, во-первых, детализировать направления эволюции кибернетических систем и, во-вторых, уточнить то место этой эволюции, где и благодаря чему осуществляется переход от неинтеллектуальных кибернетических систем к интеллектуальным.

В основе эволюционного подхода к рассмотрению многообразия кибернетических систем лежит положение о том, что идеальных кибернетических систем не существует, но существует постоянное стремление к идеалу, к большему совершенству. Важно уточнить, что конкретно в каждой кибернетической системе следует изменить, чтобы привести эту систему к более совершенному виду. Так, например, развитие технологий разработки компьютерных систем должно быть направлено на переход к таким новым архитектурным и функциональным принципам, лежащим в основе компьютерных систем, которые обеспечивают существенное снижение трудоемкости их разработки и сокращение сроков разработки, а также обеспечивают существенное повышение уровня интеллекта и, в частности, уровня обучаемости разрабатываемых компьютерных систем, например, путём перехода от поддержки обучения с учителем к реализации эффективного самообучения (к автоматизации организации самостоятельного обучения).

Проблема выделения критериев интеллектуальности систем рассмотрена в ряде работ *Finn2021*, *Nilsson N.J.HumanLAIBS-2005art*, *Kerr C..aConceMfTI-2006art*, *Анциферов С.С.ОценкКИС-2013art*. Тем не менее, для практической реализации интеллектуальных компьютерных систем необходимо детализировать и уточнить эти свойства, пытаясь свести их к более конструктивным, прозрачным и понятным для реализации свойствам.

Для детализации понятия качества кибернетической системы необходимо задать метрику качества кибернетических систем и построить иерархическую систему свойств, параметров, признаков, определяющих качество кибернетических систем.

качество кибернетической системы

⇒ *свойство-предпосылка**:

- *качество физической оболочки кибернетической системы*
- *качество информации, хранимой в памяти кибернетической системы*
- *качество решателя задач кибернетической системы*
- *гибридность кибернетической системы*

⇒ *частное свойство**:

- { • *многообразие видов знаний, хранимых в памяти кибернетической системы*
- *многообразие моделей решения задач*
- *многообразие видов сенсоров и эффекторов*

}

- *приспособленность кибернетической системы к её совершенствованию*
- *производительность кибернетической системы*
- *надежность кибернетической системы*
- *интероперабельность кибернетической системы*

Пункт 1.1.1.4. Комплекс свойств, определяющих качество физической оболочки кибернетической системы

Качество физической оболочки кибернетической системы есть интегральное качество физической, аппаратной основы кибернетической системы. Выделенное множество свойств, определяющих качество физической оболочки кибернетической системы, приведена ниже:

качество физической оболочки кибернетической системы

⇒ *свойство-предпосылка*:*

- *качество памяти кибернетической системы*
- *качество процессора кибернетической системы*
- *качество сенсоров кибернетической системы*
- *качество эффекторов кибернетической системы*
- *приспособленность физической оболочки кибернетической системы к ее совершенствованию*
- *удобство транспортировки кибернетической системы*
- *надежность физической оболочки кибернетической системы*

Важно, чтобы память обеспечивала высокий уровень гибкости указанной информационной модели. Важно также, чтобы эта информационная модель была моделью не только внешней среды кибернетической системы, но также и моделью самой этой информационной модели – описанием её текущей ситуации, предыстории, закономерностей.

качество памяти кибернетической системы

⇒ *свойство-предпосылка*:*

- *способность памяти кибернетической системы обеспечить хранение высококачественной информации*
- *способность памяти кибернетической системы обеспечить функционирование высококачественного решателя задач*
- *объём памяти*

Факт возникновения памяти в кибернетической системе является важнейшим этапом её эволюции. Дальнейшее развитие памяти кибернетической системы, обеспечивающее хранение все более качественной информации, хранимой в памяти и все более качественную организацию обработки этой информации, т.е. переход на поддержку все более качественных моделей обработки информации, является важнейшим фактором эволюции кибернетических систем.

Способность памяти кибернетической системы обеспечить функционирование высококачественного решателя задач основывается на качестве доступа к информации, хранимой в памяти кибернетической системы, логико-семантической гибкости памяти кибернетической системы, способности памяти кибернетической системы обеспечить интерпретацию широкого многообразия моделей решения задач.

Качество процессора кибернетической системы определяется его способностью обеспечить функционирования высококачественного решателя задач.

способность процессора кибернетической системы обеспечить функционирования высококачественного решателя задач

⇒ *свойство-предпосылка*:*

- *многообразие моделей решения задач, интерпретируемых процессором кибернетической системы*
- *простота и качество интерпретации процессором системы широкого многообразия моделей решения задач*
- *обеспечение процессором кибернетической системы качественного управления информационными процессами в памяти*
- *быстродействие процессора кибернетической системы*

Максимальным уровнем качества процессора кибернетической системы по параметру многообразия моделей решения задач, интерпретируемых процессором кибернетической системы, является его универсальность, т.е. его принципиальная возможность интерпретировать любую модель решения как интеллектуальных, так и неинтеллектуальных задач. Простота определяется степенью близости интерпретируемых моделей решений задач к “физическому” уровню организации процессора кибернетической системы. Качественное управление информационными процессами в памяти подразумевает грамотное сочетание таких аспектов управления процессами, как централизация и децентрализация *Melekhova O..aDecenSfCD-2018art*, синхронность и асинхронность, последовательность и параллельность.

Качество сенсоров и эффекторов кибернетической системы сводится к многообразию видов сенсоров и эффекторов кибернетической системы, т.е. к многообразию средств восприятия и воздействия на информацию о текущем состоянии внешней среды и собственной физической оболочки. Приспособленность физической оболочки кибернетической системы к её совершенствованию определяется гибкостью и стратифицированностью физической оболочки кибернетической системы.

Пункт 1.1.1.5. Комплекс свойств, определяющих качество информации, хранимой в памяти кибернетической системы

Качество информационной модели среды "обитания" кибернетической системы, в частности, определяется:

- корректностью этой модели, отсутствием в ней ошибок;
- адекватностью этой модели;
- полнотой, достаточностью находящейся в ней информации для эффективного функционирования кибернетической системы;
- структурированностью, систематизированностью.

Важнейшим этапом эволюции информационной модели среды кибернетической системы является переход от недостаточно полной и несистематизированной информационные модели среды к базе знаний.

качество информации, хранимой в памяти кибернетической системы

⇒ *свойство-предпосылка**:

- семантическая мощность языка представления информации в памяти кибернетической системы
- объём информации, загруженной в память кибернетической системы
- степень конвергенции и интеграции различного вида знаний, хранимых в памяти кибернетической системы
- стратифицированность информации, хранимой в памяти кибернетической системы
- простота и локальность выполнения семантически целостных операций над информацией, хранимой в памяти кибернетической системы
- корректность/некорректность информации, хранимой в памяти кибернетической системы
- однозначность/неоднозначность информации, хранимой в памяти кибернетической системы
- целостность/нецелостность информации, хранимой в памяти кибернетической системы
- чистота/загрязненность информации, хранимой в памяти кибернетической системы
- достоверность/недостоверность информации, хранимой в памяти кибернетической системы
- точность/неточность информации, хранимой в памяти кибернетической системы
- четкость/нечеткость информации, хранимой в памяти кибернетической системы
- определенность/недоопределенность информации, хранимой в памяти кибернетической системы

Корректность/некорректность информации есть уровень адекватности хранимой информации той среды, в которой существует кибернетическая система и информационной моделью которой эта хранимая информация является. Непротиворечивость/противоречивость информации означает уровень присутствия в хранимой информации различного вида противоречий и, в частности, ошибок. Ошибки в хранимой информации могут быть синтаксическими и семантическими, противоречащими некоторым правилам, которые явно в памяти могут быть не представлены и считаются априори истинными.

Полнота/неполнота информации — уровень того, насколько информация, хранимая в памяти кибернетической системы, описывает среду существования этой системы и используемые ею методы решения задач достаточно полно для того, чтобы кибернетическая система могла действительно решать все множество соответствующих ей задач. Чем полнее информация, хранимая в памяти кибернетической системы, чем полнее информационное обеспечение деятельности этой системы это системы, тем эффективнее (качественнее) сама эта деятельность. Полнота определяется структурированностью информации и многообразием видов знаний, хранимых в памяти кибернетической системы.

Однозначность/неоднозначность информации определяется многообразием форм дублирования информации и частотой дублирования информации.

Целостность/нецелостность информации есть уровень содержательной информативности информации, уровень того, насколько семантически связной является информация, насколько полно специфицированы все описываемые в памяти сущности (путём описания необходимого набора связей этих сущностей с другими описываемыми сущностями), насколько редко или часто в рамках хранимой информации встречаются информационные дыры, соответствующие явной недостаточности некоторых спецификаций. Примерами информационных дыр являются:

- отсутствующий метод решения часто встречающихся задач;
- отсутствующее определение используемого определяемого понятия;
- недостаточно подробная спецификация часто рассматриваемых сущностей.

Чистота/загрязненность информации означает многообразие форм и общее количество информационного мусора, входящего в состав информации, хранимой в памяти кибернетической системы. Под информационным мусором понимается информационный фрагмент, входящий в состав информации, удаление которого существенно не усложнит деятельность кибернетической системы. Примерами информационного мусора являются:

- информация, которая нечасто востребована, но при необходимости может быть легко логически выведена;
- информация, актуальность которой истекла.

Семантическая мощность языка представления информации в памяти кибернетической системы определяется гибридностью информации, хранимой в памяти кибернетической системы. Язык, информационные конструкции которого могут представить любую конфигурацию любых связей между любыми сущностями, является универсальным языком. Универсальность внутреннего языка кибернетической системы является важнейшим фактором её интеллектуальности.

Гибридность информации, хранимой в памяти кибернетической системы определяется многообразием видов знаний и степенью конвергенции и интеграции различного вида знаний.

Стратифицированность информации есть способность кибернетической системы выделять такие разделы информации, хранимой в памяти этой системы, которые бы ограничивали области действия агентов решателя задач кибернетической системы, являющиеся достаточными для решения заданных задач. Стратифицированность определяется структурированностью и рефлексивностью информации, хранимой в памяти кибернетической системы. Рефлексивность информации, хранимой в памяти кибернетической системы, т.е. наличие метаязыковых средств, является фактором, обеспечивающим не только структуризацию хранимой информации, но возможность описания синтаксиса и семантики самых различных языков, используемых кибернетической системой.

База знаний является примером информации, хранимой в памяти кибернетической системы и имеющей высокий уровень качества по всем показателям и, в частности, высокий уровень:

- семантической мощности языка представления информации хранимой в памяти кибернетической системы;
- гибридности информации, хранимой в памяти кибернетической системы;
- многообразия видов знаний, хранимых в памяти кибернетической системы;
- формализованности информации, хранимой в памяти кибернетической системы;
- структурированности информации, хранимой в памяти кибернетической системы;

Переход информации, хранимой в памяти кибернетической системы на уровень качества, соответствующий базам знаний, является важнейшим этапом эволюции кибернетических систем.

Пункт 1.1.1.6. Комплекс свойств, определяющих качество решателя задач кибернетической системы

Качество решателя задач кибернетической системы — интегральная качественная оценка множества задач, которые кибернетическая система способна выполнять в заданный момент. Основным свойством и назначением решателя задач кибернетической системы является способность решать задачи на основе накапливаемых, приобретаемых кибернетической системой различного вида навыков с использованием процессора кибернетической системы, являющегося универсальным интерпретатором всевозможных накопленных навыков. При этом качество указанной способности определяется целым рядом дополнительных факторов.

общая характеристика решателя задач кибернетической системы

⇒ *свойство-предпосылка*:*

- {• общий объем задач, решаемых кибернетической системой
- многообразие видов задач, решаемых кибернетической системой
- способность кибернетической системы к анализу решаемых задач
- способность кибернетической системы к решению задач, методы решения которых в текущий момент известны
- способность кибернетической системы к решению задач, методы решения которых ей в текущий момент не известны
- множество навыков, используемых кибернетической системой
- степень конвергенции и интеграции различного вида моделей решения задач, используемых кибернетической системой
- качество организации взаимодействия процессов решения задач в кибернетической системе
- быстродействие решателя задач кибернетической системы
- способность кибернетической системы решать задачи, предполагающие использование информации, обладающей различного рода не-факторами
- многообразие и качество решения задач информационного поиска
- способность кибернетической системы генерировать ответы на вопросы различного вида в случае, если они целиком или частично отсутствуют в текущем состоянии информации, хранимой в памяти
- способность кибернетической системы к рассуждениям различного вида
- качество целеполагания
- качество реализации планов собственных действий
- способность кибернетической системы к локализации такой области информации, хранимой в ее памяти, которой достаточно для обеспечения решения заданной задачи

- способность кибернетической системы к выявлению существенного в информации, хранимой в ее памяти
 - активность кибернетической системы
- }

Общий объем задач, решаемых кибернетической системой, определяется мощностью языка представления задач, решаемых кибернетической системой. Мощность языка представления задач прежде всего определяется многообразием видов представляемых задач (многообразием видов описываемых действий). Каждая задача есть спецификация соответствующего (описываемого) действия. Поэтому рассмотрение многообразия видов задач, решаемых кибернетической системой, полностью соответствует многообразию видов деятельности, осуществляющейся этой системой. Важно заметить, что есть виды деятельности кибернетической системы, которые определяют качество и, в частности, уровень интеллекта кибернетической системы.

Способность кибернетической системы к анализу решаемых задач предполагает оценку задачи на предмет:

- сложности достижения;
- целесообразности достижения (нужности, важности, приоритетности);
- соответствия цели существующим нормам (правилам) соответствующей деятельности.

Метод решения задач – это вид знаний, хранимых в памяти кибернетической системы и содержащих информацию, которой достаточно либо для сведения каждой задачи из соответствующего класса задач к полной системе подзадач, решение которых гарантирует решение исходной задачи, либо для окончательного решения этой задачи из указанного класса задач. Методами для решения задач могут быть не только алгоритмы, но также и функциональные программы, продукционные системы, логические исчисления, генетические алгоритмы, искусственные нейронные сети различного вида. Задачи, для которых не находятся соответствующие им методы, решаются с помощью метаметодов (стратегий) решения задач, направленных:

- на генерацию нужных исходных данных (нужного контекста), необходимых для решения каждой задачи;
- на генерацию плана решения задачи, описывающего сведение исходной задачи к подзадачам (до тех подзадач, методы решения которых системы известны);
- на сужение области решения задачи (на сужение контекста задачи, достаточного для ее решения).

Качество решения каждой задачи определяется:

- временем её решения (чем быстрее задача решается, тем выше качество её решения);
- полнотой и корректностью результата решения задачи;
- затраченными для решения задачи ресурсами памяти (объемом фрагмента хранимой информации, используемой для решения задачи);
- затраченным для решения задачи ресурсами решателя задач (количеством используемых внутренних агентов).

Таким образом, повышение качества процесса решения каждой конкретной задачи, а также каждого класса задач (путем совершенствования соответствующего метода, в частности, алгоритма) является важным фактором повышения качества решателя задач в целом.

Перспективным вариантом построения решателя задач кибернетической системы является реализация агентно-ориентированной модели обработки информации, т.е. построение решателя задач в виде многоагентной системы, агенты которой осуществляют обработку информации, хранимой в памяти кибернетической системы, и управляются этой информацией (точнее, её текущим состоянием). Особое место среди этих агентов занимают сенсорные (рецепторные) и эффекторные агенты, которые, соответственно, воспринимают информацию о текущем состоянии внешней среды и воздействуют на внешнюю среду, в частности, путем изменения состояния физической оболочки кибернетической системы.

Указанная агентно-ориентированная модель организации взаимодействия процессов решения задач в кибернетической системе по сути есть не что иное, как модель ситуационного управления процессами решения задач, решаемых кибернетической системой как в своей внешней среде, так и в своей памяти.

Быстродействие решателя задач кибернетической системы сводится ко скорости решения задач, быстродействию решателя задач, скорости реакции кибернетической системы на различные задачные ситуации. Во многом свойство определяется быстродействием процессора кибернетической системы.

Примерами задач, предполагающих использование информации, обладающей различного рода не-факторами, являются задачи проектирования, распознавания, целеполагания, прогнозирования, и т.д. Зачастую это:

- нечетко сформулированные задачи ("делай то, не знаю что");
- задачи, которые решаются в условиях неполноты, неточности, противоречивости исходных данных;
- задачи, принадлежащие классам задач, для которых практически невозможно построить соответствующие алгоритмы.

Для таких задач характерны:

- неточность и недостоверность исходных данных;
- отсутствие критерия качества результата;

- невозможность или высокая трудоемкость разработки алгоритма;
- необходимость учета контекста задачи.

Способность кибернетической системы генерировать (порождать, строить, синтезировать, выводить) ответы на самые различные вопросы и, в частности, на вопросы типа “что это такое”, на почему-вопросы, означает способность кибернетической системы объяснять (обосновывать корректность) своих действий.

Самостоятельность целеполагания есть способность кибернетической системы генерировать, инициировать и решать задачи, которые не являются подзадачами, инициированными внешними (другими) субъектами, а также способность на основе анализа своих возможностей отказаться от выполнения задачи, инициированной извне, переадресовав её другой кибернетической системе, либо на основе анализа самой этой задачи обосновать её нецелесообразность или некорректность. Повышение уровня самостоятельности существенно расширяет возможности кибернетической системы, т.е. объем тех задач, которые она может решать не только в "идеальных" условиях, но и в реальных, осложненных обстоятельствах. Способность системы адекватно расставлять приоритеты своим целям и не "распыляться" на достижение неприоритетных (несущественных) целей есть способность анализа целесообразности деятельности.

Способность кибернетической системы к выявлению существенного в информации, хранимой в ее памяти есть способность к выявлению (обнаружению, выделению) таких фрагментов информации, хранимой в памяти кибернетической системы, которые существенны (важны) для достижения соответствующих целей. Понятие существенного (важного) фрагмента информации, хранимой в памяти кибернетической системы, относительно и определяется соответствующей задачей. Тем не менее, есть важные перманентно решаемые задачи, в частности задачи анализа качества информации, хранимой в памяти кибернетической системы. Существенные фрагменты хранимой информации, выделяемые в процессе решения этих задач, являются относительными не столько по отношению к решаемой задаче, сколько по отношению к текущему состоянию хранимой информации.

Уровень активности кибернетической системы может быть разным для разных решаемых задач, для разных классов выполняемых действий, для разных видов деятельности. Чем выше активность кибернетической системы, тем (при прочих равных условиях) она больше успевает сделать, следовательно, тем выше ее качество (эффективность). Обратным свойством является понятие пассивности кибернетической системы.

Пункт 1.1.1.7. Комплекс свойств, определяющих уровень обучаемости кибернетической системы

Обучаемость кибернетической системы есть способность кибернетической системы повышать своё качество, адаптируясь к решению новых задач, качество внутренней информации модели своей среды, качество своего решателя задач и даже качество своей физической оболочки. Способность кибернетической системы к совершенствованию (к эволюции, к повышению уровня своего качества), к самосовершенствованию с различной степенью самостоятельности.

Максимальный уровень обучаемости кибернетической системы – это её способность эволюционировать (повышать уровень своего качества) максимально быстро и в любом направлении, т.е. способность быстро и без каких-либо ограничений приобретать любые новые знания и навыки.

Реализация способности кибернетической системы обучаться, т.е. решать перманентно инициированную сверхзадачу самообучения, накладывает дополнительные требования, предъявляемые к информации, хранимой в памяти кибернетической системы, к решателю задач кибернетической системы, а в перспективе также и к физической оболочке кибернетической системы.

Важнейшей характеристикой кибернетической системы является не только то, какой уровень интеллекта кибернетическая система имеет в текущий момент, какое множество действий (задач) она способна выполнять, но и то, насколько быстро этот уровень может повышаться.

обучаемость кибернетической системы

⇒ *свойство-предпосылка**:

- гибкость кибернетической системы
- стабилизированность кибернетической системы
- рефлексивность кибернетической системы
- ограниченность обучения кибернетической системы
- познавательная активность кибернетической системы
- способность кибернетической системы к самосохранению

Поскольку обучение всегда сводится к внесению тех или иных изменений в обучаемую кибернетическую систему, без высокого уровня гибкости этой системы не может быть высокого уровня её обучаемости. Гибкость возможных

самоизменений кибернетической системы определяется простотой и многообразием возможных самоизменений кибернетической системы.

При наличии стратифицированности кибернетической системы появляется возможность четкого определения области действия различных изменений, вносимых в кибернетическую систему, т.е. возможность четкого ограничения тех частей кибернетической системы, за пределы которых нет необходимости выходить для учета последствий внесенных в систему первичных изменений (осуществлять дополнительные изменения, являющиеся последствиями первичных изменений).

Рефлексивность кибернетической системы есть способность кибернетической системы к самоанализу. Конструктивным результатом рефлексии кибернетической системы является генерация в её памяти спецификации различных негативных или подозрительных особенностей, которые следует учитывать для повышения качества кибернетической системы. Такими особенностями (недостатками) могут быть выявленные противоречия (ошибки), выявленные пары синонимичных знаков, омонимичные знаки, информационные дыры.

Ограниченностю обучения кибернетической системы определяет границу между теми знаниями и навыками, которые соответствующая кибернетическая система принципиально может приобрести, и теми знаниями и навыками, которые указанная кибернетическая система не сможет приобрести никогда. Данное свойство определяет максимальный уровень потенциальных возможностей соответствующей кибернетической системы. Максимальная степень отсутствия ограничений в приобретении новых знаний и навыков – это полное отсутствие ограничений, т.е. полная универсальность возможностей соответствующих кибернетических систем.

Познавательная активность кибернетической системы — любознательность, активность и самостоятельность в приобретении новых знаний и навыков. Следует отличать способность приобретать новые знания и навыки, а также их совершенствовать, от желания это делать. Желание (целевая установка) научиться решать те или иные задачи может быть сформулировано кибернетической системой либо самостоятельно, либо извне (некоторым учителем).

познавательная активность кибернетической системы

⇒ *свойство-предпосылка**:

- способность кибернетической системы к синтезу познавательных целей и процедур
- способность кибернетической системы к самоорганизации собственного обучения
- способность кибернетической системы к экспериментальным действиям

Способность кибернетической системы к синтезу познавательных целей и процедур является способностью планировать своё обучение и управлять процессом обучения, умение задавать вопросы или целенаправленные последовательности вопросов, способность генерировать четкую спецификацию своей информационной потребности. Способность кибернетической системы к самоорганизации собственного обучения есть способность осуществлять управление своим обучением, способность кибернетической системы самой выполнять роль своего учителя. Способность кибернетической системы к экспериментальным действиям — способность к отклонениям от составленных планов своих действий для повышения качества результата или сохранении целенаправленности этих действий, способность к импровизации.

Чем выше уровень безопасности кибернетической системы, тем выше её уровень обучаемости. Способность кибернетической системы к самосохранению означает способность кибернетической системы к выявлению и устраниению угроз, направленных на снижение её качества и даже на её уничтожение, что означает полную потерю необходимого качества.

Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы

Основным свойством, характеристикой кибернетической системы является уровень ее интеллекта, который является интегральной характеристикой, определяющей уровень эффективности взаимодействия кибернетической системы со средой своего существования. Процесс эволюции кибернетических систем следует рассматривать как процесс повышения уровня их качества по целому ряду свойств и как процесс повышения уровня их интеллекта.

Кибернетическая система может быть как интеллектуальной, так и неинтеллектуальной. В свою очередь, интеллектуальные системы могут быть как слабоинтеллектуальной, так и высокоинтеллектуальной.

уровень интеллекта кибернетической системы

⇒ *свойство-предпосылка**:

- образованность кибернетической системы
- обучаемость кибернетической системы
- интероперабельность кибернетической системы

Образованность кибернетической системы есть уровень навыков, а также иных знаний, приобретенных кибернетической системой к заданному моменту.

образованность кибернетической системы

⇒ *свойство-предпосылка**:

- *качество навыков, приобретенных кибернетической системой*
- *качество информации, хранимой в памяти кибернетической системы*

Примерами образованной кибернетической системы являются:

- кибернетическая система, основанная на знаниях;
- кибернетическая система, управляемая знаниями;
- целенаправленная кибернетическая система;
- гибридная кибернетическая система;
- потенциально универсальная кибернетическая система.

Обучаемая кибернетическая система есть кибернетическая система, способная познавать среду своего обитания, то есть строить и постоянно уточнять в своей памяти информационную модель этой среды, а также использовать эту модель для решения различных задач (для организации своей деятельности) в указанной среде.

Примерами обучаемой кибернетической системы являются:

- кибернетическая система с высоким уровнем стратифицированности своих знаний и навыков;
- рефлексивная кибернетическая система;
- самообучаемая кибернетическая система;
- кибернетическая система с высоким уровнем познавательной активности.

Интеллект кибернетической системы, как и лежащий в его основе познавательный процесс, выполняемый кибернетической системой, имеет социальный характер, поскольку наиболее эффективно формируется и развивается в форме взаимодействия кибернетической системы с другими кибернетическими системами. Социально ориентированная кибернетическая система имеет достаточно высокий уровень интеллекта, чтобы быть полезным членом различных, в том числе, и человеко-машинных сообществ. Определенный уровень социально значимых качеств является необходимым условием интеллектуальности кибернетической системы. Примерами социально ориентированной кибернетической системы являются:

- кибернетическая система, способная устанавливать и поддерживать высокий уровень семантической совместности и взаимопонимания с другими системами;
- договороспособная кибернетическая система.

Все свойства, присущие кибернетическим системам, в различных кибернетических системах могут иметь самый различный уровень. Более того, в некоторых кибернетических системах некоторые из этих свойств могут вообще отсутствовать. При этом в кибернетических системах, которые условно будем называть интеллектуальными системами, все указанные выше свойства должны быть представлены в достаточно развитом виде.

§ 1.1.2. Понятие интеллектуальной многоагентной системы

⇒ *ключевое понятие**:

- *индивидуальная кибернетическая система*
- *многоагентная система*
 - ▷ *кибернетическая система*
- *иерархическая кибернетическая система*
- *агент*
- *многоагентная система*
 - := [самостоятельный компонент многоагентной системы]
 - := [кибернетическая система, являющаяся агентом по крайней мере одной многоагентной системы]
- *агент**
 - := [быть агентом заданной многоагентной системы*]
- ⇒ *второй домен**:
 - агент*
- *индивидуальный агент*
- *коллективный агент*
- *встроенный агент*
 - := [внутренний агент индивидуальной кибернетической системы]
- *специализированный агент*
 - := [интеллектуальный агент]

- **мультиагент**
:= [кибернетическая система, являющаяся агентом более чем одной многоагентной системы]

⇒ *ключевое знание**:

- Классификация многоагентных систем
- Обобщенная архитектура кибернетических систем
- Факторы, определяющие качество многоагентных систем
- Факторы, определяющие уровень интеллекта многоагентных систем

⇒ *рассматриваемый вопрос**:

- Почему переход от отдельных кибернетических систем к их коллективным, многоагентным системам, агентами которых являются заданные кибернетические системы, является одним из направлений повышения уровня интеллекта исходных кибернетических систем.
- Всегда ли объединение кибернетических систем в коллектив этих систем приводит к повышению уровня интеллекта.
- Почему не каждое соединение достаточно качественных кибернетических систем порождает качественную многоагентную систему.

⇒ *библиографическая ссылка**:

- Тарасов В.Б. о Много СкИО-2002kn
- Варшавский В.А.. Оркес ИВД-1984kn
- Hadzic M.. OntolBMAS-2009bk
- Dorri A.. MultiASaS-2018art
- Ferber J.. Agent tOaOVoMAS-2003art
- Balaji P.G.. aIntro tMAS-2010art

⇒ *подраздел**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

Переход от кибернетических систем к коллективам взаимодействующих между собой кибернетических систем, т.е. к социальной организации кибернетических систем, является важнейшим фактором эволюции кибернетических систем. Кибернетическую систему, представляющую собой коллектив взаимодействующих кибернетических систем, обладающих определенной степенью самостоятельности (самодостаточности, свободы выбора), будем называть многоагентной системой *Dorri A.. MultiASaS-2018art*.

кибернетическая система

⇒ *разбиение**:

- {• индивидуальная кибернетическая система
 - кибернетическая система, являющаяся минимальным компонентом индивидуальной кибернетической системы
 - кибернетическая система, являющаяся комплексом компонентов соответствующей индивидуальной кибернетической системы
 - сообщество индивидуальных кибернетических систем
- ⇒ *разбиение**:
- {• простое сообщество индивидуальных кибернетических систем
 - иерархическое сообщество индивидуальных кибернетических систем
- }

Агенты многоагентной системы могут (но вовсе не обязательно должны) быть интеллектуальными системами. Так, например, агенты интеллектуального решателя задач, имеющего агентно-ориентированную архитектуру, не являются интеллектуальными системами. Агентом иерархической многоагентной системы может быть другая многоагентная система *Ferber J.. Agent tOaOVoMAS-2003art*.

Многоагентная система есть коллектив взаимодействующих автономных кибернетических систем, имеющих общую среду обитания, жизнедеятельности *Hadzic M.. OntolBMAS-2009bk*.

Классификация многоагентных систем приведена ниже:

многоагентная система

⇒ *разбиение**:

- {• одноуровневая многоагентная система

- иерархическая многоагентная система
- }

Одноуровневая многоагентная система реализует либо одну модель параллельного (распределенного) решения задач соответствующего класса, либо комбинацию фиксированного числа разных и параллельно реализованных моделей решения задач. Иерархическая многоагентная система состоит из агентов, которые могут быть индивидуальные кибернетические системы, коллективы индивидуальных кибернетических систем, а также коллективы, состоящие из индивидуальных кибернетических систем и коллективов индивидуальных кибернетических систем.

В многоагентной системе с централизованным управлением специально выделяются агенты, которые принимают решения в определенной области деятельности многоагентной системы и обеспечивают выполнение этих решений путем управления деятельностью остальных агентов, входящих в состав этой системы.

В многоагентной системе с децентрализованным управлением решения принимаются коллегиально и "автоматически" (решения о признании новой кем-то предложенной информации – в том числе, об инициировании некоторой задачи, решения о коррекции (уточнении) уже ранее признанной, согласованной информации) на основе четко продуманной и постоянно совершенствуемой методики, а также на основе активного участия всех агентов в формировании новых предложений, подлежащих признанию или согласованию *Balaji P.G..alntro tMAS-2010art*. В такой многоагентной системе все агенты участвуют в управлении этой системы. Примером такой системы является оркестр, способный играть без дирижера.

Переход к многоагентным системам является важнейшим фактором повышения качества (и, в частности, уровня интеллекта) кибернетических систем, т.к. уровень интеллекта многоагентной системы может быть значительно выше уровня интеллекта каждого входящего в неё агента. Это происходит далеко не всегда, поскольку важнейшим фактором качества многоагентных систем является не только качество входящих в неё агентов, но и организация взаимодействия агентов и, в частности, переход от централизованного к децентрализованному управлению. Количество не всегда переходит в новое качество.

Качество индивидуальных кибернетических систем определяется, кроме всего прочего тем, насколько большой вклад индивидуальная кибернетическая система вносит в повышение качества тех коллективов, в состав которых она входит. Указанное свойство индивидуальных кибернетических систем будем называть уровнем их интероперабельности *Ouksel A.M..SemantGIS-1999art*.

Синергетическая кибернетическая система есть многоагентная система, обладающая высоким уровнем коллективного интеллекта, атомарными агентами которой являются индивидуальные интеллектуальные системы, имеющие высокий уровень интероперабельности *Lopes L.S..SemantIoT-2022art Hamilton Inter aKEfG-2006art*. Примером синергетической кибернетической системы является творческий коллектив, реализующий сложный научноемкий проект.

Эффективность творческого коллектива (например в области научно-технической деятельности) определяется:

- согласованностью мотивации, целевой установки всего коллектива и каждого его члена (не должно быть противоречий между целью коллектива и творческой самореализацией каждого его члена);
- эффективной организацией децентрализованного управления деятельностью членов сообщества;
- четкой, оперативной и доступной всем фиксацией документации текущего состояния содеянного и направлений его дальнейшего развития;
- уровнем трудоемкости оперативности фиксации индивидуальных результатов в рамках коллективно создаваемого общего результата;
- уровнем структурированности и, прежде всего, стратифицированности обобщенной документации (базы знаний);
- эффективностью ассоциативного доступа к фрагментам документации;
- гибкостью коллективно создаваемой базы;
- автоматизацией анализа содеянного и управления проектом.

Уровень интеллекта многоагентной системы может быть значительно ниже уровня интеллекта самого "глупого" члена этого коллектива, но может быть и значительно выше уровня интеллекта самого "умного" члена указанного коллектива. Для того, чтобы количество интеллектуальных систем переходило в существенно более интеллектуальное качество коллектива таких систем, все объединяемые в коллектив интеллектуальные системы должны иметь высокий уровень интероперабельности, что накладывает дополнительные требования, предъявляемые к информации, хранимой в памяти, а также к решателям задач интеллектуальных систем, объединяемых в коллектив.

Интероперабельность кибернетической системы есть способность кибернетической системы взаимодействовать с другими кибернетическими системами в целях создания коллектива кибернетических систем (многоагентных систем), уровень качества и, в частности, уровень интеллекта которого выше уровня качества каждой кибернетической системы, входящей в состав этого коллектива.

Для того, чтобы количество членов коллектива кибернетической системы перешло в более высокое качество самого коллектива, члены коллектива должны обладать дополнительными способностями, которые будем называть свой-

ствами интероперабельности. Основными такими свойствами являются способность устанавливать и поддерживать достаточный уровень семантической совместимости (взаимопонимания) с другими кибернетическими системами и договороспособность (способность согласовывать свои действия с другими) *Neiva F.W.TowarPiISC-2016art.*

Целенаправленный обмен информацией между кибернетическими системами существенно ускоряет процесс их обучения (процесс накопления знаний и навыков). Следовательно, способность эффективно использовать указанный канал накопления знаний и навыков существенно повышает уровень обучаемости кибернетических систем. Повышение уровня интероперабельности кибернетической системы является, с одной стороны, дополнительным повышением уровня интеллекта самой этой кибернетической системы, а также фактором повышения уровня интеллекта тех коллективов, тех многоагентных систем, в состав которых эта кибернетическая система входит.

интероперабельность кибернетической системы

⇒ *свойство-предпосылка*:*

- договороспособность кибернетической системы
- социальная ответственность кибернетической системы
- социальная активность кибернетической системы

Свойства-предпосылки уровня договороспособности кибернетической системы представлены ниже:

договороспособность кибернетической системы

⇒ *свойство-предпосылка*:*

- способность кибернетической системы к пониманию принимаемых сообщений
- способность кибернетической системы к формированию передаваемых сообщений, понятных адресатам
- способность кибернетической системы к обеспечению семантической совместимости с партнёрами
- коммуникабельность кибернетической системы
- способность кибернетической системы к обсуждению и согласованию целей и планов коллективной деятельности
- способность кибернетической системы брать на себя выполнение актуальных задач в рамках согласованных планов коллективной деятельности

Понимание информации, поступающей извне, включает в себя:

- перевод этой информации на внутренний язык кибернетической системы;
- локальную верификацию вводимой информации;
- погружение (конвергенцию, размещение) текста, являющегося результатом указанного перевода в состав хранимой информации (в частности, в состав базы знаний).

Погружение вводимой информации в состав базы знаний кибернетической системы сводится к выявлению и устранению противоречий, возникающих между погружаемым текстом и текущего состояния базы знаний. Сложность проблемы понимания вводимой вербальной информации заключается не только в сложности непротиворечивого погружения вводимой информации в текущее состояние базы знаний, но и в сложности трансляции этой информации с внешнего языка на внутренний язык кибернетической системы, т. е. в сложности генерации текста внутреннего языка, семантически эквивалентного вводимому тексту внешнего языка. Для естественных языков указанная трансляция является сложной задачей, так как в настоящее время проблема формализации синтаксиса и семантики естественных языков не решена.

Семантическая совместимость двух заданных кибернетических систем определяется согласованностью систем понятий, используемых обеими взаимодействующими кибернетическими системами. Проблема обеспечения permanentной поддержки семантической совместимости взаимодействующих кибернетических систем является необходимым условием обеспечения высокого уровня взаимопонимания кибернетических систем и, как следствие, эффективного их взаимодействия.

Коммуникабельность кибернетической системы есть способность кибернетической системы к установлению взаимовыгодных контактов с другими кибернетическими системами (в том числе, с коллективами интеллектуальных систем) путем честного выявления взаимовыгодных общих целей (интересов).

Свойства-предпосылки уровня социальной ответственности кибернетической системы представлены ниже:

социальная ответственность кибернетической системы

⇒ *свойство-предпосылка*:*

- способность кибернетической системы выполнять качественно и в срок взятые на себя обязательства в рамках соответствующих коллективов
- способность кибернетической системы адекватно оценивать свои возможности при распределении коллективной деятельности
- альтруизм/эгоизм кибернетической системы

- отсутствие/наличие действий, которые по безграмотности кибернетической системы снижают качество коллективов, в состав которых она входит
- отсутствие/наличие "осознанных", мотивированных действий, снижающих качество коллективов, в состав которых кибернетическая система входит

Свойства-предпосылки уровня социальной активности кибернетической системы представлены ниже:

социальная активность кибернетической системы

⇒ свойство-предпосылка*:

- способность кибернетической системы к генерации предлагаемых целей и планов коллективной деятельности
- активность кибернетической системы в экспертизе результатов других участников коллективной деятельности
- способность кибернетической системы к анализу качества всех коллективов, в состав которых она входит, а также всех членов этих коллективов
- способность кибернетической системы к участию в формировании новых коллективов
- количество и качество тех коллективов, в состав которых кибернетическая система входит или входила

Формирование специализированного коллектива кибернетических систем сводится к тому, что в памяти каждой кибернетической системы, входящей в коллектив, генерируется спецификация этого коллектива, включающая в себя:

- перечень весь членов коллектива;
- способности каждого из членов коллектива;
- их обязанности в рамках коллектива;
- спецификацию всего множества задач (вида деятельности), для решения (выполнения) которых сформирован данный коллектив кибернетических систем.

Каждая кибернетическая система может входить в состав большого количества коллективов, выполняя при этом в разных коллективах в общем случае разные "должностные обязанности", разные "бизнес-процессы".

§ 1.1.3. Эволюция традиционных и интеллектуальных компьютерных систем

⇒ ключевое понятие*:

- компьютерная система
- память компьютерной системы
- информационный процесс в памяти компьютерной системы
- компьютер
- интерфейс компьютерной системы
- пользовательский интерфейс компьютерной системы
- традиционная компьютерная система
- алгоритм
- абстрактная машина Тьюринга
- абстрактная машина фон-Неймана
- параллельная программа
 - := [программа, при выполнении (интерпретации) которой инициируются одновременно выполняемые информационные процессы]
- объектно-ориентированная программа
- интерпретация программы
- компиляция программы
- уровень языка программирования
- база данных
- интеллектуальная компьютерная система
- логическая программа
- функциональная программа
- продукционная программа
- искусственная нейронная сеть
- база знаний
- интеллектуальный решатель задач

⇒ *ключевое знание**:

- Классификация компьютерных систем
- Обобщённая архитектура компьютерных систем
- Классификация традиционных компьютерных систем
- Эволюция традиционных компьютерных систем
- Эволюция языков программирования
- Эволюция систем программирования
- Классификация интеллектуальных компьютерных систем
- Обобщённая архитектура интеллектуальных компьютерных систем
- Отличия интеллектуальных компьютерных систем от традиционных компьютерных систем
- Эволюция интеллектуальных компьютерных систем

⇒ *библиографическая ссылка**:

- Cho J..StramMtToCBS-2019art
- Sherif Y.S..CompuSDQAMaM-1988art
- Laird J.E..Claim aCiEHIS-2009art
- Gao R..PerfoMfISaEP-2002art

Были предложены различные системные показатели для измерения качества компьютерных систем. Поскольку компьютерные системы становятся все более сложными и включают множество подсистем или компонентов, измерение их качества в нескольких измерениях становится сложной задачей *Cho J..StramMtToCBS-2019art*. Метрики качества включают в себя набор мер, которые могут описывать атрибуты системы в терминах, не зависящих от структуры, которая приводит к этим атрибутам; эти меры должны быть выражены количественно и должны иметь значительный уровень точности и надежности *Sherif Y.S..CompuSDQAMaM-1988art*.

Целью является построение компьютерных систем, которые демонстрируют весь спектр когнитивных способностей, которые мы обнаруживаем у людей *Laird J.E..Claim aCiEHIS-2009art*.

Исследователи искусственного интеллекта определяют интеллект как неотъемлемое свойство машины *Gao R..PerfoMfISaEP-2002art*.

Глава 1.2.

Интеллектуальные компьютерные системы нового поколения

⇒ *автор**:

- Голенков В.В.
- Шункевич Д.В.
- Ковалёв М.В.
- Садовский М.Е.

⇒ *аннотация**:

[В главе рассмотрены принципы построения интеллектуальных компьютерных систем нового поколения. В качестве ключевых свойств интеллектуальных систем нового поколения выделяются их семантическая совместимость и интероперабельность. В главе рассматривается подход к обеспечению указанных свойств на основе смыслового представления информации и многоагентных моделей обработки информации.]

⇒ *подраздел**:

- § 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения
- § 1.2.2. Принципы, лежащие в основе смыслового представления информации
- § 1.2.3. Принципы, лежащие в основе многоагентных моделей решателей задач интеллектуальных компьютерных систем нового поколения
- § 1.2.4. Принципы, лежащие в основе онтологических моделей мультимодальных интерфейсов интеллектуальных компьютерных систем нового поколения
- § 1.2.5. Достиоинства предлагаемых принципов, лежащих в основе интеллектуальных компьютерных систем нового поколения

⇒ *ключевое понятие**:

-
-
-

⇒ *ключевое знание**:

-
-
-

⇒ *ключевая сущность, не являющаяся понятием и знанием**:

-
-
-

⇒ *библиографическая ссылка**:

- Летичевский А.А..ИнсерП-2003ст

⇒ *ключевая библиографическая ссылка**:

-

Введение в Главу 1.2.

Важнейшим направлением повышения уровня интеллекта индивидуальных интеллектуальных кибернетических систем является переход к коллективам индивидуальных интеллектуальных кибернетических систем и далее к иерархическим коллективам интеллектуальных кибернетических систем, членами которых являются как индивидуальные интеллектуальные кибернетические системы, так и коллективы индивидуальных интеллектуальных кибернетических систем, а также иерархические коллективы интеллектуальных кибернетических систем.

Аналогичным образом необходимо повышать уровень интеллекта и индивидуальных интеллектуальных компьютерных систем (искусственных кибернетических систем). Но при этом надо помнить, что далеко не каждое

объединение интеллектуальных кибернетических систем (в том числе и компьютерных систем) становится интеллектуальным коллективом. Для этого необходимо соблюдение дополнительных требований, предъявляемых ко всем членам интеллектуальных коллективов. Важнейшим из них является требование высокого уровня интероперабельности, то есть способности к эффективному взаимодействию с другими членами коллектива. Переход от современных интеллектуальных компьютерных систем к интероперабельным интеллектуальным компьютерным системам является ключевым фактором перехода к интеллектуальным компьютерным системам нового поколения, обеспечивающим существенное повышение уровня автоматизации человеческой деятельности.

Расширение областей применения интеллектуальных компьютерных систем требует перехода к решению компьютерных задач — задач, решение которых невозможно с помощью

- одной модели решения задач,
- одного вида знаний,
- одной интеллектуальной компьютерной системы.

Для этого необходим:

- Переход к гибридным, мультиагентным интеллектуальным компьютерным системам
 - конвергенция и интеграция различных моделей решения задач и различных видов знаний в рамках любой самостоятельной интеллектуальной компьютерной системы (с одной памятью)
- Переход к коллективам семантически совместимых самостоятельных интеллектуальных компьютерных систем
 - интероперабельность интеллектуальных компьютерных систем
 - конвергенция и интеграция различных интеллектуальных компьютерных систем как самостоятельных субъектов

§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения

⇒ ключевое понятие*:

- интеллектуальная компьютерная система нового поколения
- интероперабельная интеллектуальная компьютерная система
- гибридная интеллектуальная компьютерная система

⇒ ключевое отношение*:

- соединение интеллектуальных компьютерных систем*
:= [преобразование множества интеллектуальных компьютерных систем в коллектив, членами (агентами) которого являются эти системы*]
- глубокая интеграция интеллектуальных компьютерных систем*
:= [быть результатом преобразования множества индивидуальных интеллектуальных компьютерных систем в одну интегрированную индивидуальную интеллектуальную компьютерную систему*]

⇒ ключевой параметр*:

- интероперабельность интеллектуальных компьютерных систем[^]
- семантическая совместимость пар интеллектуальных компьютерных систем[^]

⇒ ключевое значение*:

- Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения
- Принципы, лежащие в основе интеллектуальных компьютерных систем нового поколения
- Отличие данных от знаний

Создание различных комплексов взаимодействующих интеллектуальных компьютерных систем требует повышения качества не только самих этих систем, но также и качества их взаимодействия. Интеллектуальные компьютерные системы нового поколения должны иметь высокий уровень интероперабельности, то есть высокий уровень способности к эффективному, целенаправленному взаимодействию с себе подобными и с пользователями в процессе коллективного (распределённого) и децентрализованного решения сложных задач [Yaghoobirafi K..aApprofSliADIS-2022art](#); [Ouksel A.M..SemaniGIS-1999art](#); [Lanzenberger M..MakinOTKLiSW-2008art](#); [Neiva F.W..TowarPItSC-2016art](#); [Pohl J.Inter a tNfISaHP-2004art](#); [Waters J..GlobalIUS-2009art](#). Уровень интероперабельности интеллектуальных компьютерных систем — это, образно говоря, уровень их "социализации", полезности в рамках различных априори неизвестных сообществ (коллективов) интеллектуальных систем.

Уровень интероперабельности интеллектуальных компьютерных систем — это уровень их коммуникационной (социальной) совместимости, позволяющей им самостоятельно формировать коллективы интеллектуальных компьютерных систем и их пользователей, а также самостоятельно согласовывать и координировать свою деятельность в рамках этих коллективов при решении сложных задач в частично предсказуемых условиях. Повышение уровня интероперабельности интеллектуальных компьютерных систем определяет переход к интеллекту-

альным компьютерным системам нового поколения, без которых невозможна реализация таких проектов, как *smart-предприятие*, *smart-больница*, *smart-школа*, *smart-университет*, *smart-кафедра*, *smart-дом*, *smart-город*, *smart-общество* Lopes L.S..SemantIoT-2022art; Hamilton Inter aKEftG-2006art.

интеллектуальная компьютерная система

:= [интеллектуальная искусственная кибернетическая система]

⇒ разбиение*:

- {• индивидуальная интеллектуальная компьютерная система

- интеллектуальный коллектив интеллектуальных компьютерных систем

:= [интеллектуальная многоагентная система, агенты которой являются интеллектуальными компьютерными системами]

⇒ примечание*:

[Не каждый коллектив интеллектуальных компьютерных систем может оказаться интеллектуальным, поскольку уровень интеллекта такого коллектива определяется не только уровнем интеллекта его членов, но также и эффективностью (качеством) их взаимодействия.]

⇒ разбиение*:

- {• интеллектуальный коллектив индивидуальных интеллектуальных компьютерных систем

- иерархический интеллектуальный коллектив интеллектуальных компьютерных систем

:= [интеллектуальный коллектив интеллектуальных компьютерных систем, по крайней мере одним из членов которого является интеллектуальный коллектив интеллектуальных компьютерных систем]

}

}

интеллектуальные компьютерные системы нового поколения

⇒ предъявляемые требования*:

- высокий уровень интероперабельности
- высокий уровень обучаемости
- высокий уровень гибридности
- высокий уровень способности решать интеллектуальные задачи (то есть задачи, методы решения которых и/или требуемая для их решения исходная информация априори неизвестны)
- высокий уровень синергетичности

интероперабельность[^]

:= [способность к эффективному (целенаправленному) взаимодействию с другими самостоятельными субъектами]

:= [способность к партнёрскому взаимодействию в решении комплексных задач, требующих коллективной деятельности]

:= [способность работать в коллективе (в команде)]

:= [уровень социализации]

:= [social skills]

высокий уровень интероперабельности

⇒ обеспечивается*:

- высоким уровнем взаимопонимания

⇒ обеспечивается*:

- высоким уровнем семантической совместимости заданного субъекта с другими субъектами данного коллектива

- высоким уровнем способности понимать сообщения и поведение партнеров

- высоким уровнем способности быть понятной для партнеров:

- способности понятно и обоснованно формулировать свои предложения и информацию, полезную для решения текущих задач;
- способности действовать и комментировать свои действия так, чтобы они и их мотивы были поняты партнерам;

- высоким уровнем способности к повышению уровня семантической совместимости со своими партнёрами

- высоким уровнем договороспособности, то есть способности согласовывать с партнёрами свои планы и намерения в целях своевременного обеспечения высокого качества коллективного результата

- высоким уровнем способности к децентрализованной координации своих действий с действиями партнёров в непредсказуемых (нештатных) обстоятельствах

•

высоким уровнем способности к минимизации негативных последствий конфликтных ситуаций с другими субъектами

⇒ *обеспечивается*:*

- *высоким уровнем способности к предотвращению возникновения конфликтных ситуаций*
- *соблюдением этических норм и правил, препятствующих возникновению разрушительных последствий конфликтных ситуаций*
- *высоким уровнем способности разделять ответственность с партнерами за своевременное и качественное достижение общей цели*

семантическая совместимость[^]

:= [степень согласованности (совпадения) систем понятий и других *ключевых сущностей*, используемых заданными взаимодействующими субъектами]

⇒ *примечание*:*

[Обеспечение *семантической совместимости* требует формализации *смыслового представления информации*]

способность разделять ответственность с партнерами, являющаяся необходимым условием децентрализованного управления коллективной деятельностью

⇒ *обеспечивается*:*

- *способностью к мониторингу и анализу коллективно выполняемой деятельности*
- *способностью оперативно информировать партнеров о неблагоприятных ситуациях, событиях, тенденциях, а также инициировать соответствующие коллективные действия*

обучаемость[^]

:= [способность быстро и качественно приобретать новые знания и навыки, а также совершенствовать уже приобретённые знания и навыки]

высокий уровень обучаемости

⇒ *обеспечивается*:*

- *высоким уровнем гибкости информации, хранимой в памяти интеллектуальной системы*
- *высоким уровнем качества стратификации информации, хранимой в памяти интеллектуальной системы (стратифицированностью базы знаний)*
- *высоким уровнем рефлексивности интеллектуальной системы*
- *высоким уровнем способности исправлять свои ошибки* (в том числе устранять противоречия в своей базе знаний)
- *высоким уровнем познавательной активности*
- *уровнем ограничений на вид приобретаемых знаний и навыков* (отсутствие таких ограничений означает потенциальную универсальность интеллектуальной системы)

гибридность[^]

:= [степень многообразия используемых видов знаний и моделей решения задач и уровень эффективности их совместного использования]

:= [индивидуальная способность решать *комплексные задачи*, требующие использования различных видов знаний, а также различных комбинаций различных моделей решения задач]

высокий уровень гибридности

⇒ *обеспечивается*:*

- *высокой степенью многообразия используемых видов знаний и моделей решения задач*
- *высокой степенью конвергенции и глубокой интеграции* (степенью взаимопроникновения) различных видов знаний и моделей решения задач
- *способностью неограниченно расширять уровень своей гибридности*

Подчеркнем, что *гибридность* и *интероперабельность интеллектуальных компьютерных систем нового поколения* предполагает отказ от известной парадигмы "черных ящиков", поскольку:

- всё многообразие моделей решения задач *гибридной интеллектуальной компьютерной системы* должно интерпретироваться на одной общей *универсальной платформе*;
- доступность информации о том, как устроен каждый используемый метод, модель решения задач, каждый субъект существенно повышает качество их *координации* при *совместном решении комплексных задач*;
- появляется возможность некоторые методы, модели решения задач и целевые субъекты (например, *интеллектуальные компьютерные системы*) использовать для совершенствования (повышения качества) других методов, моделей и субъектов.

Особо необходимо отметить следующие характеристики *интеллектуальных компьютерных систем нового поколения*:

- **степень конвергенции**, унификации и стандартизации *интеллектуальных компьютерных систем* и их компонентов и соответствующая этому **степень интеграции** (глубина интеграции) *интеллектуальных компьютерных систем* и их компонентов;
- **семантическая совместимость** между *интеллектуальными компьютерными системами* в целом и **семантическая совместимость** между компонентами каждой *интеллектуальной компьютерной системы* (в частности, совместимость между различными *видами знаний* и различными *моделями обработки знаний*), которые являются основными показателями степени **конвергенции** (ближения) между *интеллектуальными компьютерными системами* и их компонентами.

Особенность указанных характеристик *интеллектуальных компьютерных систем* их компонентов заключается в том, что они играют важную роль при решении всех ключевых задач современного этапа развития *Искусственного интеллекта* и тесно связаны друг с другом.

Заметим также, что перечисленные требования, предъявляемые к *интеллектуальным компьютерным системам нового поколения*, направлены на преодоление проклятия *Вавилонского столпотворения* **IIIadis2019** как внутри *интеллектуальных компьютерных систем нового поколения* (между внутренними информационными процессами решения различных задач), так и между взаимодействующими самостоятельными *интеллектуальными компьютерными системами нового поколения* в процессе коллективного решения комплексных задач.

На современном этапе эволюции *интеллектуальных компьютерных систем* для существенного расширения областей их применения и качественного повышения уровня автоматизации человеческой деятельности:

- Необходим переход к созданию **семантически совместимых интеллектуальных компьютерных систем нового поколения**, ориентированных не только на индивидуальное, но и на **коллективное** (совместное) решение комплексных задач, требующих скоординированной деятельности нескольких самостоятельных интеллектуальных компьютерных систем и использования различных моделей и методов в непредсказуемых комбинациях, что необходимо для существенного расширения сфер применения *интеллектуальных компьютерных систем*, для перехода от автоматизации локальных видов и областей человеческой деятельности к комплексной автоматизации более крупных (объединённых) видов и областей этой деятельности;
- Необходима разработка **Общей формальной теории и стандарта интеллектуальных компьютерных систем нового поколения**;
- Необходима разработка **Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения**, которая включает в себя поддержку проектирования этих систем (как начального этапа их жизненного цикла) и обеспечение их **совместимости** на всех этапах их жизненного цикла;
- Необходима **конвергенция и унификация интеллектуальных компьютерных систем нового поколения** и их компонентов;
- Необходима реализация "бесшовной", "диффузной", взаимопроникающей, **глубокой интеграции семантически смежных компонентов интеллектуальных компьютерных систем**, то есть интеграции, при которой отсутствуют чёткие границы ("швы") интегрируемых (соединяемых) компонентов, и которая может осуществляться **автоматически**. Это означает переход к **гибридным интеллектуальным компьютерным системам**;
- Необходимо соблюдение **Принципа бритвы Оккама** — максимально возможное структурное упрощение *интеллектуальных компьютерных систем нового поколения*, исключение **эклектических** решений;
- Необходима ориентация на потенциально **универсальные** (то есть способные быстро приобретать **любые** знания и навыки), **синергетические интеллектуальные компьютерные системы** с "сильным" интеллектом;

интеллектуальные компьютерные системы нового поколения

⇒ **принципы, лежащие в основе***:

- смысловое представление знаний в памяти *интеллектуальных компьютерных систем*, предполагающее отсутствие **омонимических знаков**, которые в разных контекстах обозначают разные сущности, а также отсутствие **синонимии**, то есть пар синонимичных знаков, которые обозначают одну и ту же сущность
- смысловое представление информационной конструкции в общем случае имеет нелинейный (графовый) характер представления информации, который является рафинированной **семантической сетью**
- фрактальный характер структуризации представляемых знаний
- использование **общего** для всех интеллектуальных компьютерных систем **универсального языка смыслового представления знаний** в памяти *интеллектуальных компьютерных систем*, обладающий максимально простым **синтаксисом**, обеспечивающим представление любых видов знаний и имеющий неограниченные возможности перехода от знаний к метазнаниям. Простота синтаксиса информационных конструкций указанного языка позволяет называть эти конструкции **рафинированными семантическими сетями**
-

структурно-перестраиваемая (графодинамическая) организация памяти интеллектуальных компьютерных систем, при которой обработка знаний сводится не столько к изменению состояния хранимых знаков, сколько к изменению конфигурации связей между этими знаками

- *семантически неограниченный ассоциативный доступ к информации, хранимой в памяти интеллектуальных компьютерных систем, по заданному образцу произвольного размера и произвольной конфигурации*
- *децентрализованное ситуационное управление информационными процессами в памяти интеллектуальных компьютерных систем, реализованное с помощью агентно-ориентированной модели обработки баз знаний, в котором инициирование новых информационных процессов осуществляется не путём передачи управления соответствующим априори известным процедурам, а в результате возникновения соответствующих ситуаций или событий в памяти интеллектуальной компьютерной системы, поскольку «основная проблема компьютерных систем состоит не в накоплении знаний, а в умении активизировать нужные знания в процессе решения задач» (Поспелов Д.А.). Такой многоагентный процесс обработки информации представляет собой деятельность, выполняемую некоторым коллективом самостоятельных информационных агентов (агентов обработки информации), условием инициирования каждого из которых является появление в текущем состоянии базы знаний соответствующей этому агенту *ситуации/или события*.*

«Выбор многоагентных технологий объясняется тем, что в настоящее время любая сложная производственная, логистическая или другая система может быть представлена набором взаимодействий более простых систем до любого уровня детальности, что обеспечивает фрактально-рекурсивный принцип построения многоярусных систем, построенных как открытые цифровые колонии и экосистемы ИИ. В основе многоагентных технологий лежит распределенный или децентрализованный подход к решению задач, при котором динамически обновляющаяся информация в распределенной сети интеллектуальных агентов обрабатывается непосредственно у агентов вместе с локально доступной информацией от "соседей". При этом существенно сокращаются как ресурсные и временные затраты на коммуникации в сети, так и время на обработку и принятие решений в центре системы (если он все-таки есть).»

⇐ цитата*:

Баринов И.И.. ФормиСРКпИИ-2021ст стр. 270

- Переход к *семантическим моделям решения задач*, в основе которых лежит учёт не только синтаксических (структурных) аспектов обрабатываемой информации, но также и семантических (смыслоных) аспектов этой информации — “From data science to knowledge science”
- *онтологическая модель баз знаний интеллектуальных компьютерных систем*, то есть онтологическая структуризация всей информации, хранимой в памяти интеллектуальной компьютерной системы, предполагающая четкую *стратификацию* базы знаний в виде иерархической системы *предметных областей* и соответствующих им *онтологий*, каждая из которых обеспечивает семантическую *спецификацию* всех *понятий*, являющихся ключевыми в рамках соответствующей *предметной области*
- *онтологическая локализация решения задач* в интеллектуальных компьютерных системах, предполагающая локализацию области действия каждого хранимого в памяти метода и каждого *информационного агента* в соответствии с *онтологической моделью* обрабатываемой базы знаний. Чаще всего, такой *областью действия* является одна из *предметных областей* либо одна из *предметных областей* вместе с соответствующей ей *онтологией*
- *онтологическая модель интерфейса* интеллектуальной компьютерной системы, в состав которой входит:
 - онтологическое описание *синтаксиса* всех языков, используемых интеллектуальной компьютерной системой для общения с внешними субъектами;
 - онтологическое описание *денотационной семантики* каждого языка, используемого интеллектуальной компьютерной системой для общения с внешними субъектами;
 - семейство *информационных агентов*, обеспечивающих *синтаксический анализ*, *семантический анализ* (перевод на внутренний смысловой язык) и *понимание* (погружение в базу знаний) любого введенного *сообщения*, принадлежащего любому *внешнему языку*, полное онтологическое описание которого находится в базе знаний интеллектуальной компьютерной системы;
 - семейство *информационных агентов*, обеспечивающих *синтез сообщений*, которые (1) адресуются внешним субъектам, с которыми общается интеллектуальная компьютерная система, (2) *семантически эквивалентны* заданным *фрагментам* базы знаний интеллектуальной компьютерной системы, определяющим *смысл* передаваемых *сообщений*, (3) принадлежат одному из *внешних языков*, полное онтологическое описание которого находится в базе знаний интеллектуальной компьютерной системы;
- *семантически дружественный характер пользовательского интерфейса*, обеспечиваемый (1) формальным описание в базе знаний средства управления пользовательским интерфейсом и (2) введением в состав интеллектуальной компьютерной системы соответствующих *help-подсистем*, обеспечивающих существенное снижение языкового барьера между пользователями и интеллектуальными компьютерными

- системами, что существенно повысит эффективность эксплуатации интеллектуальных компьютерных систем*
- *минимизация негативного влияния человеческого фактора* на эффективность эксплуатации интеллектуальных компьютерных систем благодаря реализации интероперабельного (партнерского) стиля взаимодействия не только между самими интеллектуальными компьютерными системами, но также и между интеллектуальными компьютерными системами и их пользователями. Ответственность за качество совместной деятельности должно быть распределено между всеми партнёрами
 - **мультимодальность** (гибридный характер) интеллектуальной компьютерной системы, что предполагает:
 - многообразие видов знаний, входящих в состав базы знаний интеллектуальной компьютерной системы;
 - многообразие моделей решения задач, используемых решателем задач интеллектуальной компьютерной системы;
 - многообразие сенсорных каналов, обеспечивающих мониторинг состояния внешней среды интеллектуальной компьютерной системы;
 - многообразие эффекторов, осуществляющих воздействие на внешнюю среду;
 - многообразие языков общения с другими субъектами (с пользователями, с интеллектуальными компьютерными системами);
 - **внутренняя семантическая совместимость** между компонентами интеллектуальной компьютерной системы (то есть максимально возможное введение общих, совпадающих понятий для различных фрагментов хранимой базы знаний), являющаяся формой **конвергенции** и глубокой интеграции внутри интеллектуальной компьютерной системы для различного вида знаний и различных моделей решения задач, что обеспечивает эффективную реализацию мультимодальности интеллектуальной компьютерной системы
 - **внешняя семантическая совместимость** между различными интеллектуальными компьютерными системами, выражаясь не только в общности используемых понятий, но и в общности базовых знаний и являясь необходимым условием обеспечения высокого уровня интероперабельности интеллектуальных компьютерных систем
 - ориентация на использование интеллектуальных компьютерных систем как когнитивных агентов в составе **иерархических многоагентных систем**
 - фрактальный характер коллективов интеллектуальных компьютерных систем нового поколения
 - **платформенная независимость интеллектуальных компьютерных систем**, предполагающая:
 - четкую стратификацию каждой интеллектуальной компьютерной системы (1) на логико-семантическую модель, представленную ее базой знаний, которая содержит не только декларативные знания, но и знания, имеющие операционную семантику, и (2) на платформу, обеспечивающую интерпретацию указанной логико-семантической модели;
 - универсальность указанной платформы интерпретации логико-семантической модели интеллектуальной компьютерной системы, что дает возможность каждой такой платформе обеспечивать интерпретацию любой логико-семантической модели интеллектуальной компьютерной системы, если эта модель представлена на том же универсальном языке смыслового представления информации;
 - многообразие вариантов реализации платформы интерпретации логико-семантических моделей интеллектуальных компьютерных систем — как вариантов, программно реализуемых на современных компьютерах, так и вариантов, реализуемых в виде универсальных компьютеров нового поколения, ориентированных на использование в интеллектуальных компьютерных системах нового поколения (такие компьютеры мы называли *ассоциативными семантическими компьютерами*);
 - легко реализуемую возможность переноса (переустановки) логико-семантической модели (базы знаний) любой интеллектуальной компьютерной системы на любую другую платформу интерпретации логико-семантических моделей;
 - изначальная ориентация интеллектуальных компьютерных систем нового поколения на использование **универсальных ассоциативных семантических компьютеров** (компьютеров нового поколения) в качестве платформы интерпретации логико-семантических моделей (баз знаний) интеллектуальных компьютерных систем

Принципы, лежащие в основе интеллектуальных компьютерных систем нового поколения

- фрактальность
 - база знаний
 - ostis-система (в рамках Экосистемы OSTIS)
- := [масштабируемое самоподобие]

интеллектуальная компьютерная система нового поколения

⇒ *принципы, лежащие в основе**:

- смысловое представление
 - универсальный язык смыслового представления
 - рафинированная семантическая сеть
- универсальная ситуационная многоагентная модель абстрактных знаний, ориентированная на обработку смыслового представления информации в ассоциативной графодинамической памяти, в которой (1) обработка информации сводится не только к изменению собственных элементов памяти, но и к изменениям кофенденциальных связей между ними, (2) инициирование информационных процессов осуществляется возникновением соответствующих ситуаций и событий в памяти
- агентно-ориентированная модель обработки знаний в памяти интеллектуальной компьютерной системы, обеспечивающая высокую степень интероперабельности между внутренними агентами интеллектуальной компьютерной системы, взаимодействующими через общую память (это, фактически, "внутренняя" интероперабельность интеллектуальной компьютерной системы нового поколения)

В настоящее время разработано большое количество различного вида *моделей решения задач*, моделей представления и обработки знаний различного вида. Но в разных *интеллектуальных компьютерных системах* могут быть востребованы разные комбинации этих моделей. При разработке и реализации различных *интеллектуальных компьютерных систем* соответствующие методы и средства должны гарантировать *логико-семантическую совместимость* разрабатываемых компонентов и, в частности, их способность использовать общие *информационные ресурсы*. Для этого, очевидно, необходима *унификация* указанных моделей.

Многообразие различных видов интеллектуальных компьютерных систем и, соответственно, многообразие используемых ими комбинаций моделей представления знаний и решения задач определяется:

- многообразием назначения интеллектуальных компьютерных систем и вида окружающей их среды;
- многообразием назначения интеллектуальных компьютерных систем и вида окружающей их среды;
- многообразием различных видов хранимых знаний; многообразием моделей обработки знаний и решений задач;
- многообразием различных видов интерфейсов (обработки сигналов, аудио, видео, эффекторных средств).

Следует выделить следующие аспекты *совместимости* моделей представления и обработки знаний в *интеллектуальных компьютерных системах*:

- синтаксический;
- семантический (согласованность систем понятий, их денотационной семантики);
- функциональный (операционный).

Следует также отличать:

- *совместимость* между компонентами *интеллектуальных компьютерных систем*;
- *совместимость* между верхним логико-семантическим уровнем используемых моделей представления и обработки знаний и различными уровнями их интерпретации вплоть до аппаратного уровня;
- *совместимость* между индивидуальными интеллектуальными компьютерными системами;
- *совместимость* между индивидуальными интеллектуальными компьютерными системами и их пользователями;
- *совместимость* между коллективами интеллектуальных компьютерных систем.

следует отличать*

Э { • *данные*
 := [информационная конструкция, обрабатываемая с помощью программы традиционного языка программирования]
 • *знание*
 := [семантически целостный фрагмент базы знаний]
 }

⇒ *отличие*:*

[Для каждого знания всегда известен язык, на котором это знание представлено и денотационная семантика которого задана. При этом указанный язык имеет достаточно большую семантическую мощность, а в идеале является универсальным языком. В отличие от этого структуризация данных для традиционных программ осуществляется в целях упрощения самих этих программ и, следовательно, для разных программ в общем случае осуществляется по-разному. Таким образом, при разработке традиционных программ представление обрабатываемых данных осуществляется в общем случае на разных языках, денотационная семантика которых нигде не документируется и известна только разработчикам программ. Другими словами, данные для разных программ имеют денотационную семантику не только разную, но еще и априори неизвестную. Поэтому это форма проявления вавилонского столпотворения в традиционных языках программирования.]

§ 1.2.2. Принципы, лежащие в основе смыслового представления информации

⇒ *ключевое понятие**:

- *смысловое представление информации*
:= [смысл]
- *семантическая сеть*
- *рафинированная семантическая сеть*
- *граф знаний*
:= [представление сложноструктурированного знания в виде графовой структуры]
- *универсальный язык семантических сетей*
:= [универсальный язык, информационными конструкциями которого являются семантические сети]

⇒ *ключевое знание**:

- *Принципы, лежащие в основе смыслового представления информации*

смысловое представление информации

:= [запись (представление) информационной конструкции на смысловом уровне]

:= [информационная конструкция синтаксическая структура которой близка её смыслу, то есть близка описываемой конфигурации связей между описываемыми сущностями]

:= [смысловое представление информационной конструкции]

⊂ *семантическая сеть*

⊂ *рафинированная семантическая сеть*

рафинированная семантическая сеть

⇒ *принципы, лежащие в основе**:

- Каждый элемент (синтаксически атомарный фрагмент) рафинированной семантической сети является знаком одной из описываемых сущностей
- Каждая сущность, описываемая рафинированной семантической сетью, должна быть представлена своим знаком, который является элементом этой сети
- В рамках каждой отдельной рафинированной семантической сети отсутствует синонимия разных знаков, а также отсутствуют омонимичные знаки
- Многообразие сущностей, описываемых рафинированными семантическими сетями, ничем не ограничивается. Соответственно этому, семантическая типология элементов рафинированных семантических сетей является весьма богатой
- Особым видом элементов рафинированных семантических систем являются знаки связей между другими элементами этих сетей. При этом, связываемыми элементами (то есть элементами, которые инцидентны указанным знакам связей) могут быть также и знаки других связей. Чаще всего знак связи между элементами рафинированной семантической сети является отражением связи между сущностями, которые обозначаются указанными элементами. Но в некоторых случаях знак связи между элементами рафинированной семантической сети может быть отражением, например, связи между одной описываемой сущностью и знаком другой описываемой сущности

§ 1.2.3. Принципы, лежащие в основе многоагентных моделей решателей задач интеллектуальных компьютерных систем нового поколения

⇒ *ключевое понятие**:

- *смысловая память*
- *графодинамическая память*
- *ассоциативная память с информационным доступом по образцу произвольного размера и конфигурации*
• *система ситуационного децентрализованного управления информационными процессами*
- *многоагентная система обработки информации в общей памяти*
- *язык смыслового представления задач*
- *универсальный язык смыслового представления знаний*
- *язык смыслового представления методов*
:= [интегрированный язык смыслового представления различного вида программ]
- *инсерционная программа*

⇒ *ключевое знание**:

- *Принципы, лежащие в основе решателей задач индивидуальных интеллектуальных компьютерных систем нового поколения*

⇒ библиографическая ссылка*:

- *Летичевский А.А..ИнсерП-2003ст*

решатель задач интеллектуальных компьютерных систем нового поколения

⇒ предъявляемые требования*:

- решатель задач интеллектуальных компьютерных систем нового поколения должен уметь решать интеллектуальные задачи, к числу которых относятся следующие виды задач:
 - некачественно сформулированная задача
:= [задача, формулировка которой содержит различные не-факторы (неполнота, нечеткость, противоречивость (некорректность) и так далее)]
 - задача, для решения которой, кроме самой формулировки задачи и соответствующего метода её решения необходима дополнительная, но априори неизвестна какая информация об объектах, указанных в формулировке (постановке) задачи. При этом указанная дополнительная информация может присутствовать, а может и отсутствовать в текущем состоянии базы знаний интеллектуальных компьютерных систем. Кроме того, для некоторых задач может быть задана (указана) та область базы знаний, использования которой достаточно для поиска или генерации (в частности, логического вывода) указанной дополнительной требуемой информации. Такую область базы знаний будем называть областью решения соответствующей задачи
 - задача, для которой соответствующий метод ее решения в текущий момент не известен. Для решения такой задачи можно:
 - переформулировать задачу, то есть сгенерировать (логически вывести) логически эквивалентную формулировку исходной задачи, для которой метод её решения в текущий момент является известным;
 - свести исходную задачу к семейству подзадач, для которых методы их решения в текущий момент известны.
- процесс решения задач в интеллектуальных компьютерных системах нового поколения реализуется коллективом информационных агентов, обрабатывающих базу знаний интеллектуальных компьютерных систем
- управление информационными процессами в памяти интеллектуальных компьютерных систем нового поколения осуществляется децентрализованным образом по принципам ситуационного управления

ситуационное управление

:= [ситуационно-событийное управление]

⇒ пояснение*:

[управление последовательностью выполнения действий, при котором условием ("триггером") инициирования указанных действий является:

- возникновение некоторых ситуаций (условий, состояний);
- и/или возникновение некоторых событий

]

ситуация

:= [структура, описывающая некоторую временно существующую конфигурацию связей между некоторыми сущностями]

:= [описание временно существующего состояния некоторого фрагмента (некоторой части) некоторой динамической системы]

событие

▷ возникновение временной сущности

:= [появление, рождение, начало существования некоторой временной сущности]

▷ исчезновение временной сущности

:= [прекращение, завершение существования некоторой временной сущности]

▷ переход от одной ситуации к другой

⇒ примечание*:

[Здесь учитывается не только факт возникновения новой ситуации, но и её предыстория — то есть та ситуация, которая ей непосредственно предшествует. Так, например, реагируя на аномальное значение какого-либо параметра, нам важно знать:

- какова динамика изменения этого параметра (увеличивается он или уменьшается и с какой скоростью);
- какие меры были предприняты ранее для ликвидации этой аномалии.

]

§ 1.2.4. Принципы, лежащие в основе онтологических моделей мультимодальных интерфейсов интеллектуальных компьютерных систем нового поколения

решатель задач индивидуальной интеллектуальной компьютерной системы нового поколения

⇒ *принципы, лежащие в основе*:*

- Смысловое представление обрабатываемых знаний
- Семантически неограниченный ассоциативный доступ к различным фрагментам знаний, хранимым в памяти интеллектуальных компьютерных систем нового поколения (доступ по заданному образцу произвольного размера и произвольной конфигурации)
- Графодинамический характер обработки знаний в памяти, при котором обработка знаний сводится не только к изменению состояния атомарных фрагментов (ячеек) памяти, но и к изменению конфигурации связей между этими атомарными фрагментами
- Ситуационное децентрализованное управление процессом обработки знаний, а также процессом организации взаимодействия интеллектуальных компьютерных систем с внешней средой
- Использование семантически мощного языка задач, обеспечивающего представление формулировок самых различных задач, которые могут решаться либо в рамках памяти интеллектуальной компьютерной системы, либо во внешней среде и которые осуществляют инициирование соответствующих процессов решения задач
- Многоагентный характер реализации процессов решения инициированных задач, в основе которого лежит иерархическая система агентов, каждый из которых активизируется при возникновении в памяти интеллектуальной компьютерной системы соответствующий ситуации или соответствующего события

§ 1.2.4. Принципы, лежащие в основе онтологических моделей мультимодальных интерфейсов интеллектуальных компьютерных систем нового поколения

⇒ *ключевое понятие*:*

- *мультимодальный интерфейс*
- *верbalный интерфейс*
- *естественно-языковой интерфейс*
- *внешний язык*
:= [язык обмена сообщениями]
- *внутренний язык*
:= [язык представления информации в памяти кибернетической системы]
- *синтаксис внешнего языка*
- *денотационная семантика внешнего языка*
- *интерфейсная задача*
- *понимание сообщения*
- *синтез сообщения*
- *невербальный интерфейс*
- *сенсор*
:= [рецептор]
- *сенсорная подсистема*
- *мультисенсорная подсистема*
- *сенсорная информация*
- *эффектор*
- *мультиэффекторная подсистема*
- *сенсо-моторная координация*

⇒ *ключевое знание*:*

- *Принципы, лежащие в основе интерфейсов интеллектуальных компьютерных систем нового поколения*

интерфейс интеллектуальной компьютерной системы нового поколения

⇒ *принципы, лежащие в основе*:*

- *интерфейс интеллектуальной компьютерной системы нового поколения* рассматривается как решатель задач частного вида — *интерфейсных задач*, основными из которых являются:
 - задачи понимания вербальной информации, приобретаемой интеллектуальной компьютерной системой (синтаксический анализ, семантический анализ и погружение в базу знаний интеллектуальной компьютерной системы)
 - задачи понимания невербальной информации, воспринимаемой сенсорными подсистемами интеллектуальной компьютерной системы (анализ изображений, анализ аудио-сигналов, погружение результатов анализа в базу знаний интеллектуальной компьютерной системы)
 - задачи синтеза сообщений, адресуемых внешним субъектам (кибернетическим системам)

- тот факт, что интерфейс *интеллектуальной компьютерной системы нового поколения* является решателем частного вида задач *интеллектуальной компьютерной системы нового поколения*, свойства, лежащие в основе решателей задач *интеллектуальной компьютерной систем нового поколения*, наследуются интерфейсами *интеллектуальной компьютерной систем нового поколения*. Из этого следует, что в основе интерфейса *интеллектуальной компьютерной систем нового поколения* лежит:
 - смысловое представление накапливаемых (приобретаемых знаний);
 - трактовка семантического анализа приобретаемой вербальной информации как процесса перевода этой информации на внутренний язык смыслового представления знаний с последующим погружением (вводом, интеграцией) результата этого перевода в состав текущего состояния базы знаний *интеллектуальной компьютерной системы нового поколения*
 - трактовка синтеза сообщений, адресуемых внешними субъектами как процесса обратного перевода некоторого фрагмента базы знаний с внутреннего языка смыслового представления информации на внешний язык, используемый для общения с заданным субъектом
 - агентно-ориентированная организация решения интерфейсных задач, реализуемая соответствующим коллективом внутренних агентов интерфейса *интеллектуальных компьютерных систем нового поколения*, взаимодействующих через общедоступную для них базу знаний *интеллектуальной компьютерной системы нового поколения*
- интерфейс *интеллектуальной компьютерной системы нового поколения* трактуется как специализированная встроенная *интеллектуальная компьютерная система нового поколения*, входящая в состав указанной выше интеллектуальной компьютерной системы, база знаний которой включает в себя:
 - онтологию синтаксиса внутреннего языка смыслового представления информации
 - онтологию денотационной семантики внутреннего языка смыслового представления информации
 - онтологию синтаксиса всех внешних языков, используемых для общения с внешними субъектами
 - онтологии денотационной семантики всех внешних языков, используемых для общения с внешними субъектами (каждая такая онтология с формальной точки зрения является описанием соответствия между текстами внешних языков и семантически эквивалентными им текстами внутреннего языка смыслового представления информации).

Подчеркнем при этом, что все указанные онтологии, входящие в состав базы знаний интерфейса интеллектуальных компьютерных систем нового поколения, как и вся остальная информация, входящая в состав этой базы знаний, представляется на внутреннем языке смыслового представления информации, который, соответственно используется в данном случае как метаязык.

интерфейс индивидуальной интеллектуальной компьютерной системы нового поколения

⇒ *принципы, лежащие в основе**:

- Интерфейс индивидуальной интеллектуальной компьютерной системы нового поколения является специализированным компонентом решателя задач интеллектуальной компьютерной системы нового поколения, то есть специализированной встроенной (в индивидуальную интеллектуальную компьютерную систему нового поколения) интеллектуальной компьютерной системой нового поколения, ориентированной на решение интерфейсных задач, к которым относятся:
 - Понимание принятых сообщений (их перевод на язык внутреннего смыслового представления информации и погружения в текущее состояние базы знаний);
 - Синтез передаваемых сообщений (перевод сформированного сообщения с внутреннего языка смыслового представления на используемый внешний язык);
 - Первичный анализ приобретаемой сенсорной информации, предполагающий распознавание некоторого семейства первичных образов и сцен;
 - Сенсомоторная координация действий, выполняемых эффекторами интеллектуальной компьютерной системы
- Мультимодальный характер интерфейса — многообразие внешних языков, видов сенсоров и эффекторов
- Формальное онтологическое описание на языке внутреннего смыслового представления информации
 - Синтаксиса и денотационной семантики всех используемых внешних языков;
 - Первичных образов и сцен (ситуаций), являющихся результатом первичного анализа приобретаемой сенсорной информации;
 - Методов низкого уровня, непосредственно интерпретируемых эффекторами интеллектуальной компьютерной системы.

Разговоры о дружественном и, в частности, адаптивном пользовательском интерфейсе ведутся давно, но это, чаще всего, касается формы ("синтаксической" стороны) пользовательского интерфейса, а не смыслового содержания взаимодействия с пользователями. В настоящее время пользовательские интерфейсы компьютерных систем (в том числе и *интеллектуальных компьютерных систем*) для широкого контингента пользователей не являются семантически (содержательно) дружественными (семантически комфортными). Организация взаимодействия пользователей с компьютерными системами (в том числе и с *интеллектуальными компьютерными системами*)

§ 1.2.5. Достоинства предлагаемых принципов, лежащих в основе интеллектуальных компьютерных систем нового поколения

является "узким местом", оказывающим существенное влияние на эффективность *автоматизации человеческой деятельности*. В основе современной организации взаимодействия пользователя с компьютерной системой лежит парадигма грамотного пользователя, который знает, чего он хочет от используемого им инструмента и несёт полную ответственность за качество взаимодействия с этим инструментом. Эта парадигма лежит в основе деятельности лесоруба во взаимодействии с топором, всадника во взаимодействии с лошадью, автводителя, летчика во взаимодействии с соответствующим транспортным средством, оператора атомной электростанции, железнодорожного диспетчера и так далее.

На современном этапе развития *Искусственного интеллекта* для повышения эффективности взаимодействия необходим переход от парадигмы грамотного управления используемым инструментом к парадигме равноправного сотрудничества, партнёрскому взаимодействию *интеллектуальной компьютерной системы* со своим пользователем. *Интеллектуальная компьютерная система* должна повернуться "лицом" к пользователю.

Семантическая дружественность пользовательского интерфейса должна заключаться в адаптивности к особенностям и квалификации пользователя, исключении любых проблем для пользователя в процессе диалога с *интеллектуальной компьютерной системой*, в перманентной заботе о совершенствовании коммуникационных навыков пользователя.

При организации взаимодействия пользователя с Глобальной сетью компьютерных системам необходимо перейти от парадигмы "многооконного" интерфейса, в каждом "окне" которого свои "правила игры" к парадигме "одного окна". Пользователь не должен знать, какое "окно" ему надо "открыть" (в какую систему ему надо войти) для удовлетворения той или иной его потребности.

Пользователь не должен знать, какая конкретно система будет решать его задачу. Пользователь должен уметь с помощью универсальных средств сформулировать свою задачу, а соответствующая компьютерная система, входящая в Глобальную сеть и способная решить эту задачу, должна сама инициироваться, реагируя на факт появления указанной задачи. Таким образом пользовательский интерфейс должен быть интерфейсом не конкретной компьютерной системой, а в целом со всей Глобальной сетью компьютерных систем.

§ 1.2.5. Достоинства предлагаемых принципов, лежащих в основе интеллектуальных компьютерных систем нового поколения

Смысловое представление информации в памяти *интеллектуальных компьютерных систем* обеспечивает устранение дублирования информации, хранимой в памяти *интеллектуальной компьютерной системы*, то есть устранение многообразия форм представления одной и той же информации, запрещение появления в одной памяти *семантически эквивалентных информационных конструкций* и, в том числе, синонимичных знаков. Это существенно снижает сложность и повышает качество:

- разработки различных моделей обработки знаний (т.к. нет необходимости учитывать многообразие форм представления одного и того же знания);
- семантического анализа и понимания информации, поступающей (передаваемой) от различных внешних субъектов (от пользователей, от разработчиков, от других *интеллектуальных компьютерных систем*);
- конвергенции и интеграции различных видов знаний в рамках каждой *интеллектуальной компьютерной системы*;
- обеспечения семантической совместимости и взаимопонимания между различными *интеллектуальными компьютерными системами*, а также между *интеллектуальными компьютерными системами* и их пользователями.

Понятие *семантической сети* нами рассматривается не как красавая метафора сложноструктурированных *знаковых конструкций*, а как формальное уточнение понятия *смыслового представления информации*, как принцип представления информации, лежащей в основе нового поколения *компьютерных языков* и самих *компьютерных систем* — *графовых языков* и *графовых компьютеров*. *Семантическая сеть* — это нелинейная (графовая) *знаковая конструкция*, обладающая следующими свойствами:

- все элементы (то есть синтаксически элементарные фрагменты) этой *графовой структуры* (узлы и связки) являются *знаками* описываемых сущностей и, в частности, *знаками связей* между этими сущностями;
- все знаки, входящие в эту *графовую структуру*, не имеют *синонимов* в рамках этой структуры;
- "внутреннюю" структуру (строение) знаков, входящих в *семантическую сеть* не требуется учитывать при ее *семантическом анализе* (понимании);
- смысл *семантической сети* определяется *дениотационной семантикой* всех входящих в нее знаков и конфигурацией *связей инцидентности* этих знаков;
- из двух *инцидентных знаков*, входящих в *семантическую сеть*, по крайней мере один является знаком связи.

Рафинированная семантическая сеть — это *семантическая сеть*, имеющая простую *синтаксическую структуру*, в которой, в частности,

- используется конечный алфавит элементов *семантической сети*, то есть конечное число синтаксически выделяемых типов (синтаксических меток), приписываемых этим элемента;
- внешние идентификаторы (в частности, имена), приписываемые элементам *семантической сети* используются только для ввода/вывода информации.

Агентно-ориентированная модель обработки информации в сочетании с *децентрализованным ситуационным управлением процессом обработки информации*, а также со *смысловым представлением информации* в памяти *интеллектуальной компьютерной системы* существенно снижает сложность и повышает качество интеграции различных *моделей решения задач* при обработке общей базы знаний. Имеется в виду одновременное использование различных *моделей решения задач* при обработке одних и тех же знаний, в частности, при решении одной и той же задачи.

Высокий уровень *семантической гибкости информации*, хранимой в памяти *интеллектуальной компьютерной системы нового поколения*, обеспечивается тем, что каждое удаление или добавление синтаксически элементарного фрагмента хранимой информации, а также удаление или добавление каждой *связи идентичности* между такими элементами имеет четкую семантическую интерпретацию.

Высокий уровень *стратифицированности информации*, хранимой в памяти *интеллектуальной компьютерной системы нового поколения*, обеспечивается онтологически ориентированной структуризацией *базы знаний интеллектуальной компьютерной системы нового поколения*.

Высокий уровень *индивидуальной обучаемости* интеллектуальных компьютерных систем нового поколения (то есть их способности к быстрому расширению своих знаний и навыков) обеспечивается:

- *семантической гибкостью информации*, хранимой в их памяти;
- *стратифицированностью* этой информации;
- *рефлексивностью* интеллектуальных компьютерных систем нового поколения.

Высокий уровень *коллективной обучаемости* интеллектуальных компьютерных систем нового поколения обеспечивается высоким уровнем их *социализации* (то есть способности к эффективному участию в деятельности различных коллективов, состоящих из *интеллектуальных компьютерных систем нового поколения* и людей) и, прежде всего, высоким уровнем их *взаимопонимания*.

Высокий уровень *интероперабельности* интеллектуальных компьютерных систем нового поколения принципиально меняет характер взаимодействия *компьютерных систем* с людьми, автоматизацию деятельности которых они осуществляют, — от управления этими средствами автоматизации к *равноправным партнерским осмысленным взаимоотношениям*.

Каждая *интеллектуальная компьютерная система нового поколения* способна:

- самостоятельно или по приглашению войти в состав коллектива, состоящего из *интеллектуальных компьютерных систем нового поколения* и/или людей. Такие коллективы создаются на временной или постоянной основе для коллективного решения сложных задач;
- участвовать в распределении (в т.ч. в согласовании распределения) задач — как "разовых" задач, так и долгосрочных задач (обязанностей);
- мониторить состояние всего процесса коллективной деятельности и координировать свою деятельность с деятельностью других членов коллектива при возможных непредсказуемых изменениях условий (состояния) соответствующей среды.

Высокий уровень интеллекта *интеллектуальных компьютерных систем нового поколения* и, соответственно, высокий уровень их самостоятельности и целенаправленности позволяет им быть полноправными членами самых различных сообществ, в рамках которых *интеллектуальные компьютерные системы нового поколения* получают права самостоятельно инициировать (на основе детального анализа текущего положения дел и, в том числе, текущего состояния плана действий сообщества) широкий спектр действий (задач), выполняемых другими членами сообщества, и тем самым участвовать в согласовании и координации деятельности членов сообщества. Способность *интеллектуальной компьютерной системы нового поколения* согласовывать свою деятельность с другими подобными системами, а также корректировать деятельность всего *коллектива интеллектуальных компьютерных систем нового поколения*, адаптируясь к различного вида изменениям среды (условий), в которой эта деятельность осуществляется, позволяет существенно автоматизировать деятельность *системного интегратора* как на этапе создания *коллектива интеллектуальных компьютерных систем нового поколения*, так и на этапе его обновления (реинжиниринга).

Достоинства *интеллектуальных компьютерных систем нового поколения* обеспечиваются:

- достоинствами языка внутреннего *смыслового кодирования информации*, хранимой в памяти этих систем;
- достоинствами организации графодинамической ассоциативной *смысловой памяти интеллектуальных компьютерных систем нового поколения*;
- достоинствами *смылового представления баз знаний* интеллектуальных компьютерных систем нового поколения и средствами онтологической структуризации баз знаний этих систем;

§ 1.2.5. Достоинства предлагаемых принципов, лежащих в основе интеллектуальных компьютерных систем нового поколения

- достоинствами *агентно-ориентированных моделей решения задач*, используемых в *интеллектуальных компьютерных системах нового поколения* в сочетании с децентрализованным управлением процессом обработки информации.

Заключение к Главе 1.2.

Кратко перечислим основные положения Главы 1.2.:

- Основным практически значимым направлением развития современных *интеллектуальных компьютерных систем* является переход к *интероперабельным интеллектуальным компьютерным системам*, способным к эффективному взаимодействию между собой и с пользователями, что:
 - обеспечивает автоматизацию решения сложных комплексных задач, для которых требуется создание временных или постоянных коллективов
 - превращает *интеллектуальные компьютерные системы* в самостоятельные активные субъекты, способные инициировать различные комплексные задачи и, собственно, инициировать для этого работоспособные коллективы, состоящие из людей и *интероперабельных интеллектуальных компьютерных систем* требуемой квалификации
- Коллективы, состоящие из самостоятельных *интероперабельных интеллектуальных компьютерных систем* и людей, имеют хорошие перспективы стать *синергетическими* системами.
- *Интероперабельность интеллектуальных компьютерных систем* обеспечивается:
 - высоким уровнем взаимопонимания и, соответственно, семантической совместимости
 - высоким уровнем договороспособности, то есть способности предварительно согласовывать свои действия с действиями других субъектов
 - высоким уровнем способности оперативно координировать свои действия с действиями других субъектов в ходе их выполнения
- К числу принципов, лежащих в основе построения *интероперабельных интеллектуальных компьютерных систем*, относятся:
 - смысловое представление знаний в памяти *интеллектуальных компьютерных систем* в виде рафинированных семантических сетей
 - использование универсального языка внутреннего смыслового представления знаний
 - графодинамическая организация обработки знаний
 - агентно-ориентированные модели решения задач
 - структуризация и стратификация баз знаний в виде иерархической системы формальных онтологий
 - семантически дружественный пользовательский интерфейс
- Для разработки большого количества интероперабельных семантически совместимых *интеллектуальных компьютерных систем*, обеспечивающих переход на принципиально новый уровень автоматизации человеческой деятельности, необходимо создание технологии, обеспечивающей массовое производство таких *интеллектуальных компьютерных систем*, участие в котором доступно широкому контингенту разработчиков (в том числе разработчиков средней квалификации и начинающих разработчиков). Основными положениями такой технологии являются
 - стандартизация *интероперабельных интеллектуальных компьютерных систем*
 - широкое использование *компонентного проектирования* на основе мощной библиотеки семантически совместимых многократно используемых (типовых) компонентов *интероперабельных интеллектуальных компьютерных систем*
- Эффективная эксплуатация *интероперабельных интеллектуальных компьютерных систем* требует создания не только *технологии проектирования* таких систем, но также и семейства технологий поддержки всех остальных этапов их жизненного цикла. Особенно это касается технологий перманентной поддержки *семантической совместимости* всех взаимодействующих *интероперабельных интеллектуальных компьютерных систем* в ходе их эксплуатации.

Глава 1.3.

Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения

⇒ автор*:

- Голенков В.В.
- Шункевич Д.В.
- Ковалёв М.В.
- Садовский М.Е.

⇒ аннотация*:

[В главе рассмотрены принципы построения комплексной технологии разработки и поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения — Технологии OSTIS.]

⇒ подраздел*:

- § 1.3.1. Технология OSTIS (Open Semantic Technology for Intelligent Systems)
- § 1.3.2. Семантически совместимые ostis-системы

§ 1.3.1. Технология OSTIS (Open Semantic Technology for Intelligent Systems)

⇒ ключевой знак*:

- Технология OSTIS
- ostis-система
- Стандарт ostis-систем
- Метасистема OSTIS
- Стандарт OSTIS
- платформа ostis-систем
- Экосистема OSTIS

⇒ ключевое знание*:

- Обобщенный жизненный цикл ostis-систем
- Принципы, лежащие в основе Технологии OSTIS

жизненный цикл интеллектуальной компьютерной системы нового поколения

⇒ включает в себя*:

- проектирование интеллектуальной компьютерной системы нового поколения
 - ⇒ включает в себя*:
 - проектирование базы знаний интеллектуальной компьютерной системы нового поколения
 - проектирование решателя задач интеллектуальной компьютерной системы нового поколения
 - проектирование интерфейса интеллектуальной компьютерной системы нового поколения
 - реализацию интеллектуальной компьютерной системы нового поколения
 - начальное обучение интеллектуальной компьютерной системы нового поколения
 - мониторинг качества интеллектуальной компьютерной системы нового поколения
 - восстановление требуемого уровня интеллектуальной компьютерной системы нового поколения
 - реинжиниринг интеллектуальной компьютерной системы нового поколения
 - обеспечение безопасности интеллектуальной компьютерной системы нового поколения
 - эксплуатация интеллектуальной компьютерной системы нового поколения конечными пользователями

Построение Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения предполагает:

- Четкое описание текущей версии стандарта интеллектуальных компьютерных систем нового поколения, обеспечивающего семантическую совместимость разрабатываемых систем;
- Создание мощных библиотек семантически совместимых и многократно (повторно) используемых компонентов разрабатываемых интеллектуальных компьютерных систем;
- Уточнение требований, предъявляемых к создаваемой комплексной технологии и обусловленных особенностями интеллектуальных компьютерных систем нового поколения, разрабатываемых и эксплуатируемых с помощью указанной технологии.

§28. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения

Создание инфраструктуры, обеспечивающей интенсивное перманентное развитие *Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения* предполагает:

- Обеспечение низкого порога входления в *технологию проектирования интеллектуальных компьютерных систем* как для пользователей технологии (то есть разработчиков прикладных или специализированных интеллектуальных компьютерных систем), так и для разработчиков самой технологии;
- Обеспечение высоких темпов развития *технологии* за счет учета опыта разработки различных приложений путем активного привлечения авторов приложений к участию в развитии (совершенствовании) *технологии*.

В основе создания предлагаемой нами *технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения* лежат следующие положения:

- реализация предлагаемой *технологии* разработки и сопровождения *интеллектуальных компьютерных систем нового поколения* в виде *интеллектуальной компьютерной метасистемы*, которая полностью соответствует *стандартам* предлагаемых интеллектуальных компьютерных систем нового поколения, разрабатываемым по предлагаемой *технологии*. В состав такой *интеллектуальной компьютерной метасистемы*, реализующей предлагаемую *технологию* входит:
 - формальное онтологическое описание текущей версии *стандарта интеллектуальных компьютерных систем нового поколения*;
 - формальное онтологическое описание текущей версии *методов и средств проектирования, реализации, сопровождения, реинжиниринга и эксплуатации интеллектуальных компьютерных систем нового поколения*.

Благодаря этому технология проектирования и реинжиниринга интеллектуальных компьютерных систем нового поколения и технология проектирования и реинжиниринга самой указанной технологии (то есть интеллектуальной компьютерной метасистемы) суть одно и тоже;

- **унификация и стандартизация интеллектуальных компьютерных систем нового поколения**, а также *методов их проектирования, реализации, сопровождения, реинжиниринга и эксплуатации*;
- перманентная эволюция *стандарта интеллектуальных компьютерных систем нового поколения*, а также *методов их проектирования, реализации, сопровождения, реинжиниринга и эксплуатации*;
- **онтологическое проектирование интеллектуальных компьютерных систем нового поколения**, предлагающее:
 - четкое согласование и оперативную формализованную фиксацию (в виде *формальных онтологий*) утвержденного *текущего состояния* иерархической системы всех *понятий*, лежащих в основе перманентно эволюционирующего *стандарта интеллектуальных компьютерных систем нового поколения*, а также в основе каждой разрабатываемой *интеллектуальной компьютерной системы*;
 - достаточно полное и оперативное документирование текущего состояния каждого проекта;
 - использование *методики проектирования "сверху-вниз"*.
- **компонентное проектирование интеллектуальных компьютерных систем нового поколения**, то есть проектирование, ориентированное на сборку *интеллектуальных компьютерных систем* из готовых компонентов на основе постоянно расширяемых библиотек *многократно используемых компонентов*;
- **комплексный характер** предлагаемой *технологии*, осуществляющей:
 - поддержку *проектирования* не только *компонентов интеллектуальных компьютерных систем нового поколения* (различных *фрагментов баз знаний, баз знаний в целом, различных методов решения задач, различных внутренних информационных агентов, решателей задач в целом, формальных онтологических описаний различных внешних языков, интерфейсов в целом*), но также и *интеллектуальных компьютерных систем* в целом как самостоятельных *объектов проектирования* с учетом специфики тех классов, которым принадлежат проектируемые *интеллектуальные компьютерные системы*;
 - поддержку не только *комплексного проектирования интеллектуальных компьютерных систем нового поколения*, но также и поддержку их реализации (сборки, воспроизведения), сопровождения, реинжиниринга в ходе эксплуатации и непосредственно самой эксплуатации.

Для создания *технологии комплексного проектирования* и комплексной поддержки последующих этапов жизненного цикла *интеллектуальных компьютерных систем нового поколения* необходимо:

- Унифицировать формализацию различных моделей представления различного вида используемой информации, хранимой в памяти *интеллектуальных компьютерных систем* и различных моделей решения интеллектуальных задач для обеспечения *семантической совместимости* и простой автоматизируемой интегрируемости различных видов знаний и моделей решения задач в *интеллектуальных компьютерных системах*. Для этого необходимо разработать базовую *универсальную абстрактную модель* представления и обработки знаний, обеспечивающую реализацию всевозможных моделей решения задач.
- Унифицировать структуризацию баз знаний интеллектуальных компьютерных систем в виде иерархической системы онтологий разного уровня.

- Унифицировать систему используемых *понятий*, специфицируемых соответствующими *онтологиями* для обеспечения *семантической совместимости и интероперабельности* различных *интеллектуальных компьютерных систем*.
- Унифицировать архитектуру *интеллектуальных компьютерных систем*,
 - обеспечивающую *семантическую совместимость*:
 - между *интеллектуальными компьютерными системами* и их *пользователями*
 - между *индивидуальными интеллектуальными компьютерными системами*;
 - между *коллективными интеллектуальными компьютерными системами*,
 - а также обеспечивающую *интероперабельность* сообществ, состоящих из:
 - *индивидуальных интеллектуальных компьютерных систем*;
 - *коллективных интеллектуальных компьютерных систем*;
 - *пользователей интеллектуальных компьютерных систем*
- Разработать *базовую модель интерпретации* всевозможных формальных моделей решения задач в интеллектуальных компьютерных системах с ориентацией на максимально возможное упрощение такой интерпретации в *компьютерах нового поколения*, которые специально предназначены для реализации индивидуальных *интеллектуальных компьютерных систем*.
- Разработать *компьютеры нового поколения*, принципы функционирования которых максимально близки к базовой абстрактной модели, обеспечивающей интеграцию всевозможных моделей представления знаний и моделей решения задач. При этом базовая машина обработки информации, лежащая в основе указанных компьютеров, должна существенно отличаться от фон-Неймановской машины и должна быть близка к базовой модели решения задач в интеллектуальных компьютерных системах для того, чтобы существенно снизить сложность интерпретации всего многообразия моделей решения задач в интеллектуальных компьютерных системах.

Реализация всех перечисленных этапов развития *технологий Искусственного интеллекта* представляет собой переход на принципиально новый технологический уклад, обеспечивающий существенное повышение эффективности практического использования результатов работ в области *Искусственного интеллекта* и существенное повышение уровня автоматизации *человеческой деятельности*.

Предложенную нами *технологию комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения* мы назвали **Технологией OSTIS** (Open Semantic Technology for Intelligent Systems). Соответственно этому *интеллектуальные компьютерные системы нового поколения*, разрабатываемые по этой технологии называются *ostis-системами*. Сама *Технология OSTIS* реализуется нами в форме специальной *ostis-системы*, которая названа нами *Метасистемой OSTIS* и база знаний которой содержит:

- Формальную теорию *ostis-систем*;
- Стандарт *ostis-систем*
 - Стандарт баз знаний *ostis-систем*
 - Стандарт внутреннего универсального языка смыслового представления знаний в памяти *ostis-систем*
 - Стандарт внутреннего представления онтологий верхнего уровня в памяти *ostis-систем*
 - Стандарт представления исходных текстов баз знаний *ostis-систем*
 - Стандарт решателей задач *ostis-систем*
 - Стандарт базового языка программирования *ostis-систем*
 - Стандарт языков программирования высокого уровня для *ostis-систем*
 - Стандарт представления искусственных нейронных сетей в памяти *ostis-систем*
 - Стандарт внутренних информационных агентов в *ostis-систем*
 - Стандарт интерфейсов *ostis-систем*
 - Стандарт внешних языков *ostis-систем*, близких к внутреннему универсальному языку смыслового представления знаний
- Стандарт *ostis-систем* и Технологии OSTIS (**Стандарт OSTIS**) Голенков В.В. СтандОТОППиЭССГИКС-2021кн;
- Ядро Библиотеки многократно используемых компонентов *ostis-систем* (**Библиотеки OSTIS**);
- Методики и инструментальные средства поддержки жизненного цикла *ostis-систем* и их компонентов.

Технология OSTIS

:= [Open Semantic Technology for Intelligent Systems]

:= [Открытая семантическая технология комплексной поддержки жизненного цикла *семантически совместимых* интеллектуальных компьютерных систем нового поколения]

:=

§ 3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения

[Модели, методики, методы и средства комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения]

:= [Теория интеллектуальных компьютерных систем нового поколения и практика компьютерной поддержки их жизненного цикла]

:= [Технологический комплекс (моделей, методик, автоматизированных методов и средств), соответствующий интеллектуальным компьютерным системам нового поколения (интероперабельным и семантически совместимым компьютерным системам)]

:= [Предлагаемая нами комплексная технология поддержки всех этапов жизненного цикла всех компонентов для всех классов (видов) интеллектуальных компьютерных систем нового поколения при перманентной поддержке их семантической совместимости]

⇒ *принципы, лежащие в основе**:

- Комплексный характер технологии, заключающийся в том, что осуществляется поддержка:
 - всех этапов жизненного цикла создаваемых продуктов
 - для всех компонентов интеллектуальных компьютерных систем нового поколения
 - для всех классов интеллектуальных компьютерных систем нового поколения
- Обеспечивается перманентная поддержка семантической совместимости между всеми создаваемыми интеллектуальными компьютерными системами нового поколения
- Ориентация на комплексную автоматизацию всего многообразия человеческой деятельности
- Реализация технологии и, соответственно, комплексная автоматизация поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения (со всеми их компонентами и классами) осуществляется в виде семейства интеллектуальных компьютерных систем нового поколения, построенных по той же технологии

Стандарт OSTIS

:= [Стандарт Технологии OSTIS]

⇒ *оригинал**:

[Основная часть базы знаний Метасистемы OSTIS]

§ 1.3.2. Семантически совместимые ostis-системы

база знаний ostis-системы

:= [sc-конструкция, которая в текущий момент времени хранится в памяти ostis-системы]

▷ база знаний индивидуальной ostis-системы

▷ база знаний корпоративной ostis-системы

▷ распределённая база знаний коллектива ostis-систем

Э База знаний Экосистемы OSTIS

⇒ *примечание**:

[Каждый sc-элемент по отношению к базе знаний каждой ostis-системы считается временной сущностью, поскольку каждый sc-элемент в какой-то момент вводится в состав базы знаний и в какой-то момент может быть из неё удалён, но при этом следует отличать временный характер самого sc-элемента от временного или постоянного характера обозначаемой им сущности]

Полная документация ostis-системы, включающая в себя и руководство пользователя, и руководство разработчика, является неотъемлемой частью каждой ostis-системы и, соответственно, обеспечивает всестороннюю информационную поддержку деятельности пользователя, а также обеспечивает персонифицированную адаптацию к каждому пользователю и повышению уровня его квалификации по использованию этой интеллектуальной компьютерной системы.

ostis-система

:= [интеллектуальная компьютерная система нового поколения, построенная по Технологии OSTIS]

:= [предлагаемое нами уточнение понятия интеллектуальной компьютерной системы нового поколения]

⇒ *разбиение**:

{• ostis-субъект

:= [самостоятельная ostis-система]

⇒ *разбиение**:

{• индивидуальная ostis-система

• коллективная ostis-система

}

- *встроенная ostis-система*
:= [*ostis*-система, являющаяся частью некоторой *индивидуальной ostis-системы*]
- }

индивидуальная ostis-система:= [минимальная самостоятельная *ostis*-система]⇒ *разбиение**:

- {• *персональный ostis-ассистент*
:= [*ostis*-система, осуществляющая комплексное адаптивное обслуживание конкретного пользователя по всем вопросам, касающимся его взаимодействия с любыми другими *ostis*-системами, а также представляющая интересы этого пользователя во всей глобальной сети *ostis*-систем]
 - *корпоративная ostis-система*
:= [*ostis*-система, осуществляющая координацию совместной деятельности *ostis*-систем в рамках соответствующего коллектива *ostis*-систем, осуществляющая мониторинг и реинжиниринг соответствующего множества *ostis*-систем и представляющая интересы этого коллектива в рамках других коллективов *ostis*-систем]
 - *индивидуальная ostis-система, не являющаяся ни персональным ostis-ассистентом, ни корпоративной ostis-системой*
- }

коллективная ostis-система:= [многоагентная система, представляющая собой коллектив индивидуальных и коллективных *ostis*-систем, деятельность которого координируется соответствующей корпоративной *ostis*-системой]⇒ *примечание**:

[В состав коллектива *ostis*-систем могут входить индивидуальные *ostis*-системы могут входить индивидуальные *ostis*-системы любого вида — в том числе, корпоративные *ostis*-системы, представляющие интересы других коллективов *ostis*-систем]

Для Технологии *OSTIS* в рамках множества всевозможных *ostis*-систем особое место занимают следующие *ostis*-системы:

- *Метасистема OSTIS*
:= [Индивидуальная *ostis*-система, являющаяся реализацией ядра Технологии *OSTIS*]
- *Экосистема OSTIS*
:= [Коллективная *ostis*-система, представляющая собой глобальный коллектив всех *ostis*-систем, взаимодействующих между собой и осуществляющих комплексную автоматизацию человеческой деятельности]

Метасистема OSTIS:= [Интеллектуальная компьютерная система нового поколения, построенная по Технологии *OSTIS* и обеспечивающая автоматизацию компьютерную поддержку жизненного цикла интеллектуальной компьютерной системы нового поколения, создаваемых также по Технологии *OSTIS*]∈ *ostis*-система⇐ *предлагаемая форма реализации**:

Технология *OSTIS*

Экосистема OSTIS:= [Глобальная сеть взаимодействующих *ostis*-систем, ориентированная на перманентно расширяемую комплексную автоматизацию самых различных видов и областей человеческой деятельности]⇒ *примечание**:

[Это основной продукт Технологии *OSTIS*, который можно рассматривать как предлагаемый нами подход к реализации Общества 5.0, Науки 5.0, Индустрии 5.0]

Заключение к Главе 1.3.

Для разработки большого количества интероперабельных семантически совместимых интеллектуальных компьютерных систем, обеспечивающих переход на принципиально новый уровень автоматизации человеческой деятельности необходимо создание технологии, обеспечивающей массовое производство таких интеллектуальных компьютерных систем, участие в котором доступно широкому контингенту разработчиков (в том числе

663. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения.

разработчиков средней квалификации и начинающих разработчиков). основными положениями такой технологии являются:

- стандартизация интероперабельных *интеллектуальных компьютерных систем*
- широкое использование *компонентного проектирования* на основе мощной библиотеки семантически совместимых многократно используемых (типовых) компонентов *интероперабельных интеллектуальных компьютерных систем*

Эффективная эксплуатация *интероперабельных интеллектуальных компьютерных систем* требует создания не только *технологии проектирования* таких систем, но также и семейства технологий поддержки всех остальных этапов их жизненного цикла. Особенно это касается технологии перманентной поддержки *семантической совместимости* всех взаимодействующих *интероперабельных интеллектуальных компьютерных систем* в ходе их эксплуатации.

Часть 2.

Смыслоное представление и онтологическая систематизация знаний в интеллектуальных компьютерных системах нового поколения

Описание к главе

Глава 2.1.

Информационные конструкции и языки

Никифоров С.А.

Гойло А.А.

⇒ *аннотация**:

[Аннотация к главе.

Список доработок:

- написать аннотацию;
- написать анализ, из статьи не подойдет;
- разбить SCn с отношениями, заданными на множестве языков^Λ на части, вынести из него текст;
- добавить все, что было в стандарте, он было искоючено из статьи, в частности - про разделители;
- пройтись по подсказкам от плагина, поправит форматирвоание,
- формализовать пр. о..
- разбить на подразделы

Список вопросов:

- нужно ли заключение?

]

§ 2.1.1. Формализация понятия информационной конструкции

знак

⇒ *разбиение**:

- {• знак, являющийся элементом дискретной информационной конструкции
• знак, являющийся неатомарным фрагментом дискретной информационной конструкции
}

Знак - фрагмент информационной конструкции, который условно представляет (изображает) некоторую описываемую сущность, которую называют денотатом знака. При этом отсутствие знака, обозначающего некоторую сущность, не означает отсутствие самой этой сущности. Это означает только то, что мы даже не догадываемся о её существовании и, следовательно, не приступили к её исследованию.

Поскольку все знаки являются дискретными информационными конструкциями, множество знаков является областью задания всех отношений, заданных на множестве дискретных информационных конструкций. Тем не менее есть как минимум одно отношение, специфичное для множества знаков.

отношение, заданное на множестве знаков^Λ

Э синонимия знаков*

Знаки являются синонимичными в том и только в том случае, если они обозначают одну и ту же сущность. При этом синонимичные знаки могут быть синтаксически эквивалентными, а могут и не быть таковыми.

знаковая конструкция

С дискретная информационная конструкция

Знаковая конструкция - дискретная информационная конструкция, которая в общем случае представляет собой конфигурацию знаков и специальных фрагментов информационной конструкции, обеспечивающих структуризацию конфигурации знаков — различного вида разделителей и ограничителей. Для некоторых знаковых конструкций и даже для некоторых языков необходимость в разделителях и ограничителях может отсутствовать.

отношение, заданное на множестве знаковых конструкций[^]

- Э знак*
- Э разделитель знаковой конструкции*
- Э разделители знаковой конструкции*
- Э ограничитель знаковой конструкции*
- Э ограничители знаковой конструкции*
- Э семантическая смежность знаковых конструкций*
- Э конкатенация знаковых конструкций*
- := [декомпозиция заданной знаковой конструкции на последовательность знаковых конструкций*]

Знак* - бинарное ориентированное отношение, связывающее знаковую конструкцию со множеством всех знаков, входящих в её состав.

Семантическая смежность знаковых конструкций* - бинарное отношение, связывающее семантически смежные знаковые конструкции. При этом знаковые конструкции T_i и T_j являются смежными в том и только в том случае, если существуют синонимичные знаки T_i и T_j , один из которых входит в состав конструкции T_i , а второй — в состав конструкции T_j

класс знаковых конструкций[^]

- Э семантически элементарная знаковая конструкция
- Э семантически связная знаковая конструкция

Семантически элементарная знаковая конструкция - знаковая конструкция, описывающая некоторую (одну) связь между некоторыми сущностями.

Семантически связная знаковая конструкция - знаковая конструкция, которую можно представить в виде конкатенации семантически элементарных знаковых конструкций, каждая из которых семантически смежна предшествующей и последующей семантически элементарной знаковой конструкции

параметр, заданный на множестве знаковых конструкций[^]

- Э семантическая связность знаковых конструкций[^]
- Э семантически связная знаковая конструкция
- Э семантически несвязная знаковая конструкция
- Э наличие разделителей и ограничителей[^]
- Э знаковая конструкция, содержащая разделители и-или ограничители
- Э знаковая конструкция без разделителей и ограничителей

Информационная конструкция - конструкция (структура), содержащая некоторые сведения о некоторых сущностях. Форма представления ("изображения", "материализации"), форма структуризации (синтаксическая структура), а также смысл* (денотационная семантика) информационных конструкций могут быть самыми различными.

Дискретная информационная конструкция — это информационная конструкция, смысл которой задается:

- множеством элементов (синтаксически атомарных фрагментов) этой информационной конструкции,
- алфавитом этих элементов — семейством классов синтаксически эквивалентных элементов информационной конструкции,
- принадлежностью каждого элемента информационной конструкции соответствующему классу синтаксически эквивалентных элементов информационной конструкции,
- конфигурацией связей инцидентности между элементами информационной конструкции.

Следствием этого является то, что форма представления элементов дискретной информационной конструкции для анализа её смысла не требует уточнения. Главным является:

- наличие простой процедуры выделения (сегментации) элементов дискретной информационной конструкции,
- наличие простой процедуры установления синтаксической эквивалентности разных элементов дискретной информационной конструкции,
- наличие простой процедуры установления принадлежности каждого элемента дискретной информационной конструкции соответствующему классу синтаксически эквивалентных элементов (т.е. соответствующему элементу алфавита).

Элементом дискретной информационной конструкции является синтаксически атомарный фрагмент (символ), входящий в состав дискретной информационной конструкции. Поскольку дискретные информационные конструкции могут иметь общие элементы (атомарные фрагменты) и даже некоторые из них могут быть частями других информационных конструкций, элемент дискретной информационной конструкции может входить в состав сразу нескольких информационных конструкций.

Далее рассмотрим отношения, заданные на множестве элементов дискретных информационных конструкций.

отношение, заданное на множестве элементов дискретных информационных конструкций[^]

- Э *элемент дискретной информационной конструкции**
- Э *синтаксическая эквивалентность элементов дискретных информационных конструкций**
- Э *инцидентность элементов дискретных информационных конструкций**

*Элемент дискретной информационной конструкции** - бинарное ориентированное отношение, каждая пара которого связывает (1) знак некоторой дискретной информационной конструкции и (2) знак одного из элементов этой дискретной информационной конструкции*.

*Синтаксическая эквивалентность элементов дискретных информационных конструкций** - отношение, связывающее синтаксически эквивалентные элементы (атомарные фрагменты) одной и той же или разных дискретных информационных конструкций, т.е. элементы, принадлежащими одному и тому же классу синтаксически эквивалентных элементов дискретных информационных конструкций*.

*Инцидентность элементов дискретных информационных конструкций** для линейных информационных конструкций – это последовательность элементов (символов), входящих в состав этих конструкций. Для дискретных информационных конструкций, конфигурация которых имеет нелинейный характер, отношение инцидентности их элементов может быть разбито на несколько частных отношений инцидентности, каждое из которых является подмножеством объединенного отношения инцидентности. Например, для двухмерных дискретных информационных конструкций это (1) инцидентность элементов информационных конструкций "по горизонтали" и (2) инцидентность элементов информационных конструкций "по вертикали".

Далее рассмотрим отношения, заданные на множестве дискретных информационных конструкций.

отношение, заданное на множестве элементов дискретных информационных конструкций[^]

- Э *неэлементарный фрагмент дискретной информационной конструкции**
- Э *алфавит дискретной информационной конструкции**
- Э *первичная синтаксическая структура дискретной информационной конструкции**
- Э *синтаксическая эквивалентность дискретных информационных конструкций**
- Э *копия дискретной информационной конструкции**
- Э *семантическая эквивалентность дискретных информационных конструкций**
- Э *семантическое расширение дискретной информационной конструкции**
- Э *синтаксис информационной конструкции**
- Э *смысл**
- Э *операционная семантика информационной конструкции**

*Неэлементарный фрагмент дискретной информационной конструкции** - бинарное ориентированное отношение, связывающее заданную дискретную информационную конструкцию с дискретной информационной конструкцией, которая является подструктурой для нее, в состав которой входит (1) подмножество элементов заданной информационной конструкции и, соответственно, (2) подмножество пар инцидентности элементов заданной информационной конструкции.

*Алфавит дискретной информационной конструкции** - бинарное отношение, связывающее дискретную информационную конструкцию с семейством попарно непересекающихся классов синтаксически эквивалентных элементов заданной дискретной информационной конструкции*

*Первичная синтаксическая структура дискретной информационной конструкции** - бинарное ориентированное отношение, связывающее дискретную информационную конструкцию с графовой структурой, которая полностью описывает ее конфигурацию и которая включает в себя: (1) знаки всех тех классов синтаксически эквивалентных элементов, которым принадлежат элементы описываемой дискретной информационной конструкции, (2) знаки всех элементов (атомарных фрагментов) описываемой информационной конструкции, (3) пары, описывающие инцидентность элементов описываемой информационной конструкции, (4) пары, описывающие принадлежность элементов описываемой информационной конструкции соответствующим классам синтаксически эквивалентных элементов этой информационной конструкции.

*Синтаксическая эквивалентность дискретных информационных конструкций**: дискретные информационные конструкции *T_i* и *T_j* являются синтаксически эквивалентными в том и только в том случае, если между конструкцией *T_i* и конструкцией *T_j* существует изоморфизм, в рамках которого каждому элементу конструкции *T_i* соответствует синтаксически эквивалентный элемент конструкции *T_j*, т.е. элемент, принадлежащий тому же классу синтаксически эквивалентных элементов дискретных информационных конструкций. И наоборот.

*Копия дискретной информационной конструкции** - бинарное ориентированное отношение, которое связывает дискретную информационную конструкцию с дискретной информационной конструкцией, которая не только синтаксически эквивалентна ей, но и содержит информацию о форме представления элементов данной копируемой информационной конструкции*

копия дискретной информационной конструкции*

\subset синтаксическая эквивалентность дискретных информационных конструкций*

Семантическая эквивалентность дискретных информационных конструкций*: информационная конструкция T_i и информационная конструкция T_j являются семантически эквивалентными в том и только в том случае, если каждая сущность (в том числе, и каждая связь между сущностями), описываемая в информационной конструкции T_i описывается также и в информационной конструкции T_j . И наоборот.

Семантическое расширение дискретной информационной конструкции*: информационная конструкция T_j является семантическим расширением информационной конструкции T_i в том и только в том случае, если каждая сущность, описываемая в T_i , описывается также и в T_j , но обратное неверно.

Синтаксис информационной конструкции* - описание того, из каких частей состоит заданная информационная конструкция и как эти части (фрагменты) связаны между собой.

Смысл*(денонационная семантика информационной конструкции*) - каждая пара которого связывает некоторую информационную конструкцию с ее явным (формальным) представлением того, какие сущности описывает данная информационная конструкция и как эти сущности связаны между собой.

Операционная семантика информационной конструкции* - бинарное ориентированное отношение, каждая пара которого связывает знак некоторой информационной конструкции со множеством правил ее трансформации - описанием того, на основании каких правил можно осуществлять действия по преобразования (обработке, трансформации) заданной информационной конструкции, оставляя ее в рамках класса синтаксически и семантически правильных информационных конструкций.

операционная семантика информационной конструкции*

\Rightarrow второй домен*:

операционная семантика информационной конструкции

Далее рассмотрим заданные на множестве дискретных информационных конструкций соответствия.

соответствие, заданное на множестве дискретных информационных конструкций

\exists соответствие между синтаксической структурой информационной конструкции и смыслом этой

конструкции*

\subset соответствие*

Соответствие, заданное на множестве дискретных информационных конструкций - множество ориентированных пар, первым компонентом которых является ориентированная пара, состоящая из (1) знака синтаксической структуры некоторой информационной конструкции и (2) знака смысловой структуры этой конструкции, а вторым компонентом которых является множество ориентированных пар, связывающих фрагменты синтаксической структуры заданной информационной конструкции (которые описывают либо структуру фрагментов заданной конструкции, либо связи между фрагментами этой конструкции) с теми фрагментами смысловой структуры заданной информационной конструкции, которые семантически эквивалентны либо синтаксически представленным фрагментам заданной информационной конструкции, либо синтаксически представленным связям между такими фрагментами.

У дискретных информационных конструкций можно выделить следующие параметры.

параметр, заданный на множестве дискретных информационных конструкций[^]

\exists размерность дискретных информационных конструкций[^]

$:=$ [типология дискретных информационных конструкций, определяемая их размерностью]

\exists линейная информационная конструкция

\exists двухмерная информационная конструкция

\exists трехмерная информационная конструкция

\exists четырехмерная информационная конструкция

\exists графовая информационная конструкция

Линейная информационная конструкция - дискретная информационная конструкция, каждый элемент которой может иметь не более двух инцидентных ему элементов (один слева, другой справа).

Двухмерная информационная конструкция - дискретная информационная конструкция, каждый элемент которой может иметь не более четырех инцидентных ему элементов (слева-справа, сверху-снизу).

Трехмерная информационная конструкция - дискретная информационная конструкция, каждый элемент которой может иметь не более шести инцидентных ему элементов (слева-справа, сверху-снизу, сзади-спереди).

Четырехмерная информационная конструкция - дискретная информационная конструкция, каждый элемент которой может иметь не более восьми инцидентных ему элементов (например, слева-справа, сверху-снизу, сзади-спреди, раньше-позже).

Графовая информационная конструкция - дискретная информационная конструкция, множество элементов которой разбивается на два подмножества — связки и узлы. При этом узлы могут иметь неограниченное количество инцидентных им связок. В некоторых графовых информационных конструкциях связки могут иметь неограниченное количество инцидентных им других связок.

параметр, заданный на множестве дискретных информационных конструкций[^]

- Э типология дискретных информационных конструкций, определяемая их носителем[^]
 - Э некомпьютерная форма представления дискретных информационных конструкций
 - ▷ аудио-сообщение
 - ▷ информационная конструкция, представленная на языке жестов
 - ▷ информационная конструкция, представленная в письменной форме
 - Э файл

Представление информационных конструкций в виде *файлов* ориентировано на представление дискретных (!) информационных конструкций. Поэтому "файловое" представление недискретных информационных конструкций (например, различного рода сигналов) предполагает "дискретизацию" таких конструкций, т.е. преобразование их в дискретные. Так преобразуются аудио-сигналы (в частности, речевые сообщения), изображения, видео-сигналы и др.

параметр, заданный на множестве дискретных информационных конструкций[^]

- Э уровень унификации представления синтаксически эквивалентных элементов дискретных информационных конструкций[^]
 - Э дискретная информационная конструкция с низким уровнем унификации представления элементов
 - ▷ аудио-сообщение
 - ▷ информационная конструкция, представленная на языке жестов
 - ▷ рукопись или её копия
 - Э дискретная информационная конструкция с высоким уровнем унификации представления элементов
 - ▷ печатный текст
 - ▷ файл

Уровень унификации представления синтаксически эквивалентных элементов дискретных информационных конструкций[^] - уровень "членораздельности" дискретных информационных конструкций. Чем выше уровень унификации представления элементов дискретных информационных конструкций, тем проще реализуется (1) процедура выделения (сегментации) элементов дискретной информационной конструкции, (2) процедура установления синтаксической эквивалентности этих элементов и (3) процедура их распознавания, т.е. процедура установления их принадлежности соответствующим классам синтаксически эквивалентных элементов.

Чем выше уровень унификации представления элементов дискретных информационных конструкций, тем проще реализуется:

- процедура выделения (сегментации) элементов дискретной информационной конструкции,
- процедура установления синтаксической эквивалентности этих элементов,
- процедура их распознавания, т.е. процедура установления их принадлежности соответствующим классам синтаксически эквивалентных элементов.

Уточнив понятия знака, знаковой конструкции, информационной конструкции, дискретной информационной конструкции и рассмотрев соответствующие отношения, можно перейти к формализации понятия "язык".

Язык - класс знаковых конструкций, для которого существуют (1) общие правила их построения и (2) общие правила их соотнесения с теми сущностями и конфигурациями сущностей, которые описываются (отражаются) указанными знаковыми конструкциями.

язык

⇒ разбиение**:

- {• язык, у которого все знаки, входящие в состав его знаковых конструкций, являются элементарными фрагментами этих конструкций
 - язык, у которого знаки, входящие в состав его знаковых конструкций, в общем случае не являются элементарными фрагментами этих конструкций
- ⇒ разбиение**:
- {• язык, знаковые конструкции которого содержат разделители и ограничители
 - язык, знаковые конструкции которого не содержат разделителей и ограничителей

}

}

Для языков, у которого все знаки, входящие в состав его знаковых конструкций, являются элементарными фрагментами этих конструкций существенно упрощаются методы обработки их текстов.

язык, знаковые конструкции которого не содержат разделителей и ограничителей - язык, знаковые конструкции такого языка состоят только из знаков.

Отношение, заданное на множестве языков[^] - отношение, область определения которого включает в себя множество всевозможных языков.

*Текст заданного языка** - бинарное отношение, связывающее язык и синтаксически правильную (правильно построенную) знаковую конструкцию данного языка. *Синтаксически корректная знаковая конструкция для заданного языка** - бинарное отношение, связывающее язык и знаковую конструкцию, не содержащую синтаксических ошибок для данного языка.

отношение, заданное на множестве языков[^]

- := [отношение, область определения которого включает в себя множество всевозможных языков]
- Э *текст заданного языка**
 - = (*синтаксически корректная знаковая конструкция для заданного языка** \cap *синтаксически целостная знаковая конструкция для заданного языка**)
 - Э *синтаксически корректная знаковая конструкция для заданного языка**
 - Э *синтаксически целостная знаковая конструкция для заданного языка**
 - Э *синтаксически неправильная знаковая конструкция для заданного языка**
 - = (*синтаксически некорректная знаковая конструкция для заданного языка** \cup *синтаксически нецелостная знаковая конструкция для заданного языка**)
 - Д *синтаксически некорректная знаковая конструкция для заданного языка**
 - Д *синтаксически нецелостная знаковая конструкция для заданного языка**
 - Э *знание, представленное на заданном языке**
 - := [семантически правильный текст заданного языка*]
 - = (*семантически корректный текст заданного языка** \cap *семантически целостный текст заданного языка**)
 - = [*истинный текст заданного языка**]
 - = [*истинное высказывание, представленное на заданном языке**]
 - Э *семантически корректный текст заданного языка**
 - := [*текст заданного языка, не содержащий семантических ошибок, противоречащих признанным закономерностям и фактам**]
 - Э *семантически целостный текст заданного языка**
 - := [*текст заданного языка, содержащий достаточную информацию для установления его истинности**]
 - Э *семантически неправильный текст для заданного языка**
 - = (*семантически некорректный текст для заданного языка** \cup *семантически нецелостный текст для заданного языка**)
 - Д *семантически некорректный текст для заданного языка**
 - Д *семантически нецелостный текст для заданного языка**
 - Э *алфавит**
 - := [*алфавит заданной информационной конструкции или заданного языка**]
 - := [*семейство классов, синтаксически эквивалентных элементов (элементарных фрагментов) заданной информационной конструкции или информационных конструкций заданного языка**]
 - Э *семейство классов синтаксически эквивалентных разделителей**
 - := [*семейство классов синтаксически эквивалентных разделителей, входящих в состав заданной информационной конструкции или в состав информационных конструкций заданного языка**]
 - Э *семейство классов синтаксически эквивалентных ограничителей**
 - := [*семейство классов синтаксически эквивалентных ограничителей, входящих в состав заданной информационной конструкции или в состав информационных конструкций заданного языка**]
 - Э *синтаксис языка**
 - := [*быть теорией правильно построенных информационных конструкций, принадлежащих заданному языку**]
 - := [*определение понятия правильно построенной информационной конструкции заданного языка**]
 - := [*синтаксические правила заданного языка**]
 - := [*быть синтаксическими правилами заданного языка **]
 - := [*Бинарное ориентированное отношение, каждая пара которого связывает знак некоторого языка с описанием синтаксически выделяемых классов фрагментов конструкций заданного языка, с описанием*

отношений, заданных на этих классах и с конъюнкцией кванторных высказываний, являющихся синтаксическими правилами заданного языка, то есть правилами, которым должны удовлетворять все синтаксические правильные (правильно построенные) конструкции (тексты) указанного языка*]

⇒ *примечание**:

[При представлении синтаксиса (синтаксических правил) заданного языка используются все те понятия, которые вводятся для представления синтаксических структур информационных конструкций, принадлежащих указанному языку. Это и синтаксически выделяемые классы фрагментов указанных информационных конструкций, и отношения, заданные на множестве таких фрагментов.]

⇒ *второй домен**:

синтаксис языка

Э *описание синтаксических понятий языка**

:= [описание синтаксически выделяемых классов фрагментов конструкций заданного языка*]

⇒ *второй домен**:

описание синтаксических понятий языка

↔ *обобщенное включение**:

синтаксис языка

Э *синтаксические правила языка**

:= [синтаксические правила заданного языка*]

⇒ *второй домен**:

синтаксические правила языка

Э *денотационная семантика языка**

:= [быть теорией морфизмов, связывающих правильно построенные информационные конструкции заданного языка с описываемыми конфигурациями описываемых сущностей*]

Э *денотационная семантика языка**

:= [семантические правила заданного языка*]

:= [быть семантическими правилами заданного языка*]

:= [Бинарное ориентированное отношение, каждая пара которого связывает знак некоторого языка с описанием основных семантических понятий заданного языка и конъюнкцией кванторных высказываний, являющихся семантическими правилами заданного языка, то есть правилами, которым должны удовлетворять семантически правильные смысловые информационные конструкции, соответствующие (семантические эквивалентные) синтаксически правильным конструкциям (текстам) заданного языка*]

⇒ *примечание**:

[При формулировке семантических правил заданного языка используются понятия, введенные в рамках базовых онтологий (онтологии самого высокого уровня, в которых рассматриваются самые общие классы описываемых сущностей, самые общие отношения и параметры).]

⇒ *второй домен**:

денотационная семантика языка

Э *описание семантических понятий языка**

⇒ *второй домен**:

описание семантических понятий языка

Э *семантические правила языка**

⇒ *второй домен**:

семантические правила языка

Э *семантическая эквивалентность языков**

:= [быть семантически эквивалентными языками*]

⇒ *определение**:

[Язык *Lj* и язык *Lj* будем считать *семантически эквивалентными языками** в том и только в том случае, если для каждого текста, принадлежащего языку *Lj*, существует *семантически эквивалентный текст**, принадлежащий языку *Lj*, и наоборот.]

Э *семантическое расширение языка**

↔ *обратное отношение**:

*семантическое сужение языка**

⇒ *определение**:

[Язык *Lj* будем считать *семантическим расширением** языка *Lj* в том и только в том случае, есть ли для каждого текста, принадлежащего языку *Lj*, существует *семантически эквивалентный текст**, принадлежащий языку *Lj*, но обратное неверно.]

Э *синтаксическое расширение языка**

:= [быть семантически эквивалентным надмножеством заданного языка*]

⇒ *определение**:

[Язык *Lj* будем считать *синтаксическим расширением** языка *Lj* в том и только в том случае, если:

- $L_j \supset Li$ (то есть все тексты языка Li являются также и текстами языка Lj , но обратное неверно);
- Язык Lj и язык Li являются семантически эквивалентными языками*.
-]
- \exists синтаксическое ядро языка*
 $\quad :=$ [быть синтаксическим ядром заданного языка*]
 $\quad :=$ [быть семантически эквивалентным подмножеством заданного языка, имеющим минимальную синтаксическую сложность*]
- \exists направление синтаксического расширения ядра заданного языка*
 $\quad :=$ [быть правилом трансформации информационных конструкций, принадлежащих заданному языку, которое описывает одно из направлений перехода от множества конструкций ядра этого языка ко множеству всех принадлежащих ему информационных конструкций*]
- \exists операционная семантика языка*
 $\quad :=$ [Бинарное ориентированное отношение, каждая пара которого связывает знак некоторого языка со множеством правил трансформации текстов этого языка*]
 \Rightarrow второй домен*: операционная семантика языка
- \exists внутренний язык*
 $\quad :=$ [быть внутренним языком для заданной системы, основанной на обработке информации, или заданного множества таких систем*]
 $\quad :=$ [быть языком внутреннего представления информации в памяти заданной системы, основанной на обработке информации или заданного класса таких систем*]
- \exists внешний язык*
 $\quad :=$ [быть внешним языком для заданной системы, основанной на обработке информации, или заданного множества таких систем*]
 $\quad :=$ [быть языком, используемым для обмена информацией заданной системы, основанной на обработке информации, или заданного множества таких систем с другими системами, основанными на обработке информации, (в том числе, и с себе подобными)*]
- \exists используемый язык*
 $\quad =$ (*внутренний язык** \cup *внешний язык**)
 $\quad :=$ [язык, используемый заданной системой, основанной на обработке информации или заданного множества таких систем*]
 $\quad :=$ [язык, носителем которого является (которым владеет) данная система, основанная на обработке информации]
- \exists используемые языки*

Параметр, заданный на множестве языков[^] - семейство классов эквивалентности языков, трактуемой в контексте того или иного свойства (характеристики), присущего языкам.

параметр, заданный на множестве языков[^]

- \exists семантическая мощность языка[^]
 $\quad :=$ [класс языков, семантически эквивалентных друг другу]
- \exists универсальный язык
 $\quad :=$ [класс всевозможных универсальных языков]
 \Rightarrow примечание*: [Очевидно, что все универсальные языки (если они действительно таковыми являются, а не только претендуют на это) семантически эквивалентны друг другу, т.е. имеют одинаковую семантическую мощность.]
- \exists уровень синтаксической сложности представления знаков в текстах языка[^]
 - \exists язык, в текстах которого все знаки представлены синтаксически элементарными фрагментами
 - \exists язык, в текстах которого знаки в общем случае представлены синтаксически неэлементарными фрагментами
- \exists использование разделителей и ограничителей в текстах языка[^]
 - \exists язык, в текстах которого не используются разделители и ограничители
 - \exists язык, в текстах которого используются разделители и ограничители
- \exists уровень сложности процедуры установления синонимии знаков в текстах языка[^]
 - \exists язык, в рамках каждого текста которого синонимичные знаки отсутствуют
 \Rightarrow пояснение*: [В текстах такого языка знак каждой описываемой сущности входит однократно.]
 - \exists язык, в рамках которого синонимичные знаки представлены синтаксически эквивалентными фрагментами текстов
 - \exists флексивный язык

- \coloneqq [язык, в рамках которого синонимичные знаки могут быть представлены синтаксически неэквивалентными фрагментами, но фрагментами, являющимися модификациями некоторого "ядра" этих фрагментов (при склонении и спряжении этих знаков).]
- \ni язык, в рамках которого синонимичные знаки в общем случае могут быть представлены синтаксически неэквивалентными текстовыми фрагментами, структура которых носит непредсказуемый характер
- \ni наличие омонимии в текстах языка[^]
- \ni язык, в текстах которого присутствует омонимия знаков
 - \coloneqq [язык, в текстах которого присутствуют синтаксически эквивалентные, не синонимичные знаки]
 - \ni язык, в текстах которого омонимия знаков отсутствует

семантически выделяемый класс дискретных информационных конструкций

- \ni синтаксическая структура информационной конструкции
 - \Leftarrow второй домен*:
 - синтаксис информационной конструкции*
 - \sqsubset первичная синтаксическая структура информационной конструкции
 - \sqsubset вторичная синтаксическая структура информационной конструкции
- \ni синтаксис языка
- \ni описание синтаксических понятий языка
- \ni синтаксические правила языка
- \ni денотационная семантика языка
- \ni описание семантических понятий языка
- \ni семантические правила языка
- \ni операционная семантика языка
- \ni смысл
 - \Leftarrow второй домен*:
 - смысл*

Смысл - явное (формальное) представление описываемых сущностей и связей между ними. Для явного представления описываемых сущностей и связей между ними требуется существенное упрощение синтаксической структуры информационных конструкций.

Язык *ostis*-системы - язык представления информационных конструкций в *ostis*-системах.

язык *ostis*-системы

- \sqsubset формальный язык
- \sqsubset универсальный язык
- \Leftarrow используемые языки*:
 - ostis*-система
- \ni SC-код
 - \coloneqq [Semantic Computer Code]
 - \Leftarrow внутренний язык*:
 - ostis*-система
 - \in универсальный язык

Для формального описания различного рода языков, включая рассматриваемые нами языки (SCg-код, SCs-код, SCn-код) используется целый ряд метаязыковых понятий.

Перечислим некоторые из них: *идентификатор*, *класс синтаксически эквивалентных идентификаторов*, *имя*, *простое имя*, *выражение*, *внешний идентификатор**, *алфавит**, *разделители**, *ограничители**, *предложения**

Синтаксис языков представления знаний в *ostis*-системах может быть формально описан различными способами. Так, например, можно использовать метаязык Бэкуса-Наура для описания синтаксиса SCs-кода или его расширение для описания синтаксиса SCn-кода. Однако значительно более логично и целесообразно описывать синтаксис всех форм внешнего отображения sc-текстов с помощью самого SC-кода. Такой подход позволит *ostis*-системам самостоятельно понимать, анализировать и генерировать тексты указанных языков на основе принципов, общих для любых форм внешнего представления информации, в том числе нелинейных.

Алфавит* - бинарное отношение, связывающее множество текстов с семейством максимальных множеств синтаксически однотипных элементарных (атомарных) фрагментов текстов, принадлежащих заданному множеству текстов.

Идентификатор – структурированный знак соответствующей (обозначаемой) сущности, который чаще всего представляет собой строку (цепочку символов), которую будем называть именем соответствующей сущности. В формальных текстах (в том числе текстах SC-кода, SCg-кода, SCs-кода, SCn-кода) основные используемые

иентификаторы не должны быть омонимичными, то есть должны однозначно соответствовать идентифицируемым сущностям. Следовательно, каждая пара идентификаторов, имеющих одинаковую структуру, должны обозначать одну и ту же сущность.

имя

С идентификатор
 := [строковый идентификатор]
 := [идентификатор, представляющий собой строку (цепочку) символов]
 ⇒ декомпозиция действия*:
 {• простое имя
 := [атомарное имя]
 := [имя, в состав которого другие имена не входят]
 • выражение
 := [неатомарное имя]
 }

Внешний идентификатор* - бинарное ориентированное отношение, каждая связка (sc-дуга) которого связывает некоторый элемент с файлом, содержимым которого является внешний идентификатор (чаще всего, имя), соответствующий указанному элементу. Понятие внешнего идентификатора является понятием относительным и важным для ostis-систем, поскольку внутреннее для ostis-систем представление информации (в виде текстов SC-кода) оперирует не идентификаторами описываемых сущностей, а знаками, структура которых никакого значения не имеет.

§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем

Существует несколько языков внешнего представления sc-текстов **Standard2021**:

- SCn-код;
- SCs-код;
- SCg-код.

SCg-код - внешний язык* ostis-систем, тексты которого представляют собой графовые структуры общего вида с точно заданной денотационной семантикой*

SCg-код

∈ язык ostis-системы
 := [Semantic Code graphical]
 ⇐ внешний язык*:
 ostis-система
 ∈ универсальный язык

SCs-код - внешний язык* ostis-систем, тексты которого представляют собой строки (цепочки) символов.

SCs-код

∈ язык ostis-системы
 := [Semantic Code string]
 ⇐ внешний язык*:
 ostis-система
 ∈ универсальный язык

SCn-код - внешний язык* ostis-систем, тексты которого представляют собой двухмерные матрицы символов, являющиеся результатом форматирования, двухмерной структуризации текстов SCs-кода.

SCn-код

∈ язык ostis-системы
 := [Semantic Code natural]
 ⇐ внешний язык*:
 ostis-система
 ∈ универсальный язык

Для представления *баз знаний ostis-систем* используется целый ряд как *универсальных языков*, так и *специализированных языков*, как *формальных языков*, так и *естественных языков*, как *внутренних языков*, обеспечивающих представление информации в памяти *ostis-систем*, так и *внешних языков*, обеспечивающих представление информации, вводимой в память *ostis-систем*, либо выводимой из этой памяти. *Естественные языки* используются исключительно для представления *файлов*, хранимых в памяти *ostis-системы* и формально специфицируемых в рамках *базы знаний* этой *ostis-системы*.

Для эксплуатации *интеллектуальных компьютерных систем*, построенных на основе *SC-кода*, кроме способа абстрактного внутреннего представления баз знаний (*SC-кода*) потребуются несколько способов внешнего изображения абстрактных *sc-текстов*, удобных для пользователей и используемых при оформлении исходных текстов *баз знаний* указанных интеллектуальных компьютерных систем и исходных текстов фрагментов этих *баз знаний*, а также используемых для отображения пользователям различных фрагментов *баз знаний* по пользовательским запросам. В качестве таких способов внешнего отображения *sc-текстов* и предлагаются указанные выше внешние языки *ostis*-систем (*SCg-код*, *SCs-код* и *SCn-код*).

Все основные внешние формальные языки, используемые *ostis*-системами (*SCg-код*, *SCs-код*, *SCn-код*) являются различными вариантами внешнего представления текстов внутреннего языка *ostis*-систем – *SC-кода*. Указанные языки являются универсальными и, следовательно, *семантически эквивалентными языками**.

При этом, каждая *ostis*-система может приобрести способность использовать любой внешний язык (как универсальный, так и специализированный, как естественный, так и искусственный), если синтаксис и денотационная семантика этого языка будут описаны в памяти *ostis*-системы на ее внутреннем языке (*SC-коде*).

§ 2.1.3. Файлы *ostis*-систем

Глава 2.2.

Смыслоное представление знаний в памяти ostis-систем

⇒ *автор**:

- Голенков В. В.
- Иващенко В. П.

⇒ *аннотация**:

[Аннотация к главе.]

⇒ *ключевое понятие**:

- *sc-элемент*
- *обозначение sc-множества*
- *обозначение sc-связки*
- *обозначение sc-синглетона*
- *обозначение sc-пары*
- *обозначение неориентированной sc-пары*
- *обозначение ориентированной sc-пары*
- *обозначение sc-пары принадлежности*
- *обозначение sc-пары нечёткой принадлежности*
- *обозначение sc-пары позитивной принадлежности*
- *обозначение sc-пары негативной принадлежности*
- *обозначение ориентированной sc-пары, не являющейся sc-парой принадлежности*
- *обозначение sc-связки, не являющейся синглетоном и парой*
- *обозначение sc-класса*
- *обозначение sc-структуры*
- *обозначение внешней сущности*
- *обозначение внутреннего файла*
- *обозначение внешней сущности, не являющейся внутренним файлом*
- *обозначение постоянной сущности*
- *обозначение временной сущности*
- *обозначение статической сущности*
- *обозначение динамической сущности*
- *sc-константа*
- *sc-переменная*
- *sc-множество*
- *sc-связка*
- *sc-синглетон*
- *sc-пара*
- *неориентированная sc-пара*
- *ориентированная sc-пара*
- *sc-пара принадлежности*
- *sc-пара нечёткой принадлежности*
- *sc-пара позитивной принадлежности*
- *sc-пара негативной принадлежности*
- *ориентированная sc-пара принадлежности, не являющаяся sc-парой принадлежности*
- *sc-связка, не являющаяся синглетоном и парой*
- *sc-класс*
- *sc-структура*
- *внешняя сущность*
- *внутренний файл*

- внешняя сущность, не являющаяся внутренним файлом
- ⇒ подраздел*:
- § 2.2.1. SC-код (*Semantic Computer Code*)
 - § 2.2.2. Базовая денотационная семантика SC-кода
 - § 2.2.3. Синтаксис SC-кода
 - § 2.2.4. Файлы ostis-системы
 - § 2.2.5. Смыслоное пространство ostis-систем

§ 2.2.1. SC-код (*Semantic Computer Code*)

Общие положения

Ниже (в § 2.2.2. *Базовая денотационная семантика SC-кода* и в § 2.2.3. *Синтаксис SC-кода*) приведено формальное описание денотационной семантики и синтаксиса SC-кода. Но сделано это будет не совсем обычно. Поскольку все элементы конструкции являются обозначениями описываемых сущностей и, в том числе, обозначениями различных выделяемых классов *sc-элементов*, (!)нам ничего не стоит(!) явно ввести различные семантически значимые и синтаксически выделяемые классы *sc-элементов* и на основе этого явно описать средствами SC-кода базовую денотационную семантику и синтаксис SC-кода. Синтаксис SC-кода задаётся семейством классов синтаксический задаваемых (выделяемых) *sc-элементов*.

Элементы, принадлежащие каждому синтаксически выделяемому классу *sc-элементов* должны иметь одинаковые синтаксические признаки (синтаксические метки). При этом очевидно, что синтаксис SC-кода существенно упростится, если синтаксически выделяемые классы *sc-элементов* будут одновременно иметь и чёткую семантическую интерпретацию. Таким образом, формализацию синтаксиса SC-кода целесообразно осуществлять после формализации базовой денотационной семантики SC-кода. Путём синтаксического выделения тех семантически выделенных классов *sc-элементов*, которые, во-первых, необходимы для кодирования *sc-конструкций* памяти ostis-систем (в *sc-памяти*) и во-вторых, обеспечивают максимально возможное упрощение обработки *sc-конструкций* (например, путём упрощения анализа часто проверяемых семантических характеристик обрабатываемых *sc-элементов*).

Особенности SC-кода как одного из возможных вариантов смыслового представления знаний (см. § 1.2.2. *Принципы, лежащие в основе смыслового представления информации*)

Понятие *sc-элемента*

Понятие *sc-конструкции*

Понятие внутреннего файла ostis-систем

Понятие *sc-идентификатора*

§ 2.2.2. Базовая денотационная семантика SC-кода

⇒ подраздел*:

- *Пункт 2.2.2.1. Семантическая классификация sc-элементов по базовым признакам*
- *Пункт 2.2.2.2. Уточнение смысла выделенных классов sc-элементов и связей между этими классами*
- *Пункт 2.2.2.3. Структура базовой семантической спецификации sc-элемента*
- *Пункт 2.2.2.4. Онтологическая формализация Базовой денотационной семантики SC-кода*

Пункт 2.2.2.1. Семантическая классификация sc-элементов по базовым признакам

К числу базовых признаков классификации *sc-элементов* относятся:

- структурный признак;
- логико-семантический признак;
- темпоральная характеристика сущностей, обозначаемых *sc-элементами*, которая в свою очередь, включает в себя:
 - постоянство или временность существования обозначаемой сущности;
 - статичность (стационарность) или динамичность (изменчивость) обозначаемой сущности.

Структурная классификация sc-элементов

=
{

sc-элемент

⇒ разбиение*:

Структурный признак классификации sc-элементов

= {• обозначение *sc-множества*

⇒ разбиение*:

{• обозначение *sc-связки*

⇒ разбиение*:

{• обозначение *sc-синглетона*

• обозначение *sc-пары*

⇒ разбиение*:

{• обозначение неориентированной *sc-пары*

• обозначение ориентированной *sc-пары*

⇒ разбиение*:

{• обозначение *sc-пары принадлежности*

⇒ разбиение*:

{• обозначение *sc-пары нечеткой принадлежности*

• обозначение *sc-пары позитивной принадлежности*

• обозначение *sc-пары негативной принадлежности*

}

• обозначение *ориентированной sc-пары, не являющейся парой принадлежности*

}

}

• обозначение *sc-связки, не являющейся синглетоном и парой*

}

• обозначение *sc-класса*

• обозначение *sc-структуры*

}

• обозначение *внешней сущности*

⇒ разбиение*:

{• внутренний файл

• внешняя сущность, не являющаяся внутренним файлом

• обозначение файла

• обозначение информационной конструкции, не являющейся ни *sc-множеством*, ни файлом

• обозначение внешней сущности, не являющейся информационной конструкцией

}

}

/* Завершили представление *Структурной классификации sc-элементов* */

Логико-семантическая классификация sc-элементов

=
{

sc-элемент

⇒ разбиение*:

- sc-константа
 - := [sc-элемент, логико-семантическим значением которого является он сам]
- sc-переменная
 - ⇒ разбиение*:
 - sc-переменная 1-го уровня
 - := [sc-элемент, областью возможных значений которого является множество sc-констант]
 - sc-переменная 2-го уровня
 - := [sc-элемент, возможными значениями которого являются переменные 1-го уровня]
 - ⇒ примечание*:
 - [такие переменные (метапеременные) необходимы для описания логических языков]
 - sc-переменная универсального типа
 - := [sc-переменная, на значения которой не накладывается никаких ограничений]

{}

}

/* Завершили представление Логико-семантической классификации sc-элементов */

Классификация sc-элементов по темпоральным характеристикам обозначаемых ими сущностей

=
{

sc-элемент

⇒ разбиение*:

Признак постоянства существования сущностей, обозначаемых sc-элементами

- = {• обозначение постоянной сущности
 - := [обозначение постоянно существующей сущности]
- обозначение временной сущности
 - ⇒ разбиение*:
 - обозначение внешней временной сущности
 - ▷ обозначение внешней ситуации
 - ▷ обозначение внешнего события
 - ▷ обозначение внешнего процесса
 - обозначение внутренней временной сущности в sc-памяти
 - ⇒ разбиение*:
 - обозначение ситуации в sc-памяти
 - := [обозначение ситуации, которая возникла или возникает в процессе обработки информации в sc-памяти]
 - обозначение события в sc-памяти
 - := [обозначение события, которое произошло или произойдет в процессе обработки информации в sc-памяти]
 - обозначение информационного процесса в sc-памяти
 - := [обозначение внутреннего процесса в sc-памяти, который происходит, произошёл или будет происходить]

}

}

⇒ разбиение*:

Признак статичности сущностей, обозначаемых sc-элементами

- = {• обозначение статической сущности
 - := [обозначение статичной сущности]
 - := [обозначение стационарной сущности]
 - := [обозначение сущности, изменения которой в рамках соответствующего отрезка времени считаются несущественными]
 - ▷ обозначение статического sc-множества
- обозначение динамической сущности
 - := [обозначение сущности изменяющейся во времени]

} \supset обозначение динамического sc-множества
 }
 } /* Завершили представление Классификации sc-элементов по темпоральным характеристикам обозначаемых ими сущностей */

Когда речь идёт о темпоральных свойствах sc-элементов, следует чётко отличать:

- временный характер присутствия любого sc-элемента в составе той базы знаний (в той sc-памяти) ostis-системы, в которой он находится (когда-то он появляется, когда-то может быть удалён из sc-памяти);
- временный характер присутствия в sc-памяти всей заданной sc-конструкции (заданного множества sc-элементов) — такую sc-конструкцию будем называть ситуацией в sc-памяти;
- временный характер существования внешней сущности, которую sc-элемент обозначает;
- статичный или динамичный (изменчивый) характер внешней сущности, обозначаемой sc-элементом динамический характер внешней сущности, предполагает наличие в sc-памяти описания процесса изменения состояния или конфигурации указанной внешней сущности;
- динамическое sc-множество (динамическую sc-конструкцию), являющееся отражением (динамической модели) соответствующего внешнего процесса (процесса, происходящего во внешней среде);
- динамическое sc-множество (динамическую sc-конструкцию), являющуюся отражением (динамической моделью) соответствующего внутреннего процесса (информационного процесса, происходящего в той же sc-памяти, в которой находится sc-элемент, обозначающий указанное динамическое sc-множество)

Структурная классификация sc-констант

\Rightarrow примечание*:

[Данная классификация полностью аналогична Структурной классификации sc-элементов, в отличие от которой она описывает структурную классификацию только константных sc-элементов (sc-констант)]

\in sc-структура

\Leftrightarrow аналог*:

Структурная классификация sc-элементов

Структурная классификация sc-констант

=

{

sc-константа

\Rightarrow разбиение*:

{• sc-множество

\Rightarrow разбиение*:

{• sc-связка

\Rightarrow разбиение*:

{• sc-синглетон

• sc-пара

\Rightarrow разбиение*:

{• неориентированная sc-пара

• ориентированная sc-пара

\Rightarrow разбиение*:

{• sc-пара принадлежности

\Rightarrow разбиение*:

{• sc-пара нечёткой принадлежности

• sc-пара позитивной принадлежности

\supset sc-пара постоянной позитивной принадлежности

\Leftarrow пересечение множеств*:

{• sc-константа

• постоянная сущность

• статическая сущность

• sc-пара позитивной принадлежности

\supset sc-пара постоянной позитивной принадлежности

\Leftarrow пересечение множеств*:

{• sc-константа

• постоянная сущность

• статическая сущность



Пункт 2.2.2.2. Уточнение смысла выделенных классов sc-элементов и связей между этими классами

Перейдём к детальному рассмотрению смысла классов *sc-элементов* (*sc-классов*), введенных в представленных выше классификациях.

Указанные *sc-классы* рассматриваются в порядке их введения в представленных выше классификациях *sc-элементов*.

Сначала поясним смысл понятий (*sc-классов*), введенных в Структурной классификации *sc-элементов* и в Структурной классификации *sc-констант*.

sc-элемент

:= [обозначение множества]

:= [*sc*-обозначение множества, представимого в SC-коде]

⇒ разбиение*:

{• *sc-множество*

:= [обозначение множества *sc-элементов*]

:= [обозначение множества, все элементы которого являются *sc-элементами*]

:= [обозначение внутренней для *sc*-памяти сущности, то есть сущности, хранимой в *sc*-памяти]

• обозначение внешней сущности

:= [обозначение синглетона внешней сущности]

:= [терминальный *sc-элемент*]

}

⇒ примечание*:

• [Каждый *sc-элемент* является обозначением соответствующего множества.]

• [Ко множествам, представимым в SC-коде, относятся либо множества, элементами которых являются *sc-элементы* (*sc-множества*), либо синглетоны, элементами которых являются сущности, не являющиеся *sc-элементами* (синглетоны внешних сущностей). Таким образом, строго говоря, не каждое множество может быть обозначено соответствующим *sc-элементом*. Но каждое множество, не являющееся *sc-множеством* или синглетоном указанного выше вида может быть однозначно преобразовано в *sc-множество* и описано средствами *SC-кода*. При этом теоретико-множественные свойства "нестандартных" для *SC-кода* множеств совпадают со свойствами соответствующих им "стандартных" для *SC-кода* множеств.]

• [Тот факт, что каждый *sc-элемент* является обозначением соответствующего множества (частным случаем которых являются синглетоны внешних описываемых сущностей), означает то, что базовым видом объектов, которыми оперирует *SC-код* на синтаксической, семантическом и логическом уровне являются множества знаков, обозначающих различные множества. В этом смысле *SC-код* имеет базовую теоретико-множественную основу.]

⇒ правила построения внешних идентификаторов *sc-элементов* заданного класса*:

Общие правила построения внешних идентификаторов *sc-элементов*

:= [Общие правила идентификации *sc-элементов*]

= {• [Принадлежность идентифицируемого *sc-элемента* некоторым классам *sc-элементов* (*sc-классам*) явно указывается во внешнем идентификаторе этого *sc-элемента* (в *SC-идентификаторе*) с помощью соответствующих условных признаков:

- если первым символом *sc-идентификатора* является знак подчёркивания, то идентифицируемый *sc-элемент* принадлежит Классу *sc-переменных*. По умолчанию считается, что идентифицируемый *sc-элемент* принадлежит Классу *sc-констант*;
- если последним символом *sc-идентификатора* является символ "звездочка", то идентифицируемый *sc-элемент* принадлежит Классу обозначений неролевых отношений;
- если последним символом *sc-идентификатора* является апостроф, то идентифицируемый *sc-элемент* принадлежит Классу обозначений ролевых отношений, каждого из которых является подмножеством Отношения принадлежности, то есть Класса всех константных позитивных пар принадлежности;
- если последним символом *sc-идентификатора* является символ "^", то идентифицируемый *sc-элемент* принадлежит Классу обозначений параметров.

]

• [Слово "обозначение" в *sc-идентификаторе* означает то, что обозначаемая сущность может быть как константной, так и переменной.]

• [В *sc-идентификаторах* можно делать следующие сокращения:

- "'sc-элемент, обозначающий ...'" — "обозначение"
- "'обозначение константного'" — "знак константного"

- "знак константного" — "константный"
- слово "константный" в sc-идентификаторах можно опускать, так как константность подразумевается по умолчанию
 -]
 - [Для каждого sc-элемента можно построить sc-идентификатор, являющийся именем собственным, которое всегда начинается с большой буквы (заглавной) буквы.]
 - [Если sc-элемент является обозначением некоторого класса sc-элементов, то этому sc-элементу можно поставить в соответствие не только имя собственное, но и имя нарицательное, которое начинается маленькой (строчной) буквы. Приведём пример семейства эквивалентных (сионимичных) sc-идентификатор, среди которых есть как имена собственные, так и имена нарицательные:
 - Множество всевозможных обозначений sc-множеств
 - Класс обозначений sc-множеств
 - обозначение sc-множества
 -]
- }

обозначение sc-множества

- \coloneqq [sc-элемент, являющийся знаком множества всевозможных обозначений sc-множеств]
 - \in *имя собственное*
- \coloneqq [Знак множества всевозможных обозначений sc-множеств]
 - \in *имя собственное*
- \coloneqq [Множество всевозможных обозначений sc-множеств]
 - \in *имя собственное*
- \coloneqq [Класс обозначений sc-множеств]
 - \in *имя собственное*
- \coloneqq [sc-элемент, являющийся обозначением множества sc-элементов]
 - \in *имя нарицательное*
- \coloneqq [sc-обозначение множества sc-элементов]
 - \in *имя нарицательное*
- \coloneqq [обозначение множества, каждый элемент которого является sc-элементом]
- \coloneqq [обозначение информационной конструкции, принадлежащей SC-коду]
- \coloneqq *часто используемый sc-идентификатор**:
 - [обозначение sc-конструкции]
- \Rightarrow *разбиение**:
 - {• *sc-множество*
 - \coloneqq [знак константного sc-множества]
 - $=$ (*обозначение sc-множества* \cap *sc-константа*)
 - *переменное sc-множество*
 - $=$ (*обозначение sc-множества* \cap *sc-переменная*)
- }

следует отличать*

- \ni {• *обозначение sc-множества*
 - \coloneqq [обозначение sc-множества, которое может быть как константным sc-множеством, так и переменным sc-множеством]
 - \coloneqq [обозначение внутренней для sc-памяти сущности]
 - \coloneqq [обозначение внутренней sc-памяти информационной конструкции (sc-конструкции)]
 - \Rightarrow *разбиение**:
 - {• *sc-множество*
 - \coloneqq [обозначение конкретного множества]
 - \coloneqq [знак множества]
 - $=$ (*sc-константа* \cap *обозначение sc-множества*)
 - \coloneqq [конкретное sc-множество]
 - \coloneqq [знак константного sc-множества]
 - \coloneqq [константное sc-множество]
 - *переменное sc-множество*
 - \coloneqq [произвольное sc-множество]
 - \coloneqq [обозначение произвольного sc-множества]
 - $=$ (*sc-переменная* \cap *обозначение sc-множества*)
 - }
- *sc-множество*

- *переменное sc-множество*
 - *обозначение внешней сущности*
 - \coloneqq [обозначение сущности, не являющейся множеством sc-элементов (sc-множеством)]
 - \supset *обозначение файла*
 - \coloneqq [обозначение файла, хранимого либо в файловой памяти той же ostis-системы, в sc-памяти которой хранится знак этого файла, либо в файловой памяти другой дополнительно указываемой компьютерной системы]
 - \supset *обозначение информационной конструкции, не являющейся ни sc-множеством, ни файлом*
 - \Rightarrow *примечание**:
[Примером такой информационной конструкции является напечатанный текст, речевое сообщение, которой следует отличать от его записи в виде аудио-файла . . .]
 - \supset *обозначение внешней сущности, не являющейся информационной конструкцией*
 - \Rightarrow *примечание**:
[Примером такой внешней сущности . . .]
- }

sc-множество

- \coloneqq [sc-конструкция]
- \coloneqq [информационная конструкция, принадлежащая SC-коду]
- \coloneqq *часто используемый sc-идентификатор**:
 - [SC-код]
 - \in *имя собственное*
- \coloneqq [Множество всевозможных sc-конструкций]

обозначение sc-связки

- \Rightarrow *разбиение**:
 - {• *sc-связка*
 - *переменная sc-связка*

sc-связка

- \coloneqq [знак связи (связки) между sc-элементами]
- \Rightarrow *примечание**:
[Если элементами sc-связки являются знаки внешних сущностей, то sc-связка является отображением (моделью) некоторой связи, которая связывает указанные внешние сущности]
- \Rightarrow *пояснение**:
 - [Понятие sc-связки — это попытка формализации понятия целостности, понятия перехода некоторой совокупности сущности в некоторое новое качество, которое не сводится к свойствам каждой сущности, входящей в эту совокупность. Таким образом, связками следует считать:
 - множество всех чисел, являющихся слагаемыми для заданного числа;
 - множество всех сотрудников заданной организации, в заданный момент времени;
 - множество всех сотрудников заданной организации, которые работают в ней;
 - множество всех точек некоторого отрезка;
 - множество всех точек некоторого треугольника.

]

- \Rightarrow *примеры**:

[Примерами sc-связок являются:

- конкретная окружность, (множество всех точек, равноудаленных от некоторой заданной точки);
- конкретный отрезок (множество всех точек, лежащих между двумя заданными точками с включением этих точек);
- конкретный линейный треугольник (множество всех точек, лежащих между каждыми двумя из трёх заданных точек с включением этих точек);
- пары граничных точек различных отрезков;
- тройки вершин различных треугольников.

]

обозначение sc-синглетона

- \Rightarrow *разбиение**:
 - {• *sc-синглeton*

- *переменный sc-синглетон*
- }

sc-синглетон

- := [sc-множество, являющееся синглетоном]
- := [одномощное sc-множество]
- := [sc-множество, имеющее мощность, равную единице]
- := [sc-элемент, обозначающий унарную sc-связку]
- := [sc-обозначение унарной sc-связки]
- := [унарная sc-связка]
- := [обозначение sc-синглетона]
- := [обозначение одномощного множества, единственный элемент которого является sc-элементом]

обозначение sc-пары

- ∈ sc-константа
 - ∈ sc-класс
 - ⇒ разбиение*:
 - {• *sc-пара*
 - := [константная sc-пара]
 - ⊆ sc-константа
 - ∈ sc-константа
 - ∈ sc-класс
 - *переменная sc-пара*
 - ∈ sc-переменная
 - ∈ sc-константа
 - ∈ sc-класс
- }

sc-пара***обозначение неориентированной sc-пары***

- := [неориентированная sc-пара]

обозначение ориентированной sc-пары

- := [ориентированная sc-пара]

обозначение sc-пары принадлежности

- := [sc-пара принадлежности]

обозначение sc-пары принадлежности

- := [sc-пара принадлежности]

обозначение sc-пары нечёткой принадлежности

- := [sc-пара нечёткой принадлежности]

обозначение sc-пары позитивной принадлежности

- := [sc-пара позитивной принадлежности]

sc-пара константной постоянной позитивной принадлежности

- := [константная позитивная постоянная sc-пара принадлежности]

sc-пара константной временной позитивной принадлежности***обозначение sc-пары негативной принадлежности***

- := [sc-пара негативной принадлежности]

обозначение sc-пары, не являющейся парой принадлежности

:= [sc-пара, не являющаяся парой принадлежности]

обозначение sc-связки, не являющейся синглетоном и парой

:= [sc-пара, не являющаяся синглетоном и парой]

обозначение sc-класса

:= [sc-класс]

⇒ разбиение*:

- {• sc-класс
- переменный sc-класс

}

⇒ разбиение*:

- {• обозначение sc-класса обозначений sc-связок
- обозначение sc-класса обозначений sc-классов
- обозначение sc-класса обозначений sc-структур
- обозначение sc-классов обозначений внешних сущностей

}

sc-класс

⇒ разбиение*:

- {• sc-класс sc-связок
 - ▷ sc-отношение
- sc-класс sc-классов
 - ▷ sc-параметр
- sc-класс sc-структур
- sc-класс внешних сущностей
 - := [sc-класс sc-элементов, являющихся знаками внешних сущностей]
- sc-класс sc-элементов разного структурного типа
 - Э пример':
 - [sc-константа]

}

Перечислим основные основные виды sc-классов

sc-класс

▷ sc-отношение

:= [sc-класс sc-связок]

▷ бинарное sc-отношение

⇒ разбиение*:

- {• бинарное неориентированное sc-отношение
- бинарное ориентированное sc-отношение
 - ▷ ролевое sc-отношение

}

▷ sc-класс sc-классов

▷ sc-параметр

▷ sc-класс sc-структур

▷ sc-класс внешних сущностей

▷ sc-класс внутренних файлов

▷ sc-класс эквивалентности

:= [фактор-множество соответствующего отношения эквивалентности]

⇒ пояснение*:

[Требованиями, предъявляемыми к каждому sc-классу являются:

- бесконечность этого sc-множества;
- наличие общего свойства, присущего всем элементам этого sc-множества, в частности, наличие его определения.

]

следует отличать*

Э {• sc-связка
• sc-класс
}

⇒ *сравнение*:

[В отличие от sc-связки принципом формирования sc-класса является наличие общего свойства, присущего всем элементам этого sc-класса и только им, (или присущего всем сущностям, которые обозначаются указанными sc-элементами). Таким общим свойством может быть определение sc-класса либо принадлежность одному из значений некоторого параметра, то есть одному из элементов фактор-множества, соответствующего некоторому отношению эквивалентности или толерантности.]

⇒ *пояснение*:

[Примерами связок являются:

- множество людей живущих сейчас (динамическое множество);
- множество сотрудников некоторой организации (динамическое множество);
- отрезок, треугольник.

Здесь речь не идёт об эквивалентности свойств самих людей и геометрических точек безотносительно к тому, в состав чего они входят. Поэтому это не является sc-классом.]

Э {• sc-класс эквивалентности

⇒ *пояснение*:

[В sc-класс входит не просто некоторое количество попарно эквивалентных между собой сущностей, а абсолютно все такие сущности]

• sc-связка попарно эквивалентных сущностей

}

Приведём пример:

следует отличать*

Э {• множество всех треугольников, подобных одному из них
 ⊂ sc-класс

• конечное множество подобных треугольников
 ⊂ sc-связка попарно эквивалентных треугольников

}

Э {• параметр

:= часто используемый sc-идентификатор*:

[sc-параметр]

⊂ класс классов

• признак различия
 := [признак классификации]

}

⇒ *пояснение*:

{

параметр

⊂ бесконечное множество

признак различия

⊂ конечное множество

Э *пример*':

• Признак конечности множеств

= {• конечное множество
• бесконечное множество
}

• Признак наличия кратных элементов

= {• мульти множество
• множество без кратных вхождений элементов
}

}

sc-класс

⇒ правила построения внешних идентификаторов sc-элементов заданного класса*:

Правила построения внешних идентификаторов sc-элементов, являющихся знаками sc-классов

- = {• [Слово "обозначение" в начале идентификатора используется тогда, когда в идентифицируемый класс sc-элементов включаются знаки как константных, так и переменных сущностей соответствующего вида]
 - [Слово "переменный" в начале идентификатора и ...]
- }

обозначение sc-структуры***sc-структура******следует отличать****

≡ {• sc-структура
• sc-связка
}

⇒ сравнение*:

- {• [В отличие от sc-связок в каждую sc-строктуру должна входить по крайней мере одна sc-связка вместе с компонентами этой sc-связки]

}

обозначение внешняя сущность***внешняя сущность***

:= [синглетон внешней сущности]

:= [обозначение синглетона внешней сущности]

:= [sc-элемент, обозначающий синглетон, элементом которого является некоторая внешняя описываемая сущность]

:= [множество обозначаемой sc-элементом, являющееся 1-мощным множеством, единственным элементом которого является сущность, внешняя по отношению к sc-конструкции, то есть сущность, не являющаяся sc-элементом]

⇒ примечание*:

- [обозначение внешней сущности, то есть sc-элемент, обозначающий этот синглетон, можно также трактовать как sc-элемент, обозначающий соответствующую внешнюю описываемую сущность, которую в свою очередь можно считать денодом указанного sc-элемента]
- [очевидно, что пара принадлежности, связывающая sc-элемент, обозначающий синглетон внешней сущности, не может быть непосредственно представлена в виде соответствующей sc-дуги принадлежности, так как второй компонент этой sc-дуги не находится в sc-памяти]

следует отличать*

≡ {• синглетон внешней сущности

• sc-синглетон

:= [синглетон, единственным элементом которого является некоторый sc-элемент]

⊂ sc-множество

:= [sc-элемент, обозначающий множество, элементами которого являются только sc-элементы]

:= [множество sc-элементов]

}

обозначение файла***файл***

внутренний файл

- := [внутренний образ (копия), внутренней информационной конструкции, хранимый в файловой памяти ostis-системы]
- := [файл ostis-системы]
- ⇒ *примечание*:*
 - [файловая память ostis-системы, хранящая различного рода информационные конструкции (образы, модели), не являющиеся sc-конструкциями, должна быть тесно связана с sc-памятью этой же ostis-системы. Как минимум каждый файл ostis-системы должен быть связан с тем sc-узлом, которых является знаком этого файла (точнее, знаком синглетона, элементом которого является указанный файл)]

Перейдем к пояснению смысла понятий используемых в *Логической классификации sc-элементов*.

sc-константа

- := [sc-элемент, обозначающий константную сущность]
 - ⇒ *сокращение*:*
 - [обозначение константной сущности]
- := [обозначение константной сущности]
- := [знак константной сущности]
 - ⇒ *сокращение*:*
 - [константная сущность]
 - ⇒ *сокращение*:*
 - [сущность]
- := [константная сущность]
- := [конкретная сущность]
- := [сущность]
- := [константный sc-элемент]
- := [sc-элемент, имеющий одно логико-семантическое значение, каковым является он сам]
- := [sc-элемент, являющийся знаком константной (конкретной, фиксированной) сущности]
- ⇒ *сокращение*:*
 - [знак константной (конкретной, фиксированной) сущности]
 - ⇒ *сокращение*:*
 - [константная (конкретная, фиксированная) сущность]
 - ⇒ *сокращение*:*
 - [константная сущность]

sc-переменная

- := [переменный sc-элемент]
- := [sc-элемент, являющийся обозначением некоторой произвольной (нефиксированной, переменной) сущности]
 - ⇒ *сокращение*:*
 - [обозначение произвольной (переменной) сущности]
 - ⇒ *сокращение*:*
 - [переменная сущность]
- ∈ sc-константа
- ∈ sc-класс
- ⇒ *примечание*:*
 - [Сам sc-элемент, имеющий внешний идентификатор "sc-переменная" является sc-константой (константным sc-элементом), которая является знаком одного из классов sc-элементов]

sc-элемент

- := [обозначение константной или переменной сущности]
- := [константная или переменная сущность]
- := [sc-константа или переменная сущность]
- := [обозначение описываемой сущности, которая может быть как константой, так и переменной сущностью, как внутренней, так и внешней sc-конструкцией для заданной ostis-системы, как информационной конструкцией, так и сущностью которая информационной конструкцией не является, как временной сущностью, так и постоянной, как динамической, так и статической сущностью]

обозначение sc-множества

⇒ разбиение*:

{• sc-множество

:= [часто используемый sc-идентификатор]

⇒ сокращение*:

[множество sc-элементов]

:= [константное (конкретное) sc-множество]

:= [обозначение (знак) конкретного множества]

⊂ sc-константа

∈ sc-константа

⇒ разбиение*:

{• множество констант

:= [множество, каждый элемент которого является константой]

:= [множество, являющееся подмножеством Множества всевозможных констант]

• множество переменных

• множество констант и переменных

:= [множество, элементами которого являются как константы, так и переменные]

• sc-множество sc-констант

:= [sc-множество, элементами которого являются только sc-константы]

• sc-множество sc-переменных

:= [sc-множество, элементами которого являются только sc-переменные]

• sc-множество sc-констант и sc-переменных

:= [произвольное множество]

:= [обозначение переменного (произвольного) sc-множества]

⊂ sc-переменная

∈ sc-константа

⇒ следует отличать*:

sc-множество sc-переменных

}

• переменное sc-множество

:= [обозначение переменного (произвольного) sc-множества]

}

Перейдем к пояснению смысла понятий, используемых в *Классификации sc-элементов по темпоральным характеристикам обозначаемых ими сущностей*.

обозначение временной сущности

⇒ разбиение*:

{• обозначение временной сущности существующей сейчас

:= [обозначение временной сущности, существующей в текущий (настоящий) момент]

• обозначение прошлой временной сущности

:= [обозначение бывшей временной сущности]

:= [обозначение временной сущности, которая уже перестала существовать, прекратила своё существование]

• обозначение будущей временной сущности

:= [обозначение временной сущности, появление которой прогнозируется (планируется, обеспечивается)]

⇒ примечание*:

[проектирование и производство новых, ранее не существующих полезных сущностей – это основное направление человеческой деятельности]

}

⇒ примечание*:

[ostis-системы должны постоянно мониторить состояние временных сущностей, так как в процессе их функционирования будущие сущности становятся настоящими, а настоящие – прошлыми]

динамическое sc-множество

:= [sc-процесс]

:= [процесс]

⇒ *определение**:

[sc-множество, у которого некоторые позитивные пары принадлежности, связывающие знак этого множества с его элементами, носят временный характер]

⇒ *примечание**:

[Сами элементы динамического sc-множества, связанные с ним временными позитивными парами принадлежности, могут обозначать как временные, так и постоянные сущности. Но чаще всего такими временными элементами динамического sc-множества являются знаки временных связок.]

⇒ *разбиение**:

- {• *внешний процесс*
 - *процесс в sc-памяти*
- }

tempоральная декомпозиция динамического sc-множества

:= [покадровая развертка динамического sc-множества]

:= [представление sc-множества в виде кортежа (последовательности) ситуаций]

следует отличать*∃ {• *временная сущность*

- *обозначение временной сущности*
- *переменная временная сущность*

}***обозначение временной сущности***⇒ *разбиение**:

- {• *временная сущность*
 - := [знак конкретной (константной) временной сущности]
 - *переменная временная сущность*
 - := [обозначение произвольной временной сущности]
}

сформированное sc-множество

:= [sc-множество, у которого в текущем состоянии sc-памяти перечислены все его элементы]

∈ *динамическое sc-множество*⇒ *примечание**:

[Очевидно, что сформированным sc-множеством может стать только конечное sc-множество]

формируемое sc-множество***sc-множество, элементы которого не известны******сформированный файл******формируемый файл******файл, структура которого не известна***

Перейдем к рассмотрению семантически выделяемых классов sc-элементов, которые необходимо ввести дополнительно к выше рассмотренным классам sc-элементов.

sc-элемент, не являющийся sc-синглтоном и sc-парой***sc-элемент, копируемый в других компьютерных системах***

:= [sc-элемент, имеющий в других компьютерных системах свои копии и/или копии обозначаемой им информационной конструкции]

отношение, заданное на множестве sc-элементов, копируемых в других компьютерных системах*

Э
Э
Э

отношение, заданное на множестве sc-элементов, имеющих копии в других компьютерных системах*

- Э ostis-система, в sc-памяти которой хранится копия заданного sc-элемента*
- Э компьютерная система, в файловой памяти которой хранится заданный файл*
 - ⇒ примечание*: [указанная компьютерная система назначается хранителем файла]
- Э ostis-система, в sc-памяти которой хранится копия знака заданного sc-множества и все известные в текущий момент его элементы*
 - ⇒ примечание*: [указанная ostis-система назначается основным хранителем указанного sc-множества]

информационная конструкция

- ⇒ разбиение*:
 - {• sc-множество
 - := [sc-конструкция]
 - := [информационная конструкция SC-кода]
 - := [внутренняя информационная конструкция ostis-системы, хранимая в её sc-памяти]
 - файл
 - := [файл ostis-системы]
 - := [внутренняя информационная конструкция ostis-системы, хранимая в её файловой памяти]
 - ⇒ примечание*: [файл, может храниться в памяти другой компьютерной системы и, в частности, в файловой памяти другой ostis-системы]
 - внешняя информационная конструкция, не являющаяся ни файлом, ни sc-конструкцией

sc-идентификатор

- := [внешний идентификатор sc-элемента]
- ▷ файл
- ⇒ разбиение*:
 - {• основной идентификатор
 - часто используемый sc-идентификатор
 - дополнительный sc-идентификатор

sc-идентификатор*

- := [Бинарное ориентированное отношение, связывающее sc-элементы с их внешними идентификаторами]

Пункт 2.2.2.3. Структура базовой семантической спецификации sc-элемента

базовая семантическая спецификация sc-элемента

:= пояснение*:

[Класс sc-структур, каждая из которых описывает базовые семантические свойства (характеристики) соответствующего (описываемого, специфицируемого) sc-элемента]

⊆ *sc-структура*

⊆ *sc-спецификация*

:= [представленная в SC-коде семантическая окрестность (спецификация) некоторого (специфицируемого) sc-элемента]

⊆ *sc-спецификация*

⇐ *второй домен*:*

*базовая семантическая спецификация sc-элемента**

:= пояснение*:

[Бинарное ориентированное отношение, каждая пара которого связывает sc-элемент с его базовой семантической спецификацией*]

:= пояснение*:

[Хранимая в sc-памяти ostis-система спецификация каждого sc-элемента, необходимая для эффективной обработки этого sc-элемента]

⇒ *примечание*:*

[базовая спецификация sc-элементов осуществляется как явно с помощью соответствующих sc-конструкций, так и неявно с помощью соответствующих семантических меток, приписываемых sc-элементам]

⇒ *пояснение*:*

[Базовая семантическая спецификация каждого sc-элемента включает в себя:

- перечисление всех тех *базовых классов sc-элементов*, которым принадлежит специфицируемый sc-элемент;
- уточнение "привязки" рассматриваемых временных сущностей к текущему моменту времени и другим моментам времени;
- уточнение того, какие важные характеристики специфицируемого sc-элемента в текущем состоянии sc-памяти и файловой памяти ostis-системы не известны.

]

Базовая семантическая спецификация sc-элемента, обозначающего временную сущность включает в себя указание дополнительных темпоральных характеристик, позволяющих уточнить темпоральные "координаты" этих временных сущностей (то есть их "координаты" во времени), а также их основные темпоральные связи с другими временными сущностями. К числу понятий, обеспечивающих описание указанных темпоральных характеристик временных сущностей, относятся:

- момент времени[^]
- Текущий момент времени
- прошлая сущность
- будущая сущность
- момент начала*
- момент завершения*
- внешняя ситуация
- ситуация в sc-памяти
- внешнее событие
- событие в sc-памяти
- внешний процесс
- процесс в sc-памяти

момент времени[^]

- ∈ *параметр*
- ∈ *параметр, заданный на множестве временных сущностей*
- := [глобальная приблизительно точная ситуация[^]]
- := [глобальная ситуация пренебрежительно малого отрезка времени[^]]
- := [множество (класс) всех временных сущностей, существующих одновременно в соответствующий момент времени[^]]
- ⇒ *примечание*:*
[момент времени, соответствующий глобальной точечной ситуации может быть задан с различной и дополнительно указываемой степенью точности — с точностью до секунды, до минуты, до часа, до даты, до календарного месяца, до календарного года и так далее. В том смысле корректнее говорить не о моменте времени, а об интервале времени, длительность которого считается пренебрежимо малой для рассмотрения описываемых процессов]

текущий момент времени

- := [Глобальная ситуация текущего (настоящего) момента времени]
- := [Глобальная ситуация, имеющая место сейчас]
- := [Класс всех сущностей, существующих в настоящий момент времени]
- ∈ *sc-синглетон*
- ∈ *динамическое sc-множество*
- ⇐ *включение множества*:*
момент времени
- ⇒ *пояснение*:*
[Из знака *текущего момента времени* (который является также знаком *sc-синглетона*) "выходит" sc-дуга (*sc-пара*) временной принадлежности, представляющая собой, образно говоря, "стрелку" внутренних часов *ostis-системы*, которая всегда указывает только на один элемент множества моментов времени, но в разные моменты времени указывает на разные элементы этого множества]

прошлая сущность

- := [временная сущность, уже завершившая своё существование]

будущая сущность

- := [прогнозируемая, планируемая или создаваемая временная сущность]

момент начала*

- := [момент времени, соответствующий началу существования заданной временной сущности]
- := [Бинарное ориентированное отношение, каждая пара которого, связывает (1) знак некоторой временной сущности и (2) глобальную точечную ситуацию (значение параметра "*момент времени[^]*"), элементом которой является условно точечная временная сущность, представляющая собой начальный этап существования временной сущности, указанной в первом компоненте рассматриваемой ориентированной пары]
- ⇒ *примечание*:*
[Начальный этап существования временной сущности (переходный процесс от небытия к реальному существованию) может рассматриваться с любой степенью детализации]
- ⇒ *первый домен*:*
временная сущность
- ⇒ *второй домен*:*
момент времени[^]

момент завершения*

- := [момент времени, соответствующий завершению существования заданной временной сущности]

ситуация

- ⇒ *разбиение*:*
 - {• *внешняя ситуация*
 - *ситуация в sc-памяти*

событие

⇒ разбиение*:

- {• внешнее событие
- событие в sc-памяти

{}

динамическое sc-множество

⇒ разбиение*:

- {• внешний процесс
 - := [процесс, происходящий в окружающей среде ostis-системы]
- процесс в sc-памяти

}

внешняя ситуация

:= [ситуация во внешней среде]

:= [ситуация одновременного существования (в соответствующий период времени) указанных временных внешних сущностей]

⊂ временная сущность

⊂ sc-структура

⊂ sc-константа

⊂ обозначение внешней ситуации

∈ sc-класс

класс внешних ситуаций***обобщенное описание******класс внешних ситуаций***

⇒ примечание*:

[В простейшем случае внешние ситуации, входящие в класс внешних ситуаций являются изоморфными]

внешний процесс

:= [temporalная детализация внешней динамической сущности]

внешнее событие

:= [факт появления (возникновения) некоторой внешней сущности (в том числе некоторой внешней ситуации) или факт завершения существования некоторой внешней сущности (в том числе некоторой внешней ситуации)]

ситуация в sc-памяти

:= [внутрення ситуация]

:= [sc-ситуация]

:= [хранимый в sc-памяти фрагмент базы знаний, рассматриваемый в контексте его появления в sc-памяти или его исчезновения (из-за удаления некоторые sc-элементов)]

класс ситуаций в sc-памяти

:= [класс внутренних ситуаций]

обобщённое описание класса ситуаций в sc-памяти***процесс в sc-памяти***

:= [внутренний процесс]

:= [информационный процесс, происходящий в sc-памяти]

:= [sc-процесс]

событие в sc-памяти

Важной частью *базовой семантической спецификации sc-элемента* является фиксация того, что ostis-система знает и чего она не знает о специфицируемом sc-элементе или об обозначенной им сущности:

- Если в спецификации sc-элемента указывается его принадлежность к некоторому классу sc-элементов, но не указывается его принадлежность одному из подклассов, на которые *разбивается* указанный выше класс, то это означает, что в текущий момент времени ostis-система этого не знает;
- Если специфицируемый sc-элемент является обозначением конечного множества sc-элементов (в частности, пары sc-элементов), и если в текущий момент времени ostis-системе не известны все этого множества (то есть специфицируемый sc-элемент не соединён соответствующими парами принадлежности со всеми элементами обозначаемого им множества sc-элементов), то этот специфицируемый sc-элемент следует отнести к sc-классу "*обозначение несформированного sc-множества*";
- Если специфицируемый sc-элемент является обозначением ориентированной sc-пары и если в текущий момент времени ostis-системе не известна направленность этой ориентированной пары sc-элементов (то есть не известно, какой элемент этой пары является первым её компонентом, а какой её элемент является её вторым компонентом), то этот специфицируемый sc-элемент следует отнести к sc-классу "*обозначение ориентированной sc-пары неизвестной направленности*".

К числу понятий, используемых для описания полноты базовой спецификации sc-элементов, относятся:

- *обозначение бесконечного sc-множества*;
- *обозначение конечного sc-множества*;
- *мощность обозначаемого sc-множества**;
- *обозначение sc-множества неизвестной мощности*;
- *обозначение sc-множества, о котором не известно, является ли оно sc-парой*;
- *обозначение sc-пары, о которой не известно, является ли она ориентированной или нет*;
- *обозначение ориентированной sc-пары неизвестной направленности*;
- *обозначение сформированного sc-множества*;
- *обозначение частично сформированного sc-множества*;
- *обозначение полностью несформированного sc-множества*;
- *обозначение сформированного файла*;
- *обозначение частично сформированного файла*;
- *обозначение полностью несформированного файла*;

Подчеркнём то, что базовую семантическую спецификацию должны иметь абсолютно все *sc-элементы*, хранимые в *sc-памяти* в текущий момент времени, в том числе и все *sc-элементы*, являющиеся ключевыми знаками в рамках *Предметной области Базовой денотационной семантики SC-кода*. Приведём пример базовой семантической sc-спецификации одного из таких sc-элементов:

обозначение sc-множества

- := [Множество всевозможных sc-элементов, обозначающих sc-множества]
 ∈ *имя собственное*
 ∈ *обозначение sc-множества*
 ⇒ *примечание*:*
 [Одним из элементов данного множества является знак, обозначающий это множество. Это означает, что данное множество является рефлексивным множеством]
 ∈ *обозначение множества sc-элементов разного структурного типа*
 ⇒ *примечание*:*
 [Элементами данного множества являются:
 • обозначения sc-синглетонов;
 • обозначения sc-пар;
 • обозначения sc-связок, не являющихся sc-синглетонами или sc-парами;
 • обозначения sc-классов;
 • обозначения sc-структур.
]
 ∈ *обозначение множества sc-элементов, содержащего как константные, так и переменные sc-элементы*
 ∈ *sc-константа*
 ⇒ *примечание*:*
 [Само данное множество является константным, несмотря на то, что его элементами являются как sc-константы, так и sc-переменные]
 ∈ *обозначение множества sc-элементов, содержащего sc-элементы, обозначающие как постоянные, так и временные сущности*
 ∈ *постоянная сущность*
 ⇒ *примечание*:*

[Следует отличать постоянство / временность сущности, обозначаемой sc-элементом и постоянство / временность sc-множества, одним из элементов которого указанный sc-элемент является]

∈ обозначение множества sc-элементов, содержащего sc-элементы, обозначающие как статические, так и динамические sc-множества

∈ статическое sc-множество

⇒ примечание*:

- [Следует отличать статичность / динамичность sc-множества, обозначаемого соответствующим sc-элементом и статичность / динамичность sc-множества, одним из элементов которого указанный выше sc-элемент является]
- [Напомним, что статический характер sc-множества означает отсутствие временных sc-пар принадлежности (временных sc-дуг принадлежности), выходящих из нака этого sc-множества]

∈ sc-класс

:= [Класс всевозможных sc-элементов, обозначающих sc-множества]

:= [Класс обозначающий sc-множеств]

⇒ примечание*:

[Следует отличать разные sc-элементы, являющиеся обозначениями соответствующих sc-множеств, и класс, элементами которого являются всевозможные такие sc-элементы]

Пункт 2.2.2.4. Онтологическая формализация Базовой денотационной семантики SC-кода

Суть онтологической формализации различных областей знаний, различных фрагментов баз знаний интеллектуальных компьютерных систем заключается в следующем:

- Выделяется достаточно большой семантически целостный фрагмент баз знаний, включающий в себя:
 - все элементы некоторого одного ключевого класса рассматриваемых объектов (объектов исследования) или конечного числа таких ключевых классов объектов исследования;
 - все связи между выделенными объектами исследования, соответствующие заданному семейству отношений, параметров и классов структур, которое условно будем называть предметом исследования.
- Указанный семантически целостный фрагмент базы знаний, являющийся чаще всего бесконечной структурой, будем называть **предметной областью**.
- Сама формальная **онтология** представляет собой формальную спецификацию выделенной **предметной области** и включает в себя следующие **частные онтологии**:
 - **структурную спецификацию предметной области**, в которой указываются роли всех ключевых элементов (ключевых знаков), входящих в состав **предметной области**. К числу таких ролей относятся:
 - *максимальный класс объектов исследования'*
 - *немаксимальный класс объектов исследования'*
 - *ключевой объект исследования'*
 - *исследуемый класс связок'*
 - *исследуемый класс классов'*
 - *исследуемый класс структур'*
 - *неисследуемый класс'*

:= [sc-класс, исследуемый в другой (смежной) предметной области]
- **теоретико-множественную онтологию**, в которой описываются теоретико-множественные связи между всеми классами (sc-классами), исследуемыми в рамках заданной (специфицируемой) **предметной области**
- **логическую онтологию**, которая включает в себя
 - определения исследуемых классов (исследуемых понятий);
 - логическую иерархию исследуемых понятий, которая связывает каждое понятие со множеством тех понятий, которые явно используются в определении этого понятия;
 - аксиомы и теоремы, описывающие свойства специфицируемой предметной области;
 - тексты доказательств теорем;
 - логическую иерархию теорем, которая связывает каждую теорему со множеством теорем, на основе которых она доказывается.
- **терминологическую спецификацию предметной области**, в которой указываются sc-идентификаторы всех ключевых sc-элементов специфицируемой **предметной области**, а также приводятся правила построения основных sc-идентификаторов для всех sc-классов (понятий), исследуемых в рамках специфицируемой **предметной области**;
- **диадатическую спецификацию предметной области**, в которой приводится дополнительная информация, предназначенная для того, чтобы пользователи и разработчики (инженеры знаний), которые используют или совершенствуют специфицируемую предметную область и её онтологию, могли быстрее усвоить их особенности (см. § 7.5.5. *Дидактические знания*)
- **проектную спецификацию предметной области и соответствующей ей онтологии**, в которой приводится информация об истории эволюции этой **предметной области и онтологии**, а также о направлениях и планах организации дальнейшего их развития.

Более подробно о предметных областях см. в § 2.5.4. *Формализация понятия предметной области*, а более детальное рассмотрение формальных онтологий, представленных в SC-коде см. в § 2.5.5. *Формализация понятия онтологии*.

Онтологическая формализация базовой денотационной семантики SC-кода трактуется нами как **формальная онтология**, представленная в SC-коде и описывающая детонационную семантику **семантически корректных sc-конструкций**. Указанную **формальную онтологию** будем называть **Базовой денотационной семантикой SC-кода**. Для того, чтобы уточнить **предметную область**, специфицируемую этой **онтологией**, введём следующие понятия:

сионимия sc-элементов

:= [Бинарное ориентированное *отношение эквивалентности*, каждая пара которого связывает два *sc-элемента*, обозначающие одну и ту же сущность*]

⇒ примечание*:

[Синонимия двух *sc-элементов* возможна только в том случае, если эти *sc-элементы хранятся в sc-памяти* (входят в состав *базы знаний*) разных ostis-систем. В рамках каждой *ostis-системы* синонимичные *sc-элементы* совпадают (отождествляются, склеиваются, считаются одним и тем же *sc-элементом*).]

отношение эквивалентности

⇐ ключевое понятие*:

§ 2.4.2. Формальная онтология связок и отношений

sc-память**база знаний ostis-системы****ostis-система**

⇒ разбиение*:

- {• индивидуальная ostis-система
 - коллективная ostis-система
- }

sc-конструкция

⇐ часто используемый sc-идентификатор*:

sc-множество

:= [информационная конструкция, представляющая собой множество sc-элементов]

▷ **sc-текст**

:= [текст SC-кода]

:= [sc-конструкция, являющаяся семантически корректной по отношению к Базовой денотационной семантике SC-кода]

:= [sc-конструкция, удовлетворяющая (соответствующая) правилам Базовой денотационной семантики SC-кода]

:= *часто используемый sc-идентификатор*:*

[SC-код]

∈ *имя собственное*

:= [Класс (Множество всевозможных) sc-текстов]

▷ **sc-знание**

sc-знание

:= [sc-текст, являющийся либо фрагментом (подструктурой) соответствующей *предметной области*, либо *высказыванием*, описывающим некоторое свойство (в частности, некоторую закономерность) этой *предметной области*]

:= [знание, представленное в SC-коде]

:= [*sc-текст*, обладающий истинным значением по отношению к соответствующей *предметной области*]

⊂ **связная sc-конструкция**

⇒ примечание*:

[Разные sc-знания могут противоречить друг другу, то есть отражать разные точки зрения на некоторую предметную область, но любое sc-знание должно быть sc-текстом, то есть не должно противоречить правилам Базовой денотационной семантики SC-кода]

интеграция sc-конструкций*

:= [объединение sc-конструкций*]

:= [объединение sc-множеств*]

⇒ примечание*:

[При интеграции sc-конструкций sc-элементы, обозначающие одну и ту же сущность, то есть синонимичные sc-элементы, считаются одинаковыми (совпадающими, тождественными) и, следовательно, должны склеиваться (отождествляться)]

SC-пространство

- := [Результат интеграции всевозможных sc-конструкций, семантически корректных по отношению к *Базовой денотационной семантике SC-кода*]
- := [Предметная область, специфицируемая (описываемая) *Базовой денотационной семантикой SC-кода*, которая является формальной онтологией, представленной средствами SC-кода]
- := [Результат интеграции всевозможных sc-текстов (текстов SC-кода)]
- := [Максимальный sc-текст]
- := [Текст SC-кода, включающий в себя всевозможные sc-тексты]
- := [Пространство sc-конструкций, семантически корректных по отношению к *Базовой денотационной семантике SC-кода*]
- ⇒ примечание*:
 - [Особенностью *SC-пространство* является то, что оно включает в себя и формальную онтологию, опи- сывающую его свойства]
 - [очевидно, что *SC-пространство* является бесконечным sc-текстом, то есть текстом, содержащим бес- конечное количество sc-элементов. В частности, в состав *SC-пространства* входят все sc-элементы, являющиеся элементами всех sc-множеств, знаки которых входят в состав *SC-пространства*]
 - [*SC-пространство* является "вместилищем" семантически корректных (по отношению к *Базовой дено- тационной семантике SC-кода*) частей баз знаний всевозможных ostis-систем и, в том числе, глобальной (объединенной) Базы знаний Экосистемы OSTIS. Подчеркнём при этом, что Экосистема OSTIS являет- ся примером распределённых иерархических ostis-систем]
 - [Тот факт, что корректная (с точки зрения *Базовой денотационной семантике SC-кода*) часть базы знаний каждой ostis-системы входит в состав *SC-пространства*, позволяет трактовать описание соотношения между текущим состоянием *базы знаний ostis-системы* и *Sc-пространством* как описание того, что указанная ostis-система в текущий момент времени не знает. Например, *ostis-система* в некоторый момент времени может не знать (1) всех элементов некоторого конкретного конечного sc-множества (конечно sc-конструкции), (2) количества элементов указанного конечного sc-множества, (3) какому подклассу заданного sc-класса принадлежит указанный элемент этого sc-класса и так далее]
 - [В памяти ostis-системы каждый sc-элемент считается в рамках этой памяти временной сущностью (имеется в виду сам sc-элемент, а не обозначаемая им сущность), поскольку он появляется в *памяти ostis-системы* и удаляется из неё независимо от того, что он обозначает. В отличие от этого в *SC-пространстве* все sc-элементы считаются постоянными (постоянно присутствующими) в рамках этого пространства]

Базовая денотационная семантика SC-кода

- := [Онтология Базовой денотационной семантики SC-кода]
- := [Формальная онтология, представленная в SC-коде и являющаяся материнской онтологией (онтологией са- мого высокого уровня) для всех остальных формальных онтологий, представленных в SC-коде]
- := [Онтология SC-пространства]
- := [Описание (представление) системы *правил построения семантически корректируемых sc-конструкций*, удовлетворяющих требованиям Базовой денотационной семантики SC-кода]
- ∈ sc-онтология
 - := [формальная онтология, представленная в SC-коде]

В состав *Базовой денотационной семантики SC-кода* включается:

- Приведенный выше текст *Пункта 2.2.2.1. Семантическая классификация sc-элементов по базовым призна- кам*
- Приведенный выше текст *Пункта 2.2.2.2. Уточнение смысла выделенных классов sc-элементов и связей между этими классами*
- Средства базовой семантической спецификации sc-элементов, рассмотренные в *Пункте 2.2.2.3. Структура базовой семантической спецификации sc-элемента*

Логическая онтология SC-пространства

- ⇐ логическая онтология*:
 - Базовая денотационная семантика SC-пространства
- ⇒ примечание*:
 - [Приведём пример правил, входящих в состав данной логической онтологии:
 - Вторыми компонентами sc-пар константной парой принадлежности могут быть sc-элементы любого типа(в том числе, и sc-переменные), но первыми компонентами таких sc-пар могут быть только константные sc-множества

- Знак *sc-situation* связан с элементами этой ситуации sc-парами константной постоянной позитивной принадлежности. То есть позитивная принадлежность считается постоянной в рамках интервала времени существования соответствующей ситуации. В этом смысле ситуацию можно считать квазистатической
- Знак атомарной логической формулы связан со всеми элементами этой формулы sc-парами константной постоянной позитивной принадлежности, в том числе, и с теми элементами атомарной формулы, которые являются sc-переменными
- Из переменного sc-множества могут выходить только переменные sc-пары принадлежности
- Не существует sc-пар принадлежности выходящих из обозначений внешних сущностей и sc-пар и другие

]

§ 2.2.3. Синтаксис SC-кода

⇒ подраздел*:

- *Пункт 2.2.3.1. Синтаксические особенности SC-кода*
- *Пункт 2.2.3.2. Синтаксическое Ядро SC-кода*
- *Пункт 2.2.3.3. Уточнение понятия синтаксически корректной sc-конструкции*
- *Пункт 2.2.3.4. Синтаксические расширения Ядра SC-кода*

Пункт 2.2.3.1. Синтаксические особенности SC-кода

Алфавит SC-кода — семейство синтаксических меток, приписываемых sc-элементам и указывающих факт принадлежности sc-элемента к соответствующему классу sc-элементов (sc-классу)

Минимальный алфавит SC-кода

⇒ *пояснение**:

[Если нам известен смысл выделяемых классов sc-элементов (sc-классов), каждый из которых в sc-памяти представлен константным sc-элементом, обозначающим этот sc-класс, то для "прочтения" и понимания sc-конструкций, хранимых в sc-памяти, достаточно синтаксически выделить только Класс константных постоянных позитивных sc-пар принадлежности (Класс базовых sc-дуг), с помощью которых каждый sc-элемент будет явно соединяться с sc-элементами, обозначающими те sc-классы, которым этот sc-элемент принадлежит. Очевидно, что таким явным способом выделить базовые sc-дуги с помощью самих этих базовых sc-дуг невозможно.]

Таким образом, любой класс sc-элементов можно выделить явно путём:

- введения sc-элемента, являющегося знаком этого класса sc-элементов (sc-класса);
- проведение постоянных позитивных sc-пар принадлежности во все sc-элементы, являющиеся элементами выделяемого sc-класса и хранимые (присутствующие) в текущем состоянии sc-памяти.

Минимальный алфавит SC-кода

⇒ *примечание**:

[Таким образом, Минимальным алфавитом SC-кода является Класс базовых sc-дуг и Класс всех остальных sc-элементов (по умолчанию)]

Тем не менее, если учитывать особенности обработки в *sc-памяти* разных классов *sc-элементов*, целесообразно сделать расширение Минимального алфавита SC-кода и, соответственно, ввести понятие *Синтаксического Ядра SC-кода*.

Пункт 2.2.3.2. Синтаксическое Ядро SC-кода

Синтаксическая структура линейных информационных конструкций (строк символов) задаётся:

- алфавитом используемых символов (элементарных, атомарных фрагментов информационной конструкции, каковыми, в частности, являются буквы), то есть семейством таких попарно непересекающихся классов синтаксически эквивалентных символов, для которых существует простая процедура, позволяющая для любого символа про его синтаксическим особенностям установить факт его принадлежности одному из указанных классов;
- бинарным ориентированным отношением, определяющим непосредственный порядок (последовательность) символов в строках символов.

Аналогично этому, синтаксическая структура sc-конструкций задаётся:

- семейством классов синтаксически эквивалентных sc-элементов, в каждый из которых входят sc-элементы с одинаковыми синтаксическими характеристиками или, условно говоря, с одинаковыми синтаксическими метриками;
- двумя бинарными ориентированными отношениями инцидентности sc-элементами, заданными на множестве всех sc-элементов:
 - Отношением инцидентности обозначений ы-пар с их компонентами
 - Отношением инцидентности обозначений ориентированных sc-пар с их вторыми компонентами
⇒ примечание*:
[Данное отношение является подмножеством Отношения инцидентности обозначений sc-пар с их компонентами]

*Отношение инцидентности обозначений sc-пар с их компонентами**

:= часто используемый sc-идентификатор*:

[пара инцидентности обозначения sc-пары с её компонентом*]

∈ имя нарицательное

⇒ примечание*:

- [Каждая пара, принадлежащая данному отношению семантически трактуется как *обозначение sc-пары принадлежности*, но синтаксически оформляется не в виде самостоятельного sc-элемента, а в виде бинарной ориентированной связи между sc-элементами, что аналогично бинарным ориентированным связям, описывающим последовательность символов в строке символов. Заметим при этом, что конфигурация sc-конструкций в отличие от строк символов не является линейной. Заметим также, что уточнение семантической интерпретации пар инцидентности sc-элементов полностью определяется семантической типологией первых компонентов этих пар инцидентности, то есть семантической типологией *обозначений sc-par*, являющихся первым и компонентами рассматриваемых пар инцидентности:
 - если указанное *обозначение sc-пары* является *sc-константой*, то соответствующая пара инцидентности трактуется как пара константной принадлежности;
 - если указанное *обозначение sc-пары* является *sc-переменной*, то соответствующая пара инцидентности трактуется как пара переменной принадлежности;
 - если указанное *обозначение sc-пары* является *обозначением постоянной сущности*, то соответствующая пара инцидентности трактуется как пара постоянной принадлежности;
 - если указанное *обозначение sc-пары* является *обозначением временной сущности*, то соответствующая пара инцидентности трактуется как пара временной принадлежности
-]
- Подчеркнём, что первыми компонентами пар инцидентности sc-элементов всегда являются *обозначения sc-par*, но вторыми компонентами пар инцидентности sc-элементов могут быть sc-элементы любого типа (в том числе, и обозначения sc-пар)]

пара инцидентности

⇒ пояснение*:

[Каждая sc-пара (константная пара sc-элементов), каждая переменная sc-пара и каждое обозначение sc-пары связывается со своими элементами не явно вводимыми константными или переменными sc-парами позитивной принадлежности, а реализуемыми на "физическом" уровне связями (парами) инцидентности. Таким образом пары инцидентности — это специальным образом синтаксически выделенные константные или переменные sc-пары позитивной принадлежности, связывающие обозначения sc-пар с элементами этих пар. Соответственно этому синтаксические особенности имеют и все обозначения sc-пар, поскольку только из них могут выходить ориентированные пары инцидентности. Поэтому с синтаксической точки зрения обозначения sc-пар будем называть *sc-коннекторами*, обозначения неориентированных sc-пар — *sc-ребрами*, а обозначения ориентированных sc-пар — *sc-дугами*. При этом из класса пар инцидентности выделим под-

класс пар, связывающих обозначения sc-дуг с теми sc-элементами, вкоторые эти дуги входят. Такую пару инцидентности будем называть **парой инцидентности входящей sc-дуги.**]

Какие *семантически выделенные классы sc-элементов* следует также рассматривать, как и *семантически выделенные классы sc-элементов*:

- **обозначение неориентированной sc-пары**
 ⇒ **примечание*:**
 [Каждый sc-элемент, принадлежащий этому классу является первым компонентом двух пар принадлежности обозначений sc-пар с их компонентами (двух пар, принадлежащих Отношению инцидентности) обозначений sc-пар с их компонентами]
- **обозначение ориентированной sc-пары не являющейся знаком постоянной позитивной sc-пары принадлежности**
 ⇒ **примечание*:**
 [Каждый sc-элемент, принадлежащий этому классу, является:
 - первым компонентом одной пары инцидентности обозначения sc-пары с её компонентом;
 - underlinепервым компонентом одной пары инцидентности обозначения ориентированной sc-пары с её вторым компонентом]
 • **постоянная позитивная sc-пара принадлежности**
 ⇒ **примечание*:**
 [Каждый элемент этого класса, как и любое другое обозначение ориентированной sc-пары, является первым компонентом пары инцидентности обозначения sc-пары с её компонентом, а также первым компонентом пары инцидентности обозначения ориентированной sc-пары с её вторым компонентом]
- **файл**
 := [знак файла]
 ⇒ **примечание*:**
 [Для sc-элементов этого класса необходимо на "синтаксическом" уровне обеспечить возможность связи этого sc-элемента с обозначаемым им файлом, хранимым в файловой памяти этой же ostis-системы]
- **класс всех остальных sc-элементов, не попавших в перечисленные выше синтаксически выделенные классы**
 ⇒ **примечание*:**
 [Данный класс включает в себя:
 - обозначение sc-синглетона;
 - обозначение sc-связки, не являющейся синглетоном или парой;
 - обозначение sc-класса;
 - обозначение sc-структуры;
 - обозначение внешней сущности, не являющейся файлом.]
]

Какие классы sc-элементов требуют специального синтаксического оформления:

- **неориентированные sc-пары** (замена принадлежности на инцидентность)
- **ориентированные sc-пары** (замена принадлежности на инцидентность с дополнительным указанием направленности)
- **константная постоянная позитивная sc-пара принадлежности**
- **константная временная позитивная sc-пара принадлежности**
- **константный внутренний файл**
 ⇒ **примечание*:**
 [Для sc-элементов данного класса необходимо обеспечить явную их связь с обозначаемыми ими файлами, хранимыми в памяти той же ostis-системы]

sc-ребро

- := [Класс sc-элементов, имеющих в рамках Ядра SC-кода синтаксическую метку обозначений неориентированной sc-пары]
- := [Синтаксическая метка обозначения неориентированной sc-пары, используемая в рамках Ядра SC-кода]
- ∈ *синтаксически выделяемый sc-класс в рамках Ядра SC-кода*

sc-дуга общего вида

базовая sc-дуга

sc-узел, являющийся знаком файла***sc-узел, не являющийся знаком файла***

\coloneqq [sc-узел общего вида]

Перечисленное семейство синтаксически выделяемых классов sc-элементов составляет *Алфавит Ядра SC-кода*.

Алфавит Ядра SC-кода

$= \{ \bullet \text{ sc-ребро}$
 $\quad \bullet \text{ sc-дуга общего вида}$
 $\quad \bullet \text{ базовая sc-дуга}$
 $\quad \bullet \text{ sc-узел, являющийся знаком файла}$
 $\quad \bullet \text{ sc-узел, не являющийся знаком файла}$
 $\quad \coloneqq \text{ [sc-узел общего вида]}$
 $\}$

Приведём синтаксическую классификацию sc-элементов

sc-элемент

\Rightarrow разбиение*:
 $\{ \bullet \text{ sc-коннектор}$
 \Rightarrow разбиение*:
 $\quad \{ \bullet \text{ sc-дуга}$
 \Rightarrow разбиение*:
 $\quad \{ \bullet \text{ базовая sc-дуга}$
 $\quad \bullet \text{ sc-дуга общего вида}$
 $\quad \}$
 $\quad \bullet \text{ sc-ребро}$
 $\quad \}$
 $\bullet \text{ sc-узел}$
 \Rightarrow разбиение*:
 $\quad \{ \bullet \text{ sc-узел, являющийся знаком файла}$
 $\quad \bullet \text{ sc-узел, не являющийся знаком файла}$
 $\quad \}$
 $\}$

sc-коннектор

\coloneqq [sc-элемент, являющийся обозначением sc-пары и имеющий синтаксическую метку обозначения sc-пары]
 \coloneqq [sc-элемент, являющийся обозначением sc-пары и связанный с элементами обозначаемой им sc-пары не sc-парами, обозначающими принадлежность, то есть не с помощью связующих sc-элементов, а с помощью синтаксически реализуемых связей инцидентности]
 \coloneqq [sc-элемент, имеющий синтаксическую метку, семантически эквивалентную sc-элементу, который является обозначением класса всех константных и переменных sc-пар (пар sc-элементов)]

sc-дуга***sc-узел******синтаксически выделяемый класс sc-элементов***

\coloneqq [класс sc-элементов, имеющих общий (одинаковый) синтаксический признак, который будем называть синтаксической меткой, каждая из которых семантически эквивалентна (синонимична) соответствующему sc-элементу, обозначающему некоторый класс sc-элементов (sc-класс)]
 \Rightarrow примечание*:

[Наличие у двух разных sc-элементов одной и той же синтаксической метки означает то, что оба эти sc-элемента принадлежат sc-классу, знаком которого является sc-элемент, семантически эквивалентный указанной метке]

\coloneqq пояснение*:

[sc-элемент, обозначающий sc-класс, принадлежность которому может быть представлена либо с помощью sc-пары позитивной принадлежности, либо с помощью соответствующей метки, приписываемой этому sc-элементу]

⇒ *примечание**:

[Приписывание sc-элементам меток ускоряет проверку принадлежности, только такие метки и надо вводить, этих элементов соответствующих классам]

синтаксически выделяемый sc-класс

:= [sc-класс, каждому sc-элементу которого приписывается соответствующая этому sc-классу синтаксическая метка, которая является неявной (синтаксической) формой указания факта принадлежности указанного sc-элемента указанному sc-классу]

⇒ *разбиение**:

- {• *синтаксически выделяемый sc-класс в рамках Ядра SC-кода*
- *синтаксически выделяемый sc-класс в рамках расширения Ядра SC-кода*

}

⇒ *примечание**:

[Каждый синтаксически выделяемый sc-класс можно считать элементом *Алфавита SC-кода*. Но, в отличие от других языков, синтаксически выделяемые sc-классы могут пересекаться]

Особенностью привычных нам языков является то, что каждый элементарный (атомарный) фрагмент информационных конструкций может иметь только одну метку, то есть может быть элементом только одного синтаксически выделяемого класса элементарных фрагментов. Семейство таких синтаксически выделяемых классов элементарных фрагментов информационных конструкций (например, букв) называют алфавитом соответствующего языка.

синтаксически выделяемый sc-класс в рамках Ядра SC-кода

:= [синтаксически выделяемый в рамках Ядра SC-кода класс sc-элементов]

:= [синтаксическая метка, приписываемая sc-элементам в рамках Ядра SC-кода]

:= [синтаксическая метка sc-элементов, выделяющая в рамках Ядра SC-кода соответствующий класс синтаксически эквивалентных sc-элементов]

:= [класс синтаксически эквивалентных sc-элементов в рамках Ядра SC-кода]

:= [синтаксический тип sc-элементов, выделяемый в рамках Ядра SC-кода]

⇒ *примечание**:

[В различных синтаксических расширениях Ядра SC-кода синтаксически выделяемые sc-классы могут пересекаться. То есть sc-элемент может принадлежать сразу несколькими синтаксически выделяемым sc-классом.]

При этом формулирование, семейства синтаксически выделяемых sc-классов (то есть семейства синтаксических меток, приписываемых sc-элементам) может осуществляться путём синтаксической классификации sc-элементов по различным признакам. Желательно при этом, чтобы такая синтаксическая классификация sc-элементов была согласована с семантической классификацией sc-элементов, которая рассмотрена в [Пункт 2.2.2.1. Семантическая классификация sc-элементов по базовым признакам](#). Другими словами каждый синтаксически выделяемый класс sc-элементов (каждая синтаксическая метка) должен иметь чёткую семантическую интерпретацию, то есть должен быть одновременно и семантически выделяемым sc-классом.

*следует отличать**

- ∃ {• *синтаксически выделяемый sc-класс в рамках Ядра SC-кода*
- *синтаксически выделяемый sc-класс в рамках какого-либо расширения Ядра SC-кода*
- *sc-класс*
 - := [Класс sc-элементов, выделяемый (задаваемый) явно с помощью sc-конструкции, состоящей (1) из sc-элемента, являющего знаком этого класса и (2) из константных постоянных позитивных sc-пар принадлежности, соединяющих указанный знак выделяемого класса sc-элементов со всеми sc-элементами, принадлежащими этому классу и хранимыми в текущем состоянии sc-памяти]
 - *денотационную семантику каждого sc-элемента, то есть соотношение между sc-элементом и тем, что он обозначает (его денотатом) и соответствующую семантическую классификацию всего множества sc-элементов*
 - *семантический тип каждого sc-элемента, то есть синтаксическую метку (значение синтаксического признака-параметра), приписываемую каждому sc-элементу и соответствующую синтаксическую классификацию всего множества sc-элементов (Алфавит sc-элементов))*

Пункт 2.2.3.3. Уточнение понятия синтаксически корректной sc-конструкции

Синтаксис SC-кода

- := [Онтология синтаксиса SC-кода]
- := [Описание правил построения *синтаксически корректных sc-конструкций*]
- := [Описание требований, предъявляемых к *синтаксически корректных sc-конструкциями*]
- ∈ *sc-онтология*

sc-множество

- := часто используемый *sc-идентификатор**:
[*sc-конструкция*]
- := [множество *sc-элементов*, которые могут быть (но не обязательно) связаны между собой бинарными ориентированными связями *инцидентности*, каждая из которых связывает некоторый *sc-конструктор* с *sc-элементами*, которые связываются этим *sc-коннектором*]
- := [информационная конструкция, каждый элемент (атомарный фрагмент) которой входит в состав некоторого текста, принадлежащего SC-коду, но при этом *конфигурация* всей указанной информационной конструкции не всегда позволяет считать текстом SC-кода, удовлетворяющим целому ряду синтаксических и семантических требований]
- ⇒ разбиение*:
 {• *синтаксически корректная sc-конструкция*
 • *синтаксически некорректная sc-конструкция*
 }

синтаксически корректная sc-конструкция

- := [синтаксически правильно построенная sc-конструкция]

правило построения синтаксически корректных sc-конструкций

- := [синтаксическое правило SC-кода]
- := [требование (одно из требований), предъявляемое к *синтаксически корректным sc-конструкциям*]

Синтаксис SC-кода

- Каждая sc-пара принадлежности, связывающая sc-элемент, обозначающий пару sc-элементов, с компонентом этой пары (то есть с sc-элементом, связываемым этой sc-парой с другими sc-элементом) синтаксически "преобразуется" из *sc-элемента*, обозначающего пару принадлежности в *пару инцидентности*, которая синтаксически уже не является sc-элементом
- Поскольку для каждого обозначения sc-пары осуществляется *синтаксическая замена* sc-пар принадлежности их элементов на пары инцидентности этих элементов соответствующие синтаксическое преобразование происходит и с самими обозначениями sc-пар — они "превращаются" в sc-коннекторы. Соответственно этому обозначения неориентированных sc-пар "преобразуется" в sc-ребра обозначения ориентированных sc-пар — в sc-дуги.

синтаксически некорректная sc-конструкция

- := [*sc-конструкция*, содержащая одну или несколько синтаксических ошибок]
- ▷ *минимальная синтаксически некорректная sc-конструкция*
 - := [*sc-конструкция*, не содержащая подструктуру, являющуюся синтаксически некорректными *sc-конструкциями*]
 - ⇒ примечание*:
[Каждой *минимальной синтаксически некорректной sc-конструкции* ставится в соответствие одно из синтаксических правил SC-кода, которому указанная *sc-конструкция* противоречит]
- ⇒ примечание*:
[Строго говоря, *синтаксически некорректные sc-конструкции* не являются *sc-текстами*, то есть информационными конструкциями, принадлежащими SC-коду]
- ⇐ *невключение**:
sc-текст
 - := [*sc-конструкция* принадлежащая SC-коду]

Пункт 2.2.3.4. Синтаксические расширения Ядра SC-кода

Способ кодирования *sc-конструкций* в различных вариантах реализации *sc-памяти* может быть различным. То есть каждому варианту реализации *sc-памяти* может соответствовать своя синтаксическая модификация *SC-кода*. При этом оно может касаться не только способа представления меток *sc-элементов*, но и (кодирования) *отношений инцидентности sc-элементов*. В любом случае каждая такая модификация должна быть чётко описана.

Зачет нужно расширить Алфавит *sc-элементов*.

- Для того, чтобы ускорить определение семантического типа каждого *sc-элемента* в ходе обработки *sc- конструкций*
- Чтобы быстро установить то, чего не хватает *базовой спецификации sc-элемента*
При этом уточнить содержание базовой *sc-спецификации sc-элемента* (что необходимо о нём знать, чтобы с ним эффективно работать). В частности, необходимо знать то, что о нём не известно.
Но при этом число меток, приписываемых *sc-элементу* должно быть минимизировано, то есть эти метки должны быть информационными (!). Для этого не надо бояться *расширения Алфавита sc-элементов*

Поскольку *SC-код* является языком внутреннего представления информации в *sc-памяти ostis-системы*, *Синтаксис SC-кода* является уточнением формы такого представления и, как следствие уточнением того, как устроена *sc-память ostis-систем*. Поскольку хранимая в *sc-памяти* информационная конструкция представляет собой множество *sc-элементов*, можно ввести понятие **ячейки sc-памяти**, каждая из которых обеспечивает хранение одного из *sc-элементов*.

ячейка sc-памяти

⇒ *пояснение**:

[Фрагмент *sc-памяти*, в котором может храниться один *sc-элемент* и который должен содержать

- набор синтаксических меток, приписываемых хранимому *sc-элементу*;
- уникальный (взаимнооднозначный) идентификатор хранимого *sc-элемента* (аналог адреса ячейки в адресной памяти)
- связи хранимого *sc-элемента* со смежными *sc-элементами* (связи инцидентности)
- ссылка на хранимый файл, если хранимый *sc-элемент* является знаком файла, хранимого в файловой памяти этой же индивидуальной *ostis-системы*

]

Формы представления меток *sc-элементов* могут быть разными:

- приписывание *sc-идентификатора* того *sc-класса* которому принадлежит данный *sc-элемент*
- формирование вектора признаков в некотором пространстве признаков (Каждому признаку ставится в соответствии свое поле, в которое записывается соответствующее значение признака — в качестве этого значения тоже можно использовать *sc-идентификатор* соответствующие значения этого признака)

§ 2.2.4. Файлы ostis-системы

§ 2.2.5. Смыслоное пространство ostis-систем

⇒ *ключевое понятие**:

- *sc-отношение*
- *бинарное sc-отношение*
- *слотовое sc-отношение*
- *sc-структура**
- *элементарно представленный элемент**
- *полносвязно представленный элемент**
- *полностью представленный элемент**
- *sc-связка**
- *sc-отношение**
- *sc-класс**
- *сущностное замыкание**
- *содержательное замыкание**
- *sc-отношение сходства по слотовым отношениям**
- *sc-отношение семантического сходства по слотовым отношениям**
- *связная sc-структура**
- *семантическое сходство sc-структур**
- *семантическое непрерывное сходство sc-структур**
- *ключевой запрос**
- *минимальный ключевой запрос**
- *полная семантическая окрестность элемента**
- *интроспективный ключевой элемент**
- *топологическое пространство*
- *топологическое пространство замыкания инцидентности коннекторов*
- *топологическое пространство синтаксического замыкания*
- *топологическое пространство сущностного замыкания*
- *топологическое пространство содержательного замыкания*
- *метрика*
- *семантическая метрика*
- *метрическое пространство*
- *метрическое конечное синтаксическое пространство*
- *метрическое конечное семантическое пространство*
- *псевдометрика*
- *псевдометрическое пространство*
- *псевдометрическое конечное семантическое пространство*

Понятие SC-пространства наряду с понятием SC-кода является необходимым для уточнения и формализации понятия смысла информационных конструкций и в унификации смыслового представления информации. В SC-пространстве можно выделять структуры, связанные как с синтаксическими свойствами текстов SC-кода, так и с его семантикой. В последнем случае речь можно вести о «смыслоном пространстве». Смысл информационной конструкции, в конечном счёте, определяется (1) внутренними связями всех элементарных фрагментов этой конструкции и (2) её внешними связями с элементами смыслового пространства (её положением в контексте). Смыслоное пространство является результатом естественного становления знаний в процессе их интеграции. Важнейшим достоинством SC-пространства является возможность уточнения таких понятий, как понятие аналогичности (сходства и отличия) различных описываемых «внешних» сущностей, аналогичности унифицированных семантических сетей (текстов SC-кода), понятие семантической близости описываемых сущностей (в том числе, и текстов SC-кода).

Следует отметить, что в силу абстрактности языков модели унифицированного семантического представления знаний и условности выбора меток элементов их текстов, нельзя исключить, что объединение двух произвольных текстов таких языков не будет текстом языка модели унифицированного семантического представления знаний. Чтобы избежать результатов подобных экспериментальных объединений с точки зрения синтаксиса или семантики, для абстрактных языков следует рассматривать множество «смыслоных пространств». Однако, для конкретных (реальных) языков может оказаться достаточным рассмотрение одного «смыслоового пространства». Далее рассмотрим:

- возможность перехода от sc-текстов к графовым структурам и от них к топологическому пространству;
- возможность перехода от sc-текстов к графовым структурам и от них к многообразию (топологическому пространству);
- возможность перехода от sc-текстов к графовым структурам и от них к метрическому пространству.

Чтобы исследовать структурные свойства SC-пространства, можно использовать уже разработанные модели пространств и связь их известными топологическими моделями например такими как графы. При этом изначально не будем принимать в расчёт динамические особенности, связанные с обработкой знаний, однако позже будет показано, что учёт динамики в процессах обработки и при становлении знаний является необходимым для вычисления семантической метрики, являющейся одним из определяющих признаков знаний. Обратимся к исследованию структурно-топологических свойств пространства.

Структурно-топологические свойства могут свидетельствовать о синтаксических или семантических зависимостях обозначений текстов языка, позволяющих упростить работу со структурами за счёт перехода к более простым структурам на уровнях управления данными или знаниями в характерных задачах управления для библиотек компонентов. На множестве элементов, образующих SC-пространство, можно изучать топологические свойства и рассматривать SC-пространство как топологическое пространство. Следует заметить, что, несмотря на то, что SC-код ориентирован на смысловое представление знаний, в силу наличия НЕ-факторов, не все смыслы могут быть представлены в некоторый момент времени и не будет известна структура соответствующего представления. Поэтому структурно-топологические свойства текстов языка представления знаний скорее определяют синтаксическое пространство, нежели семантическое (смысловое). Хотя оба могут приближаться друг к другу по мере устранения неопределённостей, вызванных НЕ-факторами.

Рассмотрим следующие виды топологических пространств:

- топологическое пространство замыкания инцидентности коннекторов;
- топологическое пространство синтаксического замыкания;
- топологическое пространство сущностного замыкания;
- топологическое пространство содержательного замыкания.

топологическое пространство

⇒ *пояснение**:

[Топологическое пространство – множество с определённым над ним множеством (семейством) (открытых) подмножеств, включая само множество и пустое множество. Для любого счётного подмножества семейства результат объединения принадлежит семейству, а для любого подмножества семейства результат пересечения также принадлежит семейству. Дополнения множеств семейства до наибольшего из множеств называются замкнутыми множествами.]

Чтобы рассмотреть более детально некоторые виды топологических пространств введём следующие понятия.

sc-отношение

⇒ *пояснение**:

[sc-отношение – sc-множество sc-связок.]

бинарное sc-отношение

⇒ *пояснение**:

[Бинарное sc-отношение – sc-множество sc-пар.]

слотовое sc-отношение

⇒ *пояснение**:

[Слотовое sc-отношение – sc-множество (ориентированных) sc-пар, которые не являются узловыми sc-парами.]

sc-структура*

⇒ *пояснение**:

[sc-структура* – sc-множество, в котором есть непустое sc-подмножество-носитель (множество первичных элементов sc-структуры*).]

элементарно представленный элемент'

⇒ *пояснение**:

[Элементарно представленный элемент' – элемент sc-структуры*, sc-множество, все элементы которого являются элементами sc-структуры*.]

полносвязно представленный элемент'

⇒ *пояснение**:

[Полносвязно представленный элемент' – элемент sc-структуры*, sc-множество, все элементы и все принадлежности которому являются элементами sc-структуры*, или sc-дуга, являющаяся элементарно представленным элементом' этой sc-структуры*.]

полностью представленный элемент'

⇒ *пояснение**:

[Полностью представленный элемент' – полносвязно представленный элемент' sc-структуры*, с любым элементом, не являющимся sc-дугой, выходящей из него, связанный принадлежащей этой sc-структуре* sc-дугой принадлежности или sc-дугой непринадлежности.]

sc-связка'

⇒ *пояснение**:

[sc-связка' – полносвязно представленный элемент' sc-структуры*, принадлежащий sc-отношению' этой sc-структуре*, являющийся sc-связкой.]

sc-отношение'

⇒ *пояснение**:

[sc-отношение' – полносвязно представленный элемент' sc-структуры*, sc-отношение, все элементы которого являются sc-связками' этой sc-структуре*].

sc-класс'

⇒ *пояснение**:

[sc-класс' – полносвязно представленный элемент' sc-структуры*, все элементы которого являются элементами sc-структуре*, не являющийся ни sc-отношением', ни sc-связкой' этой sc-структуре*].

сущностное замыкание*

⇒ *пояснение**:

[Сущностное замыкание* – наименьшая надмножество* (структура*), в котором каждый элемент является элементарно представленным']

◦ ;

содержательное замыкание*

⇒ *пояснение**:

[Содержательное замыкание* – наименьшее надмножество* (структура*), в котором каждый элемент является полносвязно представленным']

sc-отношение сходства по слотовым отношениям*

⇒ *пояснение**:

[sc-отношение сходства по слотовым sc-отношениям* – sc-отношение, являющееся рефлексивным по этим слотовым отношениям, т.е. для любого элемента, входящего в связку этого sc-отношения под одним из слотовых sc-отношений, найдётся связка этого sc-отношения, в которую он входит под каждым из этих слотовых sc-отношений.]

sc-отношение семантического сходства по слотовым отношениям*

⇒ *пояснение**:

[sc-отношение семантического сходства по слотовым отношениям* – sc-отношение сходства по слотовым sc-отношениям* si и sj, в котором каждый элемент под слотовым sc-отношением si, может быть преобразован к элементу синтаксического типа элемента под слотовым sc-отношением sj; два инцидентных sc-элемента под слотовым sc-отношением si, в рамках этого sc-отношения семантического сходства соответствуют инцидентным элементам соответственно под слотовым sc-отношением sj.]

связная sc-структура*

⇒ *пояснение**:

[Связная sc-структура* – sc-структура*, являющаяся связной.]

семантическое сходство sc-структур*

\coloneqq [семантическое подобие sc-структур*]

\Rightarrow пояснение*:

[Семантическое сходство sc-структур* – связывает sc-множество sc-структур* с sc-структурой* sc-отношением семантического сходства по слотовым sc-отношениям s_i, s_j так, что для каждой sc-структуры* из sc-множества найдётся её элемент и связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением s_i , а под слотовым sc-отношением s_j входит элемент sc-структуры*, также для каждого элемента sc-структуры найдётся связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением s_j , а под слотовым sc-отношением s_i входит элемент sc-структуры* из sc-множества.]

семантическое непрерывное сходство sc-структур*

\coloneqq [семантическое непрерывное подобие sc-структур*]

\Rightarrow пояснение*:

[Семантическое непрерывное сходство sc-структур* – связывает sc-множество sc-структур* со связной sc-структурой* sc-отношением семантического сходства по слотовым sc-отношениям s_i, s_j так, что для каждой sc-структуры* из sc-множества найдётся её элемент и связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением s_i , а под слотовым sc-отношением s_j входит элемент связной sc-структуры*, также для каждого элемента связной sc-структуры найдётся связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением s_j , а под слотовым sc-отношением s_i входит элемент sc-структуры* из sc-множества.]

ключевой запрос'

\Rightarrow пояснение*:

[Ключевой запрос' – поисковый-проверочный запрос (от одного известного элемента), который выполняется хотя бы от одного элемента и не выполняется хотя бы от одного элемента.]

минимальный ключевой запрос'

\Rightarrow пояснение*:

[Минимальный ключевой запрос' – ключевой запрос, который находит sc-подмножества множеств элементов, находимых всеми другими ключевыми запросами, которые имеют те же области известных элементов выполнимости и невыполнимости.]

полная семантическая окрестность элемента*

\Rightarrow пояснение*:

[Полная семантическая окрестность элемента* – все элементы, находимые выполнимыми минимальными ключевыми запросами от этого элемента (с учётом дизъюнктивного поиска и отрицания поиска).]

интроспективный ключевой элемент'

\Rightarrow пояснение*:

[Интроспективный (базовый) ключевой элемент' – элемент множества (из класса наименьших таких множеств) элементов такого, что любая полная семантическая окрестность любого элемента является sc-подмножеством объединения их полных семантических окрестностей]

топологическое пространство замыкания инцидентности коннекторов

\Rightarrow пояснение*:

[Топологическое пространство замыкания инцидентности коннекторов на множестве sc-элементов – топологическое пространство, все замкнутые множества которого содержат все sc-элементы этого множества, до которых есть маршрут по ориентированным связкам отношения инцидентности коннекторов.]

\Rightarrow комментарий*:

[В общем случае не удовлетворяет аксиоме отделимости по Тихонову. Прагматика рассмотрения таких пространств обуславливается операциями удаления sc-элементов и коннекторов, которым они инцидентны. Удаление sc-элемента требует удаления всех коннекторов, замыканию любой открытой окрестности которых он принадлежит.]

топологическое пространство синтаксического замыкания

\Rightarrow пояснение*:

[Топологическое пространство синтаксического замыкания на множестве sc-элементов – топологическое пространство, все замкнутые множества которого содержат все sc-элементы этого множества, до которых есть маршрут по ориентированным связкам отношения инцидентности.]

⇒ **комментарий*:**

[В общем случае не удовлетворяет аксиоме отделимости по Колмогорову. В качестве основы замкнутых множеств топологического пространства можно выделить синтаксическое замыкание, однако в силу возможности проведения дуг из любого sc-узла в любой в итоговом случае (в итоге процесса устранения НЕ-факторов) такое пространство является тривиальным (антидискретным) пространством. Отношение объединения топологических пространств синтаксического замыкания алгебраически не замкнуто на множестве топологических пространств синтаксического замыкания. По той же причине для любого неполного топологического пространства синтаксического замыкания, носитель которого является надмножеством носителя первого и которое не сохраняет замкнутые множества. В этом смысле топология на основе синтаксического замыкания не является устойчивой относительно процессов становления знаний и её рассмотрение pragматически не оправдывается. Топология полного же топологического пространства синтаксического замыкания антидискретна (тривиальна). Таким образом, у полного топологического пространства синтаксического замыкания все топологические подпространства синтаксического замыкания обладают антидискретной (тривиальной) топологией.]

топологическое пространство сущностного замыкания

⇒ **пояснение*:**

[Топологическое пространство сущностного замыкания на множестве sc-элементов – топологическое пространство, все замкнутые множества которого являются сущностными замыканиями.]

⇒ **комментарий*:**

[В общем случае не удовлетворяет аксиоме отделимости по Тихонову. В качестве носителя топологического (под)пространства можно выделить сущностное замыкание. Топологическое пространство сущностного замыкания сохраняет замкнутые множества любых топологических пространств сущностного замыкания, носитель которых является подмножеством его носителя и сущностным замыканием. Такие пространства образуют структуру топологических подпространств-топологических надпространств сущностного замыкания. Топология пространств в этой структуре разнообразна.]

топологическое пространство содержательного замыкания

⇒ **пояснение*:**

[Топологическое пространство содержательного замыкания на множестве sc-элементов – топологическое пространство, все замкнутые множества которого являются содержательными замыканиями.]

⇒ **комментарий*:**

[В общем случае не удовлетворяет аксиоме отделимости по Тихонову. В качестве носителя топологического (под)пространства можно выделить содержательное замыкание. Топологическое пространство содержательного замыкания сохраняет замкнутые множества любых топологических пространств содержательного замыкания, носитель которых является подмножеством его носителя и содержательным замыканием. Такие пространства образуют структуру топологических подпространств-топологических надпространств содержательного замыкания. Топология пространств в этой структуре разнообразна.]

Возможен переход от sc-структур к многообразиям и топологическим пространствам путём сведения sc-структур к графовым структурам, подробно вопросы сведения sc-структур к графовым структурам и далее – к многообразиям и топологическим пространствам рассмотрены в работе [].

Для более сложных структур таких, как полная семантическая окрестность, топологические свойства подлежат дальнейшему изучению.

Далее можно рассмотреть метрические пространства, в частности – конечные подпространства с полностью представленными sc-элементами.

метрика

⇒ **пояснение*:**

[Метрика – функция двух аргументов, принимающая значения на (линейно) упорядоченном носителе группы, неотрицательна, равна нейтральному элмененту (нулю) только при равенстве аргументов, симметрична, удовлетворяет неравенству треугольника.]

метрическое пространство

⇒ **пояснение*:**

[Метрическое пространство – множество, с определённой на нём функцией двух аргументов, являющейся метрикой, принимающей значения на упорядоченном носителе группы.]

семантическая метрика

:= [семантическая близость]

⇒ пояснение*:

[Семантическая метрика – метрика, определённая на знаках и выражающая количественно близость их значений.]

метрическое конечное синтаксическое пространство

⇒ пояснение*:

[Метрическое конечное синтаксическое пространство SC-кода – метрическое пространство с конечным носителем, элементами которого являются обозначения (sc-элементы), а значение метрики может быть определено через отношения инцидентности элементов без учёта их семантического типа.]

метрическое конечное семантическое пространство

⇒ пояснение*:

[Метрическое конечное семантическое пространство SC-кода – метрическое пространство с конечным носителем, элементами которого являются обозначения (sc-элементы), а значение метрики не может быть определено через отношения инцидентности элементов без учёта их семантического типа.]

псевдометрика

⇒ пояснение*:

[Псевдометрика – функция двух аргументов, принимающая значения на (линейно) упорядоченном носителе группы, неотрицательна, симметрична, удовлетворяет неравенству треугольника.]

псевдометрическое пространство

⇒ пояснение*:

[Псевдометрическое пространство – множество, с определённой на нём функцией двух аргументов, являющейся псевдометрикой, принимающей значения на упорядоченном носителе группы.]

псевдометрическое конечное семантическое пространство

⇒ пояснение*:

[Псевдометрическое конечное семантическое пространство SC-кода – псевдометрическое пространство с конечным носителем, элементами которого являются обозначения (sc-элементы), а значение псевдометрики не может быть определено через отношения инцидентности элементов без учёта их семантического типа.]

В силу неполноты выразительных средств для представления изменяющихся со временем знаний, отсутствия определённой пространственно-временной модели, наличия семантически неопределённых или слабоопределённых обозначений в текстах да и наличия недоопределённости самих текстов описанного в предыдущих разделах языка, на данном этапе в этом описании затруднительно предложить какую-либо модель метрического пространства для более сложных структур, учитывающих НЕ-факторы, связанные с пространством-временем. Это станет возможным при проявлении желания идти навстречу, готовности к конвергенции, интероперабельности и после достижения консенсуса, достаточного для соответствующего описания развития предлагаемого стандарта и языка.

Тем не менее, некоторые такие модели были успешно предложены в работе []. Предложенные модели полагались на представление, способное выразить семантику переменных обозначений и операционную семантику расширенными средствами алфавита. Для построения подобных моделей, кроме расширенных средств алфавита, предлагается полагаться на модели, описывающие процессы интеграции и становления знаний [], на средства спецификации знаний [], ориентированные на рассмотрение финитных структур, что позволяет перейти к рассмотрению сложных метрических соотношений в рамках метамодели смыслового пространства.

В современных работах в технических науках [bManin], возможно, наиболее близкими понятиями являются понятия, выражающие смысл термина «семантическое пространство» (интериорный подход (Табл. 1)). Общим во многих подходах к работе с «семантическим пространством» является рассмотрение словоформ или лексем (множеств словоформ) и их признаков (Табл. 1). В литературе [bManin] встречаются следующие подходы (Табл. 2):

- подход на основе семантических осей и пространства признаков (бинарных $\{0, 1\}^n$, монополярных $[0; 1]^n$, биполярных $[-1; 1]^n$);
- подход на основе семантических осей и нейронного кодирования места в поле смыслов (слова и словосочетания имеют области (подмножества) значений, связываясь другими частями речи как включением и пересечением, тексты соответствуют путям связанных областей, бинарное кодирование групп нейронов, распознавающих смыслы);

- подход на основе модели «смысл-текст» [bMeaningText] (отражение неполноты семантических шкал и анализ синтагм и поверхностно-синтаксической структуры);
- нейролингвистические данные отражают процессы синтеза и восприятия речи в нейронных сетях (сеть лексического синтеза), близка к модели «смысл-текст»;
- модели, построенные на основе статического анализа (корпусов) текстов (модель векторного пространства).

Статистический подход к обработке естественного языка противопоставляется интуиции и коммуникативному опыту учёных [bManin].

В основе подхода лежит семантическая статическая гипотеза, что смысл слов (лексем) определяется контекстом использования (его статистическим образом) в языке (с коммуникативной структурой) [bManin].

Модель векторного пространства семантики [bManin]. Модель рассматривается для двух случаев: большого словаря ($N \leq M$) и задачи информационного поиска ($M \leq N$). M – размер словаря, N – количество контекстов.

На основе статистики строится матрица размерности $M \times N$ частот p_{ij} появления лексемы (слова) w_i в документе (контексте, подтексты, которые могут перекрываться) c_j .

В знаменателе – оценки вероятности слова и контекста соответственно.

В случае невырожденной матрицы $r = N$ каждая такая матрица задаёт точку в грассманиане N -мерных подпространств M -мерного пространства ($N \leq M$).

В случае невырожденной матрицы $r = M$ каждая такая матрица задаёт точку в грассманиане M -мерных подпространств N -мерного пространства ($M \leq N$).

Каждый текст – точка в грассманиане [bGrassmannian], соответствующем проективному пространству , относительно одного выделенного контекста. Для всех контекстов получая ориентированную N -ку, в соответствии с порядком контекстов в текстах, можно построить маршрут (путь), соединяя геодезическими соседние точки в N -ке. Для двух текстов и это будут две ломанные, между которыми можно вычислить метрику Фреше [bFrechet], используя метрику Фубини-Штуди [bFubiniStudy] в , для этого следует параметризовать пути и через (,): .

Другой способ задать линейный порядок – это рассмотреть фильтрацию (флаг (флаговое многообразие)) [bFlagManifold] в , заданную расширяющимися контекстами. В итоге для текста получаем точки (флаги) во флаговом многообразии. Для флаговых многообразий тоже можно вычислить метрику Фубини-Штуди [bFubiniStudy].

Этот порядок соответствует временному измерению (процессу коммуникации во времени), что может быть существенным. Другой порядок может быть не зависимым от этого, например алфавитный или порядок в соответствии с законом Ципфа [bZipf], [bZipf2].

Сравнение подходов к построению «семантических пространств»

	семантические оси и пространства признаков	семантические оси и нейронное кодирование признаков	модель «смысл-текст»	нейролингвистическое кодирование	статистическая модель (модель векторного пространства семантики)
определённые семантические оси	+	+	-	-	-
динамическая (вычислительная) декомпозиция	-	+	-	+	-
семантические оси и пространства признаков семантические оси и нейронное кодирование признаков модель смысл-текст нейролингвистическое кодирование статистическая модель (модель векторного пространства семантики) определённые семантические оси + + - - динамическая (вычислительная) декомпозиция - + - + - анализ когнитивных процессов (интроспекция) - + - + - учёт НЕ-факторов (неполнота) - - + + + анализ когнитивных процессов (интроспекция)	-	+	-	+	-
учёт НЕ-факторов (неполнота)	-	-	+	+	+

Вопросы соотнесения смыслов, их формализации, развития языков в пространстве и времени рассмотрены в работах В.В. Мартынова [bMartynov], [bMartynov2], [bGordey].

Заключение к Главе 2.2.

Показано, что SC-код может быть использован в качестве метаязыка для описания собственной денотационной семантики и синтаксиса.

Типология sc-конструкций выглядит следующим образом

sc-множество

- \coloneqq [sc-конструкция]
- \coloneqq [информационная конструкция, принадлежащая SC-коду]
- \supset *sc-структура*
 - \supset *sc-текст*
 - \coloneqq часто используемый *sc-идентификатор**:
[SC-код]
 - \in имя собственное
 - \coloneqq [синтаксически целостная и синтаксически корректная (правильно построенная) информационная конструкция SC-кода]
 - \coloneqq [Класс (Множество всевозможных) sc-текстов]
 - \supset *sc-знание*
 - \coloneqq [семантически целостное и семантически корректный sc-текст, являющийся адекватным фрагментом соответствующей предметной области или её спецификации]

Перечислим аспекты представления знаний в памяти интеллектуальных компьютерных систем, которые требуют особой аккуратности:

- константный и переменный характер денотационной семантики знаков, хранимых в памяти интеллектуальных компьютерных систем;
- динамический характер знаковых конструкций, хранимых в памяти, обусловленный либо выполняемыми в памяти информационными процессами, либо динамическим характером структур внешних объектов, описываемых этими знаковыми конструкциями;
- временный характер существования внешних описываемых объектов и временный характер существования различных конфигураций знаковых конструкций и даже самих знаков, хранимых в памяти;
- неизвестность (отсутствие в памяти) востребованной информации различного вида, неполнота знаний, их недостаточность для решения актуальных задач;
- синтаксическая и семантическая некорректность, неадекватность и, в частности, противоречивость некоторых имеющихся знаний;
- наличие информационного мусора (излишеств) в имеющихся знаниях.

Глава 2.3.

Семейство внешних языков *ostis*-систем, близких языку внутреннего смыслового представления знаний

Садовский М.Е.

Голенков В.В.

Жмырко А.В.

⇒ *аннотация**:

[В главе рассматриваются понятия внешних и внутренних языков интеллектуальных компьютерных систем нового поколения. Описываются внешние языки представления знаний в рамках Технологии OSTIS, а именно SCg-код, SCs-код, SCn-код. Для каждого из внешних языков детально рассматривается его синтаксис и денотационная семантика.]

⇒ *подраздел**:

- § 2.3.1. Внешние идентификаторы sc-элементов – знаков, входящих в конструкции SC-кода
- § 2.3.2. Язык внешнего графического представления конструкций SC-кода – SCg-код (*Semantic Code graphical*)
- § 2.3.3. Язык внешнего линейного представления конструкций SC-кода – SCs-код (*Semantic Code string*)
- § 2.3.4. Язык внешнего форматированного представления конструкций SC-кода – SCn-код (*Semantic Code natural*)

⇒ *ключевое понятие**:

- sc-идентификатор
- SCg-код
- SCs-код
- SCn-код

⇒ *ключевое знание**:

- Правила построения sc-идентификаторов
- Синтаксис и денотационная семантика SCg-кода
- Синтаксис и денотационная семантика SCs-кода
- Синтаксис SCn-кода

§ 2.3.1. Внешние идентификаторы sc-элементов – знаков, входящих в конструкции SC-кода

Внешние идентификаторы *sc-элементов* (или, сокращенно *sc-идентификаторы*) необходимы *ostis-системе* исключительно для того, чтобы осуществлять обмен информацией с другими *ostis-системами* или со своими пользователями. Для того чтобы представить свою *базу знаний*, решать самые различные *задачи*, связанные с анализом текущего состояния и эволюцией своей *базы знаний*, задачи, связанные с анализом текущего состояния (текущих ситуаций) окружающей среды, принятием соответствующих решений (целей) и организацией соответствующих *действий*, направленных на выполнение принятых решений (на достижение поставленных целей), *ostis-системе* не нужны никакие внешние идентификаторы (в частности, имена) соответствующие *sc-элементам*. Но для понимания сообщений, принимаемых от других субъектов (что для *ostis-системы* означает построение *sc-текста*, *семантически эквивалентного* принятому сообщению) и для анализа сообщений, передаваемых другим субъектам (что для *ostis-системы* означает синтез *внешнего текста*, *семантически эквивалентного* заданному *sc-тексту* и удовлетворяющего некоторым дополнительным требованиям, например, эмоционального характера) *ostis-системе* необходимо знать, как в принимаемом или передаваемом сообщении изображаются (представляются) знаки, синонимичные *sc-элементам*, которые уже хранятся или могут храниться в составе базы знаний *ostis-системы*. В качестве внешних идентификаторов *sc-элементов* чаще всего используются имена (термины) соответствующих (обозначаемых) сущностей, представленные отдельными словами или словосочетаниями на различных естественных языках, но также могут использоваться иероглифы, условные обозначения, пиктограммы.

В общем случае *sc-элементу* может соответствовать несколько синонимичных ему имен на разных *естественных языках*. Более того, *sc-элементу* может соответствовать несколько синонимичных ему имен на одном и том же *естественном языке*. В этом случае одно из этих имен объявляется как основной внешний идентификатор для

соответствующего *sc-элемента* и соответствующего *естественного языка*. Основное требование, предъявляемое к таким внешним идентификаторам это отсутствие как синонимов, так и омонимов в рамках множества основных внешних идентификаторов *sc-элементов* для каждого естественного языка.

Каждый внешний идентификатор *sc-элемента*, используемый *ostis*-системой, может быть описан (представлен) в её памяти в виде *внутреннего файла ostis-системы*, т.е. в виде электронного образа всевозможных вхождений данного внешнего идентификатора во всевозможные внешние тексты соответствующего внешнего языка. В некоторых случаях явное представление в памяти не требуется, например, в случае *нетранслируемых sc-идентификаторов*.

Каждому *sc-элементу* может соответствовать целое семейство внешних идентификаторов этого *sc-элемента*, которые обычно являются терминами, именующими обозначаемую сущность. Среди этих внешних идентификаторов для каждого идентифицируемого *sc-элемента* выделяется один как основной идентификатор. А неосновные термины (имена), соответствующие этим *sc-элементам* (в том числе и классам), поясняют денотационную семантику указанного *sc-элемента*.

sc-идентификатор

- := [строка символов или пиктограмма, взаимно однозначно представляющая соответствующий *sc-элемент*, хранимый в *sc-памяти*]
- := [*внешний идентификатор sc-элемента*]
- ⇒ *разбиение**:
 - {• *простой sc-идентификатор*
 - := [*простой внешний идентификатор sc-элемента*]
 - *sc-выражение*
 - := [*сложный внешний идентификатор sc-элемента*, в состав которого входит один или несколько идентификаторов других *sc-элементов*]
- }
- ⇒ *разбиение**:
 - {• *основной sc-идентификатор*
 - := [*основной sc-идентификатор для носителей дополнительно указываемого языка общения* (например, соответствующего естественного языка)]
 - ⇒ *примечание**:
 - [*основной sc-идентификатор* является уникальным (не имеет синонимов и омонимов) в рамках соответствующего естественного языка]
 - ▷ *основной международный sc-идентификатор*
 - *неосновной sc-идентификатор*
 - ▷ (*неосновной sc-идентификатор* ∩ *пояснение*)
 - := [*неосновной sc-идентификатор, являющийся одновременно и пояснением обозначаемой сущности*]
 - ▷ (*неосновной sc-идентификатор* ∩ *определение*)
 - := [*неосновной sc-идентификатор, являющийся одновременно и определением обозначаемого понятия*]
 - ▷ *неосновной часто используемый sc-идентификатор*
- }

В качестве *основных sc-идентификаторов* могут использоваться также общепринятые международные условные обозначения некоторых сущностей, например, обозначения часто используемых функций (*sin*, *cos*, *tg*, *log*, и т.д.), единиц измерения, денежных единиц и многое другое. Формально каждый основной международный *sc-идентификатор* считается основным *sc-идентификатором* также и для каждого естественного языка, несмотря на то, что символы, используемые в основных международных *sc-идентификаторах*, могут не соответствовать алфавиту некоторых или даже всех естественных языков.

С помощью неосновных *sc-идентификаторов* указываются возможные *синонимы** соответствующего *основного sc-идентификатора*, которые в частности, могут пояснить или даже определять обозначаемую сущность, указывает на важные свойства этой сущности.

Для некоторых *sc-элементов* могут часто использоваться не только основные, но и неосновные *sc-идентификаторы* (особенно в неформальных текстах – в пояснениях, примечаниях и т.п.). Явное выделение такого класса *sc-идентификаторов* позволяет упростить семантический анализ исходных текстов баз знаний.

Системный *sc-идентификатор* – это *sc-идентификатор*, являющийся уникальным в рамках всей базы знаний Экосистемы *OSTIS* (Глобальной базы знаний). Данный *sc-идентификатор*, как правило, используется в исходных текстах базы знаний, при обмене сообщениями между *ostis*-системами, а также для взаимодействия *ostis*-системы с компонентами, реализованными с использованием средств, внешних с точки зрения Технологии *OSTIS*, например, программ, написанных на традиционных языках программирования. Алфавит системных *sc-идентификаторов* максимально упрощен для того, чтобы обеспечить удобство автоматической обработки таких *sc-идентификаторов*.

с использованием современных технических средств, в частности, запрещены пробелы и различные специальные символы.

основной sc-идентификатор

С файл-образец *ostis*-системы

↔ семантическая эквивалентность*:

[Все основные идентификаторы sc-элементов в памяти *ostis*-системы оформляются в виде копируемых фаллов-образцов *ostis*-системы.]

⇒ *пояснение**:

[Копии основных sc-идентификаторов входят в состав внешних текстов различных языков (SCg-кода, SCs-кода, SCn-кода), а также в различных падежах, склонения, спряжениях в состав файлов *ostis*-систем.]

Аналогичное утверждение справедливо и для неосновных часто используемых sc-идентификаторов. Все остальные неосновные sc-идентификаторы считаются вспомогательными файлами-экземплярами.

sc-идентификатор

⇒ разбиение*:

{• строковый sc-идентификатор

:= [sc-идентификатор, представленный строкой символов, которая является именем обозначаемой сущности]

:= [имя сущности, обозначаемой идентифицируемым sc-элементом]

:= [имя (термин, словосочетание), синонимичное соответствующему (идентифицируемому) sc-элементу и представленное в соответствующем алфавите символов]

• нестроковый sc-идентификатор

⇒ *пояснение**:

[В общем случае в качестве *sc-идентификатора* некоторого *sc-элемента* может выступать произвольный *внутренний файл ostis-системы*, например, пиктограмма, условное обозначение или даже аудиофрагмент.]

}

⇒ примечание*:

[Введенные нами sc-идентификаторы используются во всех внешних языках, близких SC-коду – в SCg-коде, в SCs-коде и в SCn-коде.]

строковый sc-идентификатор

:= [имя, приписываемое идентифицируемому sc-элементу]

:= [имя сущности, обозначаемой идентифицируемым sc-элементом]

:= [строка символов, синонимичная соответствующему идентифицируемому sc-элементу]

▷ основной строковый sc-идентификатор

:= [уникальное для каждого естественного языка внешнее имя, приписываемое идентифицируемому sc-элементу]

▷ основной русскоязычный sc-идентификатор

▷ системный sc-идентификатор

▷ основной англоязычный sc-идентификатор

▷ основной германоязычный sc-идентификатор

▷ основной франкоязычный sc-идентификатор

▷ основной италоязычный sc-идентификатор

▷ основной китайскоязычный sc-идентификатор

▷ системный sc-идентификатор

Символами, использующимися в *системном sc-идентификаторе*, могут быть буквы латинского алфавита, цифры, знак нижнего подчеркивания и знак тире. Для обеспечения интернационализации рекомендуется формировать *системные sc-идентификаторы* на основании основных англоязычных *sc-идентификаторов*. Таким образом, наиболее целесообразно формировать *системный sc-идентификатор sc-элемента* из основного англоязычного путем замены всех символов, не входящих в описанный выше алфавит на символ нижнее подчеркивание (“_”). Кроме того, заглавные буквы чаще всего заменяются на соответствующие строчные.

Для именования sc-элементов, являющихся знаками *ролевых отношений*, вместо знака “/” в *системном sc-идентификаторе* используется приставка “*trel*” и далее после нижнего подчеркивания записывается имя *ролевого отношения*.

Для именования sc-элементов, являющихся знаками *неролевых отношений*, вместо знака “*” в *системном sc-идентификаторе* используется приставка “*nrel*” и далее после нижнего подчеркивания записывается имя *неролевого отношения*.

Для именования sc-элементов, являющихся знаками классов *понятий* в *системном sc-идентификаторе* используется приставка “concept” и далее после нижнего подчеркивания записывается имя *класса*.

Для именования sc-элементов, являющихся знаками *структур* в *системном sc-идентификаторе* используется приставка “struct” и далее после нижнего подчеркивания записывается имя *структурь*.

нетранслируемый sc-идентификатор

С *системный sc-идентификатор*

:= [sc-идентификатор, не представляемый в базе знаний *ostis*-системы]

:= [sc-идентификатор, существующий только вне базы знаний *ostis*-системы]

Нетранслируемые sc-идентификаторы используются только в рамках исходных текстов *баз знаний* (в том числе, *sc.s-текстов*) и при обмене сообщениями между *ostis-системами* в тех случаях, когда необходимо в нескольких фрагментах исходного текста *базы знаний* или передаваемого сообщения использовать имя одного и того же *sc-элемента*, но при этом указанный *sc-элемент* не имеет *системного sc-идентификатора* и вводить его нецелесообразно. Использование *нетранслируемых sc-идентификаторов* позволяет повысить читабельность и структурированность исходных текстов *баз знаний*, а также позволяет обратиться к одному и тому же неименуемому (в рамках базы знаний) *sc-элементу* в разных файлах исходных текстов *баз знаний* или в разных сообщениях, передаваемых между *ostis-системами*. В качестве таких *sc-элементов* часто выступают знаки *структур* и *связок*.

Таким образом, *нетранслируемые sc-идентификаторы* существуют только вне *базы знаний ostis-системы* и при формировании базы знаний из исходных текстов или при погружении в базу знаний полученного сообщения соответствующий им *внутренний файл ostis-системы* не создается.

Нетранслируемые sc-идентификаторы строятся по тем же принципам, что и системные sc-идентификаторы, но в начале *нетранслируемого sc-идентификатора* ставится одна или несколько точек (“.”), количество которых определяет область видимости данного *нетранслируемого sc-идентификатора*.

Глобальные нетранслируемые sc-идентификаторы считаются уникальными в рамках всего имеющегося набора исходных текстов *баз знаний* и/или передаваемых между *ostis-системами* сообщений. Таким образом, *sc-элементы*, имеющие на уровне исходных текстов (в том числе, в разных файлах исходных текстов) одинаковые *глобальные нетранслируемые sc-идентификаторы*, считаются синонимичными и в *базе знаний ostis-системы* представляются одним и тем же *sc-элементом*.

Можно сказать, что областью видимости *глобальных нетранслируемых sc-идентификаторов* является весь набор (репозиторий) исходных текстов некоторой базы знаний.

В начале *глобальных нетранслируемых sc-идентификаторов* ставится одна точка.

Локальные нетранслируемые sc-идентификаторы считаются уникальными в рамках *конкретного файла* исходных текстов *баз знаний* и/или передаваемого между *ostis-системами* сообщения. Таким образом, *sc-элементы*, имеющие в рамках одного файла исходных текстов одинаковые *локальные нетранслируемые sc-идентификаторы*, считаются *синонимичными*, в то время как *sc-элементы*, имеющие в рамках *разных файлов* исходных текстов одинаковые *локальные нетранслируемые sc-идентификаторы* будут считаться *разными sc-элементами*.

Можно сказать, что областью видимости *локальных нетранслируемых sc-идентификаторов* является конкретный файл исходных текстов базы знаний.

В начале *глобальных нетранслируемых sc-идентификаторов* ставится две точки.

Уникальные нетранслируемые sc-идентификаторы используются для однократного обозначения конкретного неименуемого *sc-элемента* в рамках исходных текстов *баз знаний* и/или передаваемых между *ostis-системами* сообщений.

Кроме того, *уникальные нетранслируемые sc-идентификаторы* могут использоваться при визуализации баз знаний в виде, например, *sc.s-текстов* или *sc.p-текстов*, в тех случаях, когда необходимо визуализировать *sc-элемент*, не имеющий *sc-идентификатора*.

Каждое вхождение *уникального нетранслируемого sc-идентификатора* в исходный текст *базы знаний* или передаваемое сообщение считается обозначением *разных sc-элементов* (чаще всего, *sc-узлов*).

sc-идентификатор*

▷ *основной sc-идентификатор**

:= [Бинарное ориентированное отношение, каждая пара которого связывает *sc-элемент* с внутренним файлом *ostis-системы*, который содержит *основной sc-идентификатор* указанного *sc-элемента*.]

⇒ *примечание*:*

[Отношение, связывающее *sc-элементы* с файлами, содержащими их *основные sc-идентификаторы*, само имеет свой *основной sc-идентификатор*, который представляет собой строку символов, имеющую

вид “основной sc-идентификатор*”. Заметим, что в состав первого домена отношения “*основной sc-идентификатор**” входит также и *sc-узел*, обозначающий само это отношение.]

▷ *системный sc-идентификатор**

:= [Бинарное ориентированное отношение, каждая пара которого связывает *sc-элемент* с внутренним файлом *ostis-системы*, который содержит *системный sc-идентификатор* указанного *sc-элемента*.]

Системные идентификаторы отличаются от основных, во-первых, требованием к уникальности в рамках всей базы знаний Экосистемы *OSTIS* (а, значит, независимостью от внешнего языка), а во-вторых, более простым алфавитом, удобным для автоматической обработки.

Системные sc-идентификаторы и нетранслируемые sc-идентификаторы выполняют схожие функции, связанные с именованием *sc-элементов* на уровне исходных текстов баз знаний или передаваемых между *ostis-системами* сообщений.

Каждый *системный sc-идентификатор* представляется в базе знаний в виде *внутреннего файла ostis-системы* и связан с соответствующим *sc-элементом* парой отношения *системный sc-идентификатор**. *нетранслируемые sc-идентификаторы* не представляются в рамках *базы знаний*, не имеют соответствующих *внутренних файлов ostis-системы* и на уровне *базы знаний* никак не связаны с идентифицируемыми ими *sc-элементами*.

Пункт 2.3.1.1. Понятие простого идентификатора sc-элемента

Простой *sc-идентификатор* – идентификатор *sc-элемента*, в состав которого идентификаторы других *sc-элементов* не входят и который не содержит *транслируемую* в *SC-код* информацию об обозначаемой им сущности.

Простой строковый *sc-идентификатор* – простой *sc-идентификатор*, представляющий собой строку (цепочку) символов, которая является именем (названием) той же сущности, что и идентифицируемый *sc-элемент*. Простые строковые *sc-идентификаторы* являются наиболее распространенным видом идентификаторов, присыпываемых *sc-элементам*.

простой строковый sc-идентификатор

▷ *системный sc-идентификатор*

▷ *простой строковый идентификатор sc-переменной*

 Э *пример'*:

 [*_var1*]

▷ *простой строковый sc-идентификатор неролевого отношения*

:= [*простой строковый идентификатор sc-узла*, являющегося знаком неролевого отношения]

 Э *пример'*:

 [включение множеств*]

▷ *простой строковый sc-идентификатор ролевого отношения*

:= [*простой строковый идентификатор sc-узла*, являющегося знаком ролевого отношения]

 Э *пример'*:

 [слагаемое']

▷ *простой строковый sc-идентификатор класса классов*

:= [*простой строковый идентификатор sc-узла*, являющегося знаком класса классов]

 Э *пример'*:

 [длина^]

▷ *sc-идентификатор внешнего файла ostis-системы*

:= [*URL-идентификатор*]

 Э *пример'*:

 ["file:///home/user/image1.png"]

 ⇒ *примечание**:

 [Данный sc-идентификатор описывает абсолютный путь к файлу под названием "image1.png"]

 Э *пример'*:

 ["file://image1.png"]

 ⇒ *примечание**:

 [Данный sc-идентификатор описывает относительный путь к файлу под названием "image1.png"]

 Э *пример'*:

 ["https://conf.ostis.net/content/image1.png"]

 ⇒ *примечание**:

 [Данный sc-идентификатор описывает путь к файлу под названием "image1.png", расположенному на удаленном сервере.]

sc-идентификаторы внешних файлов *ostis*-систем предназначены для описания местоположения внешних файлов *ostis*-систем и представляют собой строку символов, которая строится в соответствии со стандартом URL, а затем берется в двойные кавычки. Кавычки нужны для однозначности определения того, где начинается и заканчивается данный *sc-идентификатор*, поскольку в общем случае в URL разрешены пробелы. Целесообразность этого обусловлена тем, что *sc-идентификаторы* данного типа часто используются в файлах исходных текстов баз знаний *ostis*-систем.

sc-идентификаторы внешних файлов *ostis*-систем с точки зрения Технологии *OSTIS* являются простыми строковыми *sc-идентификаторами*, хотя и могут содержать специальные символы, например "%" или "/". Это связано с тем, что указанные идентификаторы не несут в себе семантически значимой информации о свойствах самого *sc-элемента*, обозначенного таким *sc-идентификатором*, а только информацию о его расположении в текущем состоянии внешнего мира *ostis*-системы.

Имя нарицательное – имя, которое либо не является обозначением какого-либо класса сущностей, либо является обозначением (именем) некоторого класса сущностей, но построенным без использования нарицательного имени этого класса, либо является именем некоторого класса сущностей, построенным с использованием нарицательного имени этого класса либо путем преобразования имени нарицательного во множественное число, либо путем дополнительного использования в начале формулируемого имени таких слов или словосочетаний, как "Знак класса", "Класс", "Знак множества", "Множество", "Знак множества всевозможных", "Множество всевозможных". *Имя собственное* всегда начинается с большой буквы.

Имя нарицательное – имя некоторого класса сущностей (а, точнее, имя класса *sc-элементов*, обозначающих сущности некоторого класса), которое, с одной стороны, является знаком всего указанного класса, а, с другой стороны, соответствует (может быть приписано) любому экземпляру этого класса. *Имя нарицательное* всегда начинается с маленькой буквы.

Правила построения простых строковых *sc-идентификаторов* включают в себя:

- Алфавит символов, используемых в простых строковых *sc-идентификаторах*;
- Префиксы и суффиксы, используемые в простых строковых *sc-идентификаторах*;
- Разделители и ограничители, используемые в простых строковых *sc-идентификаторах*;
- Правила построения *имен собственных* и *имен нарицательных*, являющихся простыми строковыми *sc-идентификаторами*;
- Правила построения простых строковых *sc-идентификаторов*, определяемые различными классами идентифицируемых *sc-элементов*.

Общим правилом построения *простых sc-идентификаторов* является стремление максимально возможным образом использовать сложившуюся терминологию. Но при этом следует подчеркнуть, что необходимость исключения омонимии в *sc-идентификаторах* требует строгого формального уточнения семантической интерпретации каждого используемого термина. Особо подчеркнем то, что в *ostis-системах* процесс построения новых терминов (*sc-идентификаторов*) и процесс совершенствования существующей терминологии по отношению к процессу эволюции *баз знаний*, представленных в *SC-коде*, с технической точки зрения абсолютно не зависят друг от друга. Кроме того, следует помнить, что далеко не все sc-элементы, входящие в состав базы знаний *ostis*-системы, должны иметь соответствующие им *sc-идентификаторы* (быть идентифицированными). Существенно подчеркнуть то, что далеко не все *sc-элементы* должны иметь *простые sc-идентификаторы* по той простой причине, что для многих *sc-элементов* в случае необходимости можно достаточно легко построить идентифицирующее их *sc-выражение* (*sc-выражение*). Тем не менее, для целого ряда сущностей (например, для понятий, исторических событий и т.п.) обойтись без простых *sc-идентификаторов* очень сложно. Очевидно, что идентифицированными (именованными) должны быть все используемые понятия, вводимые в соответствующих предметных областях и специфицируемые соответствующими онтологиями. Идентифицированными также должны быть обладающие особыми свойствами ключевые экземпляры (элементы) некоторых понятий, различные социально значимые объекты (персоны, населенные пункты, географические объекты, страны, организации, библиографические источники, исторические события и многое другое).

Первым символом каждого *простого строкового sc-идентификатора*, идентифицирующего *sc-переменную* (переменный *sc-элемент*), является знак подчеркивания. При этом, если после указанного знака подчеркивания указывается *имя нарицательное* некоторого класса *sc-элементов*, то рассматриваемый *простой строковый sc-идентификатор* становится уже *sc-выражением*, содержащим информацию о том, что *областью возможных значений** идентифицируемой *sc-переменной* является указанный *класс sc-элементов*.

Последним символом простого *sc-идентификатора*, идентифицирующего *sc-узел*, обозначающий неролевое отношение, заданное на множестве *sc-элементов*, является надстрочная звездочка.

Последним символом простого *sc-идентификатора*, идентифицирующего *sc-узел*, обозначающий заданное на множестве *sc-элементов* ролевое отношение (т.е. отношение, являющееся подмножеством отношения принадлежности), является *штрих* (прямой апостроф).

Последним символом простого sc-идентификатора, идентифицирующего sc-узел, обозначающий понятие, являющееся классом классов (таковыми в частности, являются различные параметры – длина, площадь, объем, масса) является символ *циркумфлекс* ("крышка"). Однако, если отсутствует омонимичный идентификатор без этого суффикса, то указанный символ можно не приписывать.

Простой строковый sc-идентификатор может рассматриваться как последовательное сокращение sc-идентификаторов одного и того же sc-элемента с преобразованием имен собственных в имена нарицательные и наоборот.

При наличии синонимичных имен собственных и имен нарицательных при выборе *основного sc-идентификатора* преимущество имеют имена нарицательные. Так, например, основным идентификатором sc-узла, обозначающего SC-код, является термин "sc-текст", а термин "SC-код", являющийся именем собственным, считается *неосновным часто используемым sc-идентификатором*. Подчеркнем при этом, что имя нарицательное – это всегда имя некоторого класса сущностей (в частности, понятия). В конце этого имени может быть указан либо признак класса классов, либо признак неролевого отношения (класса связок), либо признак ролевого отношения (подмножества отношения принадлежности), либо не указано ничего. Последнее означает, что именуется класс, не являющийся ни классом классов, ни классом связок. А это, в свою очередь, означает, что именуемым классом является либо класс первичных (терминальных) сущностей, либо класс структур.

Одно и тоже словосочетание, которому приписываются разные дополнительные признаки, может быть основой для внешних идентификаторов разных sc-элементов.

В рамках SC-кода целесообразно вводить правила унифицированного построения *простых sc-идентификаторов* и целого ряда других классов идентифицируемых сущностей – персон, библиографических источников (публикаций), разделов баз знаний ostis-систем, файлов ostis-систем, самих ostis-систем.

Пункт 2.3.1.2. Понятие сложного идентификатора sc-элемента

sc-выражение – идентификатор, который не только обозначает соответствующую сущность, но также содержит информацию, представляющую собой по возможности однозначную спецификацию указанной сущности.

Однозначную спецификацию сущности, которая является понятием, называют определением этого понятия

sc-выражение

- := [имя соответствующей (именуемой) сущности построенное по принципу "та (тот), которая (который) указываемым образом связана с другими указываемыми сущностями"]
- := [выражение, идентифицирующее sc-элемент]
- := [идентификатор sc-элемента, в состав которого входят другие идентификаторы и денотационная семантика которого точно определяется конкретным набором правил построения таких сложных (комплексных) идентификаторов, состоящих из определенным образом связанных между собой других идентификаторов]
- := [сложный идентификатор, состоящий из других идентификаторов]
- := [идентификатор, который представляет собой конструкцию, состоящую из нескольких других идентификаторов, а также из некоторых разделителей и ограничителей, и денотационная семантика которого однозначно задается конфигурацией указанной конструкции]
- := [сложный sc-идентификатор]
- := [сложный (составной) внешний идентификатор sc-элемента]
- := [выражение, идентифицирующее sc-элемент]
- := [sc-идентификатор, в состав которого входит один или несколько простых sc-идентификаторов]

sc-выражение разбивается на:

- *sc-выражение неориентированного множества* – sc-выражение, ограниченное фигурными скобками;
- *sc-выражение структуры* – sc-выражение, обозначающее структуру, представленную на любом известном и легко определяемом языке (Русском, Английском, SCg-коде, SCs-коде, SCn-коде). *sc-выражение структуры* обозначает структуру, содержащую sc-текст, семантически эквивалентный тому тексту (на некотором известном языке), который заключен в фигурные скобки. Чаще всего такой текст записывается на формальном языке, например, SCs-коде, и может быть автоматически однозначно интерпретирован. Возможна ситуация, когда указанный текст записан на менее неформальном языке, например, Русском, но в этом случае его автоматическая интерпретация значительно усложняется и в общем случае не всегда однозначна. В текущей реализации средств разработки исходных текстов баз знаний в соответствии с более старой версией правил построения sc-выражений вместо фигурных скобок *sc-выражение структуры* ограничивается квадратными скобками со звездочками ("[*" и "*]");
- *sc-выражение ориентированного множества* – sc-выражение кортежа, ограничивающее угловыми скобками и обозначающее упорядоченное (ориентированное) множество sc-элементов, порядок которых задается последовательностью перечисляемых их sc-идентификаторов.;

- *sc-выражение внутреннего файла ostis-системы* – *sc-выражение*, обозначающее *внутренний файл ostis-системы*, визуальное представление (изображение) которого ограничивается квадратными скобками. *sc-выражение внутреннего файла ostis-системы* обозначает *внутренний файл ostis-системы*, содержимое которого заключено в квадратные скобки, ограничивающие данное *sc-выражение*. Дополнительная спецификация *внутреннего файла ostis-системы* легко осуществляется с помощью *SC-кода*. Сюда входит *язык*, на котором представлена *информационная конструкция*, являющаяся содержимым *файла*, формат кодирования, *автор** и многое другое.;
- *sc-выражение, обозначающее файл-образец ostis-системы* – *sc-выражение*, ограниченное квадратными скобками с восклицательными знаками.;
- *sc-выражение, построенное на основе бинарного отношения* – *sc-выражение*, в состав которого входят либо (1) *sc-идентификатор*, обозначающий бинарное ориентированное отношение, и (2) в круглых скобках *sc-идентификатор* одного из элементов первого домена указанного бинарного ориентированного отношения, либо (1) *sc-идентификатор*, обозначающий бинарное неориентированное отношение и (2) в круглых скобках *sc-идентификатор* одного из элементов области определения указанного бинарного неориентированного отношения. *sc-выражение*, построенное путём указания некоторого бинарного отношения (обычно функционального) и одного из его аргументов (в круглых скобках).;
- *sc-выражение, построенное на основе алгебраической операции* – *sc-выражение*, ограниченное круглыми скобками и построенное путем указания *sc-идентификаторов*, разделенных знаком алгебраической операции.;
- *sc-выражение, идентифицирующее sc-коннектор* – *sc-выражение*, ограниченное круглыми скобками и идентифицирующее *sc-коннектор*, инцидентный двум указанным *sc-элементам* и имеющий тип, задаваемый путем изображения соответствующего *sc.s-коннектора*. Для упрощения восприятия и обработки *sc-выражений, идентифицирующих sc-коннектор* вводится следующее ограничение: первым и третьим компонентом такого *sc-выражения* может быть только *простой sc-идентификатор*. В рамках *sc.s-текстов* внутри *sc-выражений, идентифицирующих sc-коннектор* допускается также использование *sc.s-модификаторов*.

Использование *sc-выражений* позволяет существенно сократить число "придумываемых" *sc-идентификаторов*, каковыми в конечном счете становятся только простые *sc-идентификаторы*, поскольку, зная то, как связан идентифицируемый *sc-элемент* с теми *sc-элементами*, которые уже имеют *sc-идентификаторы*, во многих случаях можно построить *sc-выражение*, идентифицирующее указанный *sc-элемент*. Кроме того, каждое *sc-выражение*, являясь внешним идентификатором, является также и транслируемым формальным текстом, содержащим некоторую информацию об обозначаемой ею сущности.

Очевидно, что некоторые *sc-выражения* могут интерпретироваться неоднозначно. Например, два одинаковых *sc-выражения, идентифицирующих sc-коннектор*, могут изображать как один и тот же *sc-коннектор*, так и кратные *sc-коннекторы* одного и того же вида, связывающие одни и те же два *sc-элемента*. Аналогичная неоднозначность может возникать при использовании *sc-выражений, построенных на основе бинарного отношения* (*подмножество*(S)*) и ряда других *sc-выражений*.

В то же время, некоторые *sc-выражения* являются однозначными, то есть всегда идентифицируют одну и ту же сущность в любом тексте, в состав которого входят. Например выражение "*sin(x)*" является однозначным (при условии, что "*x*" в разных текстах обозначает одно и то же). Однако, однозначность или неоднозначность *sc-выражений* зависит от их семантики, таким образом установить факт однозначности на уровне внешнего текста достаточно сложно.

Для решения проблемы неоднозначности интерпретации *sc-выражений* такого рода будем считать, что каждое вхождение какого-либо *sc-выражения* в различные тексты (например, *sc.s-тексты*) обозначает разные *sc-элементы*. После трансляции таких текстов в *sc-текст* может оказаться, что некоторые из указанных *sc-выражений* на самом деле обозначали одну и ту же сущность, в этом случае соответствующие *sc-элементы* будут "склеены", но уже на уровне *sc-текста*, а не на уровне внешнего текста.

Следует отметить, что факт совпадения *sc-выражений* в рамках некоторого внешнего текста может являться поводом для анализа идентифицируемых этими *sc-выражениями* *sc-элементов* на возможную синонимичность и явно фиксироваться, например, на этапе погружения внешнего текста в *sc-память*.

ограничитель sc-выражений

- := [ограничитель, используемый в *sc-выражениях*]
- := [ограничитель, ограничивающий *sc-выражения* или их компоненты]
- ⇒ разбиение*:
 - {• левый ограничитель *sc-выражений*
 - := [начальный ограничитель *sc-выражений*]
 - := [открывающий ограничитель *sc-выражений*]
 - правый ограничитель *sc-выражений*
 - := [конечный ограничитель *sc-выражений*]

:= [закрывающий ограничитель sc-выражений]
 }

§ 2.3.2. Язык внешнего графического представления конструкций SC-кода – SCg-код (Semantic Code graphical)

SCg-код

:= [Semantic Code graphical]
 := [Язык визуального (графического) представления баз знаний ostis-систем]
 := [Язык внешнего графического представления конструкций внутреннего языка ostis-систем]
 ∈ *графовый язык*

SCg-код представляет собой способ визуализации *sc-текстов* (информационных конструкций SC-кода) в виде рисунков этих абстрактных конструкций. Подчеркнем, что абстрактная *графовая структура* и её рисунок (графическое изображение) – это не одно и то же даже если они *изоморфны* друг другу. *SCg-код* рассматривается нами как объединение *Ядра SCg-кода*, обеспечивающего изоморфное графическое изображение любого *sc-текста*, а также нескольких направлений расширения этого ядра, обеспечивающих повышение компактности и "читабельности" текстов *SCg-кода* (*sc.g-текстов*). *SC-код* – это рассмотрение множества всевозможных графически представленных (визуализированных) графовых структур как универсального языка представления знаний с соответствующим синтаксисом и семантикой.

Основная цель *SCg-кода* – иметь четкие синтаксические графические признаки изображения *sc.g-элементов*, позволяющие легко выделить и различать такие классы *sc.g-элементов*, как:

- *sc.g-константы* (знаки константных сущностей) и *sc.g-переменные* (изображения переменных, значениями которых являются соответствующие *sc-элементы*);
- *sc.g-переменные*, значениями которых являются *sc-константы*, и *sc.g-переменные*, значениями которых являются *sc-переменные*;
- знаки постоянных (стабильных) сущностей и знаки временных (нестабильных, временно существующих, си туативных) сущностей;
- *sc.g-коннекторы* (знаки бинарных связей) и *sc.g-элементы*, не являющиеся *sc.g-коннекторами*;
- неориентированные *sc.g-коннекторы* (*sc.g-ребра*) и ориентированные (*sc.g-дуги*);
- *sc.g-дуги принадлежности* и *sc.g-дуги*, не являющиеся таковыми;
- *sc.g-дуги позитивной принадлежности*, негативной принадлежности и нечеткой принадлежности.

Пункт 2.3.2.1. Синтаксис SCg-кода

Алфавит Ядра SCg-кода

:= [Алфавит *sc.g-элементов*, графически изображающих *sc-элементы*]
 ⇐ *ключевой знак**:
 Таблица. Алфавит Ядра SCg-кода
 ⇒ *пояснение**:
 [Алфавит Ядра SCg-кода взаимно однозначно соответствует Алфавиту SC-кода. Указанное соответствие представлено в файле "Таблица. Алфавит Ядра SCg-кода".]

В таблице "SCg-текст. Алфавит SCg-кода" приведен перечень элементов Алфавита SCg-кода. Этот перечень оформлен в виде *sc.g-текста* и представляет собой изображение примеров всех введенных видов *sc.g-элементов* (по одному примеру каждого вида). При этом, указанные примеры *sc.g-элементов* разбиты на пять групп (*SCg-текст. Алфавит SCg-кода*). Первая группа (верхняя строка) включает в себя *sc.g-элементы*, для которых константность и постоянство обозначаемых ими сущностей требует дополнительного уточнения. Остальные четыре группы *sc.g-элементов* аналогичны друг другу и включают в себя соответственно:

- знаки константных постоянных сущностей;
- знаки константных временных сущностей;
- изображения *sc-переменных*, значениями которых или значениями значений которых (в случае, если значениями переменных являются переменные) являются знаки константных постоянных сущностей;
- изображения *sc-переменных*, значениями которых или значениями значений которых (в случае, если значениями переменных являются переменные) являются знаки константных временных сущностей.

Особое место в *SCg-коде* занимает изображение sc-элементов, являющихся *обозначениями пар принадлежности**, путём явного использования этого *семантически выделяемого класса sc-элементов*. Данный *sc.g-элемент* используется тогда, когда нам необходимо изобразить *sc-дугу*, о которой известно, что она является *обозначением пары принадлежности**, но неизвестно о какой принадлежности идет речь – о константной или переменной, о постоянной или временной, о позитивной, негативной или нечеткой.

Кроме *sc.g-элементов*, перечисленных в таблице “*SCg-текст. Алфавит SCg-кода*”, в состав Алфавита *SCg-кода* входят также следующие *sc.g-элементы*:

- внешние идентификаторы *sc-элементов*, идентичные (приписываемые) соответствующим *sc.g-элементам*.
- *sc.g-контура*, каждый из которых является знаком некоторого *sc-текста* (структуры, состоящей из *sc-элементов*). Каждый такой *sc-текст* может быть:
 - либо константной постоянной структурой;
 - либо константной временной структурой (ситуацией);
 - либо переменной структурой, значениями которой являются постоянные структуры изоморфной конфигурации;
 - либо переменной структурой, значениями которой являются временные структуры (ситуации) изоморфной конфигурации;
- *sc.g-рамки* увеличенного размера являются ограничителями изображения различных файлов, хранимых в памяти *ostis*-системы;
- *sc.g-шины*, являющиеся обозначениями тех же сущностей, что и инцидентные им *sc.g-элементы*.

Заметим также, что, кроме всех перечисленных элементов Алфавита *SCg-кода*, каждый из которых имеет вполне определенную денотационную семантику, для формализации синтаксиса *SCg-кода* необходимо ввести целый ряд более “мелких” синтаксических объектов, например:

- точек инцидентности *sc.g-коннекторов* с *sc.g-узлами*, с другими *sc.g-коннекторами*, с *sc.g-контурами*, с *sc.g-рамками*;
- точек инцидентности *sc.g-шин*;
- точек излома линейных *sc.g-элементов* (*sc.g-коннекторов*, *sc.g-контуров*, *sc.g-рамок*, *sc.g-шин*).

SCg-текст. Алфавит SCg-кода

 обозначение пары принадлежности*				
•	•	—→	•	
○	—→	○	—→	
◎	—→	◎	—→	
⊗	—→	⊗	—→	
⊕	—→	⊕	—→	
⊖	—→	⊖	—→	
⊙	—→	⊙	—→	
●				
◐				
□ ◇	• = = = = •	□ ◇	• :: :: :: :: •	
	• = • = • = • = •		• :: • :: • :: • :: • :: •	
▣ ◇	• = = = = ➤ •	▣ ◇	• :: = = = = ➤ •	
	• = • = • = • = • ➤ •		• :: • = • = • = • ➤ •	
☒ ◇	• - - - ➤ •	☒ ◇	• ➤ •	
	• - - - - - ➤ •		• ➤ •	
田 ◇	• - - - - - ➤ •	田 ◇	• ... - - - - - ➤ •	
	• - - - - - - ➤ •		• ... - - - - - - ➤ •	
曰 ◇	• - - - - - - ➤ •	曰 ◇	• ... - - - - - - ➤ •	
	• - - - - - - - ➤ •		• ... - - - - - - - ➤ •	
□ ◇	• - - - - - - - ➤ •	□ ◇	• ... - - - - - - - ➤ •	
	• - - - - - - - - ➤ •		• ... - - - - - - - - ➤ •	
■ ◇				
▣ ◇				

Трудно сразу поверить, что на основе такого простого алфавита можно построить удобный и универсальный графовый язык. В рамках *Документации Технологии OSTIS* мы постараемся Вас в этом убедить. Кроме того, нас не должна настороживать простота алфавита, поскольку человечество имеет большой опыт кодирования, хранения в памяти и передачи по каналам связи самых различных информационных ресурсов, используя алфавит, состоящий только из двух классов элементов – единиц и нулей.

Мы ведем речь о принципиально ином (графовом) способе кодирования информации в *компьютерных системах*, но стараемся при этом свести это кодирование к достаточно простому алфавиту хотя бы для того, чтобы искусственно не усложнять проблему создания нового поколения компьютеров, основанных на указанном способе кодирования информации.

Расширения *Ядра SCg-кода* рассмотрим как направления перехода от текстов *Ядра SCg-кода* к более компактным текстам. Но, поскольку это приводит к усложнению *Синтаксиса SCg-кода* и, в первую очередь, к расширению *Алфавита SCg-кода*, делать такие расширения необходимо обоснованно с учетом частоты встречаемости в рамках *баз знаний ostis-систем* соответствующих фрагментов.

Пункт 2.3.2.2. Денотационная семантика SCg-кода

В SCg-коде выделяются Ядро SCg-кода и его расширения. Алфавит Ядра SCg-кода – алфавит *sc.g-элементов*, графически изображаемых *sc-элементы*. Алфавит Ядра SCg-кода взаимно однозначно соответствует Алфавиту SC-кода.

Денотационная семантика Ядра SCg-кода соответствует денотационной семантике SC-кода. Это продемонстрировано на рисунке.

Таблица. Алфавит Ядра SCg-кода

Алфавит SC-кода	Алфавит Ядра SCg-кода	Изображение sc.g-элемента
<i>sc-узел общего вида</i>	<i>sc.g-узел общего вида</i>	•
<i>sc-ребро общего вида</i>	<i>sc.g-ребро общего вида</i>	—
<i>sc-дуга общего вида</i>	<i>sc.g-дуга общего вида</i>	—→
<i>базовая sc-дуга</i>	<i>базовая sc.g-дуга</i>	→
<i>внутренний файл ostis-системы</i>	<i>sc.g-узел с содержимым</i>	□

Алфавит Ядра SCg-кода представлен следующими элементами:

- **sc.g-узел общего вида** – *sc.g-элемент*, являющийся графическим изображением *sc-узла общего вида*. Все *sc-узлы*, не являющиеся знаками файлов, в тексте (конструкции) *Ядра SCg-кода*, изображаются в виде небольших чёрных кругов одинакового диаметра, который обозначим через *d*, и точная величина которого зависит от масштаба отображения *sc.g-текста*.
- **sc.g-ребро общего вида** – *sc.g-элемент*, являющийся графическим изображением *sc-ребра общего вида*. Каждое *sc-ребро* в *Ядре SCg-кода* изображается в виде широкой линии, в которой чередуются фрагменты со сплошной заливкой и без заливки, не имеющей самопересечений и имеющей общую толщину, равную примерно *0.7d*.
- **sc.g-дуга общего вида** – *sc.g-элемент*, являющийся графическим изображением *sc-дуги общего вида*. Каждая *sc-дуга* в *Ядре SCg-кода* изображается в виде широкой линии, в которой чередуются фрагменты со сплошной заливкой и без заливки, не имеющей самопересечений, имеющей общую толщину, равную примерно *0.7d* и имеющей изображение стрелочки на одном из концов этой линии.
- **базовая sc.g-дуга** – *sc.g-элемент*, являющийся графическим изображением *базовой sc-дуги*. Каждая входящая в состав *sc-текста* *базовая sc-дуга* в *Ядре SCg-кода* изображается в виде линии произвольной формы, не имеющей самопересечений, имеющей толщину *0.4d*, и имеющей изображение стрелочки на одном из ее концов.
- **внутренний файл ostis-системы** – *sc-узел*, являющийся знаком внутреннего файла *ostis-системы*, *sc-знак* внутреннего файла *ostis-системы*.
- **sc.g-узел с содержимым** – *sc.g-узел*, имеющий содержимое, *sc.g-узел*, являющийся знаком внутреннего файла *ostis-системы*, *sc.g-рамка*. *sc.g-рамка* – это всегда прямоугольник, максимальный размер которого не ограничивается, но минимальный фиксируется и соответствует *sc.g-рамке*, внутри которой обозначаемый ею *файл* не отображается. Каждый входящий в *sc-текст* *sc-узел, имеющий содержимое*, в *Ядре SCg-кода* изображается в виде прямоугольника произвольного размера с толщиной линии *0.6d*. Внутри этого прямоугольника отобра-

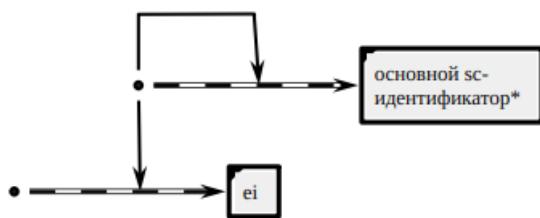
жается *файл*, обозначаемый изображаемым *sc-узлом*. Если нет необходимости изображать в тексте сам *файл*, то *sc-узел*, обозначающий такой *файл*, в *sc.g-тексте* изображается в виде прямоугольника со сторонами $2d$ по вертикали и $4d$ по горизонтали.

Пункт 2.3.2.3. Иерархическое семейство подъязыков, семантически эквивалентных SCg-коду

Первое направление расширения Ядра SCg-кода

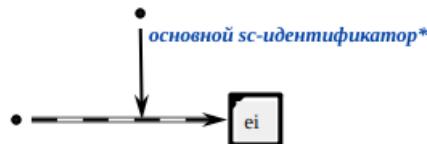
Первое направление синтаксического расширения Ядра SCg-кода – это приписывание некоторым *sc.g-элементам* основных *sc-идентификаторов** (чаще всего – строковых идентификаторов, то есть имен) *sc-элементов*, изображаемых этими *sc.g-элементами*. Указываемые идентификаторы являются уникальным для каждого идентифицируемого (именуемого) *sc.g-элемента*. Приписывание *sc.g-элементам* уникальных идентификаторов дает возможность в рамках одного *sc.g-текста* дублировать (копировать) некоторые *sc.g-узлы* при условии, если всем таким копиям будут приписаны соответствующие идентификаторы. Такое дублирование *sc.g-узлов* является дополнительным средством наглядного размещения *sc.g-текстов*. Кроме того, приписывание *sc.g-элементу* соответствующего ему основного (уникального) *sc-идентификатора** представляет собой более компактный вариант изображения *sc.g-текстов*.

Пример sc.g-текста, трансформируемого по Первому направлению расширения Ядра SCg-кода



Здесь (в левом нижнем углу приведенного *sc.g-текста*) представлен *sc.g-узел общего вида*, изображающий *sc-узел общего вида*, которому соответствует *основной sc-идентификатор** в виде строки “*ei*”.

Трансформация *sc.g-текста* по *Первому направлению расширения Ядра SCg-кода*: *sc.g-узлу общего вида* изобража-



ющему *sc-узел*, внешним идентификатором которого является строка “*основной sc-идентификатор*”* и который, соответственно является знаком *бинарного ориентированного отношения*, каждая пара которого связывает идентифицируемый *sc-элемент* с его основным внешним *sc-идентификатором*, приписывается указанный внешний идентификатор изображаемого им *sc-элемента*.

Трансформация *sc.g-текста* по Первому направлению расширения Ядра:

ei

В результате данной трансформации исходный *sc.g-текст* трансформируется в один *sc.g-узел общего вида*, которому приписывается *основной sc-идентификатор* “*ei*”.

Подчеркнем, что рассматриваемая трансформация преобразует исходный текст Ядра SCg-кода в текст, семантически эквивалентный, но принадлежащий не Ядру SCg-кода, а его расширению.

Второе направление расширения Ядра SCg-кода

Второе направление расширения Ядра SCg-кода – это уточнение типологии константных постоянных сущностей и расширение Алфавита Ядра SCg-кода, позволяющее типологию константных постоянных сущностей привести в соответствие с синтаксической типологией новых вводимых элементов Алфавита SCg-кода. Рассмотрим подробнее sc.g-элементы, знаки константных постоянных сущностей различного вида. Графическим признаком константных постоянных sc-узлов в конструкциях SCg-кода является их изображение в виде окружностей диаметра $3d$, где d – диаметр sc.g-узла общего вида. Такое изображение является более компактной записью факта принадлежности заданного sc-узла (назовем его vi) классу sc-констант и классу обозначений постоянных сущностей. Запись этого факта в Ядре SCg-кода потребует (1) явного изображения sc-узла, обозначающего класс всевозможных константных sc-элементов (класс sc-констант), (2) явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс всевозможных константных sc-элементов (класс sc-констант), с изображением заданного константного sc-узла, (3) явного изображения sc-узла, обозначающего класс всевозможных sc-элементов, обозначающих постоянные сущности, (4) явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс обозначений постоянных сущностей с изображением рассматриваемого sc-узла vi (Смотрите Файл. Изображение спецификации sc.g-элемента средствами Ядра SCg-кода и Первого расширения Ядра SCg-кода).

Второе направление расширения Ядра SCg-кода – это уточнение типологии константных постоянных сущностей и расширение Алфавита Ядра SCg-кода, позволяющее типологию константных постоянных сущностей привести в соответствие с синтаксической типологией новых вводимых элементов Алфавита SCg-кода. Рассмотрим подробнее sc.g-элементы, знаки константных постоянных сущностей различного вида. Графическим признаком константных постоянных sc-узлов в конструкциях SCg-кода является их изображение в виде окружностей диаметра $3d$, где d – диаметр sc.g-узла общего вида. Такое изображение является более компактной записью факта принадлежности заданного sc-узла (назовем его vi) классу sc-констант и классу обозначений постоянных сущностей. Запись этого факта в Ядре SCg-кода потребует:

- явного изображения sc-узла, обозначающего класс всевозможных константных sc-элементов (класс sc-констант);
- явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс sc-констант, с изображением заданного константного sc-узла;
- явного изображения sc-узла, обозначающего класс всевозможных sc-элементов, обозначающих постоянные сущности;
- явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс обозначений постоянных сущностей с изображением рассматриваемого sc-узла vi (Смотрите Файл. Изображение спецификации sc.g-элемента средствами Ядра SCg-кода и Первого расширения Ядра SCg-кода).

Общепринятая запись данного факта выглядит следующим образом:

“sc-константа $\exists vi$; постоянная сущность $\exists vi$;”

- Константные постоянные sc-ребра в конструкциях SCg-кода изображаются в виде двойной линии, каждая из которых имеет толщину примерно $d/7$, а расстояние между ними равно примерно $3d/7$.
- Константные постоянные sc-дуги изображаются в виде такой же двойной линии, но со стрелочкой. Все базовые sc-дуги, а также все sc-узлы, имеющие содержимое, по определению являются константными постоянными sc-элементами.
- Константные sc.g-узлы, изображаемые окружностями диаметра $3d$ и толщиной границы $d/5$, обозначают константные постоянные сущности, о которых мало что известно, но известно то, что они не являются парами (то есть множествами, мощность* которых равна 2) и, следовательно, не могут быть изображены в виде sc.g-дуг или sc.g-ребер. Но, если при этом об обозначаемой константной постоянной сущности (vi) известно, что она является классом сущностей, то явное указание принадлежности sc-элемента vi всевозможных классов можно заменить на специальное графическое изображение sc-элемента vi , предполагаемое указанную принадлежность. Это приводит к расширению Алфавита SCg-кода (см. Примеры sc.g-текстов, трансформируемых по Второму направлению расширения Ядра SCg-код).

Аналогичным образом (см. Примеры sc.g-текстов, трансформируемых по Второму направлению расширения Ядра SCg-код) вводятся:

- sc.g-узел, являющийся изображением класса;
- sc.g-узел, являющийся изображением класса классов;
- sc.g-узел, являющийся изображением отношения;
- sc.g-узел, являющийся изображением ролевого отношения;
- sc.g-узел, являющийся изображением структуры;
- sc.g-узел, являющийся изображением небинарной связки;
- sc.g-узел, являющийся изображением первичной сущности (терминальной сущности, которая не является множеством, а также файлом, хранимым в памяти ostis-системы).

Важное место среди константных постоянных сущностей занимают константные постоянные пары принадлежности, обозначаемое соответствующими sc.g-дугами. Такие пары принадлежности и обозначающие их sc.g-дуги

бывают позитивными, негативными и нечеткими. Константная постоянная позитивная sc.g-дуга принадлежности есть ничто иное, как *базовая sc.g-дуга*. Константная постоянная негативная sc.g-дуга принадлежности изображается в виде *базовой sc.g-дуги*, перечеркнутой штриховыми черточками. Константная постоянная нечеткая sc.g-дуга принадлежности изображается в виде "недочеркнутой" *базовой sc.g-дуги*, с каждой стороны которой отображаются штрихи, по длине равные половине от длины штрихов, которыми перечеркнута *константная постоянная негативная sc.g-дуга*.

Третье направления расширения Ядра SCg-кода

Третье направление расширения Ядра SCg-кода – это расширение его алфавита путем введения дополнительных sc.g-элементов, обозначающих *константные временные сущности* различного вида. Признаком sc.g-элементов, обозначающих *константные временные сущности* являются точечные линии (линии, состоящие из точек, размер которых равен размеру изображаемой линии и которые близко расположены друг к другу на расстоянии, равном половине их размера), с помощью которых рисуются окружности при изображении sc-узлов, а также линии при изображении sc-коннекторов.

Результатом *Третьего направления расширения Ядра SCg-кода* является введение следующих видов sc.g-элементов (см. *Примеры sc.g-текстов, трансформируемых по Третьему направлению расширения Ядра SCg-кода*).

Четвёртое направление расширения Ядра SCg-кода

Четвёртое направление расширения Ядра SCg-кода – это расширение его алфавита путем введения дополнительных элементов, обозначающих *переменные постоянные сущности* различного вида. Признаком sc.g-элементов, обозначающих сущности указанного класса, являются квадратики для изображения обозначений *переменных постоянных сущностей*, не являющихся бинарными связями, а также пунктирные и штрих-пунктирные линии для изображения *переменных постоянных бинарных связей*.

Подчеркнем, что *переменные постоянные сущности* могут отличаться друг от друга по характеру их *области значений**. Этими значениями в общем случае могут быть как *константные постоянные сущности*, так и *переменные постоянные сущности*. В любом случае, значение *переменной сущности* является либо *константной сущностью*, либо *переменной сущностью*. Если каждое значение *переменной* является константой, то такую *переменную* будем называть *переменной первого уровня*. Если каждое значение *переменной* является *переменной первого уровня*, то такую *переменную* будем называть *переменной второго уровня*.

Переменная постоянная сущность первого уровня (первичная sc-переменная), не являющаяся бинарной связью – это переменная, каждым значением которой является *константная постоянная сущность*, не являющаяся бинарной связью. Такая переменная изображается квадратиком, который ориентирован по вертикали и горизонтали. *переменная постоянная сущность второго уровня* (вторичная sc-переменная), не являющаяся бинарной связью, изображается квадратиком, повернутым на 45°.

Указанная выше семантика таких изображений приписывается по умолчанию. Это означает, что, если обозначенная sc-переменная имеет более сложную структуру области её значений (является sc-переменной третьего и выше уровня или sc-переменной, значения которой имеют различный логический уровень), то эта область должна быть специфицирована явно, при этом такая sc-переменная в SCg-коде изображается так же, как первичная sc-переменная.

Пятое направление расширения Ядра SCg-кода

Пятое направление расширения Ядра SCg-кода – это расширение его алфавита путем введения дополнительных sc.g-элементов, обозначающих *переменные временные сущности* различного вида. Указанные дополнительные sc.g-элементы аналогичны тем, которые введены в рамках *Четвертого направления расширения Ядра SCg-кода*, и отличаются только тем, что в *Пятом направлении расширении Ядра SCg-кода* речь идёт о *переменных временных сущностях*, а в *Четвертом направлении расширения Ядра SCg-кода* – о *переменных постоянных сущностях*.

Шестое направление расширения Ядра SCg-кода

Шестое направление расширения Ядра SCg-кода – это введение в SCg-код sc.g-контуров и sc.g-шин как средств структуризации sc.g-текстов и повышения наглядности при их размещении. Подчеркнем, что и sc.g-контуры, и sc.g-шины, и sc.g-рамки являются специальными видами sc.g-элементов. При этом sc.g-контуры и sc.g-рамки являются sc.g-ограничителями (ограничителями SCg-кода).

Каждый *sc.g-контур* изображается (в 2D-модификации) в виде замкнутой ломаной линии со скругленными изломами, ограничивающей некоторый фрагмент sc.g-текста и обозначает множество всех sc-элементов, sc.g-изображения которых оказались внутри этого контура. Толщина указанной линии составляет примерно $0.4d$, где d – диаметр sc.g-узла общего вида.

Обозначение множества sc-элементов, изображаемое sc.g-контуром, может быть как константным, так и переменным. Соответственно этому линия, изображающая sc.g-контур может быть:

- сплошной непунктирной линией,
- точечной непунктирной линией,
- сплошной пунктирной линией,
- точечной пунктирной линией.

Семантическим эквивалентом sc.g-контура является sc.g-узел, обозначающий структуру. Использование sc.g-контура вместо указанного sc.g-узла исключает необходимость явно изображать SC-дуги принадлежности, выходящие из этого sc.g-узла. Это существенно повышает уровень наглядности sc.g-текста.

Если представленный внутри sc.g-контура текст не является sc.g-текстом, то считается, что что на самом деле внутренностью sc.g-контура является sc.g-текст, являющийся результатом перевода предоставленного текста в SCg-код.

Каждая **sc.g-шина** представляет собой замкнутую или незамкнутую линию толщиной примерно равной диаметру *sc.g-узла общего вида*, которая инцидентна только одному sc.g-элементу и семантически ему эквивалентна. Идея введения sc.g-шин заключается в увеличении «размеров» sc.g-элементов для расширения их области инцидентности. Особенно актуально это для sc.g-узлов, имеющих большое число инцидентных им sc.g-коннекторов.

Седьмое направление расширения Ядра SCg-кода

Седьмое направление синтаксического расширения Ядра SCg-кода – это переход от 2D-изображений sc.g-текстов к 3D-изображениям.

Одним из вариантов трехмерного изображения sc.g-текстов является следующий:

- все sc.g-узлы изображаются, как и ранее, плоскими графическими примитивами. При изменении точки просмотра они всегда "поворачиваются" параллельно плоскости экрана, но их масштаб (размер на экране) при удалении от точки просмотра уменьшается;
- аналогичным "плоским" образом изображаются sc.g-рамки с их "внутренним" содержанием, а также внешние идентификаторы, приписываемые sc.g-элементам;
- sc.g-коннекторы изображаются непересекающимися линиями в трехмерном пространстве (заметим, что при изображении sc.g-текстов на плоскости пересечение sc.g-коннекторов часто снижает наглядность, "читабельность" sc.g-текстов). Т.е. sc.g-коннекторы, которые на плоскости изображаются двойными линиями, в пространстве цилиндрическими, "трубчатыми линиями" с находящейся внутри тонкой, но просвечивающейся осевой линией;
- sc.g-контур в пространстве визуализируется несколькими (!) специального вида точками – например там, где есть точки инцидентности sc.g-контура с внешними sc.g-коннекторами. При этом sc.g-контур становится виден только по команде просмотра указанного контура (указание контура – это указание одной из его точек инцидентности). По этой команде цветом выделяются все граничные точки контура (точки инцидентности) и все внутренние sc.g-элементы контура. Если просматривается несколько контуров, то используется несколько цветов.

Вторым вариантом 3D-визуализации sc.g-текстов является размещение sc.g-текстов на параллельных плоскостях (слоях) с “прошивками” между этими слоями, соединяющими синонимичные sc.g-узлы, т.е. sc.g-узлы, имеющие одинаковые приписываемые им внешние идентификаторы. Такой вариант плоской, но многослойной визуализации sc.g-текстов дает возможность широко использовать те средства просмотра и редактирования sc.g-текстов, которые разработаны для плоской их визуализации.

§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода – SCs-код (Semantic Code string)

SCs-код

- := [Semantic Code string]
- := [Язык линейного представления знаний *ostis*-систем]
- := [Множество всевозможных текстов *SCs-кода*]
- := [Язык внешнего линейного представления конструкций внутреннего языка *ostis*-систем]

SCs-код представляет собой множество линейных текстов (*sc.s-текстов*), каждый из которых состоит из предложений (*sc.s-предложений*), разделенных друг от друга двойной точкой с запятой (разделителем *sc.s-предложений*). При этом *sc.s-предложение* представляет собой последовательность *sc.s-идентификаторов*, являющихся именами описываемых *сущностей* и разделяемых между собой различными *sc.s-разделителями* и *sc.s-ограничителями*.

Пункт 2.3.3.1. Синтаксис SCs-кода

Алфавит SCs-кода

:= [Алфавит символов SCs-кода]
 := [множество символов SCs-кода]
 := [символ, используемый в текстах SCs-кода]
 := [Язык внешнего линейного представления информационных конструкций внутреннего языка ostis-систем]
 ⇐ объединение*: {• Алфавит символов, используемых в sc.s-разделителях
 • Алфавит символов, используемых в sc.s-ограничителях
 • Алфавит символов, используемых в sc-идентификаторах
 ⇐ объединение*: {• Алфавит символов, используемых в простых строковых sc-идентификаторах
 • Алфавит символов, используемых в sc-выражениях
 }
 • Алфавит символов, используемых в неоднозначных sc.s-изображениях sc-узлов
 }

Алфавит SCs-кода строится на основе современных общепринятых наборов символов, что позволяет упростить разработку средств для работы с sc.s-текстами с использованием современных технологий.

В состав sc.s-текстов, как и в состав текстов любых других языков, являющихся вариантами внешнего отображения текстов SC-кода, могут входить различные файлы, в том числе естественно-языковые или даже файлы, содержащие другие sc.s-тексты. В общем случае в таких файлах могут использоваться самые разные символы, в связи с чем будем считать, что в Алфавит SCs-кода эти символы не включаются.

Алфавит символов, используемых в sc.s-разделителях состоит из: пробел, точка с запятой, двоеточие, круглый маркер и знак равенства.

Алфавит символов, используемых в sc.s-разделителях, изображающих связь инцидентности sc-элементов состоит из: “<”, “>”, “|”, “-”.

Базовый алфавит символов, используемых в sc.s-коннекторах состоит из: “~”, знак подчеркивания, знак равенства, двоеточие, “<”, “>”, “-”, “|”, “/”.

Расширенный алфавит символов, используемых в sc.s-коннекторах состоит из: “∈”, “∃”, “⊆”, “⊇”, “⊂”, “⊃”, “≤”, “≥”, “⇐”, “⇒”, “←”, “→”, “↔”.

При необходимости комбинации указанных признаков перечисленные символы комбинируются так, как показано в параграфе "Описание sc.s-разделителей и sc.s-ограничителей".

Как в *Базовом*, так и в *Расширенном Алфавитах sc.s-коннекторов* используются следующие общие признаки, характеризующие тип изображаемого sc-коннектора:

- знак подчеркивания как признак изображений переменных sc-коннекторов (один знак подчеркивания для sc-коннекторов, являющихся первичными sc-переменными, два знака подчеркивания для sc-коннекторов, являющихся вторичными sc-переменными (sc-метапеременными));
- вертикальная черта “|” как признак изображений негативных sc-дуг принадлежности;
- косая черта “/” как признак изображений нечетких sc-дуг принадлежности;
- тильда “~” как признак изображений временных sc-дуг принадлежности.

Для упрощения процесса разработки исходных текстов баз знаний с использованием SCs-кода и создания соответствующих средств вводятся два алфавита символов. *Базовый алфавит символов, используемых в sc.s-коннекторах* включает только символы, входящие в переносимый набор символов и имеющиеся на стандартной современной клавиатуре. Таким образом, для разработки исходных текстов баз знаний, использующих только *Базовый алфавит символов, используемых в sc.s-коннекторах* достаточно обычного текстового редактора. *Расширенный алфавит символов, используемых в sc.s-коннекторах* включает также дополнительные символы, которые позволяют сделать sc.s-тексты (и sc.n-тексты) более читабельными и наглядными. Для визуализации и разработки sc.s-текстов с использованием расширенного алфавита требуется наличие специализированных средств.

Алфавит символов, используемых в sc.s-ограничителях состоит из: “(”, “)”, “*”.

Алфавит символов, используемых в неоднозначных sc.s-изображениях sc-узлов состоит из: “{”, “}”, “-”, “!”, “[”, “]”.

Описание sc.s-разделителей и sc.s-ограничителей

sc.s-разделитель и sc.s-ограничитель являются важными элементами SCs-кода.

Существует *sc.s-разделитель*, используемый для структуризации *sc.s-предложений* и *sc.s-разделитель sc.s-предложений*.

sc.s-разделить – разделитель, используемый в *sc.s-текстах*. *sc.s-разделитель* разбивается на:

- *sc.s-разделитель*, используемый для структуризации *sc.s-предложений*.
 - Разделяет *sc-идентификатор бинарного отношения* и второй компонент одной из связок этого отношения в случае, если указанное бинарное отношение и его связка связаны *константной sc-дугой принадлежности*. Представляется в виде двоеточия.
 - Разделяет *sc-идентификатор бинарного отношения* и второй компонент одной из связок этого отношения в случае, если указанное бинарное отношение и его связка связаны *переменной sc-дугой принадлежности*. Представляется в виде двойного двоеточия.
- *sc.s-разделитель sc.s-предложений*, представляется в виде двойной точки с запятой.

sc.s-ограничитель представляется в виде: (![(*)!U!(*)]!)

Круглые скобки со звездочкой ограничивают присоединенные *sc.s-предложения*, которые, в свою очередь, могут иметь в своем составе другие присоединенные *sc.s-предложения*.

Также существует *sc.s-коннектор*. Типология *sc.s-коннекторов* полностью соответствует типологии *sc.g-коннекторов*, и, тем более, *sc-коннекторов*, т.к. она учитывает устоявшиеся традиции изображения связок целого ряда конкретных отношений.

sc.s-коннектор

:= [изображение *sc-коннектора* во внешнем тексте SCs-кода или SCn-кода]

⊆ *sc.s-разделитель*

Выделяют следующие *sc.s-коннекторы*:

- *ориентированный sc.s-коннектор*,
- *неориентированный sc.s-коннектор*;
- *sc.s-коннектор*, соответствующий *sc.g-дуге принадлежности*,
- *sc.s-коннектор*, соответствующий *sc.g-коннектору*, который не является *sc.g-дугой*.

Типология *sc.s-коннекторов* полностью соответствует типологии *sc.g-коннекторов*, и, тем более, *sc-коннекторов*, т.к. она учитывает устоявшиеся традиции изображения связок целого ряда конкретных отношений.

На множестве *sc-элементов* задано бинарное ориентированное отношение инцидентности *sc-элементов*, а так же подмножество этого отношения – отношение инцидентности входящих *sc-дуг*, каждая пара которого связывает *sc-дугу* с тем *sc-элементом*, в который она входит. В *SC-коде sc-коннекторы* могут соединять между собой не только *sc-узел* с *sc-узлами*, но и *sc-узел* с *sc-коннектором* и даже *sc-коннектор* с *sc-коннектором*. В последнем случае, указывая инцидентность *sc-коннекторов*, необходимо уточнить, какой из них является соединяемым (связываемым), а какой – соединяющим (связующим). Поэтому отношение инцидентности, заданное на множестве *sc-элементов* является ориентированным. Первый компонент пары этого отношения – связующий *sc-коннектор*, а второй – связываемый *sc-элемент*. Очевидно, что связующий *sc-элемент* всегда является *sc-коннектором*, а *sc-узел* может быть только связываемым.

sc.s-разделитель, изображающий связь инцидентности *sc-элементов* разбивается на:

- знак инцидентности “правого” *sc-коннектора* – знак инцидентности *sc-коннектора*, *sc-идентификатор* которого находится справа, изображается в виде “ \vdash ”;
- знак инцидентности “левого” *sc-коннектора* – знак инцидентности *sc-коннектора*, *sc-идентификатор* которого находится слева, изображается в виде “ \dashv ”;
- знак инцидентности входящей *sc-дуги* справа – знак инцидентности *sc-дуги*, *sc-идентификатор* который находится справа, изображается в виде “ $| <$ ”;
- знак инцидентности входящей *sc-дуги* слева – знак инцидентности *sc-дуги*, *sc-идентификатор* который находится слева, изображается в виде “ $> |$ ”.

sc.s-разделитель, изображающий связь инцидентности sc-элементов

⇒ разбиение*:

- { • знак инцидентности “правого” *sc-коннектора*
 - := [знак инцидентности *sc-коннектора*, *sc-идентификатор* которого находится справа]
 - = $![-]!$
- знак инцидентности “левого” *sc-коннектора*
 - := [знак инцидентности *sc-коннектора*, *sc-идентификатор* которого находится слева]
 - = $![\dashv]!$
- знак инцидентности входящей *sc-дуги* справа
 - := [знак инцидентности *sc-дуги*, *sc-идентификатор* который находится справа]
 - = $![<]!$

- *знак инцидентности входящей sc-дуги слева*
:= [знак инцидентности sc-дуги, sc-идентификатор который находится слева]
= ![>]!
}

Указанные sc.s-разделители с точки зрения синтаксической структуры sc.s-предложений аналогичны sc.s-коннекторам, но с точки зрения их денотационной семантики в отличие от sc.s-коннекторов они не являются изображениями соответствующих sc-коннекторов

Описание изображения sc.s-коннекторов в Базовом и Расширенном алфавите

Класс sc-элементов	Изображение sc.сконнектора в Расширенном алфавите		Изображение sc.сконнектора в Базовом алфавите	
константная постоянная позитивная sc-дуга принадлежности	Ξ	Є	->	<-
константная постоянная негативная sc-дуга принадлежности	∅	∉	- >	< -
константная постоянная нечеткая sc-дуга принадлежности	/Ξ	Є/	-/>	</-
константная временная позитивная sc-дуга принадлежности	~Ξ	Є~	~>	<~
константная временная негативная sc-дуга принадлежности	~∅	∉~	~ >	< ~
константная временная нечеткая sc-дуга принадлежности	~/Ξ	Є/~	~>	</~
переменная постоянная позитивная sc-дуга принадлежности	_Ξ	Є_	->	<-_
переменная постоянная негативная sc-дуга принадлежности	_∅	∉_	- >	< -_
переменная постоянная нечеткая sc-дуга принадлежности	_Ξ	Є/_	-/>	</-_
переменная временная позитивная sc-дуга принадлежности	_~Ξ	Є~_	~>	<~_
переменная временная негативная sc-дуга принадлежности	_~∅	∉~_	~ >	< ~_
переменная временная нечеткая sc-дуга принадлежности	_~/Ξ	Є/~_	~>	</~_
метапеременная постоянная позитивная sc-дуга принадлежности	__Ξ	Є__	-->	<-__
метапеременная постоянная негативная sc-дуга принадлежности	__∅	∉__	-- >	< -__
метапеременная постоянная нечеткая sc-дуга принадлежности	__Ξ	Є/_	-->	</-_
метапеременная временная позитивная sc-дуга принадлежности	__~Ξ	Є~_	~>	<~_
метапеременная временная негативная sc-дуга принадлежности	__~∅	∉~_	~ >	< ~_
метапеременная временная нечеткая sc-дуга принадлежности	__~/Ξ	Є/~_	~>	</~_

Знак равенства является *sc.s-разделителем* двух *sc-идентификаторов*, которые идентифицируют (именуют) одну и ту же сущность и, соответственно, являются *sc-идентификаторами** (внешними уникальными изображениями) одного и того же *sc-элемента*. При этом из указанных двух *sc-идентификаторов* чаще всего один является простым *sc-идентификатором*, а второй – *sc-выражением*. Реже оба эти *sc-идентификаторы* являются *sc-выражениями*. И совсем редко оба они являются простыми *sc-идентификаторами*. Последнее обозначает то, что оба эти *sc-идентификаторы* являются основными *sc-идентификаторами** одного и того же *sc-элемента*. Пример: *SC-код = sc.s-текст;;*

Здесь первый *sc-идентификатор* является *именем собственным*, а второй – *именем нарицательным*.

При трансляции *sc.s-текста* в *SC-код* знаку равенства на некотором этапе может быть поставлено в соответствие *sc-ребро*, принадлежащее отношению *синонимии** *sc-элементов*, идентифицируемых *sc-идентификаторами*, связанными знаком равенства. Но на последующем этапе указанное *sc-ребро* удаляется, а связанные им *sc-элементы склеиваются*. Таким образом *sc-ребро*, принадлежащее отношению *синонимии** *sc-элементов*, имеет не только денотационную, но и операционную семантику.

Знак равенства с включением – изображение *sc-дуги*, принадлежащей отношению погружения*, связывающей два *sc-узла*, обозначающих *sc-тексты*, первый из которых является погружающим, а второй (в который указанная *sc-дуга* входит) является погружаемым, вводимым в состав первого *sc-текста*.

sc-дуга, принадлежащая отношению погружения*, интерпретируется как команда погружения одного *sc-текста* в состав другого. При выполнении этой команды (1) все *sc-элементы* погружающегося *sc-текста* становятся элементами, принадлежащими погружающему *sc-тексту*, (2) все синонимичные *sc-элементы*, оказавшиеся в составе погружающего *sc-текста*, склеиваются, (3) *sc-узел*, обозначающий погружаемый *sc-текст*, а так же спецификация этого *sc-текста* (включая перечень всех его *sc-элементов*) погружается в историю эволюции базы знаний вместе со спецификацией события погружения рассматриваемого *sc-текста* в состав базы знаний.

Указанные *sc.s-коннекторы* отличаются от остальных *sc.s-коннекторов* тем, что они и соответствующие им *sc-коннекторы* (*sc-ребра*, принадлежащих отношению *синонимии sc-элементов* и *sc-дуги*, принадлежащие отношению погружения одного *sc-текста* в состав другого) имеют не только денотационную, но и операционную семантику, т.к. являются командами склеивания и командами погружения.

Описание sc.s-предложений

sc.s-предложение

:= [минимальный семантически целостный фрагмент *sc.s-текста*]
:= [минимальный *sc.s-текст*]

sc.s-предложение, (1) состоящее или из двух *sc-идентификаторов*, соединенных между собой *sc.s-коннектором*, или из трех *sc-идентификаторов*, разделенных *sc.s-разделителями*, изображающими *связь инцидентности sc-элементов*, и (2) завершающееся *двойной точкой с запятой*.

Нетрудно заметить, что простые *sc.s-предложения* по сути аналогичны триплетам языка RDF (RDF-триплетам), за тем исключением, что *простое sc.s-предложение* можно "развернуть" при помощи *Операции конверсии sc.s-предложений** не меняя при этом его смысл, а RDF-триплет нельзя. Это является одной из причин, по которой, в отличие от RDF-триплетов, в простых *sc.s-предложениях sc.s-коннекторы и sc.s-разделители, изображающие связь инцидентности sc-элементов* не могут быть опущены, поскольку они в том числе показывают направление изображаемой ими связи между *sc-элементами*.

Признаком завершения любого *sc.s-предложения*, т.е. последними его символами является *двойная точка с запятой*, которую, следовательно, можно считать разделителем *sc.s-предложений*.

Выделяют следующие операции над *sc.s-предложениями*:

- **Операция конверсии *sc.s-предложения****

Каждое *sc.s-предложение* (в том числе, и *простое sc.s-предложение*) можно преобразовать в семантически эквивалентное ему *sc.s-предложение* путем конверсии ("разворота") цепочки компонентов *sc.s-предложения*. Так, например, при конверсии ("развороте") простого *sc.s-предложения* (1) первый его *sc-идентификатор* (первый компонент этого *sc.s-предложения*) становится третьим компонентом конвертированного *sc.s-предложения*, (2) второй его *sc-идентификатор* (третий компонент исходного *sc.s-предложения*) становится первым компонентом "конвертированного" *sc.s-предложения* и (3) второй компонент исходного *sc.s-предложения* (*sc.s-коннектор* или *sc.s-разделитель*, изображающий *связь инцидентности sc-элементов*, соединяющий указанные выше компоненты) остается вторым компонентом конвертированного *sc.s-предложения*, но меняет направленность (" \exists " заменяется на " \in " и наоборот, " \supset " на " \subset " и наоборот, " \Rightarrow " на " \Leftarrow " и наоборот и т.д.)

- **Операция присоединения *sc.s-предложения****

Операция соединения двух sc.s-предложений при совпадении последнего компонента первого предложения с первым компонентом второго. В результате выполнения данной операции:

- первый компонент второго sc.s-предложения удаляется;
- оставшаяся часть второго предложения окружается sc.s-ограничителем присоединенных предложений ("(*" и "*"). Разделитель sc.s-предложений (";") также попадает внутрь указанного ограничителя;
- полученная конструкция помещается между последним компонентом первого предложения и разделителем sc.s-предложений, которым заканчивалось первое предложение;
- второе предложение, таким образом, становится присоединенным sc.s-предложением.

Аналогичным образом к любому присоединенному sc.s-предложению могут "пристыковываться" другие присоединенные sc.s-предложения, в общем случае уровень вложенности не ограничен.

Присоединенные sc.s-предложения используются для того, чтобы продолжить спецификацию какого-либо sc-элемента, sc-идентификатор которого является последним компонентом в рамках какого-либо sc.s-предложения, не начиная при этом нового sc.s-предложения и, таким образом, не дублируя указанный sc-идентификатор. Внутрь присоединенных sc.s-предложений также могут встраиваться другие присоединенные sc.s-предложения, в общем случае уровень вложенности таких предложений не ограничен. Таким образом присоединенные sc.s-предложения описывают "ветвление" sc.s-предложений, при этом точками такого "ветвления" выступают sc-идентификаторы, входящие в состав этих sc.s-предложений.

Благодаря введению присоединенных sc.s-предложений появляется возможность любой sc-текст изобразить в виде одного sc.s-предложения, содержащего необходимое количество присоединенных sc.s-предложений. Таким образом, SCs-код по выразительной мощности становится эквивалентным SCn-коду.

- **Операция слияния sc.s-предложений***

Операция присоединения простого sc.s-предложения к sc.s-предложению, у которого последний sc.s-коннектор совпадает с sc.s-коннектором простого sc.s-предложения, а предшествующий указанному sc.s-коннектору sc-идентификатор совпадает с первым sc-идентификатором простого sc.s-предложения.

В результате выполнения этой операции совпадающие sc-идентификаторы и sc.s-коннекторы соединяемых sc.s-предложений "склеиваются", а последние sc-идентификаторы соединяемых sc.s-предложений становятся последними компонентами объединенного sc.s-предложения, разделенными точкой с запятой. Аналогичным образом можно присоединять сколько угодно простых sc.s-предложений.

- **Операция разложения sc.s-предложений на простые sc.s-предложения***

Каждое sc.s-предложение можно разложить на множество простых sc.s-предложений, т.е. представить в виде последовательности простых sc.s-предложений.

- **Операция разложения sc.s-предложений на простые sc.s-предложения с sc.s-разделителем, изображающим связь инцидентности sc-элементов***

Каждое sc.s-предложение (в том числе и простое sc.s-предложение с sc.s-коннектором) можно представить в виде семантически эквивалентной последовательности простых sc.s-предложений с sc.s-разделителем, изображающим связь инцидентности sc-элементов.

Данная операция осуществляет однозначное (!) формирование множества простых sc.s-предложений указанного вида.

Операции, заданные на множестве sc.s-предложений можно разделить на три группы:

- группа операций конверсии sc.s-предложений, состоящая из одной операции;
- группа операций соединения sc.s-предложений;
- группа операций декомпозиции sc.s-предложений и, в частности, операций разложения sc.s-предложений.

компонент sc.s-предложения*

Каждое sc.s-предложение представляет собой последовательность (1) sc-идентификаторов, (2) sc.s-коннекторов или sc.s-разделителей, изображающих связь инцидентности sc-элементов, (3) точек с запятыми, (4) ограничителей присоединенных sc.s-предложений, завершаемая двойной точкой с запятой. При этом непосредственно соседствовать друг с другом не могут ни sc-идентификаторы, ни sc.s-коннекторы, ни, очевидно, точки с запятыми и ограничители присоединенных sc.s-предложений.

Между sc-идентификаторами в рамках sc.s-предложения может находиться либо точка с запятой, либо sc.s-коннектор, либо sc.s-разделитель, изображающий связь инцидентности sc-элементов. Слева и справа от sc.s-коннектора и от sc.s-разделителя, изображающего связь инцидентности sc-элементов, должны находиться sc-идентификаторы.

Указанные sc-идентификаторы, sc.s-коннекторы и sc.s-разделители, изображающие связь инцидентности sc-элементов, считаются компонентами соответствующего sc.s-предложения. Понятие "быть компонентом sc.s-предложения" является относительным понятием (отношением), т.к. в состав некоторых компонентов sc.s-

предложения (в состав *sc-идентификаторов*, являющихся *sc.s-выражениями*, ограничиваемыми фигурными или квадратными скобками) могут входить других *sc.s-предложения*, состоящие из своих компонентов.

*sc.s-модификатор**

Это дополнительный вид компонентов *sc.s-предложений*. Каждый *sc.s-модификатор*, являющийся компонентом некоторого *sc.s-предложения*, представляет собой *sc-идентификатор*, обозначающий множество (чаще всего, отношение), которому принадлежит sc-коннектор, изображенный *sc.s-коннектором*, который предшествует указанному *sc-идентификатору*. Признаком *sc.s-модификатора* является *двоеточие* (или *двойное двоеточие*), которое ставится после *sc.s-модификатора* и отделяет его либо от следующего за ним другого *sc.s-модификатора* для этого же *sc.s-коннектора*, либо от следующего за ним *sc-идентификатора*, соответствующего sc-элементу, который инцидентен sc-коннектору, изображенному *sc.s-коннектором*, находящимся левее рассматриваемого *sc-идентификатора* после одного или нескольких *sc.s-модификаторов*. Обычное ("одинарное") *двоеточие* обозначает, что sc-элемент, изображенный соответствующим *sc.s-модификатором*, связан с *sc-коннектором*, изображенным левее этого *sc.s-модификатора*, *базовой sc-дугой* (*константной постоянной позитивной sc-дугой принадлежности*), *двойное двоеточие* обозначает, что указанные элементы связаны *переменной постоянной позитивной sc-дугой принадлежности*.

sc.s-текст

- := [конкатенация *sc.s-предложений*]
- := [последовательность *sc.s-предложений*, разделяемых *двойными точками с запятой*]

sc.s-предложение является минимальным *sc.s-текстом*. Смысл *sc.s-текста* (а также *sc.s-текста, включенного в структуру* не зависит от порядка *sc.s-предложений* в этих *sc-текстах*. Т.е. перестановка *sc.s-предложений* в рамках таких *sc.s-текстов* смысла этих *sc.s-текстов* не меняет (т.е. приводит к семантически эквивалентным *sc.s-текстам*), но сильно влияет на трудоемкость человеческого восприятия (на "читабельность" этих текстов).

Пункт 2.3.3.2. Денотационная семантика SCs-кода

Таблица. Алфавит sc.s-коннекторов, соответствующих sc.g-дугам принадлежности

Класс sc-элементов	Изображение sc.g-коннектора	Изображение sc.сконнектора в Расширенном алфавите	Изображение sc.сконнектора в Базовом алфавите		
константная постоянная позитивная sc-дуга принадлежности	• —————→ •	Ξ	Є	->	<-
константная постоянная негативная sc-дуга принадлежности	• —————→ •	∅	∉	- >	< -
константная постоянная нечеткая sc-дуга принадлежности	• —————→ •	/Ξ	Є/	-/ >	</ -
константная временная позитивная sc-дуга принадлежности	•→ •	~Ξ	Є~	~>	<~
константная временная негативная sc-дуга принадлежности	•+.....+.....+> •	~∅	∉~	~ >	< ~
константная временная нечеткая sc-дуга принадлежности	•+.....+.....+> •	~/Ξ	Є/~	~>	</~
переменная постоянная позитивная sc-дуга принадлежности	• — — — — > •	_Ξ	_Є	_>	<_
переменная постоянная негативная sc-дуга принадлежности	• — — — — > •	_∅	∉_	_ >	< _
переменная постоянная нечеткая sc-дуга принадлежности	• — — , — — , > •	_Ξ	Є/_	_>	</_
переменная временная позитивная sc-дуга принадлежности	• > •	_~Ξ	Є~_	_~>	<~_
переменная временная негативная sc-дуга принадлежности	• > •	_~∅	∉~_	_~ >	< ~_
переменная временная нечеткая sc-дуга принадлежности	• > •	_~Ξ	Є/~_	_~>	</~_
метапеременная постоянная позитивная sc-дуга принадлежности	• — — — — — — > •	_Ξ	Є_	_>	<_
метапеременная постоянная негативная sc-дуга принадлежности	• — + — + — + — + > •	_∅	∉_	_ >	< _
метапеременная постоянная нечеткая sc-дуга принадлежности	• — + — , — — , > •	_Ξ	Є/_	_>	</_
метапеременная временная позитивная sc-дуга принадлежности	• > •	_~Ξ	Є~_	_~>	<~_
метапеременная временная негативная sc-дуга принадлежности	• + + + + > •	_~∅	∉~_	_~ >	< ~_
метапеременная временная нечеткая sc-дуга принадлежности	• + + + + > •	_~Ξ	Є/~_	_~>	</~_

Таблица. Алфавит sc.s-коннекторов, соответствующих sc.g-коннекторам, которые не являются sc.g-дугами принадлежности

Изображение sc-коннектора в SCg	Изображение sc.s-коннектора в Расширенном алфавите		Изображение sc.s-коннектора в Базовом алфавите	
	\leftrightarrow		\Leftrightarrow	
	\rightarrow	\leftarrow	\gg	\ll
	\Leftrightarrow		\Leftrightarrow	
	\Rightarrow	\Leftarrow	\Rightarrow	\Leftarrow
	$\sim\leftrightarrow$		$\sim\langle=\rangle$	
	$\sim\Rightarrow$	$\Leftarrow\sim$	$\sim\Rightarrow$	$\Leftarrow\sim$
	$_{}\leftrightarrow$		$_{}\langle=\rangle$	
	$_{}\Rightarrow$	$\Leftarrow__$	$_{}\Rightarrow$	$\Leftarrow__$
	$_{}\sim\leftrightarrow$		$_{}\sim\langle=\rangle$	
	$_{}\sim\Rightarrow$	$\Leftarrow\sim__$	$_{}\sim\Rightarrow$	$\Leftarrow\sim__$
	$_{}\leftrightarrow$		$_{}\langle=\rangle$	
	$_{}\Rightarrow$	$\Leftarrow__$	$_{}\Rightarrow$	$\Leftarrow__$
	$_{}\sim\leftrightarrow$		$_{}\sim\langle=\rangle$	
	$_{}\sim\Rightarrow$	$\Leftarrow\sim__$	$_{}\sim\Rightarrow$	$\Leftarrow\sim__$
	\sqsupseteq	\sqsubseteq		
	\sqsubset	$\sqsubseteq_$		
	\sqsubset	\sqsubset		
	\sqsubset	$\sqsubseteq_$		

Изображение sc-коннектора в SCg	Изображение sc.s-коннектора в Расширенном алфавите	Изображение sc.s-коннектора в Базовом алфавите
	\geq	\leq
	$- \geq$	$- \leq$
	$>$	$<$
	$- >$	$- <$
	$::=$	
	$- ::=$	
	$=$	
	$\supseteq =$	$= \subset$

Примеры синтаксической трансформации sc.s-предложений с использованием Расширенного алфавита sc.s-коннекторов и соответствующие семантически эквивалентные конструкции в SCg-коде

$[si \Rightarrow \text{включение}^*: sj]$

\Rightarrow синтаксическая трансформация*:

$[si \supseteq sj]$

\Leftrightarrow семантическая эквивалентность*:



$[si __ \Rightarrow \text{включение}^*:: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ \supseteq sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si __ \Rightarrow \text{включение}^*:: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ \supseteq sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

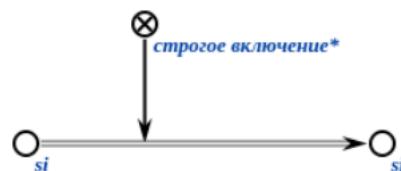
$[si \Rightarrow \text{строгое включение}^*: sj]$

\Rightarrow синтаксическая трансформация*:

$[si \supset sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

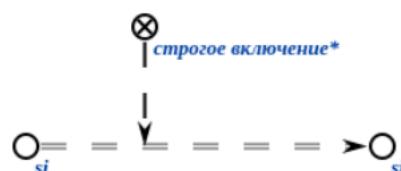
$[si __ \Rightarrow \text{строгое включение}^*:: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ \supset sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si __ \Rightarrow \text{строгое включение}^* :: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ \supset sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si \Rightarrow \text{порядок величин}^*: sj]$

\Rightarrow синтаксическая трансформация*:

$[si \geq sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si __ \Rightarrow \text{порядок величин}^* :: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ \geq sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si __ \Rightarrow \text{порядок величин}^* :: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ \geq sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

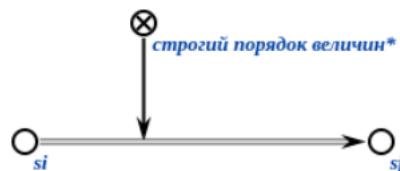
$[si \Rightarrow \text{строгий порядок величин*: } sj]$

\Rightarrow синтаксическая трансформация*:

$[si > sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

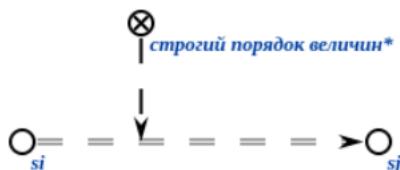
$[si_ \Rightarrow \text{строгий порядок величин*: } sj]$

\Rightarrow синтаксическая трансформация*:

$[si_ > sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

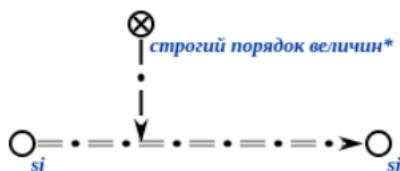
$[si_ _ \Rightarrow \text{строгий порядок величин*: } sj]$

\Rightarrow синтаксическая трансформация*:

$[si_ _ > sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si \Rightarrow \text{внешний идентификатор*: } sj]$

\Rightarrow синтаксическая трансформация*:

$[si := sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si __ \Rightarrow \text{внешний идентификатор}^*:: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ := sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

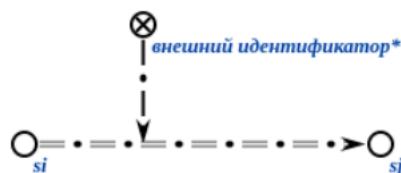
$[si __ \Rightarrow \text{внешний идентификатор}^*:: sj]$

\Rightarrow синтаксическая трансформация*:

$[si __ := sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si \Leftrightarrow \text{синонимия}^*: sj]$

\Rightarrow синтаксическая трансформация*:

$[si = sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

$[si \Rightarrow \text{погружение}^*: sj]$

\Rightarrow синтаксическая трансформация*:

$[si \supseteq sj]$

\Leftrightarrow семантическая эквивалентность*:

[



]

Аналогичным образом может быть описана трансформация предложений, содержащих любые классы sc.s-коннекторов, за исключением тех классов sc.s-коннекторов, которые соответствуют классам sc-коннекторов, входящим в Ядро SC-кода.

В общем случае *sc-элементы*, инцидентные *sc-коннекторам*, классы которых описаны в данном примере, могут быть как *sc-константами*, так и *sc-переменными* (в том числе *sc-метапеременными*). При этом как *переменному sc-коннектору* может соответствовать *константный sc-узел*, так и *константному sc-коннектору* может соответствовать *переменный sc-узел* (например, если возникает необходимость переменному sc-узлу присвоить

*внешний идентификатор**). Последняя ситуация встречается не очень часто и возникает в случае, когда область определения соответствующего *отношения* имеет непустое пересечение с классом *sc-переменных*.

Описание примеров выполнения операций, заданных на множестве sc.s-предложений

С семантической точки зрения *sc.s-предложение* представляет собой описание некоторого *маршрута* в соответствующем sc-тексте, который является графовой структурой специального вида и структура которого описывается (изображается) с помощью *sc.s-предложений*. Указанный маршрут "проводится" по sc-коннекторам и по связям инцидентности sc-элементов, если маршрут проходит через инцидентные sc-коннекторы. В описании указанного маршрута могут дополнительно указываться множества (чаще всего отношения), которым принадлежат sc-коннекторы, входящие в описываемый маршрут. Кроме того, указанный маршрут в начале и/или в конце может иметь разветвления, когда какой-либо sc-элемент одинаково инцидентен нескольким однотипным sc-коннекторам, соединяющим указанный sc-элемент с некоторыми другими sc-элементами.

Таким образом каждое указанное разветвление состоит из неограниченного числа ветвей, каждая из которых состоит из одного sc-коннектора и одного связываемого им sc-элемента.

Пункт 2.3.3.3. Иерархическое семейство подъязыков, семантически эквивалентных SCs-коду

В рамках SCs-кода выделяют Ядро SCs-кода и направления расширения Ядра SCs-кода.

Ядро SCs-кода

:= [Подъязык SCs-кода, который использует минимальный набор синтаксических средств, но при этом имеет семантическую мощность, эквивалентную мощности SCs-кода в целом]

В Ядре SCs-кода:

- используются только *простые sc-идентификаторы*, в том числе *sc-идентификаторы внешних файлов ostissистем* (sc-выражения не используются);
- используются только *sc.s-разделители*, изображающие связь инцидентности sc-элементов, а также sc.s-коннектор, изображающий константную постоянную позитивную пару принадлежности ("∈" и "Э" в Расширенном алфавите и ">" и "<" в Базовом алфавите). Другие *sc.s-коннекторы* не используются;
- не используются *sc.s-модификаторы* и, соответственно, двоеточия, являющиеся признаком завершения *sc.s-модификаторов*;
- используются только *простые sc.s-предложения*, которые, как следует из вышеуказанных свойств Ядра SCs-кода, либо состоят из двух *простых sc-идентификаторов*, соединяемых sc.s-коннектором, изображающим константную постоянную позитивную пару принадлежности, либо трех *простых sc-идентификаторов*, разделенных *sc.s-разделителями*, изображающими связь инцидентности sc-элементов.

Из перечисленных свойств Ядра SCs-кода следует, что для представления (изображения) любого sc-текста средствами Ядра SCs-кода необходимо для всех (!) sc-элементов этого sc-текста (кроме константных постоянных позитивных пар принадлежности) построить соответствующие им простые *sc-идентификаторы*, т.е. необходимо проименовать все указанные sc-элементы. В свою очередь, тип каждого используемого sc-элемента (кроме константных постоянных позитивных пар принадлежности) задается явно путем указания принадлежности этих элементов соответствующим классам sc-элементов, в том числе классам, входящим в Ядро SC-кода.

Как видно из приведенного описания, Ядро SCs-кода соответствует Ядру SCg-кода, за исключением того, что в Ядре SCg-кода нет необходимости именовать все изображаемые sc-элементы, а также в Ядре SCg-кода присутствуют графические изображения для sc-элементов, принадлежащих соответствующим классам Ядра SC-кода и эту принадлежность нет необходимости указывать явно.

Очевидно, что широко практически применять Ядро SCs-кода для записи больших фрагментов баз знаний неудобно и неэффективно. Тем не менее, с практической точки зрения Ядро SCs-кода может использоваться, например, для обмена информацией со сторонними средствами представления графовых конструкций, рассчитанными на представление информации в виде триплетов (например, RDF-хранилищ). Для обеспечения возможности более широкого практического использования необходимы синтаксические расширения Ядра SCs-кода в целях:

- минимизации числа идентифицируемых (именуемых) sc-элементов путем использования *sc-выражений* и ликвидации необходимости идентифицировать (именовать) все (!) sc-элементы;
- сокращения текста путем минимизации числа повторений одного и того же *sc-идентификатора* путем соединения *sc.s-предложений*;
- повышение уровня наглядности, "читабельности" sc.s-текстов.

Первое направление расширения Ядра SCs-кода

:= [Первое направление расширения Ядра SCs-кода и всех иных его расширений]

По сравнению с **Ядром SCs-кода** в *Первом направлении расширения Ядра SCs-кода* вместо *sc-идентификаторов*, являющихся идентификаторами (именами), которые взаимно однозначно соответствуют синонимичным им (представляемым ими) *sc-коннекторам*, вводятся *sc.s-коннекторы*, каждый из которых соответствует не одному конкретному *sc-коннектору*, а некоторому классу однотипных *sc-коннекторов*. Очевидно, что это ликвидирует необходимость каждому sc-коннектору приписывать уникальный *sc-идентификатор*. Кроме того, *Алфавит sc.s-коннекторов* включает в себя элементы этого Алфавита (классы синтаксически эквивалентных sc.s-коннекторов), которые соответствуют всем (!) элементам Алфавита *sc-коннекторов*, но при этом дополнительно включают в себя и другие элементы Алфавита *sc.s-коннекторов*, которые соответствуют часто используемым семантически явно выделяемым классам *sc-коннекторов*. К таким дополнительно вводимым классам *sc.s-коннекторов* относятся константные sc.s-коннекторы включения множеств (" \supset " или " \subset "), переменные sc.s-коннекторы включения множеств (" $_ \supset$ " или " $\subset _$ "), *sc.s-коннектор* синонимии ("="), *sc.s-коннектор* погружения ("= \subset " или " $\supset =$ ") и др.

Заметим, что указанное расширение Алфавита *sc.s-коннекторов* аналогично расширенному Алфавиту *sc.g-коннекторов* в SCg-коде и ликвидирует необходимость (как и в SCs-коде) явно специфицировать (средствами SCs-кода) синтаксически выделяемые классы *sc.s-коннекторов*.

Второе направление расширения Ядра SCs-кода

Во *Втором направлении расширения Ядра SCs-кода* вводятся модификаторы *sc.s-коннекторов* (*sc.s-модификаторы*), которые позволяют достаточно компактно дополнительно специфицировать *sc-коннекторы*, изображаемые (представляемые) соответствующими *sc.s-коннекторами*. Речь идет о такой часто востребованной форме спецификации *sc-коннекторов*, как указание множества (возможно, нескольких множеств), которому принадлежит специфицируемый *sc-коннектор* (чаще всего, таким множеством является *бинарное отношение* (в частности, *ролевое отношение*) или *квазибинарное отношение*).

sc.s-модификатор*

- \in *отношение*
 - \coloneqq [относительное понятие]
 - \coloneqq [*модификатор sc.s-коннектора**]
- sc-идентификатор*, который (1) находится либо между *sc.s-коннектором* и *двоеточием*, либо между *двоеточиями* и (2) обозначает множество (чаще всего, отношение), которому принадлежит *sc-коннектор*, изображаемый ближайшим предшествующим *sc.s-коннектором*. Два подряд идущих двоеточия ("::") обозначают, что указанное множество связано с указанным *sc-коннектором* переменной позитивной постоянной sc-дугой принадлежности.

Очевидно, что, если не использовать *sc.s-модификаторы*, указанного вида спецификация *sc-коннекторов* средствами SCs-кода будет выглядеть значительно более громоздкой.

Третье направление расширения Ядра SCs-кода

В *Третьем направлении расширения Ядра SCs-кода* осуществляется переход от использования только *простых sc-идентификаторов* к использованию как *простых sc-идентификаторов*, так и *sc-выражений*, а также к использованию *sc.s-представлений* некоторых *неидентифицируемых sc-узлов*. Это существенно сокращает число придумываемых *простых sc-идентификаторов*, т.к. каждое *sc-выражение* в конечном счете — это комбинация *простых sc-идентификаторов*, построенная по правилам, которые достаточно легко семантически интерпретируются. Если проводить аналогию с SCg-кодом, то очевидно, что *sc-выражение*, ограничиваемое фигурными скобками, есть не что иное, как информационная конструкция, ограниченная *sc.g-контуром*, а *sc-выражение*, ограничиваемое квадратными скобками есть не что иное, как информационная конструкция, ограниченная *sc.g-рамкой*. Отличие здесь заключается в том, что круглыми и квадратными скобками можно ограничивать только линейные информационные конструкции (цепочки символов).

sc.s-представление неидентифицируемого sc-узла

- \coloneqq [*изображение* (представление) неидентифицируемого (неименуемого) *sc-узла* в *sc.s-тексте*]
- \coloneqq [*sc.s-обозначение* неименуемой сущности, не являющейся парой, обозначаемой *sc-коннектором*]
- \coloneqq [*sc.s-представление* *sc-узла*, не являющееся *sc-идентификатором* (именем этого *sc-узла*)]

Если одно и то же обозначение неименуемой сущности встречается в разных sc.s-предложениях, то считается, что это обозначения разных сущностей, т.е. изображения разных *sc-узлов*.

Четвертое направление расширения Ядра SCs-кода

В *Четвертом направлении расширения Ядра SCs-кода* осуществляется переход от использования только *простых sc.s-предложений* к использованию также *sc.s-предложений*, построенных с помощью *Операции присоединения sc.s-предложения**. В результате этого, благодаря "склеиванию" одинаковых *sc-идентификаторов*, а также "склеиванию" синтаксически эквивалентных *sc.s-коннекторов* с одинаковыми *sc.s-модификаторами* (несмотря на то,

что эти "склеиваемые" sc.s-коннекторы соответствуют разным sc-коннекторам), существенно сокращается число копий используемых sc-идентификаторов и sc.s-коннекторов с их sc.s-модификаторами.

Пятое направление расширения Ядра SCs-кода

В Пятом направлении расширения Ядра SCs-кода разрешается использование присоединенных sc.s-предложений. В результате этого sc.s-тексты становятся более компактными и удобными для восприятия за счет снижения числа дублируемых sc-идентификаторов и более широких возможностей их структуризации.

§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода – SCn-код (Semantic Code natural)

SCn-код

:= [Язык структурированного представления знаний *ostis-систем*]

:= [Язык внешнего форматированного представления конструкций внутреннего языка *ostis-систем*]

SCn-код является языком структурированного внешнего представления текстов SC-кода и представляет собой синтаксическое расширение SCs-кода, направленное на повышение наглядности и компактности текстов SCs-кода.

SCn-код позволяет перейти от линейных текстов SCs-кода к форматированным и фактически двухмерным текстам, в которых появляется декомпозиция исходного линейного текста SCs-кода на строчки, размещенные "по вертикали". При этом начало всех строчек текста фиксировано и определяется известным и ограниченным набором правил, что дает возможность использовать это при форматировании sc.n-текста (текста, принадлежащего SCn-коду).

Пункт 2.3.4.1. Синтаксис SCn-кода

Алфавит символов SCs-кода является также алфавитом символов и SCn-кода, т.е. алфавиты* этих языков совпадают.

SCn-код – язык, каждый *текст* которого задается:

- множеством входящих в него *символов*;
- отношением порядка (последовательности) *символов* по "горизонтали";
- отношением порядка(последовательности) *символов* по "вертикали".

sc.n-текст

:= [текст SCn-кода]

:= [последовательность предложений SCn-кода]

:= [последовательность предложений SCn-кода, каждое из которых не является частью какого-либо другого предложения из этой последовательности]

Важной особенностью SCn-кода является "двуухмерный" характер его текстов. Это проявляется в том, что для каждого фрагмента текста SCn-кода важное значение имеет величина отступа от левого края *строчки*.

Символ, входящий в состав *двуухмерного текста*, в общем случае может иметь четыре "соседних" *символа*:

- *символ*, находящийся от него слева в рамках той же *строчки*;
- *символ*, находящийся от него справа в рамках этой же *строчки*;
- *символ*, находящийся строго над ним в предыдущей *строчке*;
- *символ*, находящийся строго под ним в следующей *строчке* текста.

Благодаря тому, что в состав sc.n-текстов могут входить и sc.s-тексты, и sc.g-тексты (ограниченные sc.n-контуром), SCn-код можно считать интегратором различных внешних языков представления знаний. Это дает возможность при визуализации и разработке базы знаний ostis-системы недостатки одного из предлагаемых вариантов внешнего представления sc-текстов (SCg-кода, SCs-кода, SCn-кода) компенсировать достоинствами других вариантов.

страница sc.n-текста

:= [страница, на которой размещается sc.n-текст]

Если sc.n-текст является частью какого-либо другого файла, разделяемого на страницы, например, публикации какой-либо части базы знаний, то sc.n-страницей считается только часть страницы, на которой изображен sc.n-текст, в то время как страница указанного файла может быть больше за счет, например, белых полей по краям страницы, необходимых для последующей распечатки.

строчка sc.n-текста

Максимальное количество символов в строчках sc.n-текста для каждого sc.n-текста фиксировано и определяется конкретным вариантом размещения sc.n-текста. При этом, в зависимости от отступов в рамках конкретного sc.n-предложения, строчка sc.n-текста может начинаться не с левого края sc.n-текста (но всегда с какой-то из вертикальных линий разметки) и иметь произвольную длину, ограничиваемую правой границей sc.n-страницы.

линия разметки sc.n-текста

- := [табуляционная линия sc.n-текста]
- := [вертикальная линия разметки sc.n-текста]
- := [вертикальная табуляционная линия]
- := [вертикальная линия, используемая для упрощения восприятия sc.n-текстов и показывающая уровень отступа для компонентов sc.n-предложений]

1-я линия разметки ограничивает левый край sc.n-страницы, 2-я линия разметки располагается примерно между 5 и 6 символами строчки и т.д. Расстояние между линиями разметки может меняться в зависимости от размера шрифта, однако в рамках одного sc.n-текста всегда остается одинаковым. Общее количество линий разметки ограничивается максимально возможной шириной sc.n-страницы в конкретном файле *ostis*-системы, содержащем данный sc.n-текст.

следует отличать*

- Э { • страница sc.n-текста
 - строчка sc.n-текста
 - строка
}

Все компоненты sc.s-текстов используются также и в sc.n-текстах:

- sc-идентификаторы;
- sc.s-коннекторы;
- модификаторы sc.s-коннекторов с соответствующими разделителями (двоеточиями)
- разделители, используемые в sc-выражениях, обозначающих sc-множества, заданные перечислением элементов с соответствующими разделителями (*точкой с запятой* или *круглым маркером*);
- *круглые маркеры* в перечислениях идентификаторов sc-элементов, связанных однотипными sc-коннекторами с однотипными модификаторами с заданным sc-элементом;
- разделители предложений (двойные точки с запятой) (опускаются при преобразовании sc.s-предложений в sc.n-предложения);
- ограничители присоединенных sc.s-предложений (опускаются при преобразовании sc.s-предложений в sc.n-предложения).

В отличие от sc.s-текстов в sc.n-текстах:

- добавляются новые виды sc-выражений (а именно – sc-выражений, имеющих двухмерный характер);
- добавляется новый вид разделителей предложений – пустая строчка;
- меняется размещение предложений с учетом двухмерного характера такого размещения.

В SCn-коде по сравнению с SCs-кодом добавляются новые виды sc-выражений:

- *sc-выражение*, представляющее собой двухмерный sc.n-текст, ограниченный sc.n-контуром или sc.n-рамкой. Каждый sc.n-контур изображается условно в виде *открывающей фигурной скобки* и расположенной строго под ней через несколько строчек *закрывающей фигурной скобки*. Внутри указанных скобок (начиная от линии вертикальной разметки, на которой расположены сами скобки, и до правого края страницы) размещается sc.n-текст. Полученный sc.n-контур является изображением структуры, являющейся результатом трансляции указанного sc.n-текста в SC-код. Каждая sc.n-рамка изображается аналогичным образом, только вместо *фигурных скобок* в ней используются *квадратные скобки*, либо *квадратные скобки с восклицательным знаком* (в случае файла-образца);
- *sc-выражение*, представляющее собой двухмерный sc.g-текст, ограниченный sc.n-контуром или sc.n-рамкой.
- *sc-выражение*, представляющее собой ограниченное sc.n-рамкой двухмерное графическое изображение *информационной конструкции*, закодированной в некотором файле *ostis*-системы. Такой *информационной конструкцией* может быть таблица, рисунок, фотография, диаграмма, график и многое другое.

Нетрудно заметить, что sc.n-контур является, по сути, двухмерным эквивалентом *sc-выражения структуры*, а sc.n-рамка – двухмерным эквивалентом *sc-выражения внутреннего файла ostis-системы* или *sc-выражения, обозначающего файл-образец ostis-системы*.

sc.n-рамка

- := [ограничитель изображения файла *ostis*-системы, используемый в sc.n-предложениях]

Обозначается с помощью квадратных скобок: "[", "] ".

С формальной точки зрения *sc.n-рамка* всегда представляет собой одну строчку sc.n-текста. Это означает, что *sc.n-рамка* не может быть синтаксически разделена на части в рамках того *sc.n-текста*, в котором она используется, и внутрь нее не могут вставляться, например, *присоединенные sc.n-предложения* или какой-либо другой текст (за исключением случаев, когда *sc.n-рамка* содержит *sc.n-текст*, но в этом случае указанный *sc.n-текст* все равно будет рассматриваться как целостный внешний файл, а не как фрагмент окружающего его *sc.n-текста*).

sc.n-контур

:= [используемый в sc.n-предложениях ограничитель, являющийся изображением структуры]

Обозначается с помощью фигурных скобок: "{ ", " } ".

Понятие *sc.n-предложение* является естественным обобщением понятия *sc.s-предложение*. Более того, аналогичным для *sc.s-предложений* образом вводятся понятия:

- *простого sc.n-предложения*
- *сложного sc.n-предложения*
- *sc.n-предложения, содержащего присоединенные sc.n-предложения*
- *sc.n-предложения, не содержащего присоединенные sc.n-предложения*
- *присоединенного sc.n-предложения*
- *неприсоединенного sc.n-предложения*

Если каждое *неприсоединенное sc.s-предложение либо* является первым предложением *sc.s-текста*, либо начинается после разделителя sc.s-предложений (двойной точки с запятой), то каждое неприсоединенное sc.n-предложение начинается с начала новой строчки.

Если каждое *присоединенное sc.s-предложение* начинается либо после открывающего ограничителя присоединенных *sc.s-предложений* (открывающей круглой скобки со звездочкой), либо после разделителя *sc.s-предложений*, то каждое присоединенное sc.n-предложение начинается с новой строчки под *sc-идентификатором*, которым завершается то *sc.n-предложение* (и соответственно, *sc.s-предложение*), в которое встраивается данное *присоединенное sc.n-предложение*.

Первый *sc-идентификатор*, входящий в состав *sc.n-предложения* до *sc.s-коннектора* выделяется жирным курсивом; В *sc.n-предложениях* двойная точка с запятой не используется в качестве признака завершения этих предложений и, соответственно, не используется в качестве разделителя *sc.n-предложений*. Таким разделителем является *пустая строка*.

Благодаря двухмерности SCn-кода появляются более широкие возможности (степени свободы) для наглядного и компактного размещения *sc.n-предложений*.

При оформлении *sc.n-предложения* осуществляется четкая табуляция всех присоединенных к нему *sc.n-предложений*, присоединяемых к исходному "по вертикали". Вертикальная линия табуляции задает левую границу исходного (максимального) *sc.n-предложения* или левую границу присоединенного *sc.n-предложения*, присоединяемого "по вертикали". Левая граница *sc.n-предложения* задает начало первого *sc-идентификатора*, входящего в состав этого *sc.n-предложения*, а также начало *sc.s-коннектора*, инцидентного указанному *sc-идентификатору* и размещаемого строго под этим *sc-идентификатором*. Расстояние между вертикальными табуляционными линиями фиксировано и примерно равно максимальной длине *sc.s-коннектора*.

Пункт 2.3.4.2. Денотационная семантика языка внешнего форматированного представления информационных конструкций внутреннего языка ostis-систем

В отличие от *sc.s-текстов*: в *sc.n-текстах* *sc.s-коннектор* может быть инцидентен предшествующему *sc-идентификатору* (как простому, так и *sc-выражению*) не только "по горизонтали", но и "по вертикали". Для этого *sc.s-коннектор* размещается строго под предшествующим ему *sc-идентификатором*.

Кроме того "по вертикали" *sc-идентификатор* может быть инцидентен не одному, а нескольким sc.s-коннекторам, которые последовательно "по вертикали" размещаются под указанным sc-идентификатором. Это позволяет в рамках одного *sc.n-предложения* представлять произвольное число "ответвлений" от каждого *sc-идентификатора*, т.е. произвольное число *sc.s-коннекторов*, инцидентных этому *sc-идентификатору*; Каждый *sc-идентификатор*, включая *sc-выражение*, ограничиваемого фигурными или квадратными скобками, должен размещаться сразу правее вертикальной разметочной линии, если под ним размещается *sc.s-коннектор*.

Каждый *sc.s-коннектор* выделяется жирным некурсивным шрифтом и, если он находится под инцидентным ему sc-идентификатором, размещается строго между двумя вертикальными разметочными линиями, прижимаясь при этом к левой из этих двух разметочных линий.

Поскольку по отношению к SCn-коду SCs-код является *синтаксическим ядром языка**, SCn-код можно рассматривать как результат интеграции нескольких направлений расширения SCs-кода, в основе которых лежат правила синтаксической трансформации sc.s-текстов и sc.n-текстов, ориентированные на повышение эффективности использования тех возможностей обеспечения наглядности и компактности sc.n-текстов, которые открываются при переходе от линейности sc.s-текстов к двухмерности sc.g-текстов.

Глава 2.4.

Представление формальных онтологий базовых классов сущностей в ostis-системах

Бутрин С.В.
Шункевич Д.В.

⇒ *аннотация**:
[Аннотация к главе.]

Правки:

- Ввести понятия для предметной области и онтологии
- Пересмотреть материал по главе 2.4, где можно объединить, убрать дублирование
- Добавить больше примеров и их описание

Вопросы:

- Что должна включать аннотация?
- Где должно быть сравнение с аналогами?
- Где вводится понятие базы знаний? Должно ли оно быть?

Для обеспечения совместного использования различных видов знаний, входящих в состав базы знаний, необходимо обеспечить их совместимость с указанной базой знаний, которая включает семантическую совместимость, что подразумевает однозначную и единую для всех фрагментов базы знаний трактовку используемых понятий.

Среди многообразия средств представления знаний к наиболее эффективным относятся онтологии [Давыденко И.Т. Модели МСРГ 2017dc](#). Суть такого подхода при проектировании базы знаний состоит в рассмотрении базы знаний как иерархической системы выделенных предметных областей и соответствующих им онтологий. Однако онтологически можно по разному специфицировать знания. Чтобы решить эту проблему проектируются онтологии верхнего уровня.

Применение современных онтологий верхнего уровня при разработке баз знаний интеллектуальных компьютерных систем сопряжено с проблемами обеспечения их совместимости. Поскольку изначальной целью создания онтологий верхнего уровня являлось обеспечение совместимости онтологий предметных областей и прикладных онтологий, а не самих интеллектуальных систем.

Такими проблемами являются:

- свобода трактовки понятий, вызванная отсутствием их четкого определения;
- отсутствие единой технологии проектирования баз знаний на основе онтологий верхнего уровня;
- отсутствие принадлежности онтологий верхнего уровня к какой-либо технологии, что не позволяет использовать их в качестве многократно используемых компонентов;

Поэтому возникает необходимость в разработке такой системы онтологии верхнего уровня, которая смогла бы обеспечить семантическую совместимость между большим количеством онтологий различных предметных областей.

Предлагаемый подход подразумевает разработку семейств Предметных областей и онтологий, которые бы содержали описания всех необходимых базовых классов сущностей для построения базы знаний интеллектуальной компьютерной системы.

К таким Предметным областям и онтологиям относятся:

- Предметная область и онтология множеств
- Предметная область и онтология связок и отношений
- Предметная область и онтология параметров, величин и шкал
- Предметная область и онтология чисел и числовых структур

- Предметная область и онтология структур
- Предметная область и онтология темпоральных сущностей
- Предметная область и онтология темпоральных сущностей баз знаний ostis-систем
- Предметная область и онтология семантических окрестностей
- Предметная область и онтология предметных областей
- Предметная область и онтология онтологий
- Предметная область и онтология логических формул, высказываний и формальных
- Предметная область и онтология внешних информационных конструкций и файлов ostis-систем
- Глобальная предметная область действий и задач и соответствующая ей онтология методов и технологий

Данные предметные области являются часть Ядра базы знаний, которое должно быть в каждой интеллектуальной системе. Это ядро гарантирует совместимость интеллектуальных компьютерных систем за счет общего понятийного аппарата. В зависимости от специфики конкретных систем могут выделяться различные Ядра базы знаний, но неизменным должна оставаться наличие базовая части, включающей в себя предметные области и онтологии указанные выше.

§ 2.4.1. Формальная онтология множеств

множество

- ⇒ разбиение*:
 - {• конечное множество
 - бесконечное множество
- ⇒ разбиение*:
 - {• множество без кратных элементов
 - мультимножество
- ⇒ разбиение*:
 - {• связка
 - класс
 - := [sc-элемент, обозначающий класс sc-элементов]
 - := [sc-знак множества sc-элементов, эквивалентных в том или ином смысле]
 - структура
 - := [sc-знак множества sc-элементов, в состав которого входят sc-связки или структуры, связывающие эти sc-элементы]
- ⇒ разбиение*:
 - {• четкое множество
 - нечеткое множество
- ⇒ разбиение*:
 - {• множество первичных сущностей
 - множество множеств
 - множество первичных сущностей и множеств
- ⇒ разбиение*:
 - {• рефлексивное множество
 - нерефлексивное множество
- ⇒ разбиение*:
 - {• сформированное множество
 - несформированное множество
- ⇒ разбиение*:
 - {• кортеж
 - неориентированное множество

}

Под **множеством** понимается соединение в некое целое M определённых хорошо различимых предметов из нашего созерцания или нашего мышления (которые будут называться «элементами» множества M).

множество – мысленная сущность, которая связывает одну или несколько сущностей в целое.

Более формально **множество** – это абстрактный математический объект, состоящий из элементов. Связь множеств с их элементами задается бинарным ориентированным отношением **принадлежность***.

Множество может быть полностью задано следующими тремя способами:

- путем перечисления (явного указания) всех его элементов (очевидно, что таким способом можно задать только конечное множество)
- с помощью определяющего высказывания, содержащего описание общего характеристического свойства, которым обладают все те и только те объекты, которые являются элементами (т.е. принадлежат) задаваемого множества.
- с помощью теоретико-множественных операций, позволяющих однозначно задавать новые множества на основе уже заданных (это операции объединения, пересечения, разности множеств и др.)

Для любого семантически ненормализованного **множества** существует единственное семантически нормализованное **множество**, в котором все элементы, не являющиеся знаками множеств, заменены на знаки множеств.

принадлежность*

- := [принадлежность элемента множеству*]
- := [отношение принадлежности элемента множеству*]
- ∈ бинарное отношение
- ∈ ориентированное отношение

принадлежность* – это бинарное ориентированное отношение, каждая связка которого связывает множество с одним из его элементов. Элементами отношения **принадлежность*** по умолчанию являются *позитивные sc-дуги принадлежности*.

класс

- := [класс sc-элементов]
- ⇒ разбиение*:
 - {• класс первичных sc-элементов
 - класс множеств

класс – множество элементов, обладающих какими-либо явно указываемыми общими свойствами.

кортеж

- := [вектор]

кортеж – это множество, представляющее собой упорядоченный набор элементов, т.е. такое множество, порядок элементов в котором имеет значение. Пары принадлежности элементов **кортежу** могут дополнительно принадлежать каким-либо *ролевым отношениям*, при этом, в рамках каждого **кортежа** должен существовать хотя бы один элемент, роль которого дополнительно уточнена *ролевым отношением*.

включение*

- := [включение множеств*]
- := [быть подмножеством*]
- ∈ бинарное отношение
- ∈ ориентированное отношение
- ∈ транзитивное отношение
- ⇒ область определения*:
 - множество
- ⊂ строгое включение*

включение* – это бинарное ориентированное отношение, каждая связка которого связывает два множества. Будем говорить, что *Множество Si включает** в себя *Множество Sj* в том и только том случае, если каждый элемент *Множества Sj* является также и элементом *Множества Si*.

объединение*

- \coloneqq [объединение множеств*]
- \in квазибинарное отношение
- \in ориентированное отношение

объединение* – это *квазибинарное ориентированное отношение*, областью определения которого является семейство всевозможных множеств. Будем говорить, что *Множество Si* является объединением *Множество Sj* и *Множество Sk* тогда и только тогда, когда любой элемент *Множество Si* является элементом или *Множество Sj* или *Множество Sk*.

разбиение*

- \coloneqq [разбиение множества*]
- \coloneqq [объединение попарно непересекающихся множеств*]
- \coloneqq [декомпозиция множества*]
- \in квазибинарное отношение
- \in ориентированное отношение
- \in отношение декомпозиции

разбиение* – это *квазибинарное ориентированное отношение*, областью определения которого является семейство всевозможных множеств. В результате разбиения множества получается множество попарно непересекающихся множеств, объединение которых есть исходное множество.

Семейство множеств $\{S_1 \dots, S_n\}$ является разбиением множества S_i в том и только том случае, если:

- семейство $\{S_1 \dots, S_n\}$ является семейством *попарно непересекающихся множеств*;
- семейство $\{S_1 \dots, S_n\}$ является покрытием множества S_i (или другими словами, множество S_i является *объединением* множеств, входящих в указанное выше семейство)

пересечение*

- \coloneqq [пересечение множеств*]
- \in квазибинарное отношение
- \in ориентированное отношение

пересечение* – это операция над множествами, аргументами которой являются два или большее число множеств, а результатом является множество, элементами которого являются все те и только те сущности, которые одновременно принадлежат каждому множеству, которое входит в семейство аргументов этой операции.

Будем говорить, что *Множество Si* является пересечением *Множество Sj* и *Множество Sk* тогда и только тогда, когда любой элемент *Множество Si* является элементом *Множество Sj* и элементом *Множество Sk*.

разность множеств*

- \in бинарное отношение
- \in ориентированное отношение

разность множеств* – это *бинарное ориентированное отношение*, связывающее между собой *ориентированную пару*, первым элементом которой является уменьшаемое множество, вторым – вычитаемое множество, и множество, являющееся результатом вычитания вычитаемого из уменьшаемого, в которое входят все элементы первого множества, не входящие во второе множество.

симметрическая разность множеств*

- \in бинарное отношение
- \in ориентированное отношение

симметрическая разность множеств* – это *бинарное ориентированное отношение*, связывающее между собой *пару* множеств и множество, являющееся результатом симметрической разности элементов указанной пары. Будем называть *Множество Si* результатом симметрической разности *Множества Sj* и *Множества Sk* тогда и только тогда, когда любой элемент *Множества Si* является или элементом *Множества Sj* или *Множества Sk*, но не является элементом обоих множеств.

декартово произведение*

- \coloneqq [декартово произведение множеств*]
- \coloneqq [прямое произведение множеств*]
- \in бинарное отношение
- \in ориентированное отношение

декартово произведение* – это *бинарное ориентированное отношение* между *ориентированной парой* множеств и множеством, элементами которого являются всевозможные упорядоченные пары, первыми элементами которых являются элементы первого из указанных множеств, вторыми – элементы второго из указанных множеств.

булеан*

- := [булеан множества*]
- := [семейство всевозможных подмножеств заданного множества*]
- ∈ *бинарное отношение*
- ∈ *ориентированное отношение*

булеан* – это *бинарное ориентированное отношение* между множеством и некоторым семейством множеств, каждое из которых является подмножеством первого множества.

мощность множества

- := [кардинальное число]
- := [общее число вхождений элементов в заданное множество]
- := [класс эквивалентности, элементами которого являются знаки всех тех и только тех множеств, которые имеют одинаковую мощность]
- := [класс эквивалентности, соответствующий отношению быть парой множеств, имеющих одинаковую мощность (равномощность множеств)]
- := [величина мощности множеств]
- := [трансфинитное число]
- := [мощность по Кантору]
- ∈ *параметр*

мощность множества – это *параметр*, элементами которых являются *множества*, имеющие одинаковое количество элементов. Значением данного параметра является числовая величина, задающая количество элементов, входящих в данный класс множеств, т.е. по сути, количество *позитивных sc-дуг принадлежности*, выходящих из данного *множества*.

Для двух множеств, имеющих одинаковую мощность, существует взаимно-однозначное соответствие между ними (между множествами вхождений элементов в эти множества – на случай мульти множеств).

§ 2.4.2. Формальная онтология связок и отношений

Предметная область связок и отношений

- ∈ *предметная область*

связь

- := [связка sc-элементов]
- := [sc-связка]

связь – множество, являющееся абстрактной моделью связи между описываемыми сущностями, которые или знаки которых являются элементами этого множества.

Напомним, что все элементы множества, представленного в SC-коде, являются знаками, но описываемыми сущностями могут быть не только сущности, обозначаемые sc-элементами, но и сами эти sc-элементы.

⇒ *разбиение*:*

- {• *бинарная связь*
- *небинарная связь*

}

⇒ *разбиение*:*

- {• *неориентированная связь*
- *ориентированная связь*

}

бинарная связь

⇒ *разбиение*:*

- {• *sc-коннектор*
- *неатомарная бинарная связь*

}

Данное разбиение осуществляется на основе синтаксического признака, а не семантического, поскольку каждый *sc-коннектор* может быть записан в памяти при помощи семантически эквивалентной конструкции, содержащей знак самой связи и пары принадлежности, ведущие к ее элементам, уточненные, при необходимости ролевыми отношениями.

sc-коннектор

:= [атомарная бинарная связь]

Каждый *sc-коннектор* представлен в *sc-памяти* одним *sc-элементом* и семантически эквивалентен конструкции, содержащей знак некоторой *бинарной связи* и пары принадлежности, ведущие к элементам этой связи, уточненные, при необходимости ролевыми отношениями.

Такая конструкция может быть обозначена *sc-коннектором* только в случае, когда роли компонентов соответствующей бинарной связи указываются только при помощи *числовых атрибутов* 1' и 2' или не уточняются вообще.

неатомарная бинарная связь

неатомарная бинарная связь – *бинарная связь*, роли компонентов которой не могут быть заданы только при помощи *ролевых отношений* 1' и 2', или не заданы совсем, а требуют дополнительного уточнения при помощи более частных ролевых отношений.

небинарная связь

небинарная связь – связь, имеющая больше двух элементов.

неориентированная связь

▷ *неориентированное множество*

⇒ *пояснение**:

[*неориентированная связь* – связь, все элементы которой имеют одинаковые роли (при этом соответствующее ролевое отношение, как правило, явно не указывается).]

ориентированная связь

▷ *кортеж*

⇒ *пояснение**:

[*ориентированная связь* – связь, в которой с помощью ролевых отношений, указываются роли компонентов этой связи.]

отношение

:= [класс связей]

:= [класс sc-связок]

:= [множество отношений]

:= [Множество всевозможных отношений]

⇒ *определение**:

[*отношение*, заданное на множестве M – это подмножество декартового произведения этого множества самого на себя некоторое количество раз]

В более широком смысле *отношение* – это математическая структура, которая формально определяет свойства различных объектов и их взаимосвязи.

⇒ *разбиение**:

{• класс равномощных связок
• класс связок разной мощности
}

⇒ *разбиение**:

{• бинарное отношение
• небинарное отношение
}

⇒ *разбиение**:

{• ориентированное отношение
• неориентированное отношение
}

\Rightarrow разбиение*:

- {• ролевое отношение
- неролевое отношение

}

класс равномощных связок

- \coloneqq [класс связок фиксированной арности]
 \coloneqq [отношение, обладающее свойством арности]
 \supset унарное отношение
 \supset бинарное отношение
 \supset тернарное отношение
 \Rightarrow определение*:

[**класс равномощных связок** – класс связок, имеющих одинаковую мощность.]

класс связок разной мощности

- \coloneqq [отношение нефиксированной арности]
 \subset небинарное отношение
 \Rightarrow определение*:

[**класс связок разной мощности** – класс связок, имеющих разную мощность.]

унарное отношение

- \coloneqq [отношение арности один]
 \coloneqq [одноместное отношение]
 \coloneqq [множество синглетонов]
 \Rightarrow определение*:

[**унарное отношение** – это множество таких отношений на множестве M , являющихся любым подмножеством множества M .]

бинарное отношение

- \coloneqq [отношение арности два]
 \coloneqq [двухместное отношение]
 \supset квазибинарное отношение
 \supset отношение порядка
 \supset отношение толерантности
 \Rightarrow разбиение*:
 - {• рефлексивное отношение
 - антирефлексивное отношение
 - частично рефлексивное отношение
}
- \Rightarrow разбиение*:
 - {• симметричное отношение
 - антисимметричное отношение
 - частично симметричное отношение
}
- \Rightarrow разбиение*:
 - {• транзитивное отношение
 - антитранзитивное отношение
 - частично транзитивное отношение
}
- \Rightarrow разбиение*:
 - {• ролевое отношение
 - неролевое бинарное отношение
}
- \Rightarrow определение*:

[**бинарное отношение** – это множество таких отношений на множестве M , являющихся подмножеством декартова произведения множества M .]

Если **бинарное отношение R** задано на множестве M и два элемента этого множества a и b связаны данным отношением, то будем обозначать такую связь как aRb .

квазибинарное отношение

- \Rightarrow пояснение*:

[**квазибинарное отношение** – множество ориентированных пар, первые компоненты которых являются связками.]

Таким образом, *sc-дуги*, принадлежащие **квазибинарным отношениям**, всегда выходят из связок.

⇒ *sc-утверждение**:

[В область определения квазибинарного отношения будем включать:

- вторые компоненты ориентированных пар, принадлежащих этому отношению;
- элементы первых компонентов ориентированных пар, принадлежащих этому отношению;
- других элементов область определения квазибинарного отношения не содержит.

]

небинарное отношение

⇒ *пояснение**:

[**небинарное отношение** – это множество отношений, хотя бы одна из связок каждого из которых имеет значение мощности больше двух.]

ориентированное отношение

⇒ *определение**:

[**ориентированное отношение** – это множество таких отношений, каждая связка которых является кортежем.]

неориентированное отношение

⇒ *определение**:

[**неориентированное отношение** – это множество таких отношений, каждая связка которых является неориентированным множеством.]

связанное отношение*

∈ **бинарное отношение**

⇒ *определение**:

[**связанное отношение*** *R* на множестве *A* – это **бинарное отношение**, в котором для каждой пары элементов *a* и *b* этого множества выполняется одно из двух отношений: *aRb* или *bRa*.]

отношение порядка

⇒ *разбиение**:

- {• *отношение строгого порядка*
- *отношение нестрогого порядка*

}

⇒ *определение**:

[**отношение порядка** – это **бинарное отношение**, обладающее свойством транзитивности и антисимметричности.]

отношение строгого порядка

⇒ *определение**:

[**отношение строгого порядка** – это **отношение порядка**, обладающее свойством антирефлексивности.]

отношение нестрогого порядка

⇒ *определение**:

[**отношение нестрогого порядка** – это **отношение порядка**, обладающее свойством рефлексивности.]

отношение толерантности

⇒ *определение**:

[**отношение толерантности** – это **бинарное отношение**, принадлежащее классам *симметричное отношение* и *рефлексивное отношение*.]

отношение эквивалентности

:= [максимальное семейство отношений эквивалентности]

⊂ **отношение толерантности**

⇒ *определение**:

[**отношение эквивалентности** – это **отношение толерантности**, принадлежащее классу *транзитивных отношений*]

⇒ *примечание**:

[Каждое отношение эквивалентности уточняет то, что мы считаем эквивалентными сущностями, т.е. то, на какие сходства этих сущностей мы обращаем внимание и какие их отличия мы игнорируем (не учитываем).]

ролевое отношение

:= [атрибут]

:= [атрибутивное отношение]

:= [отношение, которое задает роль элементов в рамках некоторого множества]

:= [отношение, являющееся подмножеством отношения принадлежности]

⇐ *семейство подмножеств**:

принадлежность*

⊂ бинарное отношение

▷ числовой атрибут

⇒ *пояснение*:

[**ролевое отношение** – это отношение, являющееся подмножеством отношения принадлежности.]

⇒ *правило идентификации экземпляров*:

[В конце каждого *идентификатора*, соответствующего экземплярам класса **ролевое отношение**, не являющегося системным, ставится знак “’”.

Например:

ключевой экземпляр'

Из-за ограничений в разрешенном алфавите символов, в системном идентификаторе не может быть использовать знак “’”, поэтому в начале каждого *системного идентификатора*, соответствующего экземплярам класса **ролевое отношение** ставится префикс “rrel_”.

Например:

rrel_key_sc_element]

числовой атрибут

:= [порядковый номер]

:= [номер компонента ориентированной связки]

∈ 1'; 2'; 3'; 4'; 5'; 6'; 7'; 8'; 9'; 10'

⇒ *пояснение*:

[**числовой атрибут** – *ролевое отношение*, задающее порядковый номер элемента некоторой ориентированной связки, не уточняя при этом семантику такой принадлежности. Во многих случаях бывает достаточно использовать числовые атрибуты, чтобы различать компоненты связки, семантика каждого из которых дополнительно оговаривается, например, при определении отношения, которому данная связка принадлежит.]

неролевое отношение

⇒ *разбиение*:

{• небинарное отношение
• неролевое бинарное отношение
}

⇒ *пояснение*:

[**неролевое отношение** – отношение, не являющееся подмножеством отношения принадлежности.]

⇒ *правило идентификации экземпляров*:

[В конце каждого *идентификатора*, соответствующего экземплярам класса **неролевое отношение**, не являющегося системным, ставится знак “*”.

Например:

*включение**

Из-за ограничений в разрешенном алфавите символов, в системном идентификаторе не может быть использовать знак “*”, поэтому в начале каждого *системного идентификатора*, соответствующего экземплярам класса **неролевое отношение** ставится префикс “nrel_”.

Например:

nrel_inclusion]

неролевое бинарное отношение

⇒ *пояснение*:

[**неролевое бинарное отношение** – бинарное отношение, не являющееся ролевым отношением.]

арность

\coloneqq [арность отношения]

\in параметр

\Rightarrow пояснение*:

[*арность* – это параметр, каждый элемент которого представляет собой класс *отношений*, каждая связка которых имеет одинаковую *мощность*. Значение данного *параметра* совпадает со значением *мощности* каждой из таких связок.]

\Rightarrow описание примера*:

область определения*

\coloneqq [область определения отношения*]

\in бинарное отношение

\Rightarrow пояснение*:

[*область определения** – это *бинарное отношение*, связывающее отношение со множеством, являющимся его областью определения.]

Областью определения отношения будем называть результат теоретико-множественного объединения всех связок этого отношения, или, другими словами, результат теоретико-множественного объединения всех множеств, являющихся доменами данного отношения.]

атрибут отношения*

\coloneqq [ролевой атрибут, используемый в связках заданного отношения*]

\in бинарное отношение

\Rightarrow пояснение*:

[*атрибут отношения** – это *бинарное отношение*, связывающее заданное отношение с *ролевым отношением*, используемым в данном отношении для уточнения роли того или иного элемента связок данного отношения.]

домен*

\coloneqq [домен отношения по заданному атрибуту*]

\in бинарное отношение

\Rightarrow пояснение*:

[*домен** – это *бинарное отношение*, связывающее связку отношения *атрибут отношения** со множеством, являющимся доменом заданного отношения по заданному атрибуту. Множество *di* является доменом отношения *ri* по атрибуту *ai* в том и только том случае, если элементами этого множества являются все те и только те элементы связок отношения *ri*, которые имеют в рамках этих связок атрибут *ai*.]

композиция отношений*

\in квазибинарное отношение

\Rightarrow определение*:

[*композиция отношений** – это *квазибинарное отношение*, связывающее два бинарных отношения с отношением, являющимся их композицией. Под композицией бинарных отношений *R* и *S* будем понимать множество $\{(x, y) | \exists z(xSz \wedge zRy)\}$]

метаотношение

\Rightarrow определение*:

[Метаотношение – это *отношение*, в каждой связке которого есть по крайней мере один компонент, являющийся знаком некоторого *отношения*.]

отношение декомпозиции

\ni разбиение*

\ni декомпозиция раздела*

\ni декомпозиция абстрактного объекта*

соответствие*

\coloneqq [наличие соответствия*]

\in бинарное отношение

\Rightarrow разбиение*:

{• *соответствие между непересекающимися множествами**
 • *соответствие между строго пересекающимися множествами**
 • *соответствие, область отправления и область прибытия которого совпадают**
 }

\Rightarrow разбиение*:

{• *всюду определенное соответствие**
 • *частично определенное соответствие**
 }

⇒ *разбиение*:*
 {• *сюръекция**
 • *несюръективное соответствие**
 }

⇒ *разбиение*:*
 {• *однозначное соответствие**
 • *неоднозначное соответствие**
 }

⇒ *определение*:*
 [*соответствие** – бинарное ориентированное отношение, каждая пара которого связывает два множества и указывает на наличие некоторого отношения, связывающего элементы этих двух множеств.]

отношение соответствия*∈ *бинарное отношение*⇒ *определение*:*[*отношение соответствия** – бинарное отношение, связывающее ориентированную пару множеств, на которых задано *соответствие** и некоторое подмножество декартова произведения* этих множеств.]**область отправления'**

:= [область отправления соответствия']

:= [область определения соответствия']

:= [первый компонент пары в отношении соответствия']

∈ *ролевое отношение*⇒ *определение*:*[*область отправления'* – ролевое отношение, указывающее на первый компонент пары в рамках отношения *соответствие**.]**область прибытия'**

:= [область прибытия соответствия']

:= [область значений соответствия']

∈ *ролевое отношение*⇒ *определение*:*[*область прибытия'* – ролевое отношение, указывающее на второй компонент пары в рамках отношения *соответствие**.]**образ'**

:= [образ соответствия']

∈ *ролевое отношение*⇒ *определение*:*[*образ'* – ролевое отношение, указывающее на второй компонент каждой пары в рамках множества пар, которое является вторым компонентом *отношения соответствия**.]**прообраз'**

:= [прообраз соответствия']

∈ *ролевое отношение*⇒ *определение*:*[*прообраз'* – ролевое отношение, указывающее на первый компонент каждой пары в рамках множества пар, которое является первым компонентом *отношения соответствия**.]**всюду определенное соответствие***

:= [полное соответствие*]

:= [наличие *всюду определенного соответствия**]⇒ *определение*:*[*всюду определенное соответствие** – это *соответствие**, при котором существует *образ'* для каждого элемента *области отправления'* данного *соответствия**.]**частично определенное соответствие***

:= [наличие частично определенного соответствия*]

⇒ *определение*:*

[частично определенное соответствие* – это *соответствие**, при котором существует *образ'* для некоторых, но не всех элементов *области отправления'* данного *соответствия**.]

сюръективное соответствие*

:= [наличие сюръективного соответствия*]

:= [сюръекция*]

⇒ *определение*:*

[сюръективное соответствие* – это *соответствие**, при котором существует *прообраз'* для каждого элемента *области прибытия'* данного *соответствия**.]

несюръективное соответствие*

:= [наличие несюръективного соответствия*]

⇒ *определение*:*

[несюръективное соответствие* – это *соответствие**, при котором не для каждого элемента *области прибытия'* данного *соответствия** существует *прообраз'*.]

однозначное соответствие*

:= [наличие однозначного соответствия*]

:= [функциональное соответствие*]

:= [функция*]

⇒ *определение*:*

[однозначное соответствие* – это *соответствие**, при котором каждому элементу из *области отправления'* соответствия ставится не более, чем один элемент из *области прибытия'* соответствия.]

множество сочетаний*

:= [множество всевозможных сочетаний*]

:= [множество всевозможных сочетаний заданной арности из элементов заданного множества*]

:= [множество всех неориентированных связок заданной арности*]

:= [множество всех подмножеств заданной мощности*]

:= [семейство всевозможных сочетаний*]

⇒ *определение*:*

[множество сочетаний* – отношение, связывающее некоторое множество и семейство всевозможных множеств, имеющих значение мощности, меньше либо равное мощности исходного множества и состоящих из тех же элементов, что и это множество.]

⇒ *утверждение*:*

[Мощность **множества сочетаний*** может быть вычислена как $n!/(k!(n-k)!)$, где n – мощность исходного множества, k – мощность элементов множества сочетаний.]

множество размещений*

⇒ *определение*:*

[множество размещений* – отношение, связывающее некоторое множество и семейство всевозможных кортежей, имеющих значение мощности, меньше либо равное мощности исходного множества и состоящих из тех же элементов, что и это множество.]

⇒ *утверждение*:*

[Мощность **множества размещений*** может быть вычислена как $n!/(n-k)!$, где n – мощность исходного множества, k – мощность элементов множества сочетаний.]

множество перестановок*

⊂ **множество размещений***

⇒ *определение*:*

[множество перестановок* – отношение, связывающее некоторое множество и семейство всевозможных кортежей, равномощных исходному множеству и состоящих из тех же элементов, что и это множество.]

⇒ *утверждение*:*

[Мощность **множества перестановок*** может быть вычислена как $n!$, где n – мощность исходного множества.]

§ 2.4.3. Формальная онтология параметров, величин и шкал

Предметная область параметров, величин и шкал

- := [Предметная область параметров и классов эквивалентности, являющихся их элементами (значениями, величинами)]
- ∈ предметная область
- ∃ максимальный класс объектов исследования':
параметр

параметр

- := [характеристика]
- := [свойство]
- := [класс классов]
- ⇒ разбиение*:
 - {• измеряемый параметр
 - неизмеряемый параметр

Каждый *параметр* представляет собой класс, являющийся семейством всевозможных классов эквивалентности или толерантности, задаваемых либо *отношением эквивалентности*, либо *отношением толерантности* (симметричным, рефлексивным, но частично транзитивным). Так, например, элементами (значениями, величинами) *параметра длины* являются либо классы эквивалентности, задаваемые отношением эквивалентности “иметь точно одинаковую длину*”, либо классы толерантности, задаваемые отношением вида “иметь приблизительно одинаковую длину с указываемой точностью*”, либо интервальные классы, задаваемые бинарными отношениями вида “иметь длину, находящуюся в одном и том же указываемом интервале*” (например, от 1 метра до 2 метров).

Примерами параметров как отношений эквивалентности являются:

- равновеликость геометрических фигур (по длине, площади, объему – в зависимости от размерности этих фигур);
- иметь одинаковый цвет (быть эквивалентными по цвету);
- эквивалентность, по вкусу, запаху, твердости и т.д.

Заметим, что среди элементов (значений, величин) параметра могут встречаться пересекающиеся множества (классы), но объединение всех элементов каждого параметра есть не что иное, как класс всевозможных сущностей, обладающих этим параметром (свойством, характеристикой). Например, класс всех сущностей, имеющих длину, класс всех сущностей, обладающих цветом.

Каждый конкретный параметр (характеристика), т.е. каждый элемент класса всевозможных параметров (характеристик) есть, по сути, признак классификации сущностей, обладающих это характеристикой, по принципу эквивалентности (одинаковости значения) этой характеристики. Например, параметр *цвет* разбивает множество сущностей имеющих цвет на классы, каждый из которых включает в себя сущности, имеющие одинаковый цвет. Параметр может разбиваться на классы для уточнения некоторого свойства, например элементами параметра цвет будут классы, соответствующие конкретным цветам (синий, красный и т.д.), в свою очередь каждый конкретный цвет может включать более частные классы, уточняющие данное свойство, например, темно-синий, светло-красный и т.д.

Другими словами, каждому множеству сущностей может ставиться в соответствие набор (семейство) параметров (параметрическое пространство), которыми обладают сущности этого множества – аналог семейства отношений, определенных (заданных) на этом множестве. Часто бывает важно построить такое параметрическое пространство, "точки" которого взаимнооднозначно соответствуют параметризуемым сущностям (например, набор параметров, позволяющих однозначно идентифицировать, установить личность каждого человека).

Таким образом, для каждого используемого элемента (значения) какого-либо параметра, необходимо явно указывать спецификацию этого значения (точное значение, неточное значение, интервальное значение, точность, интервал).

*область определения параметра**

- := [множество всех тех и только тех сущностей, которые являются компонентами значений заданного параметра*]
- := [множество всех тех и только тех сущностей, которые обладают заданным параметром*]
- ⇐ включение*:
объединение*

измеряемый параметр

- := [количественный параметр]
- := [семейство измеряемых величин]
- := [семейство классов эквивалентности, каждому из которых может быть поставлено в соответствие числовое значение]

Каждый ***измеряемый параметр*** представляет собой *параметр*, значение (элемент, экземпляр) которого трактуется как *величина*, которой можно поставить в соответствие ее числовое значение на основании выбранной единицы измерения и/или точки отсчета (нулевой отметки выбранной шкалы).

▷ *параметр, измеряемый по шкале*

неизмеряемый параметр

- := [качественный параметр]

ориентированный параметр

- := [упорядоченный параметр]
- ▷ *параметр, измеряемый по шкале*
- := [параметр, на значениях которого может быть задано некоторое отношение порядка, семантика которого уточняется в зависимости от семантики параметра]

величина

- := [значение количественного параметра]
- := [значение измеряемого параметра]
- := [класс сущностей, имеющих одинаковое значение соответствующего параметра]
- ⇒ *включение**:
 - *точная величина*
 - *неточная величина*
 - *интервальная величина*

Каждая ***величина*** представляет собой однозначный и независящий от шкалы измерения результат измерения некоторой характеристики у некоторой сущности.

Каждой ***величине*** можно поставить в соответствие ее числовое значение на основании выбранной единицы измерения и точки отсчета (нулевой отметки выбранной шкалы, в случае, если измерение осуществляется по шкале).

Нельзя путать значение параметра (***величину***) и значение величины по некоторой шкале, которое может быть скалярным и векторным.

точная величина

- := [точное значение параметра]
- := [множество всех точных значений параметра]
- := [значение параметра, являющееся семейством классов эквивалентности, соответствующим некоторому отношению эквивалентности]
- := [класс эквивалентности]

Каждая ***точная величина*** имеет одно фиксированное значение в некоторой единице измерения или по какой-либо шкале. При этом считается, что все элементы такого класса имеют одинаковое значение данного параметра и отклонениями можно пренебречь.

Каждой ***точной величине*** можно поставить в соответствие группу *неточных величин*, являющихся не разбиениями, а покрытиями того же множества, но с разной степенью точности.

неточная величина

- := [множество неточных значений параметра]
- := [приблизительная величина]
- := [приблизительное значение параметра]
- := [значение параметра в интервале с нефиксированными границами]

Каждой ***неточной величине*** ставится в соответствие ее значение в некоторой единице измерения или по какой-либо шкале, а также дополнительно указывается *точность**, т.е. возможное отклонение от данного значения.

интервальная величина

- := [интервальное значение параметра]
- := [значение параметра в интервале с фиксированными границами]

:= [интервал значения параметра из множества пересекающихся интервалов разной длины, имеющих нефиксированные границы]

Каждая **интервальная величина** представляет собой класс сущностей, находящихся в рамках точно заданного интервала, минимальная и максимальная точка которого являются *точными величинами*. Результатом *измерения** такой величины является ориентированная пара, первым компонентом которой является левая (меньшая) граница интервала, вторым компонентом – правая (большая) граница интервала.

эталон'

:= [образец']

\in ролевое отношение

Ролевое отношение **эталон'** указывает на тот элемент значения некоторого параметра, который в рамках данного класса эквивалентности считается эталонным, то есть он используется как образец при определении данного параметра.

эталон' может задаваться как для измеряемых, так и для неизмеряемых параметров, например, эталон метра или эталон красоты.

измерение*

:= [значение параметра*]

:= [значение заданной величины заданного параметра*]

:= [измерение как соответствие*]

:= [результат измерения заданной величины в заданной единице измерения и по заданной шкале*]

:= [бинарное ориентированное отношение, связывающее различные величины с результатами их измерения в различных единицах измерения и по различным шкалам*]

Связки отношения **измерение*** связывают величину и ее значение в некоторой единице измерения (в том числе, в интервале) или по некоторой шкале. Конкретная единица измерения или шкала указывается дополнительно при помощи соответствующего отношения. Одной величине может соответствовать только одно значение в каждой возможной единице измерения или одна точка на некоторой шкале.

точность*

:= [отклонение*]

:= [степень точности неточного значения параметра*]

\in бинарное отношение

Связки отношения **точность*** связывают *неточную величину* и *точную величину* того же класса, задающую максимальное возможное отклонение указанной *неточной величины* от своего значения.

единица измерения*

\in бинарное отношение

:= [единица по шкале*]

:= [единичная отметка по шкале*]

Связки отношения **единица измерения*** связывают знак конкретного *измерения с фиксированной единицей измерения* и некоторую *точную величину*, входящую в тот же конкретный *параметр*, что и первый компонент связок этого конкретного измерения, и которая используется в данном случае в качестве единицы измерения.

измерение с фиксированной единицей измерения

\Leftarrow семейство подмножеств*:

*измерение**

Каждая **измерение с фиксированной единицей измерения** представляет собой подмножество отношения *измерение** и характеризуется некоторой **единицей измерения***, которая является элементом того же параметра (семейством сущностей, имеющих значение данного параметра, совпадающее с этой единицей измерения).

измерение по шкале

:= [шкала]

\Leftarrow семейство подмножеств*:

*измерение**

Каждая **измерение по шкале** представляет собой подмножество отношения **измерение*** и характеризуется не единицей измерения, а некоторой точкой отсчета для данной **шкалы**. Результатом **измерения по шкале** будет некоторая точка шкалы, отстоящая от точки отсчета на определенное расстояние в нужную сторону (меньшую или большую). Понятно, что это расстояние может быть измерено любыми единицами измерения, но его величина при этом останется неизменной.

Не стоит путать измерение по **измерение по шкале**, которое зависит от **нулевой отметки***, с измерением изменения того же **параметра**, которое характеризуется единицей измерения и не зависит от точки отсчета. Например, не стоит путать дату по некоторому календарю, соответствующую **началу** какого-либо процесса, и **длительность** этого процесса, которая не зависит от выбранного календаря.

Каждое **арифметическое выражение на величинах** представляет собой **связку**, компонентами которой являются элементы или подмножества некоторого **количественного параметра**.

арифметическая операция на величинах

\Leftarrow семейство подмножеств*:
 \quad арифметическое выражение на величинах

Каждая **арифметическая операция на величинах** представляет собой **отношение**, элементами которого являются **арифметические выражения на величинах**, то есть множество **арифметических выражений на величинах** какого-либо одного вида.

сумма величин*

\coloneqq [сложение величин*]
 \in арифметическая операция на величинах
 \in квазибинарное отношение

сумма величин* – это **арифметическая операция на величинах**, аналогичная **арифметической операции сумма*** для чисел.

Первым компонентом связки отношения **сумма величин*** является подмножество некоторого **количественного параметра** (слагаемые **величины**), содержащее два или более элемента, вторым компонентом – элемент этого же **количественного параметра**, значение которого в любой **единице измерения*** является результатом сложения значений всех слагаемых **величин** в той же **единице измерения***. При несовпадении единиц измерения слагаемых величин необходимо воспользоваться соотношениями между **единицами измерения**, которые задаются при помощи операций **произведение величин*** и **возвведение величин в степень***.

произведение величин*

\coloneqq [умножение величин*]
 \in арифметическая операция на величинах
 \in квазибинарное отношение

произведение величин* – это **арифметическая операция на величинах**, аналогичная **арифметической операции произведения*** для чисел.

Первым компонентом связки отношения **произведение величин*** является **связка**, элементами которой являются либо **величины количественных параметров**, либо **числа**, но при этом хотя бы один элемент должен быть **величиной**. Вторым компонентом является **величина количественного параметра**.

Операция **произведение величин*** может быть использована для задания соотношения между **единицами измерения*** в рамках одного **количественного параметра**.

возвведение величин в степень*

\in арифметическая операция на величинах
 \in бинарное отношение

возвведение величин в степень* – это **арифметическая операция на величинах**, аналогичная **арифметической операции возведение в степень*** для чисел.

Первым компонентом связки отношения **возвведение величин в степень*** является ориентированная пара, первым компонентом которой является **величина количественного параметра** (основание степени), вторым – **число** (показатель степени); Вторым компонентом связки отношения **возвведение величин в степень*** является **величина количественного параметра** (результат возведения в степень).

действие. измерение

- \coloneqq [измерение как действие]
- \coloneqq [действие, направленное на установление связи, принадлежащей отношению измерение* и связывающей величину, которая принадлежит заданному параметру, и которой принадлежит заданная сущность, и соответствующее значение этой величины на некоторой шкале]
- \coloneqq [действие, направленное на решение задачи измерения заданного параметра у заданной сущности]
- \Leftarrow включение*:
действие

задача. измерение

- \coloneqq [спецификация действия измерения]
- \coloneqq [спецификация действия, целью которого является измерение заданного параметра у заданной сущности]
- \Leftarrow включение*:
задача

§ 2.4.4. Формальная онтология чисел и числовых структур**Предметная область чисел и числовых структур**

- \in предметная область
- \ni максимальный класс объектов исследования':
число

число

- \coloneqq [множество чисел]
- \subset абстрактная терминальная сущность

число – это основное понятие математики, используемое для количественной характеристики, сравнения, нумерации объектов и их частей. Письменными знаками для обозначения чисел служат *цифры*.

цифра

- \coloneqq [множество цифр]
- \subset внутренний файл ostis-системы
- \Rightarrow включение*:
 - арабская цифра
 - римская цифра

цифра – это множество файлов, обозначающих вхождение этой цифры во всевозможные записи чисел с помощью этой цифры.

натуральное число

- \coloneqq [множество натуральных чисел]
- \subset целое число

натуральное число – это подмножество множества *целых чисел*, которые используются при счете предметов.

целое число

- \coloneqq [множество целых чисел]
- \subset рациональное число
- \subset целое число

целое число – это подмножество множества *рациональных чисел*, получаемых объединением *натуральных чисел* с множеством чисел, *противоположных** *натуральным* и *нулюм*.

рациональное число

- \coloneqq [множество рациональных чисел]
- \subset действительное число
- \subset целое число

рациональное число – это число, представляемое *обыкновенной дробью*, где числитель — *целое число*, а знаменатель — *натуральное число*.

дробь

:= [множество дробей]

⇒ включение*:

- обыкновенная дробь
- десятичная дробь

дробь — это число, состоящее из одной или нескольких равных частей (долей) единицы

обыкновенная дробь

:= [множество обыкновенных дробей]

:= [множество простых дробей]

обыкновенная дробь — запись *рационального числа* в виде $\pm \frac{m}{n}$ или $\pm m/n$, где $n \neq 0$. Горизонтальная или косая черта обозначает знак деления, в результате которого получается частное. Делимое называется числителем дроби, а делитель — знаменателем.

десятичная дробь

:= [множество десятичных дробей]

десятичная дробь — разновидность дроби, которая представляет собой способ представления действительных чисел в виде $\pm d_m \dots d_1 d_0, d_{-1} d_{-2} \dots$, где , — десятичная запятая, служащая разделителем между целой и дробной частью числа, d_k — десятичные цифры.

иррациональное число

:= [множество иррациональных чисел]

⊂ действительное число

иррациональное число — это *вещественное число*, которое не является рациональным, то есть не может быть представлено в виде дроби, где числитель — *целое число*, знаменатель — *натуральное число*. Любое **иррациональное число** может быть представлено в виде бесконечной непериодической десятичной дроби.

действительное число

:= [вещественное число]

:= [множество вещественных чисел]

⇐ объединение*:

- {• рациональное число
- иррациональное число

}

⊂ комплексное число

⇒ разбиение*:

- {• положительное число
- отрицательное число
- {Нуль}

}

действительное число — это множество чисел, получаемое в результате объединения иррациональных и *рациональных чисел*.

арифметическое выражение

:= [множество арифметических выражений]

Каждое **арифметическое выражение** представляет собой *связку*, компонентами которой являются *числа* или *множества чисел*.

арифметическая операция

:= [множество арифметических операций]

⇐ семейство подмножеств*:

- арифметическое выражение

Каждая **арифметическая операция** представляет собой *отношение*, элементами которого являются *арифметические выражения*, то есть множество *арифметических выражений* какого-либо одного вида.

сумма*

- \coloneqq [сложение*]
- \in арифметическая операция
- \in квазибинарное отношение

сумма* – это арифметическая операция, в результате которой по данным числам (слагаемым) находится новое число (сумма), обозначающее столько единиц, сколько их содержится во всех слагаемых.

Первым компонентом связки отношения **сумма*** является **множество чисел** (слагаемых), содержащее два или более элемента, вторым компонентом – **число**, являющееся результатом сложения.

Отдельно отметим, что каждая связка отношения **сумма*** вида $a = b+c$ может также трактоваться и как запись о вычитании чисел, например $b = a-c$, в связи с чем **арифметическая операция** разности чисел отдельно не вводится.

произведение*

- \coloneqq [умножение*]
- \in арифметическая операция
- \in квазибинарное отношение

произведение* – это **арифметическая операция**, в результате которой один аргумент складывается столько раз, сколько показывает другой, затем результат складывается столько раз, сколько показывает третий и т.д.

Первым компонентом связки отношения **произведение*** является **множество чисел** (множителей), содержащее два или более элемента, вторым компонентом – **число**, являющееся результатом произведения.

Отдельно отметим, что каждая связка отношения **произведение*** вида $a = b*c$ может также трактоваться и как запись о делении чисел, например $b = a/c$, в связи с чем **арифметическая операция** деления чисел отдельно не вводится.

числовая структура

- \subset структура

числовая структура – структура, в состав которой входят знаки **арифметических выражений**, а также знаки их элементов и связи между выражениями и их элементами.

система счисления

- \in параметр

Каждая **система счисления** представляет собой класс синтаксически эквивалентных файлов, хранимых в памяти, каждый из которых может являться идентификатором какого-либо **числа**.

Каждая **система счисления** характеризуется алфавитом, т.е. конечным множеством символов (цифр), которые допускается использовать при построении файлов принадлежащих данной **системе счисления**.

§ 2.4.5. Формальная онтология темпоральных сущностей

Предметная область темпоральных сущностей

- \coloneqq [Предметная область темпоральных связей и отношений]
- \coloneqq [Предметная область временных сущностей]
- \in предметная область
- \ni максимальный класс объектов исследования':
временная сущность

временная сущность

- \coloneqq [временно существующая сущность]
- \coloneqq [нестационарная сущность]
- \coloneqq [сущность, имеющая и/или начало, и/или конец своего существования]
- \coloneqq [sc-элемент, являющийся знаком некоторой временно существующей сущности]
- \coloneqq [сущность, обладающая темпоральными характеристиками (длительностью, начальным моментом, конечным моментом и т.д.)]
- \Rightarrow разбиение*:
 - {• прошлая сущность

- настоящая сущность
 - будущая сущность
- }
- ⇒ разбиение*:
- {• временная связь
- темпоральная структура
- := [структура, содержащая хотя бы одну временную сущность]
- ⇒ включение*:
- структура
- ⇒ примечание*:
- [Следует отличать:
- временный характер самой структуры как sc-элемента;
 - временный характер sc-элементов, принадлежащих данной структуре, и сущностей, обозначаемых этими sc-элементами;
 - временный характер пар принадлежности, связывающих структуру с ее элементами.
-]
- := [структура, описывающая темпоральные свойства (свойства, связанные со временем) окружающей среды, частью которой являются также и различные базы знаний кибернетических систем (в том числе и собственная база знаний).]
- ⇒ разбиение*:
- {• ситуация
- := [статическая темпоральная структура]
 - процесс
 - := [динамическая структура]
 - := [динамическая темпоральная структура]
- }
- материальная сущность
- }
- ⇒ разбиение*:
- {• непрерывная временная сущность
- ⇒ разбиение*:
- {• точечная временная сущность
- := [атомарная временная сущность]
 - := [условно мгновенная временная сущность]
 - := [временная сущность, длительность существования которой в данном контексте считается несущественной (пренебрежительно малой)]
- длительная непрерывная временная сущность
- }
- дискретная временная сущность
- := [временная сущность, которая может быть декомпозирована на последовательность точечных временных сущностей]
- := [временная сущность, которой соответствует некоторый временной ряд параметров (состояний) точечных временных сущностей, на которые декомпозируется исходная временная сущность]
- прерывистая временная сущность
- := [временная сущность, являющаяся результатом соединения нескольких не только точечных временных сущностей]
- := [временная сущность с прерываниями]
- }
- ⇒ примечание*:
- [Следует отметить, что приведенная классификация *временных сущностей* характеризует не столько сами *временные сущности*, сколько наши знания о них и степень детализации знаний об этих сущностях, с которой они описаны в базе знаний. Так, если для решения конкретных задач не важно, как изменилась некоторая *временная сущность* в рамках какого-либо периода времени, а важно только ее начальное и конечное состояние, то она может рассматриваться как *точечная временная сущность*. Впоследствии же та же *временная сущность* может быть рассмотрена и описана с большей степенью детализации, и таким образом, уже не будет точечной.]

Следует отличать:

- временный характер сущности, обозначаемой *sc-элементом*;

- временный характер существования самого *sc-элемента* в рамках *sc-памяти*, поскольку в ходе обработки информации каждый *sc-элемент* может быть удален из *sc-памяти*;
- временный характер описываемых ситуаций, событий и самих процессов;
- временный характер хранения в *sc-памяти* тех *sc-конструкций*, которые являются самими описаниями соответствующих ситуаций, событий и процессов.

Следует отличать, например, *материальную сущность* (некоторый физический или, в частности, биологический объект) от различных динамических структур (*процессов*), которые с той или иной степенью детализации и в том или ином ракурсе отражают (описывают) динамику изменений этой *материальной сущности*.

При этом сам *процесс* как уточнение динамики некоторой последовательности ситуаций и событий, также является сущностью, принадлежащей к классу *временных сущностей*.

прошлая сущность

- := [сущность, существовавшая в прошлом времени]
- := [сущность прошлого времени]
- := [сущность, завершившая свое существование]

настоящая сущность

- := [сущность, существующая в текущий момент времени]
- := [сущность, существующая сейчас]
- := [сущность настоящего времени]

будущая сущность

- := [возможно будущая сущность]
- := [прогнозируемая времененная сущность]
- := [временная сущность, которая может существовать в будущем]
- := [сущность, которая может или должна начать свое существование в будущем времени]
- ⇒ *включение**:
инициированное действие

Каждой *будущей сущности* можно поставить в соответствие вероятность ее возникновения.

временная связь

- := [нестационарная связь]
- := [временно существующая связь]

Каждая *временная связь* представляет собой *связку*, принадлежащую множеству *временных сущностей*.

Понятие *временной связи* не следует путать с понятием *темпоральной связи*, которая сама является *постоянной сущностью*, описывающей то, как связаны во времени некоторые *временные сущности*.

ситуация

- := [состояние]
- := [временная структура]
- := [временно существующая структура]
- := [квазистационарная структура]
- := [состояние некоторой динамической системы, описываемое с некоторой степенью детализации (подробности)]
- := [квазистационарная структура, существующая временно (в течение некоторого отрезка времени)]

Под ситуацией понимается *структура*, содержащая, по крайней мере, один элемент, который является *временной сущностью*. Наличие в рамках ситуации нескольких *временных сущностей* говорит о том, что существует момент времени (в прошлом, настоящем или будущем), в который все они существуют одновременно.

процесс

- := [процесс преобразования некоторой временной сущности из одного состояния в другое]
- := [процесс перехода от одной ситуации к другой]
- := [абстрактный процесс]
- := [информационная модель некоторых преобразований]
- := [динамическая sc-модель]
- := [динамическая структура]
- ⇒ *включение**:
воздействие

Каждый **процесс** определяется (задается) либо возникновением или исчезновением связей, связывающих заданную *временную сущность* с другими сущностями, либо возникновением или исчезновением связей, связывающих части указанной *временной сущности* с другими сущностями.

Многим **процессам** можно поставить в соответствие *ситуацию*, которая является его *начальной ситуацией** и *ситуацию*, которая является его *конечной ситуацией**, то есть указать *ситуации*, переход между которыми осуществляется в ходе **процесса**.

При этом знаки тех *временных сущностей*, с которыми связаны изменения, описываемые некоторым **процессом**, входят в данный **процесс** как элементы и при необходимости уточняются дополнительными *ролевыми отношениями*.

⇒ *разбиение**:

- {• *процесс в sc-памяти*
 - *процесс во внешней среде ostis-системы*
- }

Каждой **материальной сущности** можно поставить в соответствие различные **процессы**, описывающие ее преобразование из одного состояния в другое.

Поскольку **процесс** представляет собой изменяющуюся во времени динамическую структуру, то полностью представить процесс в базе знаний в общем случае не представляется возможным. Однако, можно ввести sc-элемент, обозначающий конкретный процесс, с необходимой степенью детализации описать его декомпозицию на более частные подпроцессы и/или описать ситуации, соответствующие состояниям процесса в разные моменты времени. В данном случае можно провести некоторую аналогию с *бесконечными множествами*, все элементы которых физически не могут быть представлены в базе знаний одновременно, тем не менее, само множество и некоторые из его элементов могут быть описаны с необходимой степенью детализации.

воздействие

:= [процесс, осуществляющийся на основе (в результате) воздействия одной сущности на другую]

⇒ *включение**:

действие

Каждому **воздействию** может быть поставлена в соответствие (1) некоторая *воздействующая сущность**, т.е. сущность, осуществляющая **воздействие** (в частности, это может быть некоторое физическое поле), и (2) некоторый **объект воздействия***, т.е. сущность, на которую воздействие направлено. Если **воздействие** связано с *материальной сущностью*, то его объектом воздействия является либо сама эта *материальная сущность*, либо некоторая ее пространственная часть.

исходная ситуация*

:= [начальная ситуация процесса*]

:= [начальная ситуация*]

∈ *бинарное отношение*

⇒ *первый домен**:

процесс

⇒ *второй домен**:

ситуация

Связки отношения **исходная ситуация*** связывают некоторый **процесс** и некоторую *ситуацию*, являющуюся начальной для этого **процесса**, и, как правило, изменяемой в течение выполнения этого **процесса**.

Первым компонентом каждой связки отношения **исходная ситуация*** является знак **процесса**, вторым – знак начальной *ситуации*.

причинная ситуация*

⊂ *начальная ситуация**

Под причинной ситуацией понимается такая *начальная ситуация**, которая обладает достаточной полнотой для однозначного задания инициируемого **процесса**.

конечная ситуация*

:= [конечная ситуация процесса*]

:= [результативная ситуация*]

∈ *бинарное отношение*

⇒ *первый домен*:

процесс
 \Rightarrow *второй домен**:
ситуация

Связки отношения **конечная ситуация*** связывают некоторый *процесс* и некоторую *ситуацию*, ставшую результатом выполнения этого *процесса*, то есть его следствием.

Первым компонентом каждой связки отношения **конечная ситуация*** является знак *процесса*, вторым – знак *конечной ситуации*.

точечный процесс

\coloneqq [атомарный процесс]
 \coloneqq [условно мгновенный процесс]
 \coloneqq [процесс, длительность которого в данном контексте считается несущественной (пренебрежимо малой)]
 \subset точечная времененная сущность

элементарный процесс

\coloneqq [процесс, детализация которого на входящие в него подпроцессы в текущем контексте не производится]
 \supset точечный процесс

Элементарные процессы могут иметь длительность и, следовательно, не обязательно являются атомарными процессами.

Понятия *точечного процесса* и *элементарного процесса*, как и понятие *точечной временной сущности* в целом, характеризуют не столько характеристики самого *процесса*, сколько степень наших знаний о нем и степень детализации описания процесса в базе знаний. Так, очевидно, что любой процесс, протекающий в компьютерной системе, может быть при необходимости детализирован до уровня команд процессора, затем до уровня микропрограмм и даже до уровня физических процессов (изменения физических характеристик сигналов), однако чаще всего такая детализация не требуется.

событие

\subset точечная времененная сущность
 \coloneqq [точечная времененная сущность, являющаяся началом и/или завершением какой-либо временной сущности (например, процесса)]
 \coloneqq [границчная точка временной сущности]

начало*

\coloneqq [быть начальным событием заданной временной сущности*]
 \Rightarrow *первый домен**:
временная сущность
 \Rightarrow *второй домен**:
событие
 \coloneqq [быть начальной точечной временной частью заданной временной сущности*]

завершение*

\coloneqq [конец*]
 \coloneqq [быть конечным событием заданной временной сущности*]
 \coloneqq [быть конечной точечной временной частью заданной временной сущности*]
 \Rightarrow *первый домен**:
временная сущность
 \Rightarrow *второй домен**:
событие

событие*

\in бинарное отношение
 \Rightarrow пояснение*:

[Связки отношения **событие*** связывают знак процесса и ориентированную пару, первым компонентом которой является знак *начальной ситуации** данного процесса, вторым компонентом – знак *конечной ситуации** данного процесса.]

детализация процесса*

\coloneqq

[Бинарное ориентированное отношение, каждая связка которого связывает некоторый процесс с более детальным его описанием, что предполагает представление декомпозиции этого процесса на систему взаимосвязанных его подпроцессов (в том числе элементарных).]

⇒ *пример**:

Переход от процесса, соответствующего какой-либо программе, к рассмотрению декомпозиции этого процесса (протокола) в терминах языка программирования высокого уровня, затем переход для каждого из полученных подпроцессов (операторов языка высокого уровня) к детализации выполнения этих подпроцессов на уровне машинных операций, выполняемых процессором компьютера (на уровне ассемблера), и далее к детализации выполнения подпроцессов уровня машинных операций к подпроцессам на уровне языка микропрограммирования. Таким образом, детализация процесса может быть иерархической, вплоть до уровня элементарных процессов.

отношение

⇒ *разбиение**:

- {• класс временных связей
- класс постоянных связей
- класс временных и постоянных связей

}

класс временных связей

:= [отношение, все связки которого являются нестационарными]

В общем случае **класс временных связей** не является *ситуативным множеством*, поскольку факт принадлежности некоторой *временной связи* такому классу следует считать постоянным, а не временным, поскольку временность/постоянство связи и ее семантический тип, задаваемый классом (отношением), это принципиально разные параметры (характеристики, признаки) любой связи.

класс постоянных связей

:= [отношение, все связки которого являются стационарными]

класс временных и постоянных связей

:= [отношение, некоторые (но не все) связки которого являются нестационарными]

множество

⇒ *разбиение**:

- {• ситуативное множество
- неситуативное множество
- частично ситуативное множество

}

ситуативное множество

:= [полностью ситуативное множество]

Под **ситуативным множеством** понимается постоянное множество, у которого все выходящие из него связи принадлежности являются *временными сущностями*.

В частности, ситуативное множество может использоваться как вспомогательная динамическая структура, которая содержит элементы некоторых структур, обрабатываемые в данный момент, например, это может быть копия некоторого множества, из которой постепенно удаляются элементы по мере их просмотра и обработки. В случае, когда такая структура содержит всего один элемент, ее можно считать указателем на данный элемент, при этом в разные моменты времени это могут быть разные элементы.

последний добавленный sc-элемент'

∈ ролевое отношение

неситуативное множество

⇒ *пояснение**:

[Под **неситуативным множеством** понимается постоянное множество, у которого все выходящие из него связи принадлежности являются *постоянными сущностями*.]

частично ситуативное множество

⇒ *пояснение**:

[Под **частично ситуативным множеством** понимается постоянное множество, у которого некоторые (но не все) выходящие из него связи принадлежности являются *временными сущностями*.]

темпоральная связь

- := [связь во времени]
- := [постоянная связь, описывающая связь во времени между временными сущностями]

темпоральное отношение

- ⇐ семейство подмножеств*:
- темпоральная связь**
- := [класс темпоральных связей]
- := [отношение, задающее темпоральные связи между временными сущностями]
- Э **темпоральное включение***
- Э **темпоральное объединение***
- Э **темпоральная декомпозиция***
- Э **темпоральная последовательность***
- ⇒ разбиение*:
 - {• **темпоральная смежность***
 - **темпоральная последовательность с промежутком***
 - **темпоральная последовательность с пересечением***

темпоральное включение*

- ⇒ пояснение*:

[Связки отношения **темпоральное включение*** связывают две *временные сущности*, период существования одной из которых полностью включается в период существования второй.

Первым компонентом каждой связки отношения **темпоральное включение*** является знак *временной сущности*, длительность существования которой больше.]

- ▷ **темпоральная часть***
- ▷ **темпоральное включение без совпадения начальных и конечных моментов***
- ▷ **темпоральное совпадение***
- ▷ **темпоральное включение с совпадением начальных моментов***
- ▷ **темпоральное включение с совпадением конечных моментов***

темпоральная часть*

- := [этап (период) заданной временной сущности*]
- := [этап процесса существования временной сущности*]
- ▷ **начальный этап***
- ▷ **конечный этап***
- ▷ **промежуточный этап***
- ▷ **подпроцесс***
 - ⇒ **первый домен*:**
 - process
 - ⇒ **второй домен*:**
 - process

Связки отношения **темпоральная часть*** связывают две *временные сущности*, одна из которых является частью другой, например, действие и одно из его поддействий. Соответственно, период существования одной из этих сущностей всегда будет включаться в период существования другой (большой).

В отличие от более общего отношения **темпоральное включение***, связки которого могут связывать любые *временные сущности*, связки отношения **темпоральная часть*** связывают только *временные сущности*, одна из которых является частью другой.

следует отличать*

- Э {• **темпоральная часть***
 - ▷ **подпроцесс***
 - **темпоральное включение***
 - ⇒ **примечание*:**
- [Связь *темпорального включения** может связывать абсолютно разные *временные сущности*, существующие в общем случае в разных местах, а не только *временные сущности*, одна из которых

является частью другой. Хотя формально и можно объединить любые разные *временные сущности* в одну общую *временную сущность*, далеко не всегда имеет смысл это делать.]

}

temporalное включение без совпадения начальных и конечных моментов*

:= [строгое темпоральное включение*]

temporalное совпадение*

:= [совпадение начала и завершения*]

∈ отношение эквивалентности

temporalное объединение*

:= [преобразование нескольких временных сущностей в одну общую временную сущность, которая может оказаться прерывистой или даже дискретной*]

↔ аналог*:

объединение множеств*

⇒ примечание*:

[С формальной точки зрения объединять можно любые временные сущности. Но делать это надо только тогда, когда это имеет смысл, точно так же, как и в случае объединения множеств.]

temporalная декомпозиция*

:= [Темпоральное отношение, связывающее временную сущность и множество смежных во времени временных сущностей, которые являются темпоральными частями исходной сущности и результатом темпорального объединения которых является исходная сущность*]

↔ аналог*:

разбиение*

temporalная смежность*

:= [сразу позже*]

:= [смежность во времени*]

:= [строгая темпоральная последовательность (без темпорального промежутка)*]

:= [темпоральная последовательность без промежутка*]

temporalная последовательность с промежутком*

:= [позже*]

temporalная последовательность с пересечением*

начало[^]

:= [одновременность начинаний[^]]

:= [класс одновременно начавшихся сущностей[^]]

∈ параметр

Каждый элемент множества **начало** представляет собой класс *временных сущностей*, у которых совпадает момент начала их существования. Конкретное значение данного *параметра* может быть как *точной величиной*, так и *неточной величиной* или *интервальной величиной*.

завершение[^]

:= [конец[^]]

:= [одновременность завершений[^]]

:= [класс одновременно завершившихся сущностей[^]]

∈ параметр

Каждый элемент множества **завершение** представляет собой класс *временных сущностей*, у которых совпадает конечный момент их существования (момент завершения существования). Конкретное значение данного *параметра* может быть как *точной величиной*, так и *неточной величиной* или *интервальной величиной*.

одновременность[^]

:= [параметр, значениями (элементами) которого являются классы либо одновременно существующих (происходящих) точечных временных сущностей, одновременность которых рассматривается с заданной степенью точности, либо одновременно начинающихся и заканчивающихся длительных процессов]

Важно отметить, что элементами некоторого значения параметра *одновременности* с заданной точностью могут быть только те временные сущности, которые и начались, и завершились в течение периода времени, заданного указанным значением этого параметра, но при этом начало и завершение этих временных сущностей не обязательно должно совпадать с началом и завершением указанного периода времени. Так, например, можно ввести значение параметра *одновременности* “2022 год по Григорианскому календарю”, элементами которого будут все временные сущности, начавшие и закончившие свое существование в рамках 2022 года. При этом не обязательно, чтобы эти временные сущности начались именно в полночь 1 января 2022 года и закончились в полночь 1 января 2023 года, это могут быть временные сущности, существовавшие, например, в течение июля 2022 года.

следует отличать*

- Ξ {• *temporalное совпадение**
 - ∈ *отношение эквивалентности*
 - *одновременность[^]*
 - := [фактор-множество для отношения *temporalное совпадение**]

длительность[^]

- := [класс временных сущностей, имеющих одинаковую длительность[^]]
- ∈ *параметр*
- Ξ *тысячелетие*
- Ξ *век*
- Ξ *год*
- Ξ *месяц*
- Ξ *день*
- Ξ *час*
- Ξ *минута*
- Ξ *секунда*

Каждый элемент множества **длительность** представляет собой класс *временных сущностей*, у которых совпадает длительность их существования. Конкретное значение данного *параметра* может быть как *точной величиной*, так и *неточной величиной* или *интервальной величиной*.

§ 2.4.6. Формальная онтология ситуаций и событий, описывающих динамику баз знаний ostis-систем

Обработка информации в *sc-памяти* (т.е. динамика базы знаний, хранимой в *sc-памяти*) в конечном счете сводится:

- к появлению в *sc-памяти* новых актуальных *sc-узлов* и *sc-коннекторов*;
- к логическому удалению актуальных *sc-элементов*, т.е. к переводу их в неактуальное состояние (это необходимо для хранения протокола изменения состояния базы знаний, в рамках которого могут описываться действия по удалению *sc-элементов*);
- к возврату логически удаленных *sc-элементов* в статус актуальных (необходимость в этом может возникнуть при откате базы знаний к какой-нибудь ее прошлой версии);
- к физическому удалению *sc-элементов*;
- к изменению состояния актуальных (логически не удаленных *sc-элементов*) – *sc-узел* может превратиться в *sc-ребро*, *sc-ребро* может превратиться в *sc-дугу*, *sc-дуга* может поменять направленность, *sc-дуга* общего вида может превратиться в *константную стационарную sc-дугу принадлежности*, и т.д.;

Подчеркнем, что временный характер самого *sc-элемента* (т.к. он может появиться или исчезнуть) никак не связан с возможно временным характером сущности, обозначаемой этим *sc-элементом*. Т.е. временный характер самого *sc-элемента* и временный характер сущности, которую он обозначает – абсолютно разные вещи.

Таким образом, следует четко отличать динамику внешнего мира, описываемого базой знаний, а динамику самой базы знаний (динамику внутреннего мира). При этом очень важно, чтобы описание динамики базы знаний также входило в состав каждой базы знаний.

К числу понятий, используемых для описания динамики базы знаний относятся:

- логически удаленный *sc-элемент*;
- сформированное множество;

- вычисленное число;
- сформированное высказывание;

Предметная область ситуаций и событий, описывающих динамику баз знаний ostis-систем

- \coloneqq [Предметная область, описывающая динамику базы знаний, хранимой в sc-памяти]
- \in предметная область
- \exists максимальный класс объектов исследования':
ситуация

sc-элемент

- \Leftarrow разбиение*:
 - {• настоящий sc-элемент
 - логически удаленный sc-элемент
}

настоящий sc-элемент

- \in ситуативное множество

логически удаленный sc-элемент

- \in ситуативное множество

число

- \Rightarrow разбиение*:
 - {• невычисленное число
 - вычисленное число
}

невычисленное число

- \in ситуативное множество

вычисленное число

основное понятие

- \coloneqq [основное понятие для данной ostis-системы]
- \in ситуативное множество

К **основным понятиям** относятся те понятия, которые активно используются в системе и могут быть ключевыми элементами sc-агентов. К **основным понятиям** относятся также все неопределяемые понятия.

неосновное понятие

- \coloneqq [дополнительное понятие]
- \coloneqq [вспомогательное понятие]
- \coloneqq [неосновное понятие для данной ostis-системы]
- \in ситуативное множество

Каждое **неосновное понятие** должно быть строго определено на основе **основных понятий**. Такие **неосновные понятия** используются только для понимания и правильного восприятия вводимой информации, в том числе, для выравнивания онтологий. Ключевым элементом sc-агентов **неосновные понятия** быть не могут.

- \Rightarrow правило идентификации экземпляров*:

[В случае, когда некоторое понятие полностью перешло из **основных понятий** в неосновные, то есть стало **неосновным понятием**, и соответствующее ему **основное понятие** (через которое оно определяется) в рамках некоторого внешнего языка имеет одинаковый с ним основной идентификатор, то к идентификатору **неосновного понятия** спереди добавляется знак #. Если при этом соответствующее **основное понятие** имеет в идентификаторе знак \$, добавленный в процессе перехода, то этот знак удаляется. Если указанные понятия имеют разные основные идентификаторы в рамках этого внешнего языка, то никаких дополнительных средств идентификации не используется.]

Например:

#трансляция sc-текста
#scp-программа]

понятие, переходящее из основного в неосновное

\in *ситуативное множество*

понятие, переходящее из неосновного в основное

\in *ситуативное множество*

\Rightarrow *правило идентификации экземпляров**:

[В случае, когда текущее *основное понятие* и соответствующее ему *понятие, переходящее из неосновного в основное* в рамках некоторого внешнего языка имеют одинаковый основной идентификатор, то к идентификатору понятия, переходящего из неосновного в основное спереди добавляется знак \$. Если указанные понятия имеют разные основные идентификаторы в рамках этого внешнего языка, то никаких дополнительных средств идентификации не используется.

Например:

\$трансляция sc-текста

\$scp-программа]

специфицированная сущность

\Rightarrow *разбиение**:

- {• недостаточно специфицированная сущность
 - достаточно специфицированная сущность
 - средне специфицированная сущность
- }

К *достаточно специфицированным сущностям* предъявляются следующие требования:

- если сущность не является понятием, то для нее должны быть указаны
 - различные варианты обозначающих ее внешних знаков;
 - классы, которым она принадлежит;
 - связки, которыми она связана с другими сущностями (с указанием соответствующего отношения);
 - значения параметров, которыми она обладает;
 - те разделы базы знаний, в которых указанная сущность является ключевой;
 - предметные области, в которые данная сущность входит.
- если специфицированная сущность является понятием, то для нее должны быть указаны:
 - различные варианты внешних обозначений этого понятия;
 - предметные области, в которых это понятие исследуется;
 - определение понятия;
 - пояснения
 - разделы базы знаний, в которых это понятие является ключевым;
 - описание примера – пример экземпляра понятия.

событие в sc-памяти

\supset *событие*

элементарное событие в sc-памяти

\subset *событие в sc-памяти*

\Rightarrow *разбиение**:

- {• событие добавления sc-дуги, выходящей из заданного sc-элемента
 - событие добавления sc-дуги, входящей в заданный sc-элемент
 - событие добавления sc-ребра, инцидентного заданному sc-элементу
 - событие удаления sc-дуги, выходящей из заданного sc-элемента
 - событие удаления sc-дуги, входящей в заданный sc-элемент
 - событие удаления sc-ребра, инцидентного заданному sc-элементу
 - событие удаления sc-элемента
 - событие изменения содержимого файла
- }

Под *элементарным событием в sc-памяти* понимается такое *событие*, в результате выполнения которого изменяется состояние только одного sc-элемента.

точечный процесс

:= [атомарный процесс]

:= [условно мгновенный процесс]

:= [процесс, длительность которого в данном контексте считается несущественной (пренебрежимо малой)]

элементарный процесс

:= [процесс, детализация которого на входящие в него подпроцессы в текущем контексте не производится]

Глава 2.5.

Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий

⇒ *авторы**:

- Голенков В.В.
- Банцевич К.А.

⇒ *аннотация**:

[Глава посвящена онтологическому подходу к проектированию баз знаний интеллектуальных компьютерных систем нового поколения. Данный подход основан на представлении базы знаний как иерархической структуры взаимосвязанных предметных областей и их онтологий, построенных на базе онтологий верхнего уровня.]

⇒ *подраздел**:

- § 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем
- § 2.5.2. Формализация понятия структуры
- § 2.5.3. Формализация понятия семантической окрестности
- § 2.5.4. Формализация понятия предметной области
- § 2.5.5. Формализация понятия онтологии

⇒ *ключевое понятие**:

- база знаний
- онтология верхнего уровня
- знание
- структура
- семантическая окрестность
- предметная область
- онтология

⇒ *библиографическая ссылка**:

- ...

Введение в Главу 2.5.

Развитие информационных технологий привело к расширению многообразия используемой информации, и в следствии, к необходимости создания интеллектуальных систем, способных оперировать объемными информационными ресурсами. Важнейшими видами таких ресурсов являются базы знаний.

База знаний представляет собой систематизированную совокупность знаний, хранимую в памяти интеллектуальной компьютерной системы и достаточную для обеспечения целенаправленного (целесообразного, адекватного) функционирования (поведения) этой системы как в своей внешней среде, так и в своей внутренней среде (в собственной базе знаний).

Важным этапом разработки баз знаний интеллектуальных систем является их структуризация. Структуризация базы, т.е. выделение в ней различных связанных между собой подструктур, необходима по целому ряду причин. В частности, это необходимо для обеспечения их синтаксической совместимости, что подразумевает унификацию формы представления знаний.

На сегодняшний день существуют десятки моделей представления знаний. Каждая из которых адаптирована для представления знаний определенного вида, в то время как при создании интеллектуальных систем часто возникает необходимость представить различные виды знаний в рамках одной базы. Однако, в настоящее время ни одна из существующих моделей, взятых в отдельности, не может этого обеспечить.

В связи с этим возникает необходимость в создании универсальной структурированной модели представления знаний, которая позволила бы представлять любые виды знаний в унифицированном виде.

На сегодняшний день наиболее эффективным средством структуризации различных областей знаний являются онтологии. Суть онтологического подхода при проектировании базы знаний заключается в рассмотрении структуры базы знаний как иерархической системы выделенных предметных областей и соответствующих им онтологий.

Однако, онтологически существует множество способов, которыми можно описать реальный мир таким, каким он есть. Решением данной проблемы является использование при проектировании баз знаний интеллектуальных компьютерных систем онтологий верхнего уровня.

Грамотно построенная онтология верхнего уровня позволит обеспечить широкую синтаксическую совместимость между большим количеством онтологий для различных предметных областей. Поскольку термины предметно-ориентированных онтологий подчинены терминам онтологии более высокого уровня.

Перечень и анализ существующих онтологий верхнего уровня подробно представлен в Главе 2.4.. Из представленного анализа можно сделать вывод, что попытки создать универсальную онтологию верхнего уровня, способную обеспечить совместимость интеллектуальных компьютерных систем, не привели к ожидаемым результатам, поскольку имеют ряд ключевых недостатков:

- Каждая из представленных онтологий является монолитной структурой, в которой нет четкой локализации на отдельные небольшие онтологии.
Главной задачей при проектировании фрагментов баз знаний с использованием онтологического подхода является выделение онтологий таким образом, чтобы они давали возможность относительно независимой эволюции каждого фрагмента. Структура данных онтологий верхнего уровня представляет собой иерархию, состоящую из большого количества различных понятий. Данный вид структуризации приводит к ситуации, когда необходимость внесения изменений в одном месте обязательно повлечет за собой невозможность редактирования другой части онтологии. В силу вышесказанного данный вид структуризации делает онтологии неудобными для их использования при разработке различных интеллектуальных систем;
- Рассматриваемые онтологии верхнего уровня не являются частью комплексной технологии.
Поскольку рассматриваемые онтологии не являются частью какой-то комплексной технологии, они не могут рассматриваться как часть библиотеки многократно используемых компонентов. Что, в свою очередь, приводит к неудобствам в виде необходимости адаптации используемых онтологий для каждой конкретной системы.
- Не существует технологий проектирования баз знаний на основе данных онтологий верхнего уровня.
Отсутствие технологий проектирования баз знаний затрудняет разработку интеллектуальных систем.

Отсутствие удовлетворительного решения этих проблем приводит к несовместимости разрабатываемых интеллектуальных компьютерных систем. Исходя из вышесказанного, возникает необходимость в построении такой системы онтологий верхнего уровня, которая могла бы обеспечить синтаксическую совместимость между большим количеством онтологий различных предметных областей баз знаний интеллектуальных компьютерных систем.

Для решения указанных проблем необходимо согласовать трактовку таких понятий, как *структура, семантическая окрестность, предметная область, онтология*, поскольку данные понятия являются базовыми классами сущностей, составляющими основу для структуризации баз знаний интеллектуальных систем.

Онтологическая модель, построенная на основе данных понятий, станет Ядром базы знаний, обеспечивающим совместимость интеллектуальных систем, за счет унифицированного представления знаний. Следует отметить, что в зависимости от специфики разрабатываемых систем их базы знаний могут расширяться, однако, онтологическая модель, лежащая в основе Ядра, позволит обеспечить дальнейшую совместимость разрабатываемых систем.

§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем

В рамках модели баз знаний ostis-систем выделяются синтаксически корректные (для соответствующего языка) и семантически целостные информационные конструкции. Такие конструкции будем называть знаниями.

знание

- := [синтаксически корректная (для соответствующего языка) и семантически целостная информационная конструкция]
⊆ информационная конструкция
⇒ покрытие*:
 вид знаний
 := [Множество всевозможных видов знаний]

Тот факт, что семейство видов знаний является покрытием Множества всевозможных знаний, означает то, что каждое знание принадлежит по крайней мере одному выделенному нами виду знаний.

вид знаний

- ⊖ спецификация
 := [описание заданной сущности]
 ⊇ спецификация материальной сущности

- ▷ спецификация обратной сущности, не являющейся множеством
 - ▷ спецификация геометрической точки
 - ▷ спецификация числа
- ▷ спецификация множества
 - ▷ спецификация связи
 - ▷ спецификация структуры
 - ▷ спецификация класса
 - ▷ спецификация класса сущностей, не являющихся множествами
 - ▷ спецификация отношения
 - := [спецификация класса связей (связок)]
 - ▷ спецификация класса классов
 - ▷ спецификация параметра
 - ▷ спецификация класса структур
 - ▷ спецификация понятий
 - ▷ пояснение
 - ▷ определение
 - ▷ утверждение
 - := [утверждение, описывающее свойства экземпляров (элементов) специфицируемого понятия]
 - := [закономерность]
 - ▷ семантическая окрестность
 - ▷ однозначная спецификация
 - ▷ сравнительный анализ
 - ▷ достоинства
 - ▷ недостатки
 - ▷ структура специфицируемой сущности
 - ▷ принципы, лежащие в основе
 - ▷ обоснование предлагаемого решения
 - := [аргументация предлагаемого решения]

Э сравнение

Э высказывание

 - ▷ фактографическое высказывание
 - ▷ закономерность

Э формальная теория

Э предметная область

Э предметная область и онтология

 - := [предметная область и её онтология]
 - := [предметная область и соответствующая ей объединенная онтология]

Э метазнание

 - := [спецификация знания]
 - ▷ аннотация
 - ▷ введение
 - ▷ предисловие
 - ▷ заключение
 - ▷ онтология
 - ▷ онтология предметной области
 - ▷ структурная онтология предметной области
 - ▷ теоретико-множественная онтология предметной области
 - ▷ логическая онтология предметной области
 - ▷ терминологическая онтология предметной области
 - ▷ объединенная онтология предметной области

Э задача

 - := [спецификация действия]

Э план

Э протокол

Э результативная часть протокола

Э метод

Э технология

Э база знаний

Даже небольшой перечень видов знаний свидетельствует об огромном многообразии видов знаний.

Знания разделяются на *декларативные* и *процедурные*. Под *декларативными знаниями* понимаются знания, которые имеют только *денотационную семантику*, которая представляется в виде семантической *спецификации* системы понятий, используемых в этом знании. Под *процедурным знанием* понимается знание, имеющее не только *денотационную семантику*, но и *операционную семантику*, которая представляется в виде семейства *спецификаций агентов*, осуществляющих интерпретацию *процедурного знания*, направленную на решение некоторой инициированной задачи.

В рамках *Технологии OSTIS* также выделяются отношения, заданные на множестве знаний.

отношение, заданное на множестве знаний

- ∅ *дочернее знание**
 - := [знание, которое от "материнского" знания наследует все описанные там свойства объектов исследования]
 - ∅ *дочерний раздел**
 - := [частный раздел]*
 - ∅ *дочерняя предметная область и онтология**
- ∅ *спецификация**
 - := [быть знанием, которое является спецификацией (описанием) заданной сущности]
- ∅ *онтология**
 - := [быть семантической спецификацией заданного знания*]
- ∅ *семантическая эквивалентность**
- ∅ *следовательно**
 - := [логическое следствие*]
- ∅ *логическая эквивалентность**

Важным видом знаний являются базы знаний.

база знаний

- := [совокупность знаний, хранимых в памяти интеллектуальной компьютерной системы и достаточных для того, чтобы указанная система удовлетворяла соответствующим предъявляемым к ней требованиям (в частности, чтобы она имела соответствующий уровень интеллекта)]
- := [систематизированная совокупность знаний, хранимая в памяти интеллектуальной компьютерной системы и достаточная для обеспечения целенаправленного (целесообразного, адекватного) функционирования (поведения) этой системы как в своей внешней среде, так и в своей внутренней среде (в собственной базе знаний)]

Одним из основных факторов, определяющими качество интеллектуальной компьютерной системы, является качественная структуризация (систематизация) и стратификация базы знаний интеллектуальной компьютерной системы

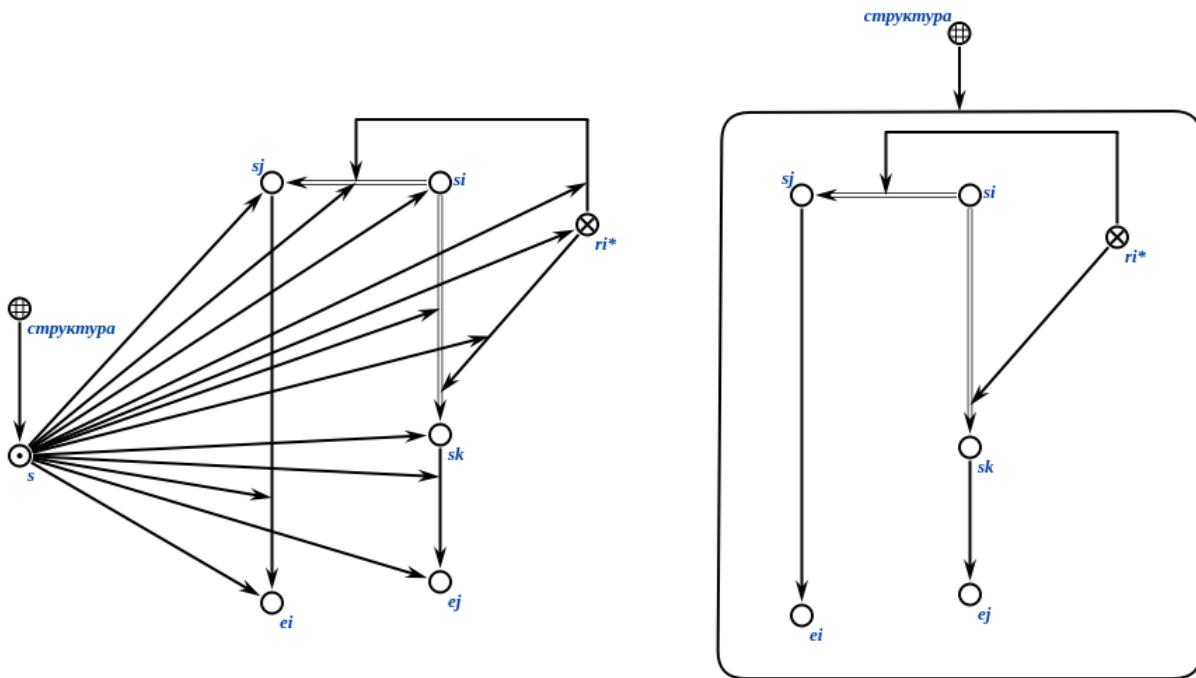
§ 2.5.2. Формализация понятия структуры

Существующие подходы к разработке *баз знаний* основываются на рассмотрении в качестве объектов спецификации конкретных элементов базы знаний (классов, экземпляров, отношений и др.). Однако при накоплении больших объемов информации в *базе знаний* возникает необходимость выделять целые фрагменты *базы знаний* и иметь возможность их специфицировать, рассматривая как отдельные сущности.

Такой фрагмент базы знаний назван *структурой* (*sc-структурой*). Под *структурой* будем понимать множество *sc-элементов*, удаление одного из которых может привести к нарушению целостности этого множества. Каждая *структура* представляет собой *sc-элемент*, обозначающий некоторый текст SC-кода, который может быть объектом спецификации, в том числе входить в состав других структур, быть связанным с другими сущностями различными отношениями.

Понятие *структуры* является основой для представления знаний, метазнаний и их структуризации, а также является одним из наиболее общих (с точки зрения уточнения семантики) понятий при описании свойств какого-либо объекта.

Структура может изображаться путем явного указания всех пар принадлежности элементов этой структуре, а также в виде контура, содержащего все элементы, входящие в состав этой структуры, что продемонстрировано на рисунке [Пример полного и сокращенного представления структуры в SCg-коде](#).



= Пример полного и сокращенного представления структуры в SCg-коде

Рассмотрим типологию *структур*, описываемых в базе знаний.

Структуре, представленной в SC-коде, поставим в соответствие орграф, вершинами которого являются sc-элементы, а дугами – связи отношений инцидентности, связывающие sc-коннекторы с инцидентными им sc-элементами, которые являются компонентами указанных sc-коннекторов. Если полученный таким способом орграф является связным орграфом, то исходную структуру будем считать *связной структурой*. Если полученный таким способом орграф не является связным орграфом, то исходную структуру будем считать *несвязной структурой*.

структура

- ⇒ разбиение*:
 - {• связная структура
 - несвязная структура

Под *тривиальной структурой* понимается *структура*, не содержащая в качестве элементов связок. В свою очередь, под *нетривиальной структурой* понимается *структура*, среди элементов которой есть хотя бы одна связка.

структура

- ⇒ разбиение*:
 - {• тривиальная структура
 - нетривиальная структура

По признаку стационарности/нестационарности выделяются *динамические структуры* (процессы) – структуры, состав которых меняется с течением времени, и *статические структуры* – структуры, состав которых не меняется с течением времени.

структура

- ⇒ разбиение*:
 - {• процесс
 - := [динамическая структура]
 - := [нестационарная структура]
 - статическая структура
 - := [стационарная структура]

196 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий

$\{ \quad := \text{ [структура, не изменяющаяся во времени]} \\ \}$

По признаку времени существования выделяются временные структуры и постоянно существующие структуры.

структура

\Rightarrow разбиение*:

$\{ \bullet \text{ временная структура} \\ := \text{ [(структура} \cap \text{временная сущность)}] \\ \bullet \text{ постоянно существующая структура} \\ := \text{ [(структура} \cap \text{постоянная сущность)}] \\ \}$

В рамках заданной структуры ее элементы можно классифицировать по заданным признакам:

- насколько полно в рамках заданной структуры представлено множество, обозначаемое заданным sc-элементом вместе с соответствующими дугами принадлежности;
- существуют ли в рамках заданной структуры sc-элементы, обозначающие множества, являющиеся надмножествами того множества, которое обозначается заданным sc-элементом;
- уровень («этаж») иерархии перехода от знаков к метазнакам для заданного sc-элемента в рамках заданной структуре.

Для формального представления структур используются понятия, описывающие роли элементов в рамках структуры. Элемент структуры' – неосновное понятие, ролевое отношение, указывающее на все элементы каждой структуры. Пример описания элементов структуры представлен на рисунке [Пример описания элементов структуры на SCg](#).

элемент структуры'

\Rightarrow разбиение*:

$\{ \bullet \text{ непредставленное множество}' \\ := \text{ [множество, не представленное в рамках данной структуры']} \\ \bullet \text{ полностью представленное множество}' \\ := \text{ [множество, полностью представленное в рамках данной структуры']} \\ \bullet \text{ частично представленное множество}' \\ := \text{ [множество, частично представленное в рамках данной структуры']} \\ \bullet \text{ элемент структуры, не являющийся множеством}' \\ \}$

\Rightarrow разбиение*:

$\{ \bullet \text{ максимальное множество}' \\ \bullet \text{ немаксимальное множество}' \\ \}$

максимальное множество'

\Rightarrow пояснение*:

[максимальное множество' – ролевое отношение, связывающее структуре со знаком множества, для которого не существует множества, которое было бы надмножеством указанного множества и знак которого был бы элементом этой же структуры.]

немаксимальное множество'

\Rightarrow пояснение*:

[немаксимальное множество' – ролевое отношение, связывающее структуре со знаком множества, для которого в рамках данной структуре существует множество, являющееся надмножеством указанного множества.]

первичный элемент'

$\begin{aligned} &:= \text{ [первичный элемент данной структуры']} \\ &:= \text{ [sc-элемент первого уровня в рамках данной структуры']} \\ &\in \text{ ролевое отношение} \\ &\subset \text{ элемент структуры'} \\ &\Rightarrow \text{ пояснение*}: \end{aligned}$

[**первичный элемент'** – ролевое отношение, указывающее на элемент *структурь*, являющийся либо терминальным элементом, либо знаком множества, такого что не существует другого элемента этой же структуры, который был бы элементом множества, обозначаемого первым из указанных элементов структуры. При этом соответствующая пара принадлежности может существовать, но в состав данной структуры не входить.]

вторичный элемент'

:= [вторичный элемент данной структуры']

:= [элемент данной структуры имеющий семантический уровень более 2']

:= [непервичный элемент']

∈ ролевое отношение

⊂ элемент структуры'

⇒ пояснение*:

[**вторичный элемент'** – ролевое отношение, указывающее на элемент структуры, обозначающий множество, все или некоторые элементы которого являются элементами указанной структуры.]

▷ элемент второго уровня'

элемент второго уровня'

∈ ролевое отношение

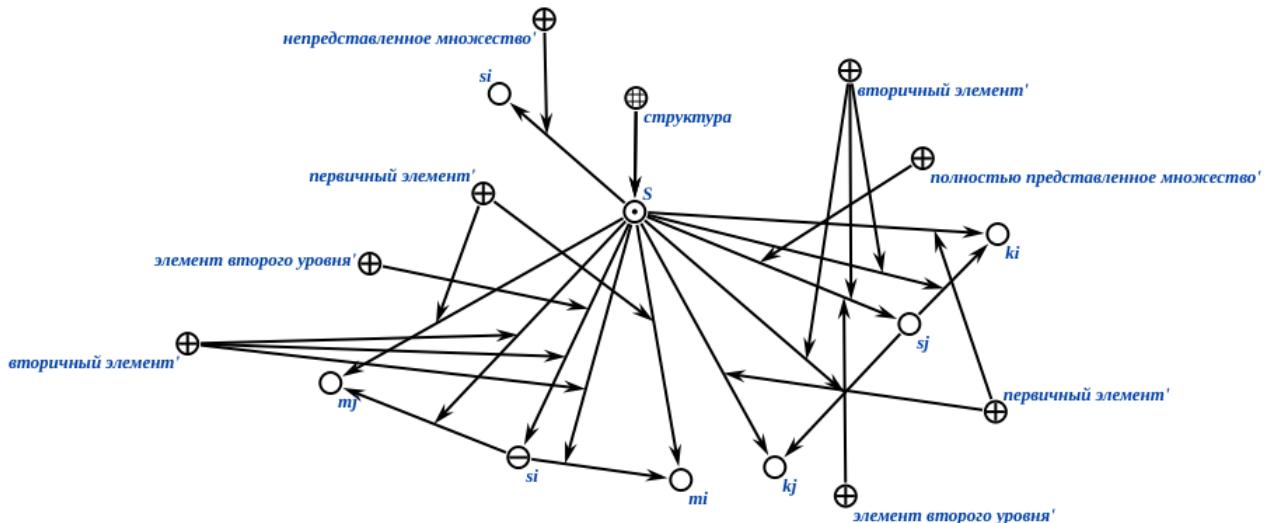
⇒ пояснение*:

[**элементом второго уровня'** в рамках заданной *структурь* может быть связка первичных элементов, тривиальная структура из первичных элементов или класс первичных элементов.]

структура второго уровня'

⇒ пояснение*:

[**структурой второго уровня** - структура, среди элементов которой есть хотя бы один **элемент второго уровня'**.]



= Пример описания элементов структуры на SCg

Тот факт, что в качестве формальной основы представления знаний в SC-коде лежит теория графов и теория множеств, позволяет анализировать не только внешние связи рассматриваемого фрагмента базы знаний с другими элементами базы знаний, но и внутреннюю структуру этих фрагментов с необходимой степенью детализации, т. е. выявлять в базе знаний аналогии, сходства, различия, строить различные виды соответствий между фрагментами.

Между *структурами* можно определять ряд соответствий, таких как *гомоморфизм*, *полиморфизм*, *автоморфизм*, *изоморфизм*, а также *аналогичность структур*, что позволяет фиксировать факт наличия некоторой аналогии, сходства и различия некоторых подструктур рассматриваемых *структур*.

полиморфность*

⊂ соответствие*

∈ бинарное отношение

⇒ пояснение*:

198 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий

[**полиморфность*** - это *соответствие*, заданное на *структур*, при котором каждому элементу из области определения соответствия (первой *структур*) ставится в соответствие один или более элемент из области значения соответствия (второй *структур*), при этом существует хотя бы один элемент области определения соответствия, которому соответствуют два или более элемента из области значения соответствия.]

полиморфизм*

∈ бинарное отношение

гомоморфность*

:= [гомоморфность структур*]

⊂ *соответствие**

∈ бинарное отношение

⇒ *пояснение*:*

[**гомоморфность*** - это *соответствие*, заданное на *структур*, при котором каждому элементу из области определения соответствия (первой *структур*) ставится в соответствие только один элемент из области значения соответствия (второй *структур*).]

гомоморфизм*

∈ бинарное отношение

изоморфность*

:= [изоморфное соответствие*]

:= [изоморфность структур*]

⊂ *гомоморфность**

∈ бинарное отношение

⇒ *пояснение*:*

[**изоморфность*** - это *гомоморфность**, при которой для каждого элемента из области значения существует ровно один соответствующий элемент из области определения.]

изоморфизм*

∈ бинарное отношение

автоморфность*

⊂ *гомоморфность**

∈ бинарное отношение

⇒ *пояснение*:*

[**автоморфность*** - это *изоморфность**, у которой область определения соответствия и область значения соответствия совпадают.]

автоморфизм*

∈ бинарное отношение

Отдельное внимание стоит уделить соответствуанию *аналогичность структур*, которое фиксирует факт наличия некоторой аналогии на подструктурах (подмножествах) указанных структур. Каждой ориентированной паре, принадлежащей *аналогичности структур*, может быть поставлено в соответствие множество пар, задающих *сходства* некоторых подструктур и *различия* некоторых подструктур исходных структур. Пример данного отношения представлен на рисунке [Пример отношения аналогичности структур](#).

аналогичность структур*

⊂ *соответствие**

∈ бинарное отношение

⇒ *пояснение*:*

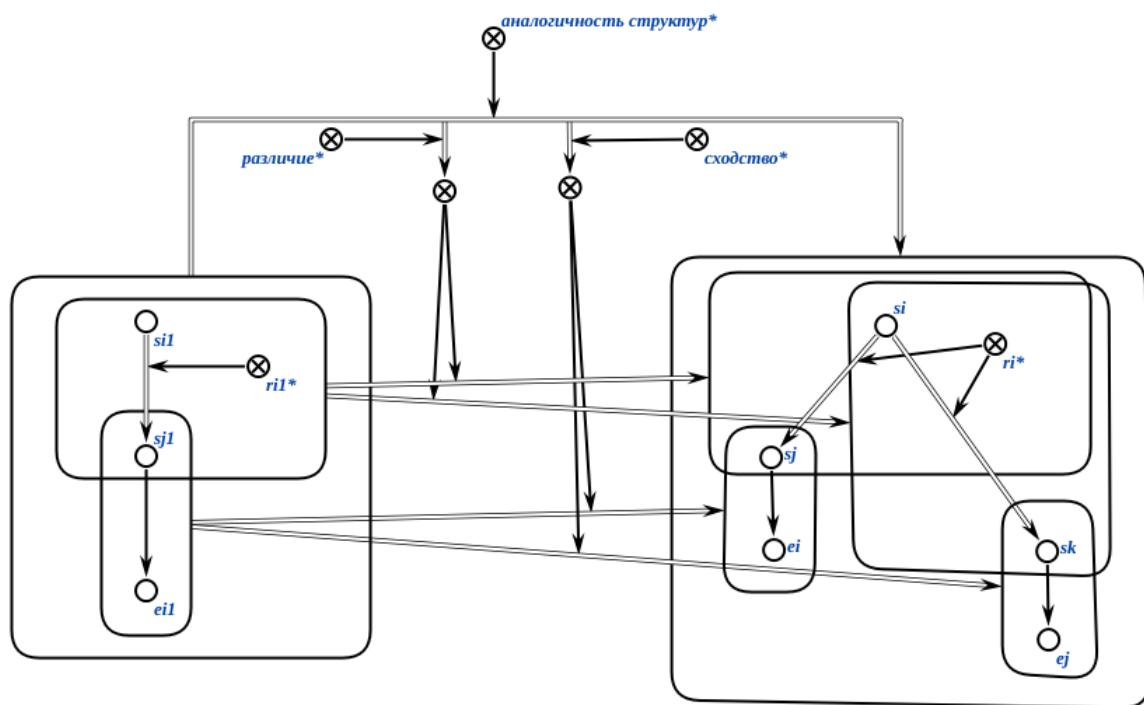
[**аналогичность структур*** - *соответствие**, задаваемое на структурах, и фиксирующее факт наличия некоторой аналогии на подструктурах (подмножествах) указанных структур. Каждой ориентированной паре, принадлежащей *аналогичности структур** может быть поставлено в соответствие множество пар, задающих *сходства** некоторых подструктур и *различия** некоторых подструктур исходных структур.]

сходство*

∈ бинарное отношение

различие*

\in бинарное отношение



= Пример отношения аналогичности структур

Соответствия на структурах являются частным случаем метазнаний. Используя такие соответствия, можно описывать в базе знаний и анализировать в дальнейшем, например, сходства и отличия различных фрагментов базы знаний, в том числе различных предметных областей.

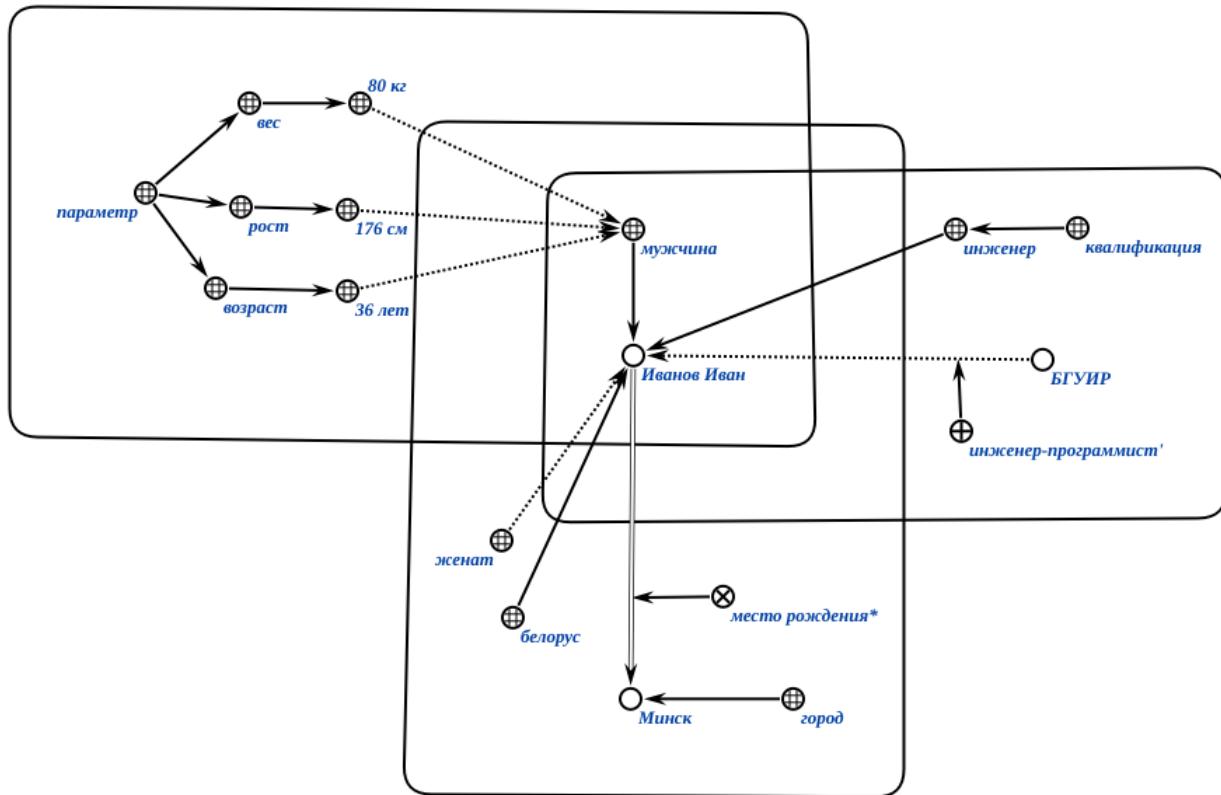
§ 2.5.3. Формализация понятия семантической окрестности

Для спецификации отдельных сущностей в рамках базы знаний вводится понятие *семантической окрестности*.

Семантическая окрестность представляет собой спецификацию заданной сущности, знак которой указывается как ключевой элемент этой спецификации. В отличие от других видов знаний, семантическая окрестность имеет только один ключевой элемент.

Набор признаков, по которым можно специфицировать сущности, различен. Кроме того, может возникнуть необходимость специфицировать одну и ту же сущность в различных аспектах и явно фиксировать эти аспекты в базе знаний.

Так, например, одну и ту же персону можно описывать с профессиональной, медицинской, гражданской и других точек зрения, как представлено на рисунке [Пример описания семантических окрестностей в SC-коде](#).



= Пример описания семантических окрестностей в SC-коде

Понятие *семантической окрестности*, как и любой другой *семантически* выделяемый класс знаний, абсолютно не зависит от языка представления знаний. Этим языком может быть не только SC-код или другой *формальный язык представления знаний* или даже *естественный язык*, тексты которых в памяти ostis-системы представляются в виде *файлов*.

Возможность описания различных свойств одного и того же объекта достигается за счет выделения различных классов семантических окрестностей и выделения набора признаков, определяющих тот или иной класс семантических окрестностей.

Перечислим основные виды *семантических окрестностей*.

семантическая окрестность

- := [sc-окрестность]
- := [семантическая окрестность, представленная в виде sc-текста]
- := [sc-текст, являющийся семантической окрестностью некоторого sc-элемента]
- := [спецификация заданной сущности, знак которой указывается как ключевой элемент этой спецификации]
- := [описание заданной сущности, знак которой указывается как ключевой элемент этой спецификации]
- С *знание*
- ▷ *семантическая окрестность по инцидентным коннекторам*
- ▷ *полная семантическая окрестность*
- ▷ *базовая семантическая окрестность*
- ▷ *специализированная семантическая окрестность*

семантическая окрестность по инцидентным коннекторам

- ▷ *семантическая окрестность по выходящим дугам*
- ▷ *семантическая окрестность по входящим дугам*

:= *пояснение**:

[вид *семантической окрестности*, в которую входят все коннекторы, инцидентные заданному элементу, а также все элементы, инцидентные указанным коннекторам.]

семантическая окрестность по выходящим дугам

- ▷ *семантическая окрестность по выходящим дугам принадлежности*

:= *пояснение**:

[вид семантической окрестности, в которую входят все дуги, выходящие из заданного sc-элемента и вторые компоненты этих дуг. Также указывается факт принадлежности этих дуг каким-либо отношениям.]

семантическая окрестность по выходящим дугам принадлежности

:= *пояснение**:

[вид семантической окрестности, в которую входят все дуги принадлежности, выходящие из заданного sc-элемента, а также их вторые компоненты. При необходимости может указываться факт принадлежности этих дуг каким-либо ролевым отношениям.]

семантическая окрестность по входящим дугам

▷ *семантическая окрестность по входящим дугам принадлежности*

:= *пояснение**:

[вид семантической окрестности, в которую входят все дуги, входящие в заданный sc-элемент, а также их первые компоненты. Также указывается факт принадлежности этих дуг каким-либо отношениям.]

семантическая окрестность по входящим дугам принадлежности

:= *пояснение**:

[вид семантической окрестности, в которую входят все дуги принадлежности, входящие в заданный sc-элемент, а также их первые компоненты. При необходимости может указываться факт принадлежности этих дуг каким-либо ролевым отношениям.]

Многообразие видов семантических окрестностей свидетельствует о многообразии семантических видов описаний различных сущностей.

Различают также полную и базовую семантические окрестности.

полная семантическая окрестность

:= [полная спецификация некоторой описываемой сущности]

Структура *полной семантической окрестности* определяется прежде всего семантической типологией описываемой сущности. Так, например, для понятия в *полную семантическую окрестность* необходимо включить следующую информацию (при наличии):

- варианты идентификации на различных внешних языках (sc-идентификаторы);
- принадлежность некоторой предметной области с указанием роли, выполняемой в рамках этой предметной области;
- теоретико-множественные связи заданного понятия с другими sc-элементами;
- определение или пояснение;
- высказывания, описывающие свойства указанного понятия;
- задачи и их классы, в которых данное понятие является ключевым;
- описание типичного примера использования указанного понятия;
- экземпляры описываемого понятия.

Для понятия, являющегося отношением дополнительно указываются:

- домены;
- область определения;
- схема отношения;
- классы отношений, которым принадлежит описываемое отношение.

базовая семантическая окрестность

:= [минимально достаточная семантическая окрестность]

:= [минимальная спецификация описываемой сущности]

Структура *базовой семантической окрестности* определяется прежде всего семантической типологией описываемой сущности. Так, например, для понятия в *базовую семантическую окрестность* необходимо включить следующую информацию (при наличии):

- варианты идентификации на различных внешних языках (sc-идентификаторы);
- принадлежность некоторой предметной области с указанием роли, выполняемой в рамках этой предметной области;
- определение или пояснение.

Для понятия, являющегося отношением дополнительно указываются:

202 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий

- домены;
- область определения;
- описание типичного примера связки указанного отношения (спецификация типичного экземпляра).

Также выделяется *специализированная семантическая окрестность* - вид *семантической окрестности*, набор связей для которой уточняется отдельно для каждого типа такой окрестности.

специализированная семантическая окрестность

- ▷ *пояснение*
 - := [sc-пояснение]
 - := *пояснение**:
 - [знак sc-текста, поясняющего описываемую сущность.]
- ▷ *примечание*
 - := [sc-примечание]
 - := *пояснение**:
 - [знак sc-текста, являющегося примечанием к описываемой сущности. В примечании обычно описываются особые свойства и исключения из правил для описываемой сущности.]
- ▷ *правило идентификации экземпляров*
 - := [правило идентификации экземпляров заданного класса]
 - := *пояснение**:
 - [sc-текст являющийся описанием правил построения идентификаторов элементов заданного класса.]
- ▷ *терминологическая семантическая окрестность*
 - := *пояснение**:
 - [семантическая окрестность, описывающая внешнюю идентификацию указанной сущности, т.е. её sc-идентификаторы]
- ▷ *теоретико-множественная семантическая окрестность*
 - := *пояснение**:
 - [описание связи описываемого множества с другими множествами с помощью теоретико-множественных отношений]
- ▷ *логическая семантическая окрестность*
 - := *пояснение**:
 - [семантическая окрестность, описывающая семейство высказываний, описывающих свойства данного понятия или какого-либо конкретного экземпляра некоторого понятия]
- ▷ *описание типичного экземпляра*
- ▷ *описание декомпозиции*

Понятие *семантической окрестности*, дополненное уточнением таких понятий, как семантическое расстояние между знаками (семантическая близость знаков), радиус семантической окрестности, является перспективной основой для исследования свойств смыслового пространства.

§ 2.5.4. Формализация понятия предметной области

Важнейшим этапом разработки баз знаний является процесс выделения описываемых *предметных областей* и их представления в базе знаний.

Понятие **предметной области** является важнейшим методологическим приемом, позволяющим выделить из всего многообразия исследуемого Мира только определенный класс исследуемых сущностей и только определенное семейство отношений, заданных на указанном классе. То есть осуществляется локализация, фокусирование внимания только на этом, абстрагируясь от всего остального исследуемого Мира.

Каждой *предметной области* можно поставить в соответствие:

- семейство соответствующих ей онтологий разного вида;
- множество семантических окрестностей, описывающих объекты исследования этой предметной области.

Предметные области являются основой структуризации смыслового пространства, средством локализации, фокусирования внимания на свойствах наиболее важных классов описываемых сущностей, которые становятся классами объектов исследования в предметных областях.

предметная область

- := [sc-модель предметной области]
- := [sc-текст предметной области]

- \coloneqq [sc-граф предметной области]
- \coloneqq [представление предметной области в SC-коде]
- \subset знание
- \subset бесконечное множество

Предметная область – это результат интеграции (объединения) частичных семантических окрестностей, описывающих все исследуемые сущности заданного класса и имеющих одинаковый (общий) предмет исследования (то есть один и тот же набор отношений, которым должны принадлежать связи, входящие в состав интегрируемых семантических окрестностей).

Предметная область - структура, в состав которой входят:

- основные исследуемые (описываемые) объекты – первичные и вторичные;
- различные классы исследуемых объектов;
- различные связи, компонентами которых являются исследуемые объекты (как первичные, так и вторичные), а также, возможно, другие такие связи – то есть связи (как и объекты исследования) могут иметь различный структурный уровень;
- различные классы указанных выше связок (то есть отношения);
- различные классы объектов, не являющихся ни объектами исследования, ни указанными выше связками, но являющимися компонентами этих связок.

При этом все классы, объявленные исследуемыми понятиями, должны быть полностью представлены в рамках данной предметной области вместе со своими элементами, элементами элементов и т.д. вплоть до терминальных элементов.

Каждому типу знаний можно поставить в соответствие предметную область, которая является результатом интеграции всех знаний данного типа. Эти знания становятся объектами исследования в рамках указанной предметной области.

предметная область

- \coloneqq [система связей некоторого множества объектов исследования, ключевыми элементами которой являются:
 - классы (точнее, знаки классов) объектов исследования (объектов, описываемых этой предметной областью);
 - конкретные объекты исследования, обладающие особыми свойствами;
 - классы связей, входящих в состав рассматриваемой системы – отношения, заданные на множестве элементов рассматриваемой системы;
 - параметры, заданные на множестве элементов рассматриваемой системы;
 - классы структур, являющихся фрагментами рассматриваемой системы.
]
- \coloneqq [структура, представляющая собой множество связей (точнее, знаков связей) и соответствующее множество компонентов этих связей, к числу которых относится:
 - элементы (экземпляры) некоторых заданных классов объектов исследования (первичных исследуемых сущностей);
 - сами связи, входящие в состав указанной структуры;
 - введенные классы объектов исследования;
 - введенные отношения (классы связей);
 - введенные параметры (классы классов эквивалентных сущностей);
 - значения параметров (и, в частности, величины для измеряемых параметров);
 - введенные структуры, являющиеся фрагментами (подструктурами) рассматриваемой структуры;
 - введенные классы подструктур рассматриваемой структуры.
]

Выделяемые в рамках *базы знаний* интеллектуальной системы *предметные области* и соответствующие им *онтологии* – это, своего рода, семантические страты, кластеры, позволяющие "разложить" все хранимые в памяти знания по "семантическим полочкам" при наличии четких критериев, позволяющих однозначно определить то, на какой "полочке" должны находиться те или иные знания.

По уровню исследовательского внимания понятия в рамках предметной области могут выполнять следующие роли:

роль элемента предметной области

- := [ролевое отношения, связывающее предметные области с их ключевыми знаками]
- := [роль ключевого элемента (знака ключевой сущности) предметной области]
- := [роль ключевого знака предметной области]
- Э *класс объектов исследования'*
 - := [быть классом первичных (для данной предметной области) объектов исследования']
- Э *максимальный класс объектов исследования'*
 - := [класс объектов исследования, для которого в заданной (!) предметной области отсутствует другой класс объектов исследования, который был бы его надмножеством']
- Э *ключевой объект исследования'*
 - := [особый объект исследования']
 - := [быть знаком особого исследуемого объекта в рамках заданной предметной области']
 - := [объект исследования, обладающий особыми свойствами']
- Э *понятие, используемое в предметной области'*
 - := [понятие, используемое в заданной предметной области не в качестве одного из объектов исследования, а в качестве ключевого понятия']
- Э *первичный исследуемый элемент предметной области'*
 - := [знак первичного объекта исследования в рамках заданной предметной области']
- Э *вторичный исследуемый элемент предметной области'*
 - := [знак вторичного объекта исследования в рамках предметной области']
- Э *неисследуемый элемент предметной области'*
 - := [вспомогательный элемент предметной области, исследуемый в другой (смежной) предметной области']

Выделяются следующие типы предметных областей:

предметная область

- ⇒ разбиение*:
 - {• *статическая предметная область*
 - := [стационарная предметная область]
 - := [*предметная область*, в которой связи между сущностями, входящими в ее состав, не зависят от времени (не меняются во времени), элементами **статической предметной области** не могут быть *временные сущности*]
 - *квазистатическая предметная область*
 - := [*предметная область*, решение задач в которой не требует учета темпоральных свойств объектов исследования]
 - *динамическая предметная область*
 - := [нестационарная предметная область]
 - := [*предметная область*, которая описывает изменение состояния (в том числе внутренней структуры) объектов исследования и/или изменение конфигурации связей между объектами исследования]
 - := [*предметная область*, в которой некоторые связи между сущностями, входящими в ее состав, меняются со временем (то есть носят ситуационный, нестационарный характер, другими словами, являются *временными сущностями*)]
 - }
- ⇒ разбиение*:
 - {• *первичная предметная область*
 - := [*предметная область*, объектами исследования которой являются внешние сущности (обозначаемые первичными sc-элементами)]
 - *вторичная предметная область*
 - := [метапредметная область]
 - := [*предметная область*, объектами исследования которой являются *sc-множества* (отношения, параметры, структуры, классы структур, знания, языки и др.)]
- }

Во всем многообразии предметных областей особое место занимают:

- *Предметная область предметных областей*, объектами исследования которой являются всевозможные предметные области, а предметом исследования являются – всевозможные ролевые отношения, связывающие предметные области с их элементами, отношения, связывающие предметные области между собой, отношение, связывающее предметные области с их онтологиями;
- *Предметная область сущностей*, являющаяся предметной областью самого высокого уровня и задающая базовую семантическую типологию sc-элементов (знаков, входящих в тексты SC-кода);

- Семейство *предметных областей*, каждая из которых задает семантику и синтаксис некоторого *sc-языка*, обеспечивающего представление *онтологий* соответствующего вида (например, теоретико множественных онтологий терминологических онтологий);
- Семейство *предметных областей верхнего уровня*, в которых классами объектов исследования являются весьма "крупные" классы сущностей. К таким классам, в частности, относятся:
 - класс всевозможных материальных сущностей,
 - класс всевозможных множеств,
 - класс всевозможных связей,
 - класс всевозможных отношений,
 - класс всевозможных структур,
 - класс всевозможных темпоральных (нестационарных) сущностей,
 - класс всевозможных действий (воздействий, акций),
 - класс всевозможных параметров (характеристик),
 - класс знаний всевозможного вида и т.п.

Важно отметить, что *предметную область* также можно считать *семантической окрестностью*, если считать ее центром знак сущности, которая является максимальным классом объектов исследования.

§ 2.5.5. Формализация понятия онтологии

Для формальной спецификации соответствующей предметной области, ориентированной на описание свойств и взаимосвязей понятий, входящих в состав указанной предметной области, используется такой вид знаний, как *онтология*.

Онтологии являются важнейшим *видом знаний*, обеспечивающих семантическую систематизацию *знаний*, хранимых в памяти *интеллектуальных компьютерных систем* (в т. ч. *ostis-систем*), и, соответственно, семантическую структуризацию *баз знаний*.

онтология

- := [sc-онтология]
- := [семантическая спецификация любого знания, имеющего достаточно сложную структуру, любого целостного фрагмента базы знаний – предметной области, метода решения сложных задач некоторого класса, описания истории некоторого вида деятельности, описания области выполнения некоторого множества действий (области решения задач), языка представления методов решения задач и т.д.]
- := [семантическая спецификация некоторого достаточно информативного ресурса (знания)]
- С спецификация
- С метазнание
- ∈ вид знаний
- := [важнейший вид метазнаний, входящих в состав базы знаний]
- := [спецификация (уточнение) системы понятий, используемых в соответствующем (специфицируемом) знании]

Онтология включает в себя:

- типологию специфицируемого знания;
- связи специфицируемого знания с другими знаниями;
- спецификацию ключевых понятий, используемых в специфицируемом знании, а также ключевых экземпляров некоторых таких понятий.

Важно отметить, что если *спецификация* может специфицировать (описывать) любую *сущность*, то *онтология* специфицирует только различные *знания*. При этом наиболее важными объектами такой спецификации являются *предметные области*.

Основная цель построения *онтологии* – семантическое уточнение (пояснение, а в идеале – определение) такого семейства знаков, используемых в заданном *знании*, которых достаточно для понимания смысла всего специфицируемого *знания*. Как выясняется, количество знаков, смысл которых определяет смысл всего специфицируемого *знания*, не является большим.

онтология

- ⇒ разбиение*:
 - {• неформальная онтология
 - формальная онтология
 - := [онтология, представленная на формальном языке]
 - :=

206 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий

[формальное описание денотационной семантики (семантической интерпретации) специфицируемого знания]

}

Очевидно, что при отсутствии достаточно полных формальных онтологий невозможно обеспечить семантическую совместимость (интегрируемость) различных знаний, хранимых в базе знаний, а также приобретаемых извне.

Онтология чаще всего трактуется как спецификация концептуализации (спецификация системы *понятий*) заданной *предметной области*. Здесь имеется в виду описание теоретико-множественных связей (прежде всего, классификации) используемых *понятий*, а также описание различных закономерностей для сущностей, принадлежащих этим *понятиям*. Тем не менее, важными видами спецификации *предметной области* являются также:

- описание связей специфицируемой *предметной области* с другими *предметными областями*;
- описание терминологии специфицируемой *предметной области*.

онтология предметной области

:= [описание денотационной семантики языка, определяемого (задаваемого) соответствующей (специфицируемой) *предметной области*]
:= [информационная надстройка (метаинформация) над соответствующей (специфицируемой) *предметной областью*, описывающая различные аспекты этой *предметной области* как достаточно крупного, самодостаточного и семантически целостного фрагмента *базы знаний*]
:= [метаинформация (метазнание) о некоторой *предметной области*]

Онтологию предметной области можно трактовать, с одной стороны, как *семантическую окрестность* соответствующей *предметной области*, с другой стороны, как *объединение* определённого вида *семантических окрестностей* всех *понятий*, используемых в рамках указанной *предметной области*, а также, возможно, ключевых экземпляров указанных *понятий*, если таковые экземпляры имеются.

Каждая конкретная онтология заданного вида представляет собой семантическую окрестность соответствующей (специфицируемой) предметной области. Каждому *виду онтологий* однозначно соответствует *предметная область*, фрагментами которых являются конкретные *онтологии* этого вида. Следовательно, каждому *виду онтологий* соответствует свой специализированный sc-язык, обеспечивающий представление *онтологий* этого вида.

онтология предметной области

⇒ разбиение*:
{• частная онтология предметной области
 := [онтология, представляющая спецификацию соответствующей предметной области в том или ином аспекте]
• объединённая онтология предметной области
 := [онтология *предметной области*, являющаяся результатом объединения всех известных частных онтологий этой предметной области]
}

Каждая *частная онтология* является фрагментом *предметной области*, включающей в себя все(!) частные онтологии, принадлежащие соответствующему *виду онтологии*. При этом указанная *предметная область*, в свою очередь, также имеет соответствующую ей *онтологию*, которая уже является не метазнанием (как любая онтология), а метаметазнанием (спецификацией метазнания).

частная онтология предметной области

⇒ разбиение*:
{• структурная спецификация предметной области
 := [вид метазнаний, описывающих соответствующие этому виду метазнаний свойства *предметных областей*]
 := [схема предметной области]
• теоретико-множественная онтология предметной области
 := [sc-спецификация заданной предметной области в рамках *Предметной области множеств*]
• логическая онтология предметной области
 := [sc-текст формальной теории заданной предметной области]
• терминологическая онтология предметной области
}

структурная спецификация предметной области

:= [структурная онтология предметной области]

- := [ролевая структура ключевых элементов предметной области]
- := [схема ролей понятий предметной области и её связи со смежными предметными областями]
- := [схема предметной области]
- := [спецификация предметной области с точки зрения теории графов и теории алгебраических систем]
- := [описание внутренней (ролевой) структуры *предметной области*, а также её внешних связей с другими *предметными областями*]
- := [описание ролей ключевых элементов предметной области (прежде всего, понятий – концептов), а также "место" специфицируемой предметной области в множестве себе подобных]
- := [семантическая окрестность знака *предметной области* в рамках самой этой *предметной области*, включающая в себя все *ключевые знаки*, входящие в состав *предметной области* (ключевые понятия и ключевые объекты исследования предметной области) с указанием их ролей (свойств) в рамках этой *предметной области* и семантическая окрестность знака специфицируемой *предметной области* в рамках *Предметной области предметных областей*, включающая в себя связи специфицируемой *предметной области* с другими семантически близкими ей *предметными областями* (дочерними и родительскими, аналогичными в том или ином смысле (например, изоморфными), имеющими одинаковые *классы объектов исследования* или одинаковые наборы *исследуемых отношений*)]

теоретико-множественная онтология предметной области

- := [семантическая окрестность специфицируемой *предметной области* в рамках *Предметной области множеств*, описывающая теоретико-множественные связи между *понятиями* специфицируемой *предметной области*, включая связи *отношений* с их *областями определения* и *доменами*, связи используемых *параметров* и классов структур их *областями определения*]
- := [онтология описывающая:
 - классификацию объектов исследования специфицируемой предметной области;
 - соотношение областей определения и доменов используемых отношений с выделенным классами объектов исследования, а также с выделенными классами вспомогательных (смежных) объектов, не являющихся объектами исследования в специфицируемой предметной области;
 - спецификацию используемых отношений и, в том числе, указание того, все ли связи этих отношений входят в состав специфицируемой предметной области.

]

Теоретико-множественная онтология предметной области включает в себя:

- теоретико-множественные связи (в т.ч. таксономию) между всеми используемыми понятиями, входящими в состав специфицируемой предметной области;
- теоретико-множественную спецификацию всех *отношений*, входящих в состав специфицируемой предметной области (ориентированность, арность, область определения, домены и т.д.);
- теоретико-множественную спецификацию всех параметров, используемых в предметной области (области определения параметров, шкалы, единицы измерения, точки отсчета);
- теоретико-множественную спецификацию всех используемых классов структур.

логическая онтология предметной области

- := [формальная теория заданной (специфицируемой) предметной области, описывающая с помощью переменных, кванторов, логических связок, формул различные свойства экземпляров понятий, используемых в специфицируемой предметной области]

Логическая онтология предметной области включает в себя:

- формальные определения всех понятий, которые в рамках специфицируемой предметной области являются определяемыми;
- неформальные пояснения и некоторые формальные спецификации (как минимум, примеры) для всех понятий, которые в рамках специфицируемой предметной области являются неопределеняемыми;
- иерархическую систему понятий, в которой для каждого понятия, исследуемого в специфицируемой предметной области либо указывается факт неопределеняемости этого понятия, либо указываются все понятия, на основе которых даётся определение данному понятию. В результате этого множество исследуемых понятий разбивается на ряд уровней:
 - неопределеняемые понятия;
 - понятия 1-го уровня, определяемые только на основе неопределеняемых понятий;
 - понятия 2-го уровня, определяемые на основе понятий, изменяющих 1-й уровень и ниже;
 - и т.д.
- формальную запись всех аксиом, т.е. высказываний, которые не требуют доказательств;
- формальную запись высказываний, истинность которых требует обоснования (доказательства);

- формальные тексты доказательства истинности высказываний, представляющие собой спецификацию последовательности шагов соответствующих рассуждений (шагов логического вывода, применения различных правил логического вывода);
- иерархическую систему высказываний, в которой для каждого высказывания, истинного по отношению к специфицируемой предметной области, либо указывается аксиоматичность этого высказывания, либо перечисляются все высказывания, на основе которых доказывается данное высказывание. В результате этого множество высказываний, истинных по отношению к специфицируемой предметной области, разбивается на ряд уровней:
 - аксиомы;
 - высказывания 1-го уровня, доказываемые только на основе аксиом;
 - высказывания 2-го уровня, доказываемые на основе высказываний, находящихся на 1-м уровне и ниже.
- формальная запись гипотетических высказываний;
- формальное описание логико-семантической типологии высказываний – высказываний о существовании, о несуществовании, об однозначности, высказывания определяющего типа (которые можно использовать в качестве определений соответствующих понятий);
- формальное описание различного вида логико-семантических связей между высказываниями (например, между высказыванием и его обобщением);
- формальное описание аналогии
 - между определениями;
 - между высказываниями любого вида;
 - между доказательствами различных высказываний.

терминологическая онтология предметной области

- := [онтология, описывающая правила построения терминов (sc-идентификаторов), соответствующих sc-элементам, принадлежащим специфицируемой предметной области, а также описывающая различного рода терминологические связи между используемыми терминами, характеризующие происхождение этих терминов]
- := [система терминов заданной предметной области]
- := [тезаурус соответствующей предметной области]
- := [словарь соответствующей (специфицируемой) предметной области]
- := [фрагмент глобальной *Предметной области sc-идентификаторов* (внешних идентификаторов sc-элементов), обеспечивающий терминологическую спецификацию некоторой предметной области]

Теперь подробнее рассмотрим понятие *объединённой онтологии предметной области*.

объединённая онтология предметной области

- := [объединение всех частных онтологий, соответствующих одной предметной области]
- ⇐ обобщённое объединение*:
 - {• структурная спецификация предметной области
 - теоретико-множественная онтология предметной области
 - логическая онтология предметной области
 - терминологическая онтология предметной области}

предметная область и онтология

- := [интеграция некоторой *предметной области* с соответствующей ей объединённой онтологией]
- := [предметная область & онтология]
- ⇐ обобщённое объединение*:
 - {• предметная область
 - объединённая онтология предметной области}
- := [sc-текст, являющийся объединением некоторой предметной области, представленной в SC-коде, и объединённой онтологии этой предметной области, также представленной в SC-коде]
- := [интеграция предметной области и всех онтологий, специфицирующих эту предметную область]
- := [совокупность различных *фактов* о структуре некоторой области деятельности некоторых *субъектов*, а также различного вида *знаний*, специфицирующих эту область деятельности]
- := [факты и знания о некоторой области деятельности]
- := [sc-модель предметной области и всевозможных онтологий, специфицирующих эту предметную область (и, в первую очередь, её ключевых понятий) в разных ракурсах]
- := [целостный с логико-семантической точки зрения фрагмент базы знаний ostis-системы, акцентирующий внимание на конкретном классе объектов исследования и на конкретном аспекте их рассмотрения]

Предметные области и онтологии являются основным видом разделов баз знаний, обладающих высокой степенью их независимости друг от друга и четкими правилами их согласования, что обеспечивает их семантическую (понятную) совместимость в рамках всей базы знаний.

Заключение к Главе 2.5. Структура баз знаний *ostis*-систем: иерархическая система предметных областей и соответствующих им онтологий

Каждому из представленных понятий соответствуют *предметные области и онтологии*, в котором данное понятие является максимальным классом объектов исследования:

- **Предметная область знаний и баз знаний *ostis*-систем**

*Предметная область знаний и баз знаний *ostis*-систем*

- ∈ *предметная область*
- Э *максимальный класс объектов исследования'*:
 - знание*
- Э *класс объектов исследования'*:
 - *вид знаний*
 - *отношение, заданное на множестве знаний*

- **Предметная область структур**

Предметная область структур

- ∈ *предметная область*
- Э *максимальный класс объектов исследования'*:
 - структура*
- Э *класс объектов исследования'*:
 - *связная структура*
 - *несвязная структура*
 - *тривиальная структура*
 - *нетривиальная структура*
- Э *исследуемое отношение'*:
 - *элемент структуры'*
 - *непредставленное множество'*
 - *полностью представленное множество'*
 - *элемент структуры, не являющийся множеством'*
 - *полиморфизм**
 - *гомоморфизм**
 - *изоморфность**
 - *аналогичность структур**

- **Предметная область семантических окрестностей**

Предметная область семантических окрестностей

- ∈ *предметная область*
- Э *максимальный класс объектов исследования'*:
 - семантическая окрестность*
- Э *класс объектов исследования'*:
 - *полная семантическая окрестность*
 - *базовая семантическая окрестность*
 - *специализированная семантическая окрестность*
 - *терминологическая семантическая окрестность*
 - *пояснение*
 - *примечание*
 - *теоретико-множественная семантическая окрестность*
 - *логическая семантическая окрестность*

- **Предметная область предметных областей**

210 2.5. Структура баз знаний *ostis*-систем: иерархическая система предметных областей и соответствующих им онтологий

В состав ***Предметной области предметных областей*** входят структурные спецификации всех ***предметных областей***, входящих в состав базы знаний *ostis*-системы, в том числе, самой ***Предметной области предметных областей***. Таким образом, ***Предметная область предметных областей*** является, во-первых, рефлексивным множеством, во-вторых, рефлексивной предметной областью, то есть ***предметной областью***, одним из объектов исследования которой является она сама.

Предметная область предметных областей

- := [Предметная область, объектами исследования которой являются предметные области]
- ∈ рефлексивное множество
- ∃ максимальный класс объектов исследования':
 - предметная область
- ∃ класс объектов исследования':
 - статическая предметная область
 - динамическая предметная область
 - понятие
 - нетривиальная структур
- ∃ исследуемое отношение':
 - исследуемое понятие'
 - максимальный класс объектов исследования'
 - немаксимальный класс объектов исследования'
 - класс исследуемых структур'
 - дочерняя предметная область*
 - понятие, исследуемое в дочерней предметной области'

- ***Предметная область онтологий***

Предметная область онтологий

- ∈ предметная область
- ∃ максимальный класс объектов исследования':
 - онтология
- ∃ класс объектов исследования':
 - структурная спецификация предметной области
 - частная онтология предметной области
 - объединенная онтология предметной области
 - теоретико-множественная онтология предметной области
 - логическая онтология предметной области
 - онтология предметной области

Глава 2.6.

Смыслоное представление логических формул и высказываний в различного вида логиках

Шункевич Д.В.

Василевская А.П.

Орлов М.К.

Ивашенко В.П.

⇒ *аннотация**:

[Аннотация к главе.]

Появление формальных систем было обусловлено осознанием того факта, что совершенно различные системы, будь то технические, социальные, экономические или биологические, обладают глубоким сходством. Действительно, каждая конкретная система состоит из каких-то первичных (базовых) элементов, обладающих какими-то свойствами. Затем, исходя из наличия исходных описаний, можно логическим путём вывести описание новых свойств, причём утверждения о наличии исходных или выведенных свойств воспринимаются как истинные на основании смысла определений данных элементов.

формальная теория — это множество высказываний, которые считаются истинными в рамках данной **формальной теории**.

Аксиоматические системы — это системы с наличием определённого числа исходных заранее выбранных и фиксированных высказываний, называемых аксиомами.

Высказывания могут быть как фактографическими, так и логическими формулами. Некоторые высказывания считаются аксиомами, а другие доказываются на основе других высказываний в рамках этой же **формальной теории**.

Каждая формальная теория интерпретируется (т.е. ее высказывания являются истинными) на какой-либо *предметной области*, которая является максимальным из *фактографических высказываний* (их *объединением**, входящих в состав этой **формальной теории**).

Каждой **формальной теории** соответствует одна *предметная область*, которая входит в нее под атрибутом *предметная область'*.

Каждая формальная теория может рассматриваться как конъюнктивное высказывание, априори истинное (с чьей-то точки зрения) при интерпретации на соответствующей предметной области.

Каждая формальная теория задаётся алфавитом, формулами, аксиомами, правилами вывода.

Предметная область является максимальным фактографическим высказыванием **формальной теории**, которая интерпретируется на данной *предметной области*.

аксиома — это высказывание, истинность которого не требует доказательства в рамках рассматриваемой формальной теории.

теорема — это высказыванием, истинность которого доказывается в рамках рассматриваемой формальной теории.

высказывание

⇒ *разбиение**:

- {• *атомарное высказывание*
- *неатомарное высказывание*

}

⇒ *разбиение**:

- {• *фактографическое высказывание*
- *логическая формула*

}

Под **высказыванием** понимается некоторая *структура* (в которую входят *sc-константы* из некоторой предметной области и/или *sc-переменные*) или *логическая связка*, которая может трактоваться как истинная или ложная в рамках какой-либо *предметной области*.

Истинность **высказывания** задается путем указания принадлежности знака этого высказывания *формальной теории*, соответствующей данной *предметной области*. Ложность высказывания задается путем указания принадлежности знака *отрицания** этого высказывания данной *формальной теории*.

Явно указанная непринадлежность **высказывания** *формальной теории* может говорить как о его ложности в рамках данной теории (если это указано рассмотренным выше образом), так и о том, что данное **высказывание** вообще не рассматривается в данной *формальной теории* (например, использует понятия, не принадлежащие данной *предметной области*).

Одно и то же **высказывание** может быть истинно в рамках одной *формальной теории* и ложно в рамках другой.

высказывание формальной теории'

⇒ разбиение*:

- {• истинное высказывание'
 - нечеткое высказывание'
 - бессмысленное высказывание'
- }

истинное высказывание – высказывание, знак которого принадлежит изучаемой формальной теории. Нечеткое высказывание – высказывание, возможно истинное или ложное в рамках изучаемой формальной теории (высказывание, возможно истинное или ложное в рамках данной формальной теории). **бессмысленное высказывание** – высказывание, не рассматриваемое в рамках данной формальной теории. Высказывание является бессмысленным в рамках заданной формальной теории, если в какое-либо *атомарное высказывание* в его составе (или в само это высказывание, если оно является атомарным) входит какая-либо *sc-константа*, не являющаяся элементом предметной области, описываемой указанной *формальной теорией*.

атомарное высказывание

С структура

⇒ разбиение*:

- {• атомарное фактографическое высказывание
 - атомарная логическая формула
- }

атомарное высказывание – это *высказывание*, которое содержит хотя бы один *sc-элемент*, не являющийся знаком другого *высказывания*.

неатомарное высказывание – это *высказывание*, в состав которого входят только знаки других *высказываний*. Следует отметить, что мы не можем говорить об истинности либо ложности **неатомарного высказывания** в рамках какой-либо *формальной теории*, в случае, когда невозможно установить истинность либо ложность любого из его элементов в рамках этой же *формальной теории*.

Под *фактографическим высказыванием* понимается:

- *атомарное высказывание*, в состав которого не входит ни одна *sc-переменная*;
- *неатомарное высказывание*, все элементы которого также являются *фактографическими высказываниями*.

высказывание*

:= [Бинарное ориентированное отношение, каждая *пара* которого связывает (1) знак некоторой *предметной области* и (2) знак некоторой *логической формулы*.]

⇒ разбиение*:

- {• ложное высказывание*
 - высказывание неизвестной истинности*
 - ▷ гипотеза
 - истинное высказывание*
- }

⇒ предъявляемое требование*:

[Все *sc-элементы*, входящие в состав *предметной области*, описываемой высказыванием, (включая и *sc-переменные*, которые, хоть и редко, но могут входить в состав некоторых *предметных областей*) *sc-элементами* для всех высказываний, соответствующих этой *предметной области*.]

⇒ предъявляемое требование*:

[Все *sc-константы*, входящие в состав всех *атомарных логических формул*, входящих в состав всех *высказываний*, описывающих некоторую *предметную область* должны входить в состав описываемой *предметной области*.]

следует отличать*

- Э {• **высказывание***
 - ∈ **бинарное ориентированное отношение**
 - := [быть высказыванием, описывающим заданную предметную область*]
 - ⇒ **сокращение*:**
[быть высказыванием*]
 - **высказывание**
 - ⊂ **логическая формула**
 - := [Второй домен отношения "быть высказыванием"]
- }

логическая формула

- ⇒ **разбиение*:**
 - {• **атомарная логическая формула**
 - **неатомарная логическая формула**
- }
- ⇒ **разбиение*:**
 - {• **открытая логическая формула**
 - **замкнутая логическая формула**
- }

Под **логической формулой** понимается:

- **атомарное высказывание**, в состав которого входит хотя бы одна *sc-переменная*;
- **неатомарное высказывание**, хотя бы один элемент которого является **логической формулой**.

Под **атомарной логической формулой** понимается **атомарное высказывание**, которое является **логической формулой**.

атомарная логическая формула – это логическая формула, которая не содержит логических связок.

По умолчанию **атомарная логическая формула** трактуется как **высказывание** о существовании, то есть наличия в памяти значений, соответствующих всем *sc-переменным*, входящим в состав данной формулы и не попадающих под действие какого-либо другого квантора (указанного явно или по умолчанию). Таким образом, на все *sc-переменные*, входящие в состав **атомарной логической формулы** и не попадающие под действие какого-либо другого квантора, неявно накладывается квантор **существования***.

неатомарная логическая формула

- ⇒ **разбиение*:**
 - {• **общезначимая логическая формула**
 - **противоречивая логическая формула**
 - **нейтральная логическая формула**
- }
- ⇒ **разбиение*:**
 - {• **выполнимая логическая формула**
 - **невыполнимая логическая формула**
- }

Под **неатомарной логической формулой** понимается **неатомарное высказывание**, которое является **логической формулой**.

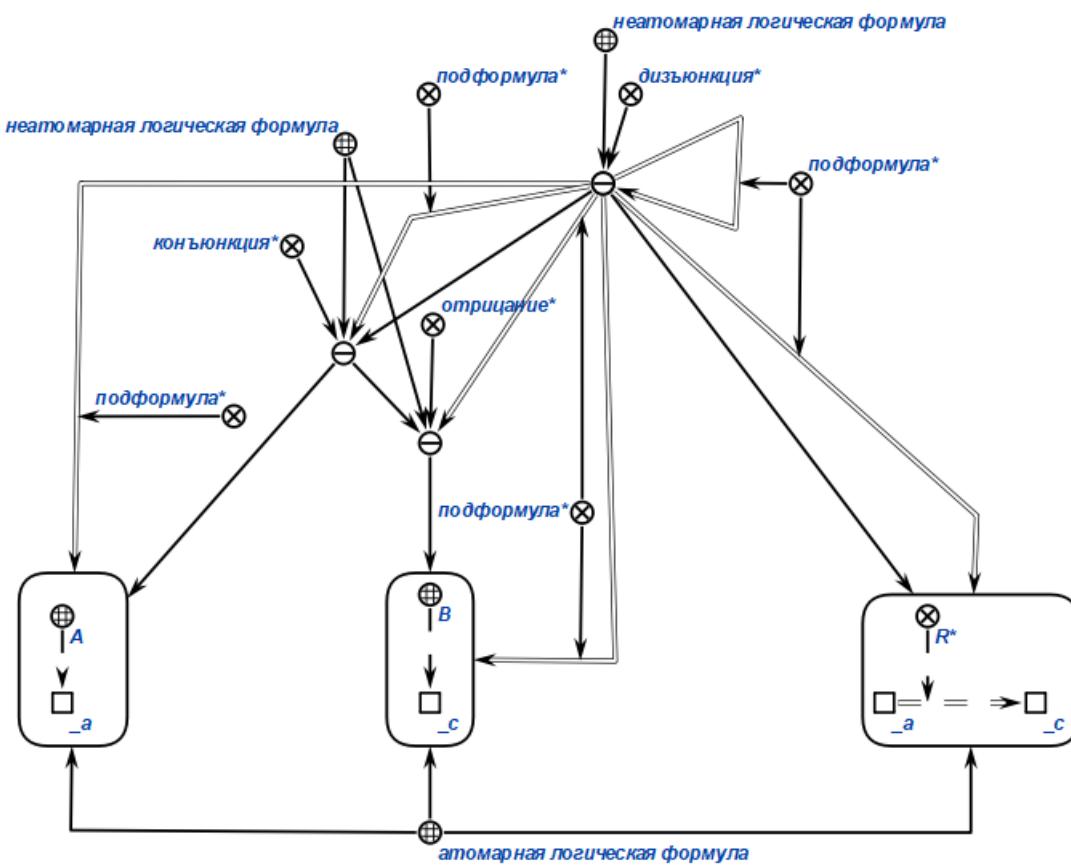
Для того, чтобы рассмотреть типологию **неатомарных логических формул**, будем говорить, что исследуется истинность самой **неатомарной логической формулы** и всех ее **подформул*** в рамках одной и той же **формальной теории**, при этом не важно, какой именно. Также считается, что в рассматриваемой **формальной теории** каждая **подформула*** рассматриваемой **неатомарной логической формулы** в рамках этой **формальной теории** может однозначно трактоваться как либо истинная, либо ложная. В противном случае мы не можем говорить об истинности либо ложности исходной **неатомарной логической формулы** в рамках этой **формальной теории**.

Будем называть **подформулой*** **неатомарной логической формулы** *fj* любую логическую формулу *fj*, являющуюся элементом исходной формулы *fj*, а также любую **подформулу*** формулы *fj*.

подформула*

- := [частная формула*]
- ∈ **бинарное отношение**
- ∈ **ориентированное отношение**

\in транзитивное отношение



= Формализация примера подформулы

утверждение – это семантическая окрестность некоторой логической формулы, в которую входит полный текст этой логической формулы, а также факт принадлежности этой логической формулы некоторой *формальной теории*.

Знак логической формулы, семантическая окрестность которой представляет собой утверждение, является *главным ключевым sc-элементом'* в рамках этого **утверждения**. Знаки понятий соответствующей *предметной области*, которые входят в состав какой-либо *подформулы** указанной логической формулы, будут *ключевыми sc-элементами'* в рамках этого **утверждения**.

Полный текст некоторой логической формулы включает в себя:

- знак самой этой логической формулы;
- знаки всех ее *подформул**;
- элементы всех логических формул, знаки которых попали в данную структуру;
- все пары принадлежности, связывающие логические формулы, знаки которых попали в данную структуру, с их компонентами.

Таким образом, факт принадлежности (истинности) логической формулы нескольким *формальным теориям* будет порождать новое утверждение для каждой такой *формальной теории*. Текст **утверждения** входит в состав *логической онтологии*, соответствующей *предметной области*, на которой интерпретируется главный *ключевой sc-элемент'* данного утверждения.

Правило идентификации экземпляров. **утверждения** в рамках *Русского языка* именуются по следующим правилам:

- в начале идентификатора пишется сокращение Утв.;
- далее в круглых скобках через точку с запятой перечисляются основные идентификаторы *ключевых sc-элементов'* данного **утверждения**. Порядок определяется в каждом конкретном случае в зависимости от того, свойства каких из этих понятий описывает данное **утверждение** в большей или меньшей степени.

Могут быть исключения для **утверждений**, названия которых закрепились исторически, например, *Теорема Пифагора*, *Аксиома о прямой и точке*.

Определение – это утверждение, главным ключевым sc-элементом' которого является связка эквиваленции*, однозначно определяющая некоторое понятие на основе других понятий.

Каждое определение имеет ровно один ключевой sc-элемент' (не считая главного ключевого sc-элемента').

Для одного и того же понятия в рамках одной *формальной теории* может существовать несколько *утверждений об эквиваленции**, однозначно задающих некоторое понятие на основе других, однако только одно такое *утверждение* в рамках этой *формальной теории* может быть отмечено как **определение**. Остальные *утверждения об эквиваленции** могут трактоваться как *пояснения* данного понятия.

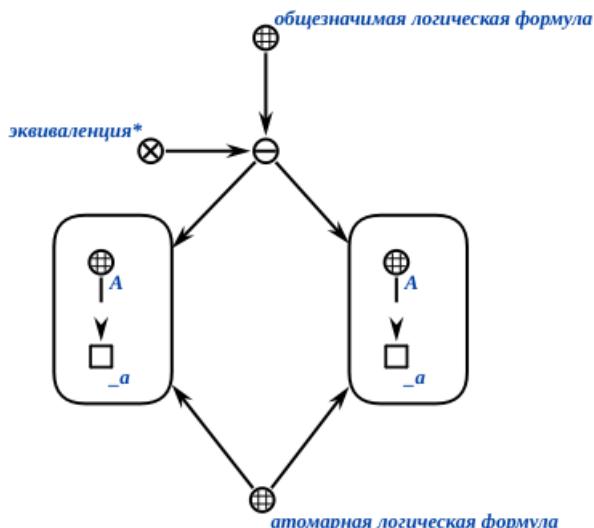
Правило идентификации экземпляров. **определения** в рамках *Русского языка* именуются по следующим правилам:

- в начале идентификатора пишется сокращение **Опр.:**
- далее в круглых скобках через точку с запятой записывается основной идентификатор *ключевого sc-элемента'* данного **определения**.

общезначимая логическая формула

- := [тождественно истинная логическая формула]
 ⊂ выполнимая логическая формула
 ⊂ тавтология

Общезначимая логическая формула – это логическая формула, для которой не существует *формальной теории*, в рамках которой она была бы ложной с учетом истинности и ложности всех ее подформул* в рамках этой же *формальной теории*.



= Формализация закона тождества

противоречивая логическая формула

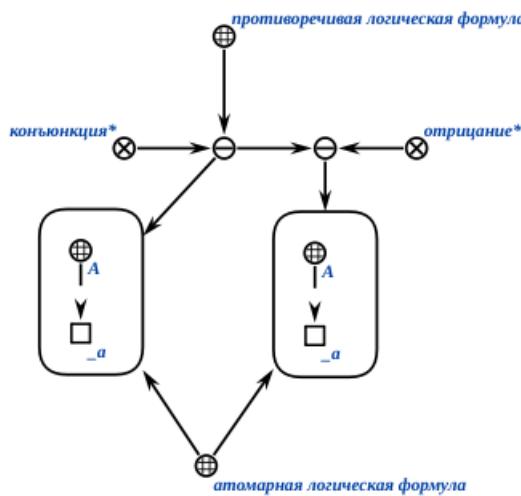
- := [тождественно ложная логическая формула]
 ⊂ невыполнимая логическая формула
 ⊂ тавтология

Противоречивая логическая формула – это логическая формула, для которой не существует *формальной теории*, в рамках которой она была бы истинной с учетом истинности и ложности всех ее подформул* в рамках этой же *формальной теории*.

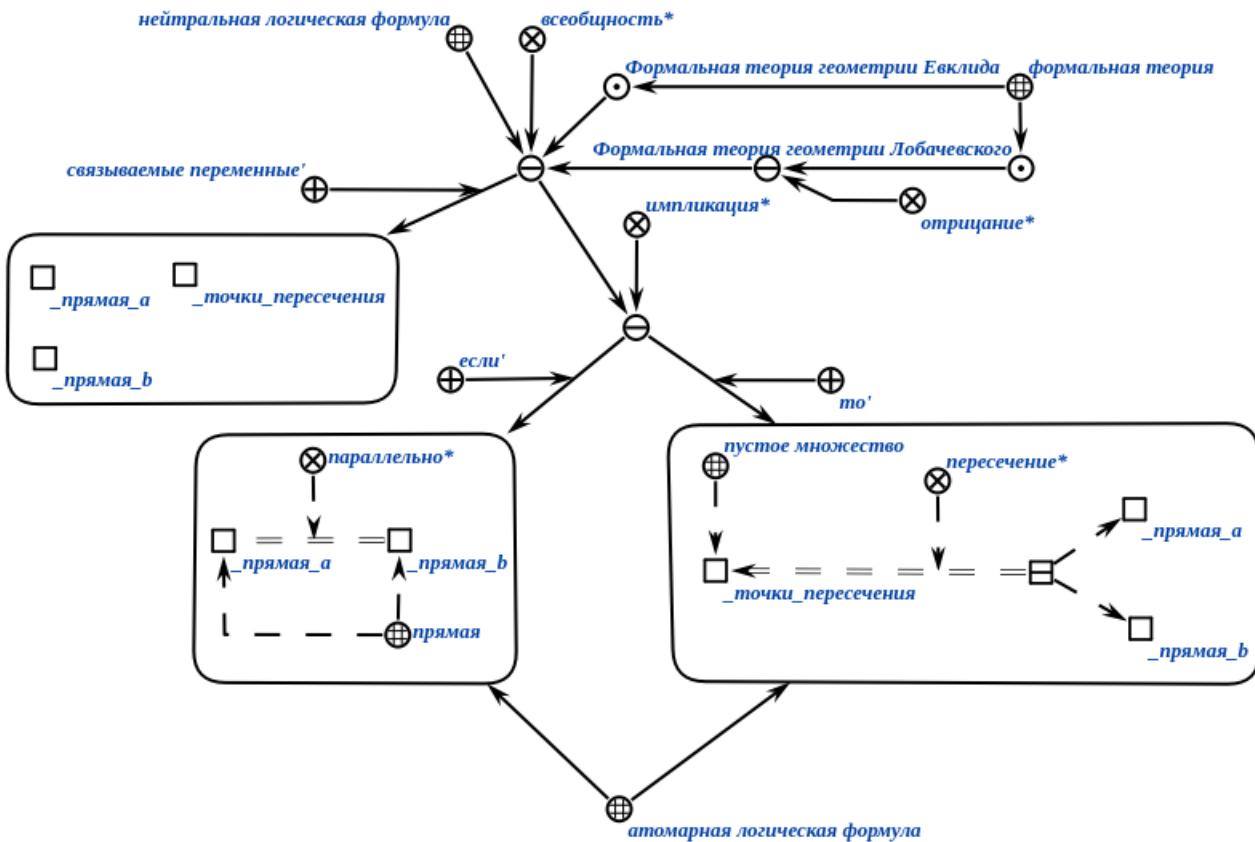
нейтральная логическая формула

- ⊂ выполнимая логическая формула

нейтральная логическая формула – это логическая формула, для которой существует хотя бы одна *формальная теория*, в рамках которой эта формула ложна, и хотя бы одна *формальная теория*, в рамках которой эта формула истинна.



= Формализация закона противоречия



= Формализация нейтральной логической формулы

В евклидовой геометрии в плоскости через точку, не лежащую на данной прямой, можно провести одну и только одну прямую, параллельную данной. В геометрии Лобачевского данный постулат является ложным. В сферической геометрии все прямые пересекаются.

непротиворечивая логическая формула

:= [выполнимая логическая формула]

\Leftarrow объединение*:

- {• нейтральная логическая формула
- общезначимая логическая формула

}

непротиворечивая логическая формула – это логическая формула, для которой существует хотя бы одна *формальная теория*, в рамках которой эта формула истинна.

необщезначимая логическая формула

\coloneqq [невыполнимая логическая формула]

\Leftarrow объединение*:

- {• нейтральная логическая формула
- противоречивая логическая формула

}

необщезначимая логическая формула – это логическая формула, для которой существует хотя бы одна *формальная теория*, в рамках которой эта формула ложна.

Тавтология – это логическая формула, которая является либо только истинной, либо только ложной в рамках всех *формальных теорий*, в которых можно установить ее истинность или ложность. Тавтология – это такая логическая формула, которая является либо *общезначимой логической формулой*, либо *противоречивой логической формулой*.

логическая связка*

\coloneqq [неатомарная логическая формула]

\coloneqq [логический оператор*]

\coloneqq [пропозициональная связка*]

\in класс связок разной мощности

\Leftarrow семейство подмножеств*:

неатомарное высказывание

логическая связка* – это отношение (класс связок), связками которого являются *высказывания*, а областью определения которого является множество *высказываний*, при этом само это отношение и некоторые его подмножества могут быть *классами связок разной мощности*.

конъюнкция*

\coloneqq [логическое и*]

\coloneqq [логическое умножение*]

\subset логическая связка*

\in неориентированное отношение

\in класс связок разной мощности

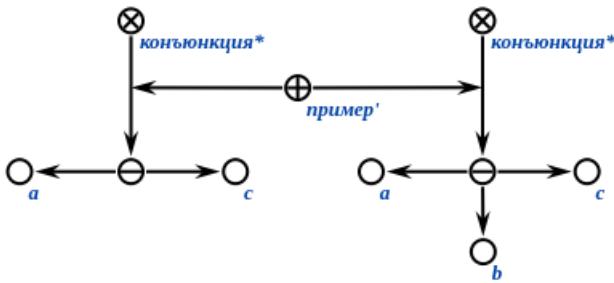
\in неунарное отношение

\Rightarrow область определения*:

логическая формула

конъюнкция* – это множество конъюнктивных *высказываний*, каждое из которых истинно в рамках некоторой *формальной теории* только в том случае, когда все его компоненты истинны в рамках этой же *формальной теории*.

конъюнкция* атомарных формул может быть заменена на атомарную формулу, полученную путём объединения исходных атомарных формул.

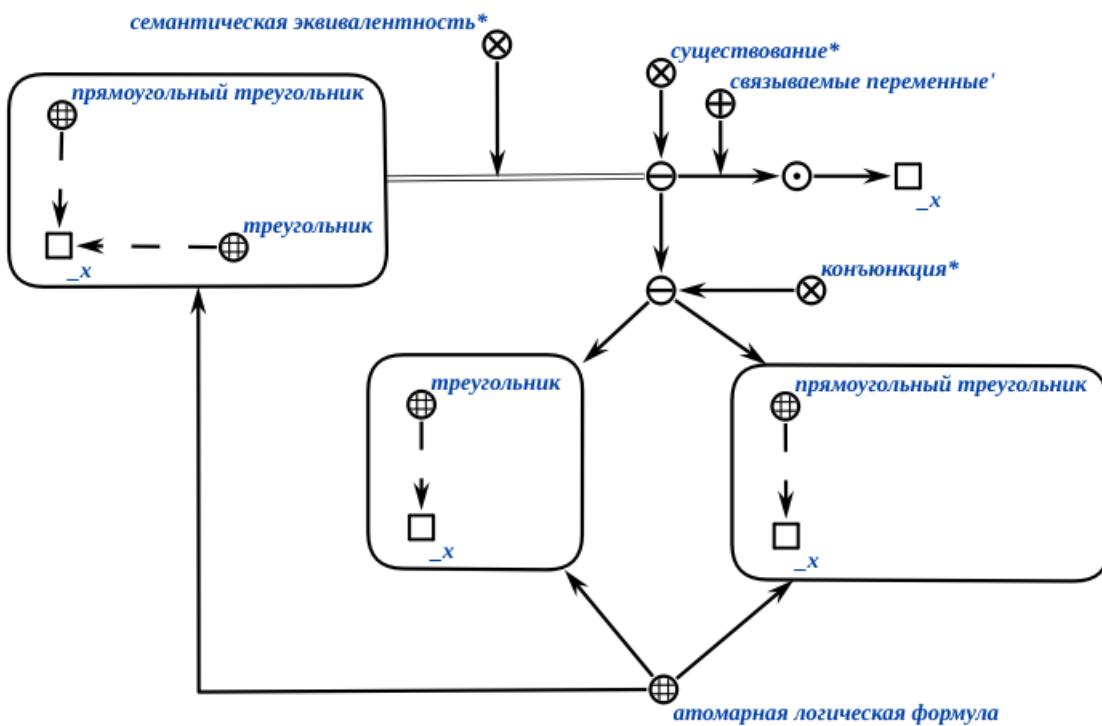


= Формализация примера конъюнкции

дизъюнкция*

\coloneqq [логическое или*]

\coloneqq [логическое сложение*]



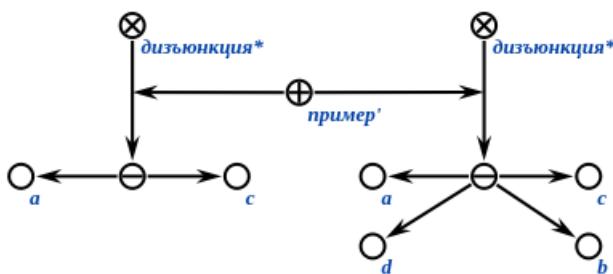
= Формализация примера конъюнкции в геометрии

\Rightarrow пояснение*:

[Данные конструкции эквивалентны по принципу $\exists xT(x) \wedge \exists xPT(x) \implies \exists x(T(x) \wedge PT(x))$]

- \coloneqq [включающее или*]
- \subset логическая связка*
- \in неориентированное отношение
- \in класс связок разной мощности
- \in неунарное отношение
- \Rightarrow область определения*:
- логическая формула

дизъюнкция* – это множество дизъюнктивных высказываний, каждое из которых истинно в рамках некоторой *формальной теории* только в том случае, когда хотя бы один его компонент является истинным в рамках этой же *формальной теории*.

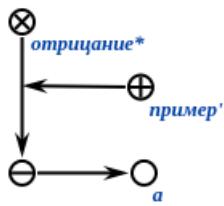


= Формализация примера дизъюнкции

отрицание*

- \subset логическая связка*
- \subset синглетон
- \in унарное отношение
- \Rightarrow область определения*:
- логическая формула

отрицание* – это множество *высказываний* об отрицании, каждое из которых истинно в рамках некоторой *формальной теории* только в том случае, когда его единственный элемент является ложным в рамках этой же *формальной теории*.

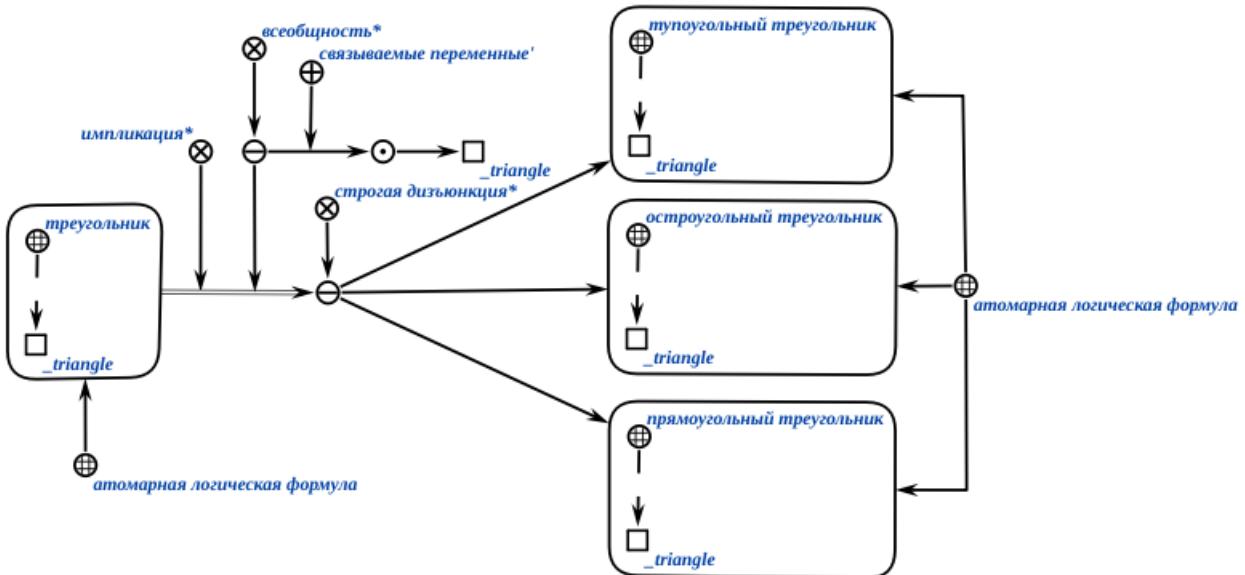


= *Формализация примера отрицания*

строгая дизъюнкция*

- := [сложение по модулю 2*]
- := [исключающее или*]
- := [альтернатива*]
- ⊆ логическая связка*
- ∈ неориентированное отношение
- ∈ класс связок разной мощности

строгая дизъюнкция* – это множество строго дизъюнктивных *высказываний*, каждое из которых истинно в рамках некоторой *формальной теории* только в том случае, когда ровно один его компонент является истинным в рамках этой же *формальной теории*.



= *Формализация примера строгой дизъюнкции в геометрии*

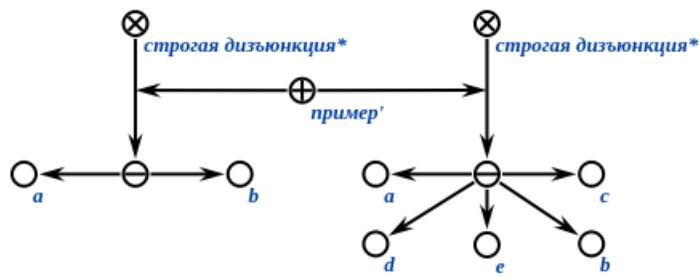
⇒ *пояснение*:*

[Данная неатомарная логическая формула содержит следующую информацию: для любых переменных $_triangle$ если $_triangle$ является треугольником, то $_triangle$ является или тупоугольным треугольником, или остроугольным треугольником, или прямоугольным треугольником.]

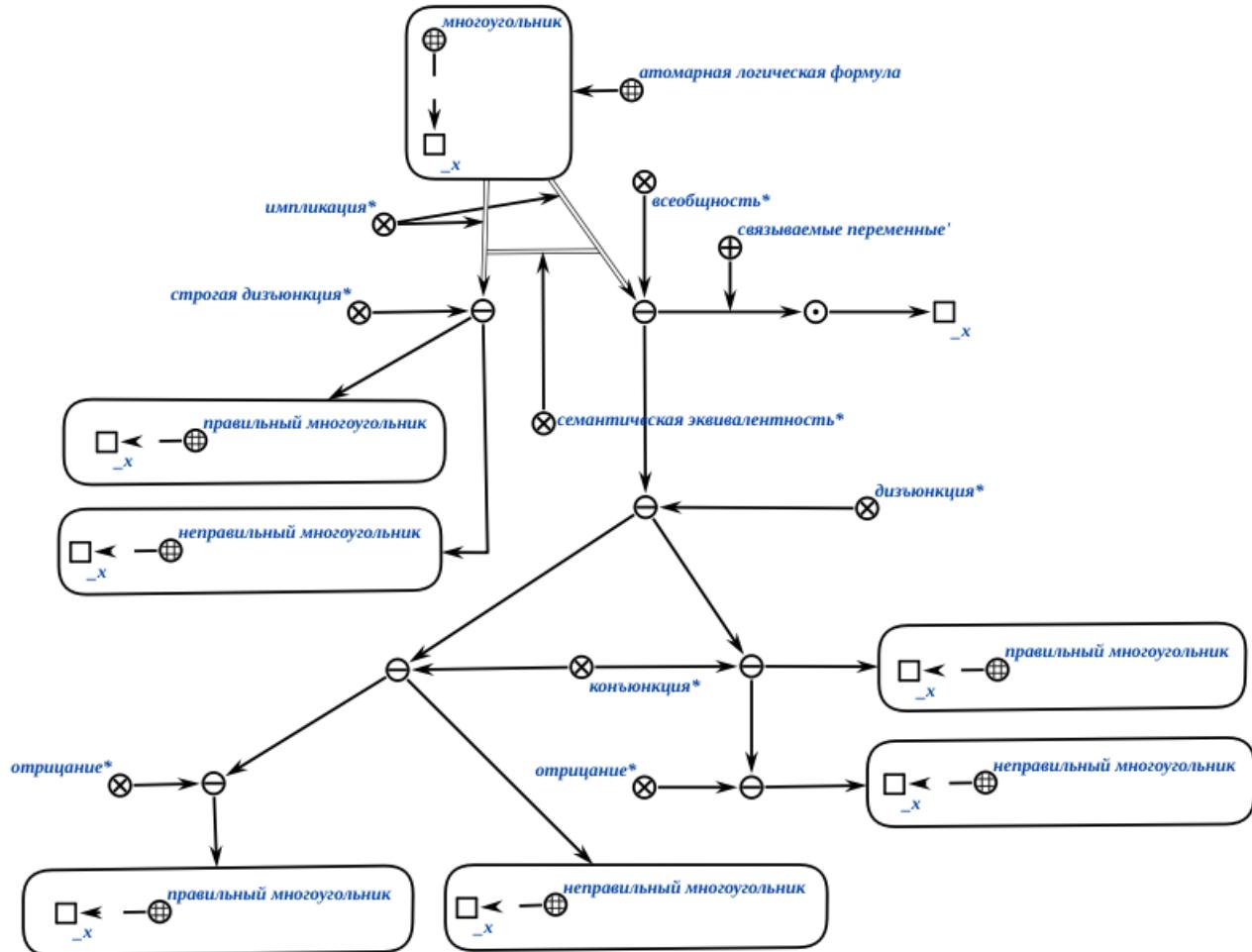
строгая дизъюнкция* может быть представлена как *дизъюнкция конъюнкции отрицания* первой логической формулы и второй логической формулы и *конъюнкции* первой логической формулы и *отрицания* второй логической формулы. Также она может быть представлена и в виде *конъюнкции дизъюнкций* двух логических формул и их *отрицаний*.

импликация*

- := [логическое следование*]



= Формализация примера строгой дизъюнкции



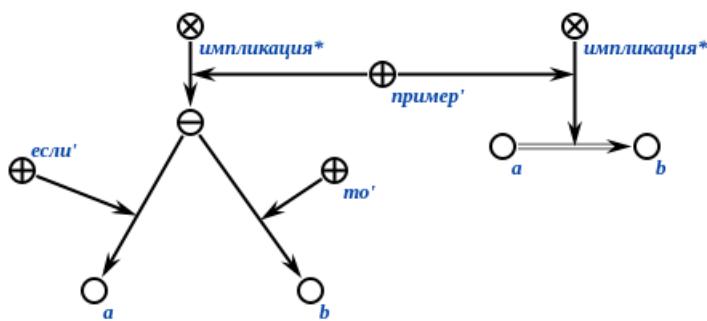
= Формализация примера строгой дизъюнкции

- \subseteq логическая связка*
- \in бинарное отношение
- \in ориентированное отношение
- \Rightarrow область определения*:
- логическая формула

импликация* – это множество импликативных *неатомарных высказываний*, каждое из которых состоит из посылки (первый компонент *высказывания*) и следствия (второй компонент *высказывания*).

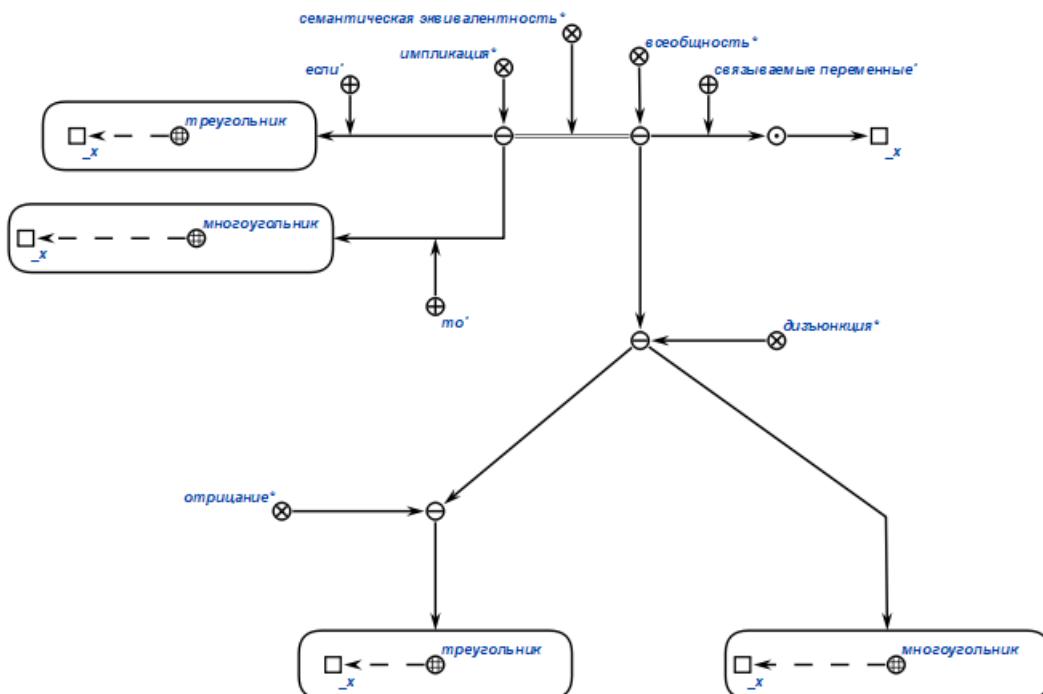
Каждое импликативное высказывание должно в рамках некоторой *формальной теории* в том случае, когда его посылка истинна, а заключение должно в рамках этой же *формальной теории*. В других случаях такое *высказывание* истинно.

По умолчанию на все переменные, входящие в обе части высказывания об *импликации** (или хотя бы одну из *подформул** каждой части) неявно накладывается квантор *всеобщности**, при условии, что эти переменные не связаны другим квантором, указанным явно.



= Формализация примера импликации

импликация* может быть представлена как *дизъюнкция отрицания* первой логической формулы и второй логической формулы или же как *отрицание конъюнкции* первой логической формулы и *отрицания* второй логической формулы.



= Формализация примера импликации

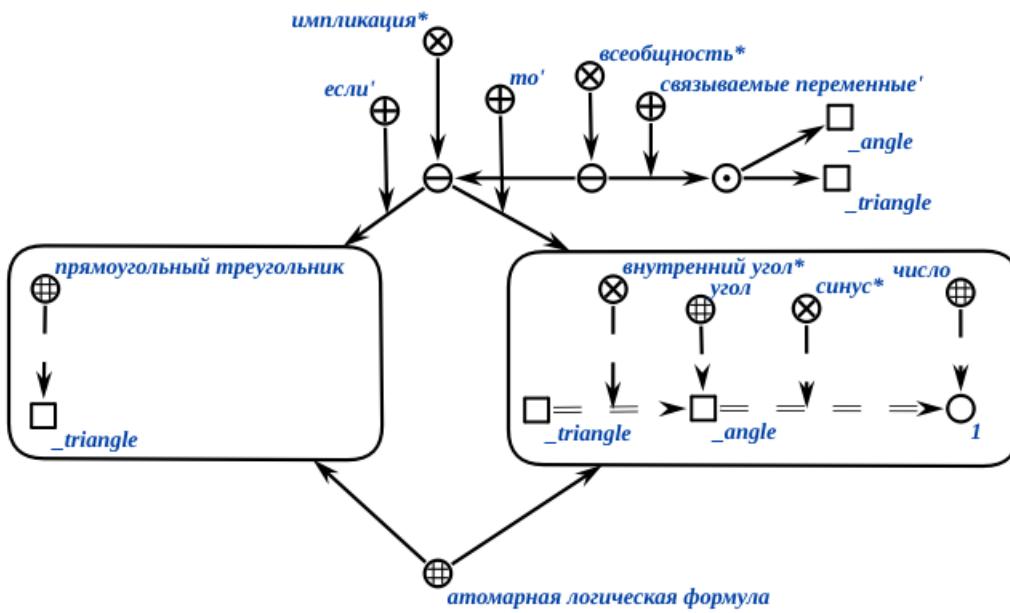
если'
:= [посылка']
 $\subset 1'$
 \in ролевое отношение

Если' – это ролевое отношение, используемое в связках импликации* для указания посылки.

то'
:= [следствие']
 $\subset 2'$
 \in ролевое отношение

То' – это ролевое отношение, используемое в связках импликации* для указания следствия.

эквиваленция*
:= [эквивалентность*]



= Формализация примера импликации

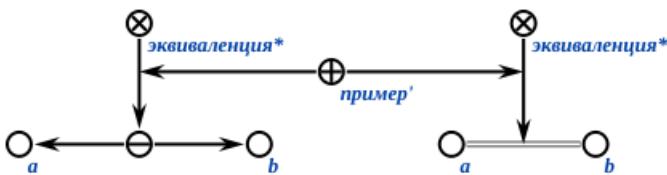
\Rightarrow пояснение*:

[Данная неатомарная логическая формула содержит следующую информацию: для любых переменных $_triangle$ и $_angle$ если $_triangle$ является прямоугольным треугольником, то синус его внутреннего угла $_angle$ равен единице.]

- \subset логическая связка*
- \in бинарное отношение
- \in неориентированное отношение
- \Rightarrow область определения*:
логическая формула

эквиваленция* – это множество высказываний об эквивалентности, каждое из которых истинно в рамках некоторой *формальной теории* только в тех случаях, когда оба его компонента одновременно либо истинны в рамках этой же *формальной теории*, либо ложны.

По умолчанию на все переменные, входящие в обе части высказывания об **эквиваленции*** (или хотя бы одну из подформул* каждой части) неявно накладывается квантор *всеобщности**, при условии, что эти переменные не связаны другим квантором, указанным явно.



= Формализация примера эквиваленции

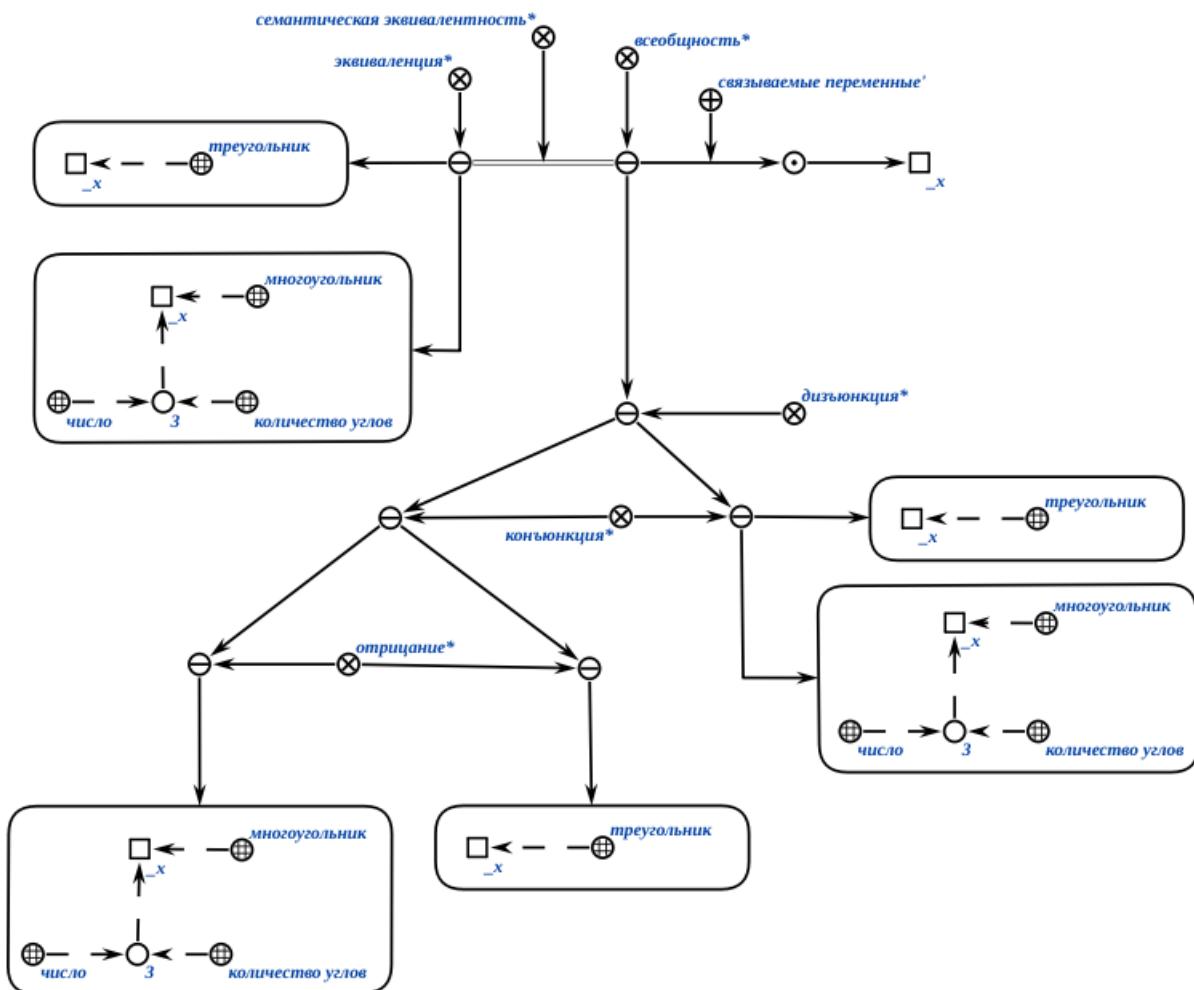
эквиваленция* двух логических формул может быть представлена как дизъюнкция конъюнкции этих двух логических формул и конъюнкции отрицаний этих двух логических формул.

квантор

- \subset логическая связка*

квантор — это *отношение*, каждая связка которой задает истинность множества логических формул, входящих в ее состав, при выполнении дополнительных условий, связанных с некоторыми из переменных, входящих в состав этих логических формул.

Будем говорить, что переменные связаны **квантором** или попадают под область действия данного **квантора** (имея в виду конкретную связку конкретного **квантора**).



= Формализация примера эквиваленции

В состав каждой связки каждого **квантора** входит *атомарная формула*, являющаяся *тривиальной структурой*, в которой перечислены переменные, связанные данным **квантором**.

всеобщность*

- := [квантор всеобщности*]
- := [квантор общности*]
- ∈ квантор
- ∈ ориентированное отношение
- ∈ класс связок разной мощности

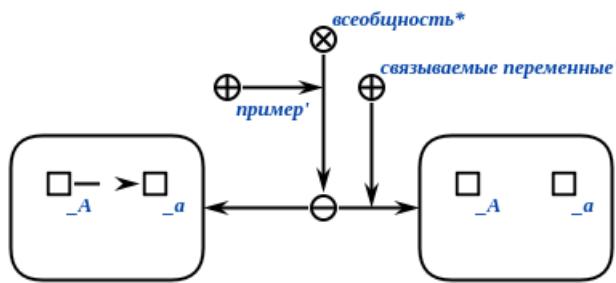
всеобщность* – это **квантор**, для каждой связки которого, истинной в рамках некоторой *формальной теории*, выполняется следующее утверждение: все формулы, входящие в состав этой связки истинны в рамках этой же *формальной теории* при всех (любых) возможных значениях всех элементов множества *связываемых переменных'* входящего в эту связку.

Каждая связка квантора **всеобщность*** может быть представлена как **конъюнкция*** (потенциально бесконечная) исходных логических формул, входящих в состав этой связки, в каждой из которых все *связанные переменные'* заменены на их возможные значения.

Квантор **всеобщности*** зачастую обозначается " \forall " и читается как "для всех", "для каждого", "для любого" или "все", "каждый", "любой".

формула существования

- := [существование*]
- ⇐ разбиение*:
 - атомарная логическая формула
 - неатомарное существование*



= Формализация примера всеобщности

}

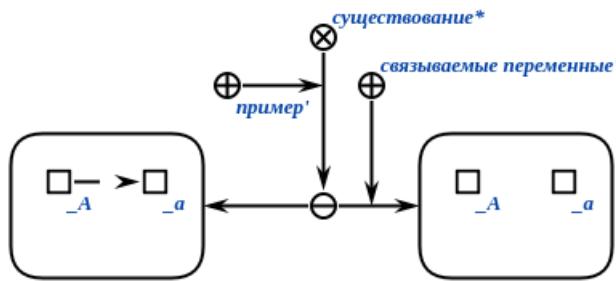
неатомарное существование*

- := [квантор неатомарного существования*]
- ∈ квантор
- ∈ ориентированное отношение
- ∈ класс связок разной мощности

неатомарное существование* – это квантор, для каждой связки которого, истинной в рамках некоторой *формальной теории*, выполняется следующее утверждение: существуют значения всех элементов множества *связываемых переменных'* входящего в эту связку, такие, что все формулы, входящие в состав этой связки истинны в рамках этой же *формальной теории*.

Каждая связка квантора **неатомарное существование*** может быть представлена как *дизъюнкция** (потенциально бесконечная) исходных логических формул, входящих в состав этой связки, в каждой из которых все *связанные переменные'* заменены на их возможные значения.

Квантор **существования*** зачастую обозначается " \exists " и читается как "существует", "для некоторого", "найдется".



= Формализация примера неатомарного существования

число значений переменной

- ∈ параметр

Каждый элемент **параметра число значений переменной** представляет собой класс ориентированных пар, первым компонентом которых является знак логической формулы, вторым – sc-переменная, имеющая в рамках данной логической формулы ограниченное известное число значений, при которых данная формула является истинной в рамках соответствующей *формальной теории*.

Отметим, что в случае *атомарной логической формулы* каждая такая связка связывает знак формулы и знак принадлежащей ей sc-переменной, т.е. является, по сути, частным случаем пары принадлежности. В случае *неатомарной логической формулы* указанная sc-переменная может принадлежать любой из *подформул** исходной формулы.

измерением* значения параметра **число значений переменной** является некоторое **число**, задающее количество значений sc-переменных в рамках логической формулы.

кратность существования

- \in *параметр*
- \Rightarrow *область определения параметра**:
- формула существования*
- \exists *единственное существование*

Каждый элемент *параметра кратности существования* представляет собой класс логических *формул существования*, для которых при интерпретации на соответствующей *предметной области* существует ограниченное общее для всех таких формул число комбинаций значений переменных, при которых указанные формулы являются истинными в рамках соответствующей *формальной теории*. *измерением** каждого значения *кратности существования* является некоторое число, задающее количество таких комбинаций.

единственное существование

- \coloneqq [однократное существование]
- \coloneqq [формула существования и единственности]

Единственное существование зачастую обозначается " $\exists!$ " и читается как "существует и единственный".

логическая формула и единственность

- \subset *логическая формула*
- \subset *единственное существование*

Каждый элемент множества *логическая формула и единственность* представляет собой *логическую формулу* (*атомарную* или *неатомарную*), для которой дополнительно уточняется, что при ее интерпретации на некоторой предметной области существует только один набор значений переменных, входящих в эту формулу (или ее *подформулы**), при котором указанная логическая формула истинна в рамках *формальной теории*, в которую входит данная *предметная область*.

связываемые переменные' – это *ролевое отношение*, которое связывает связку конкретного *квантора* с множеством переменных, которые связаны этим квантором.

открытая логическая формула – это *логическая формула*, в рамках которой (и всех ее *подформул**) существует хотя бы одна переменная, не связанная никаким *квантором*.

замкнутая логическая формула – это *логическая формула*, в рамках которой (и всех ее *подформул**) не существует переменных, не связанных каким-либо *квантором*.

§ 2.6.1. Смыслоное представление логических формул и высказываний в прикладных логиках

§ 2.6.2. Смыслоное представление логических формул и высказываний в неклассических логиках

Глава 2.7.

Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах

Никифоров С.А.

Гойло А.А.

⇒ *аннотация**:

[Аннотация к главе.

Список доработок:

- написать аннотацию;
- вставить отредактированный анализ из статьи (в частности, тут не нужно говорить про sem web и нужно убрать пару опасных высказываний, погорячился);
- формализовать все обобщенные правила,
- адекватное определения лексемы, пересмотреть термины для исключения возможности появления омонимии / синонимии,
- примеры и пояснения,
- сказать что-то про отношения на лексемах,
- выделить пр. о. лексики,
- пройтись по подсказкам от плагина, поправит форматирование,
- формализовать пр. о..

Список вопросов:

- нужно ли формализовать все частные правила, их несколько десятков.

]

Проблема совместности результатов исследований также остро стоит и в лингвистике – науке, в которой существует множество различных теорий, часто несовместимых друг с другом. В лингвистических исследованиях используются разные варианты разметки данных, нет одного подхода к структуризации корпусов текстов и различаются способы представления данных в них **Farrar2002ACO**, **Chiarcos2012**.

В качестве решения проблемы несовместимости различных способов описания данных в лингвистике предлагались варианты стандартизации форматов такого описания. Примером могут служить Text Encoding Initiative – консорциум по стандартизации представления текстов в цифровом виде **Text_Encoding_Initiative** и гайдлайны экспертной группы по стандартизации представления языковых данных EAGLES (например, рекомендации по разметке корпусов текстов **EAGLES_Recommendations**). Однако ни один из таких стандартов не получил распространения и не стал использоваться лингвистами повсеместно **Ide2010WhatDI**.

Вместо создания рекомендаций по разметке языкового материала в качестве более эффективного средства решения указанных выше проблем предлагается создание онтологий **schalley_2019**, **mccrae_2015**. Помимо того, что онтология верхнего уровня для предметной области языкознания может служить связующим звеном между различными лингвистическими теориями, она также представляет собой формализованное описание лингвистических концептов, представленное в удобном для компьютеров формате, что обусловливает ее применимость в системах, способных понимать аннотированные языковые данные, совершать интеллектуальный поиск по корпусам текстов, а также потенциально выполнять анализ существующих лингвистических исследований **Farrar2002ACO**.

В качестве такой онтологии ПрО лингвистики выступает The General Ontology of Linguistic Description (GOLD) **gold**. В этой онтологии формализованы наиболее базовые категории и отношения, используемые в лингвистике, а сама онтология интегрирована с онтологией верхнего уровня Suggested Upper Merged Ontology (SUMO) **pease_2002_sumo**. Авторы GOLD пишут, что создавали онтологию в первую очередь для того, чтобы решить проблему интероперабельности данных лингвистической типологии и для того, чтобы с ее помощью экспертные системы могли обрабатывать научные данные по естественным языкам – т.е. целью создателей онтологии не являлось непосредственно решение задач из области обработки текстов на естественном языке **farrar_2003** (с.4).

Онтологией естественных языков, нацеленной непосредственно на использование при решении задач по обработке ЕЯ, является Ontologies of Linguistic Annotation (OLiA) **chiarcos-2012-ontologies**. Основной идеей онтологии является обеспечение совместимости разметки языковых данных, полученных в результате выполненного компью-

терными системами анализа текстов на естественном языке с соответствующими им лингвистическими концептами из онтологии – в отличие от других лингвистических онтологий, OLiA предоставляет не только инвентарь концептов и отношений, но и необходимую спецификацию интеграции этих элементов с разметкой языковых данных (например, в корпусах). **chiarcos-2012-ontologies**.

При создании онтологий естественного языка, встает вопрос о статусе спецификации лингвистической информации в таких онтологиях. Дж. Бейтман выделяет три типа онтологий в зависимости от интегрированности естественноязыковой информации в онтологию **Bateman_1997**:

1. онтологии, представляющие собой абстрактную семантико-концептуальную репрезентацию знаний о мире, которая используется непосредственно в качестве денотационной семантики для синтаксиса и лексики естественного языка;
2. онтологии, в которых есть отдельная спецификация денотационной семантики естественного языка, которая служит интерфейсом между синтаксисом естественных языков и собственно концептуальной онтологией;
3. онтологии, представляющие собой абстрактную спецификацию знаний о реальном мире вне зависимости от ограничений естественного языка

Популярность в сфере обработки естественного языка приобрел второй тип онтологии **Bateman_1997**, так как он, в отличие от третьего подхода, который совсем не формализует лингвистическую информацию, позволяет специфицировать больше информации о естественных языках. Так, одна из самых популярных онтологий, используемых в системах для обработки естественного языка, the Generalized Upper Model **Bateman2002TheGU**, является онтологией второго типа **Bateman_1997**. П. Буйтелаар и др. подчеркивают, что всем формальным онтологиям необходима связь с языковой информацией для решения таких задач как выделение информации из ЕЯ-текстов, автоматизированное заполнение онтологий и генерации текста на естественном языке **Buitelaar_2009**.

Так как использование онтологий в NLP позволяет задать семантику получаемым в результате обработки естественного языка данным и потенциально повысить качество NLP-анализа, начинается переход к созданию движимых онтологиями систем обработки естественных языков **Kostareva2016UsingOM**, **nevзорова_2019**. Онтологии естественного языка активно применяются для генерации ЕЯ-текстов на основе некоторой онтологии предметной области **cimiano-et-al-2013-exploiting**, **Bouayad_2014**.

Онтологический подход также используется в системах естественноязыковых запросов для баз данных, в которых запрос на естественном языке транслируется в язык запросов по онтологиям конкретных предметных областей, конструкции которого затем транслируются в SQL для обеспечения взаимодействия с реляционными базами данных **saha_2016**.

Кроме того, спецификация лингвистической информации в виде онтологий помогает решать задачу автоматизированного создания онтологий на основе естественноязыковых текстов **SHAMSFARD200417**.

Создаются онтологии частных областей лингвистики: например, онтология пространственных выражений в естественных языках **BATEMAN20101027**, онтология темпоральных сущностей на основе естественного языка **Moens_1987**, онтологии конкретных естественных языков **Dobrov_2018**. При использовании онтологий для обработки естественного языка необходимо "связать" концепты из онтологии с лексикой конкретного ЕЯ. Для этого создаются различные расширения существующих языковых БД, таких как WordNet **wordnet**, VerbNet **verbnet** и FrameNet **framenet**, направленные на их использование совместно с онтологиями верхнего уровня (например, **pease_fellbaum_2010**). Активные разработки идут в сфере создания онтологий словарного состава естественных языков, в результате которых появилось множество формализованных описаний лексики **matsukawa-yokota-1991-developments**. Так как распространенные базы данных лексики ЕЯ не являются онтологиями и не имеют достаточной степени формализации (например, WordNet), создаются онтологии, являющиеся своего рода "надстройкой" над такими базами данных, самой известной из которых является lemon **McCrae_2012**.

Многие из приведенных выше онтологий созданы с использованием технологии Semantic Web **sem_web**, который является внешней технологией по отношению к существующим решениям для обработки естественных языков, поэтому последним приходится обращаться к ней с помощью API и стандартизованных языков запросов (в частности, SPARQL) **Bouayad_2014**.

Стоит отметить, что несмотря на активное развитие в направлении применения онтологий для обработки ЕЯ, многие популярные NLP-библиотеки (например, NLTK **nltk** и spaCy **spacy**) в принципе не поддерживают использование онтологий, а большинство инструментов для разметки естественноязыковых текстов используют обычно свой формат, что требует использования специфичных для таких инструментов парсеров и конвертеров, чтобы данные можно было применить при решении каких-либо задач **Erekhinskaya2020TenWO**.

Таким образом, в настоящее время в данной области можно выделить следующие проблемы:

1. Отсутствие унификации (стандартизации) приведенных выше решений приводит к существенным накладным расходам на их интеграцию и значительно усложняет построение различных систем с их использованием в силу большой трудоемкости их интеграции **Standard2021**, **GolenkovProblems2021**.
2. Несмотря на то, что онтологии потенциально способствуют решению широкого круга задач в сфере обработки естественного языка, большинство движимых онтологий NLP-систем сконцентрированы на решении

специализированных задач (например, только генерации текста, только заполнения онтологии или только обеспечения поиска с помощью естественного языка).

3. Создано довольно большое количество частных лингвистических онтологий, формализующих, однако, лишь некий подраздел предметной области лингвистики (в особенности лексики), что отчасти вытекает из предыдущего пункта. В то же время, существующие лингвистические онтологии верхнего уровня (например, OLiA) все равно не до конца решают проблему унификации, т.к. им требуется вводить промежуточный уровень для интеграции полученных в результате NLP-анализа данных с фрагментами онтологии.

Так как используемый в технологии OSTIS язык – sc-код – обладает достаточной экспрессивностью для описания знаний любого вида, а сама технология нацелена на создание интероперабельных интеллектуальных систем нового поколения, естественно-языковые интерфейсы ostis-систем смогут справляться с широким кругом задач по обработке текстов на естественных языках – будь то синтез естественно-языковых текстов в целом, ведение диалога в диалоговых системах, поиск с использованием естественного языка, выделение информации из текстов и т.п. При этом в то время как в текущем состоянии сферы обработки естественных языков данные классы задач выполняются зачастую специализированными средствами и требуют дополнительных затрат на обеспечение потенциальной совместимости с конкретными компьютерными системами, в рамках технологии OSTIS для их решения будет использоваться один универсальный язык смыслового представления знаний, на котором будут написаны как компоненты решателя задач, так и онтология языков и конкретных предметных областей, что позволит решить проблему интероперабельности.

Более того, онтология естественных языков, разработанная в рамках такой технологии, могла бы быть использована не только для решения прикладных задач по обработке естественного языка, но и для обеспечения интероперабельности данных, полученных в ходе лингвистических исследований, что было бы ценным вкладом в область теоретической лингвистики.

Наконец, онтологию естественных языков можно рассматривать в качестве подмножества онтологии языков вообще (как естественных, так и искусственных и формальных), чего не делают рассмотренные выше существующие онтологии. Это позволит концептуализировать естественный язык в одной системе с языками программирования и более тесно связать используемые в соответствующих предметных областях понятия для более эффективного решения задач по обработке естественного языка в интеллектуальных компьютерных системах.

Цель данной работы – предложить базовые средства формального описания синтаксиса и денотационной семантики различных языков в виде фрагмента онтологии языков и информационных конструкций, который можно будет использовать при проектировании интеллектуальных компьютерных систем нового поколения.

§ 2.7.1. Формализация естественных языков

Как уже говорилось выше, для использования достижений лингвистики при проектировании интеллектуальных компьютерных систем требуется представить полученные результаты в формальном виде. В данном разделе мы предложим формализацию основных лингвистических концептов, выполненную на формальном языке представления знаний – sc-коде.

язык

- ⇒ *разбиение**:
- { • *естественный язык*
 - ⇒ *пояснение**:
 - [Естественный язык представляет собой язык, который не был создан целенаправленно]
- *искусственный язык*
 - ⇒ *пояснение**:
 - [Искусственный язык представляет собой язык, специально разработанный для достижения определенных целей]
 - Э *Эсперанто*
 - Э *Python*
 - ▷ *сконструированный язык*
 - ⇒ *пояснение**:
 - [Сконструированный язык представляет собой искусственный язык, предназначенный для общения людей]
 - Э *Эсперанто*
- }
- ▷ *международный язык*
 - ⇒ *пояснение**:

[Международный язык представляет собой естественный или искусственный язык, использующийся для общения людей из разных стран]

- Э Английский язык
- Э Русский язык

плановый язык

⇐ пересечение*:

- {• сконструированный язык
- международный язык

}

язык общения

⇐ объединение*:

- {• естественный язык
- сконструированный язык

}

Э Английский язык

Э Русский язык

Э Эсперанто

⇐ объединение*:

- {• корневой язык

⇒ пояснение*:

[Корневой язык представляет собой язык, для которого характерно полное отсутствие словоизменения и наличие грамматической значимости порядка слов, состоящих только из корня.]

Э Английский язык

- агглютинативный язык

⇒ пояснение*:

[Агглютинативный язык характеризуется развитой системой употребления суффиксов, приставок, добавляемых к неизменяемой основе слова, которые используются для выражения категорий числа, падежа, рода и др.]

Э Английский язык

- флексивный язык

⇒ пояснение*:

[Для флексивного языка характерно развитое употребление окончаний для выражения категорий рода, числа, падежа, сложная система склонения глаголов, чередование гласных в корне, а также строгое различие частей речи.]

Э Русский язык

- профлексивный язык

⇒ пояснение*:

[Для профлексивного языка характерны агглютинация (в случае именного словоизменения), флексия и чередование гласных (аблаут) (в случае глагольного словоизменения).]

}

§ 2.7.2. Формализация синтаксиса естественных языков

Лексема – минимальная единица языка, имеющая семантическую интерпретацию и обозначающая концепт, отражающий взгляд на мир некоторого языкового сообщества **glossary**.

Грамматическая категория – система противопоставленных друг другу рядов грамматических форм с однородными значениями. В рамках нашей формализации предлагается представить грамматические категории как классы ролевых отношений, каждый из которых соответствует определенному грамматическому значению. Следует отметить, что приводится онтология основных грамматических категорий, часто встречающихся в естественных языках, а не всех возможных.

грамматическая категория

Э лицо

⇐ семейство подмножеств*:

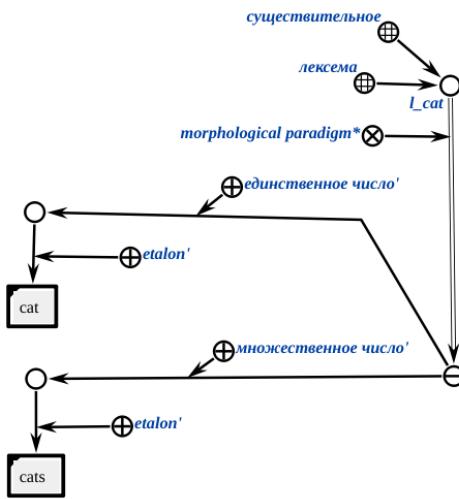
ролевое отношение

Э первое лицо'

Э второе лицо'

- Э *третье лицо'*
- Э *число*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *единственное число'*
 - Э *множественное число'*
 - Э *двойственное число'*
 - Э *тройственное число'*
 - Э *научальное число'*
- Э *род*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *мужской род'*
 - Э *средний род'*
 - Э *женский род'*
- Э *падеж*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *именительный падеж'*
 - Э *родительный падеж'*
 - Э *дательный падеж'*
 - Э *винительный падеж'*
 - Э *творительный падеж'*
 - Э *предложный падеж'*
 - Э *звательный падеж'*
 - Э *абсолютивный падеж'*
 - Э *эргативный падеж'*
- Э *время*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *настоящее время'*
 - Э *прошедшее время'*
 - Э *будущее время'*
- Э *наклонение*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *изъявительное наклонение'*
 - Э *повелительное наклонение'*
 - Э *сослагательное наклонение'*
 - Э *условное наклонение'*
- Э *залог*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *действительный залог'*
 - Э *страдательный залог'*
 - Э *средний залог'*
 - Э *возвратный залог'*
 - Э *взаимный залог'*
- Э *вид*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *совершенный вид'*
 - Э *несовершенный вид'*
 - Э *общий вид'*
 - Э *прогрессивный вид'*
 - Э *перфектный вид'*
- Э *степень сравнения*
 - ⇐ семейство подмножеств*:
 - ролевое отношение*
 - Э *положительная степень сравнения'*
 - Э *сравнительная степень сравнения'*
 - Э *превосходная степень сравнения'*

Пример формализации части приведенных выше отношений на языке sc.g приведен на рисунке 4.2.4.



= Пример спецификации лексемы в базе знаний.

Часть речи – категория, представляющая собой класс синтаксически эквивалентных знаков ЕЯ.

часть речи

⇐ семейство подмножеств*:

лексема

Э существительное

Э прилагательное

Э глагол

Э наречие

Э предлог

Э комплементатор

Э вспомогательный глагол

Э детерминант

морфологическая парадигма* - бинарное ориентированное отношение, связывающее лексему и множество ее словоформ.

Словоформа – подмножество лексемы, которому принадлежат все вхождения лексемы с определенными грамматическими значениями. В рамках нашей онтологии словоформа понимается несколько иначе, чем принято в лингвистике, так как все вхождения лексемы в технологии OSTIS являются файлами.

При формализации синтаксиса в основном использовались стандартные положения генеративной грамматики adger2003core, jackendoff1977x, haegeman1994introduction, carnie2012syntax.

Дистрибуция знака — это подмножество синтаксических правил, в которые входит данный знак.

Составляющая – элемент множества С подмножеств кортежа вхождений лексем S, которое содержит в качестве элементов как сам S, так и все вхождения лексем в S, таким образом, что любые два подмножества, входящие в С, либо не пересекаются, либо одно из них включается в другое.

Непосредственно составляющая – есть множество составляющих S, в которое входят составляющие А и В. В является непосредственно составляющей А если и только если В является подмножеством А и нет такой составляющей С, которая является подмножеством А и подмножеством которой является В.

Элементарная составляющая – элемент кортежа вхождений лексем L, являющихся непосредственно составляющими множества составляющих С и не имеющими дочерних составляющих.

Синтаксическая группа – класс составляющих, в который входят составляющие с вершинами, принадлежащими к одной части речи. Синтаксические группы представляют собой либо синглетон (минимально включают в себя вершину), либо упорядоченную пару, состояющую из вершины и другой синтаксической группы.

Вершина — составляющая, дистрибуция которой совпадает с дистрибуцией всей синтаксической группы.

составляющая \Rightarrow разбиение*:

- {• синтаксическая группа
- вершина

}

синтаксическая группа \Rightarrow разбиение*:

- {• именная группа

 \Rightarrow пояснение*:

[Именная группа – синтаксическая группа, вершиной которой является существительное.]

- глагольная группа

 \Rightarrow пояснение*:

[Глагольная группа – синтаксическая группа, вершиной которой является глагол.]

- группа прилагательного

 \Rightarrow пояснение*:

[Группа прилагательного – синтаксическая группа, вершиной которой является прилагательное.]

- наречная группа

 \Rightarrow пояснение*:

[Наречная группа – синтаксическая группа, вершиной которой является наречие.]

- предложная группа

 \Rightarrow пояснение*:

[Предложная группа – синтаксическая группа, вершиной которой является предлог.]

- группа комплементатора

 \Rightarrow пояснение*:

[Группа комплементатора – синтаксическая группа, вершиной которой является комплементатор.]

- временная группа

 \Rightarrow пояснение*:

[Временная группа – синтаксическая группа, вершиной которой является вспомогательный либо модальный глагол.]

- группа детерминанта

 \Rightarrow пояснение*:

[Группа детерминанта – синтаксическая группа, вершиной которой является детерминант.]

}

 \Rightarrow разбиение*:

- {• максимальная проекция вершины

- промежуточная проекция вершины

}

При этом для упрощения могут быть введены более узкие классы, являющиеся пересечением приведенных выше, например *максимальная проекция вершины группы детерминанта*.

максимальная проекция вершины группы детерминанта \Leftarrow пересечение*:

- {• группа детерминанта

- максимальная проекция вершины

}

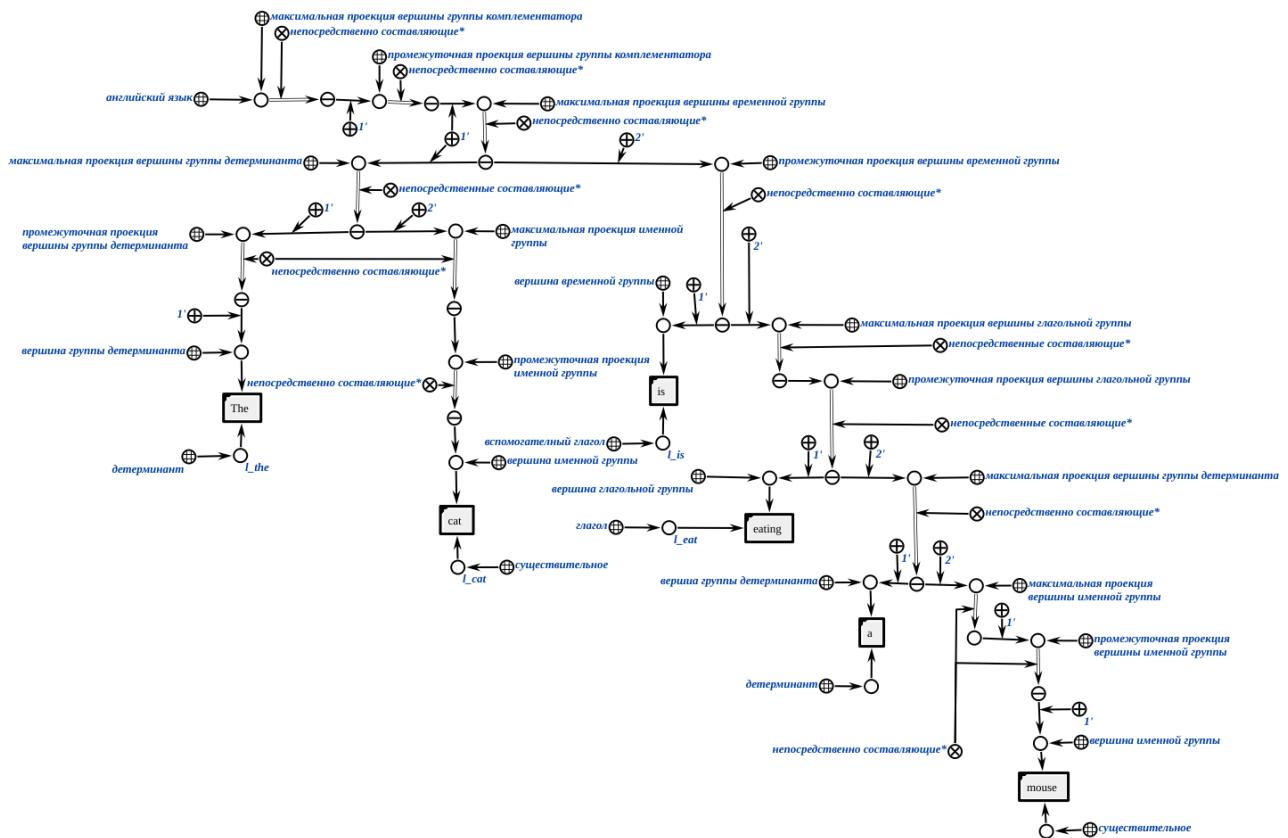
Пример синтаксической структуры предложения, записанный с применением введенных выше понятий представлен на рисунке 2.7..2.

Структуры синтаксических групп не являются произвольными – элементы внутри группы могут граничить только с определенными множествами элементов. Ниже приводятся возможные структуры синтаксических групп. Знак $>$ следует читать как "состоит из". В скобках указаны optionalные элементы.

Группа детерминанта:

- Максимальная проекция вершины группы детерминанта $->$ (Максимальная проекция вершины группы детерминанта) Промежуточная проекция вершины группы детерминанта
- Промежуточная проекция вершины группы детерминанта $->$ Вершина группы детерминанта (Максимальная проекция вершины именной группы)

Именная группа:



= Пример синтаксической структуры предложения.

- Максимальная проекция вершины именной группы -> (Максимальная проекция вершины группы детерминанта) Промежуточная проекция вершины именной группы
 - Промежуточная проекция вершины именной группы -> (Максимальная проекция вершины группы прилагательного) Промежуточная проекция вершины именной группы ИЛИ Промежуточная проекция вершины именной группы (Максимальная проекция вершины предложной группы)
 - Промежуточная проекция вершины именной группы -> Вершина именной группы (Максимальная проекция вершины предложной группы)

Глагольная группа:

- Максимальная проекция вершины глагольной группы -> Промежуточная проекция вершины глагольной группы
 - Промежуточная проекция вершины глагольной группы -> Промежуточная проекция вершины глагольной группы (Максимальная проекция вершины предложной группы) ИЛИ Промежуточная проекция вершины глагольной группы (Максимальная проекция вершины наречной группы)
 - Промежуточная проекция вершины глагольной группы -> Вершина глагольной группы (Максимальная проекция вершины именной группы)

Наречная группа:

- Максимальная проекция вершины наречной группы -> Промежуточная проекция вершины наречной группы
 - Промежуточная проекция вершины наречной группы -> (Максимальная проекция вершины наречной группы)
Промежуточная проекция вершины наречной группы
 - Промежуточная проекция вершины наречной группы -> Вершина наречной группы (Максимальная проекция вершины предложной группы)

Группа прилагательного:

- Максимальная проекция вершины группы прилагательного -> Промежуточная проекция вершины группы прилагательного
 - Промежуточная проекция вершины группы прилагательного -> (Максимальная проекция вершины наречной группы) + Промежуточная проекция вершины группы прилагательного
 - Промежуточная проекция вершины группы прилагательного -> Вершина группы прилагательного (Максимальная проекция вершины предложной группы)

Предложная группа:

- Максимальная проекция вершины предложной группы -> Промежуточная проекция вершины предложной группы
- Промежуточная проекция вершины предложной группы -> Промежуточная проекция вершины предложной группы (Максимальная проекция вершины предложной группы) ИЛИ (Максимальная проекция вершины наречной группы) Промежуточная проекция вершины предложной группы
- Промежуточная проекция вершины предложной группы -> Вершина предложной группы (Максимальная проекция вершины именной группы)

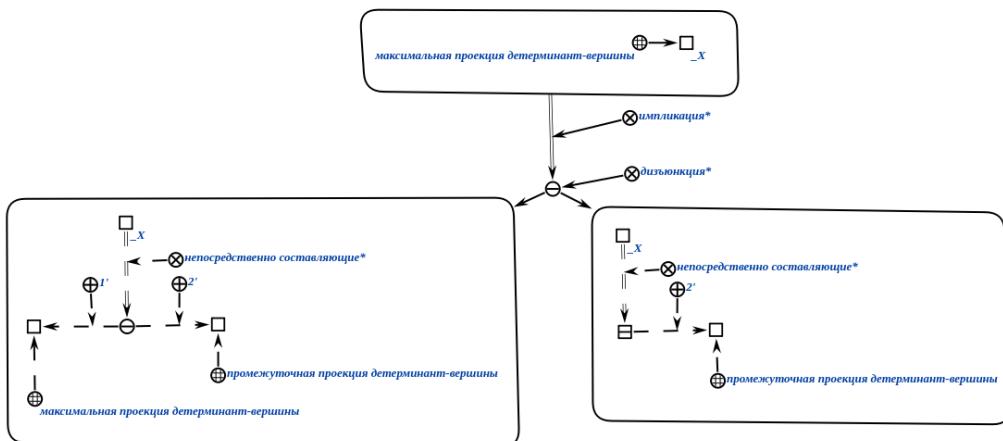
Временная группа:

- Максимальная проекция вершины временной группы -> (Максимальная проекция вершины группы детерминанта) Промежуточная проекция вершины временной группы
- Промежуточная проекция вершины временной группы -> Вершина временной группы (Максимальная проекция вершины глагольной группы)

Группа комплементатора:

- Максимальная проекция вершины группы комплементатора -> (Максимальная проекция вершины некоторой синтаксической группы) Промежуточная проекция вершины группы комплементатора
- Промежуточная проекция вершины группы комплементатора -> Вершина группы комплементатора Максимальная проекция вершины временной группы

В формальном виде данные правила можно представить следующим образом (см. рисунок 2.7..3).



= Пример правила структуры синтаксической группы.

Комплмент – синтаксическая группа, являющаяся сестрой вершины. Сестрами считаются составляющие, являющиеся непосредственно составляющими одной и той же составляющей.

Адъюнкт – синтаксическая группа, являющаяся дочерью (непосредственно составляющей) промежуточной проекции и сестрой промежуточной проекции вершины той же синтаксической группы.

Спецификатор – синтаксическая группа, являющейся дочерью максимальной проекции и сестрой промежуточной проекции.

Приведенные выше правила структуры синтаксических групп можно обобщить и свести к трем более абстрактным.

Правило спецификатора: $XP \rightarrow (YP) X'$

Правило адъюнкта: $X' \rightarrow X' (ZP) \mid X' \rightarrow (ZP) X'$

Правило комплемента: $X' \rightarrow X (WP)$

Формальное представление данных правил аналогично приведенному на рисунке 2.7..3.

§ 2.7.3. Формализация денотационной семантики естественных языков

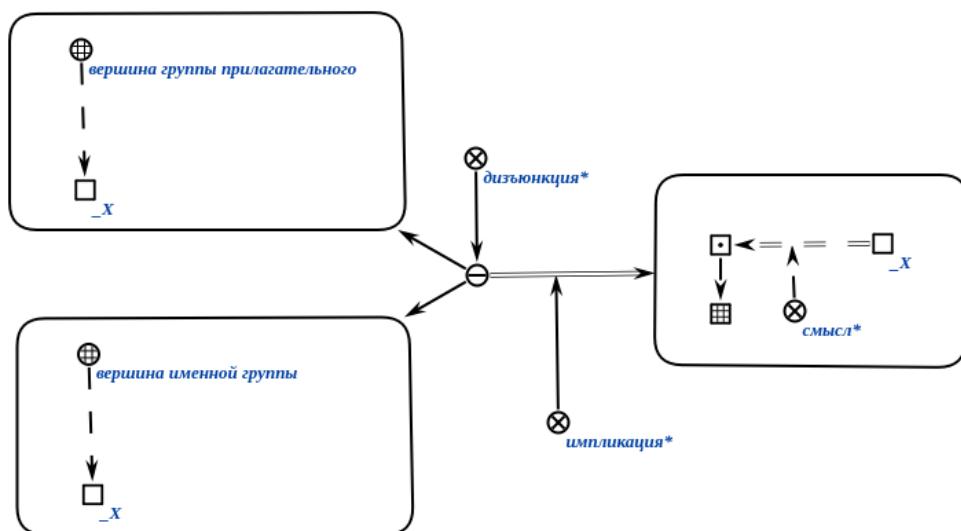
Денотационная семантика языка специфицирует интерпретацию элементов синтаксиса данного языка и представляет собой множество формул, описывающих то, каким образом знаковым конструкциям языка ставятся в соответствие обозначаемые ими сущности и конфигурации отношений между этими сущностями.

Денотационная семантика естественных языков должна обладать свойством композициональности – т.е. интерпретация всего высказывания должна выводиться из интерпретации отдельных его частей. Таким образом, необходимо предоставить формальное описание интерпретации элементов синтаксиса ЕЯ, представленных в предыдущем разделе, а также описание правил совмещения интерпретации отдельных элементов для получения смысла всего высказывания.

В данной главе мы предлагаем вариант формализации денотационной семантики естественных языков в рамках технологии OSTIS, для составления которой использовались стандартные положения формальной семантики [heim1998semantics](#), [Winter+2016](#), [portner2008formal](#).

Рассмотрим примеры правил, реализующих денотационную семантику языка. Приведенные ниже правила должны применяться последовательно и позволяют получить смысл текста естественного языка по его синтаксической структуре, "поднимаясь" по дереву составляющих от вершин к максимальным проекциям.

На рисунке [Правило интерпретации вершины группы прилагательного и вершины именной группы](#) приведено правило, по которому происходит интерпретация вершин именной группы и группы прилагательного. Смыслом таких вершин является класс, например: прилагательному "черный" соответствует множество черных объектов, а существительному "кот" – множество котов.



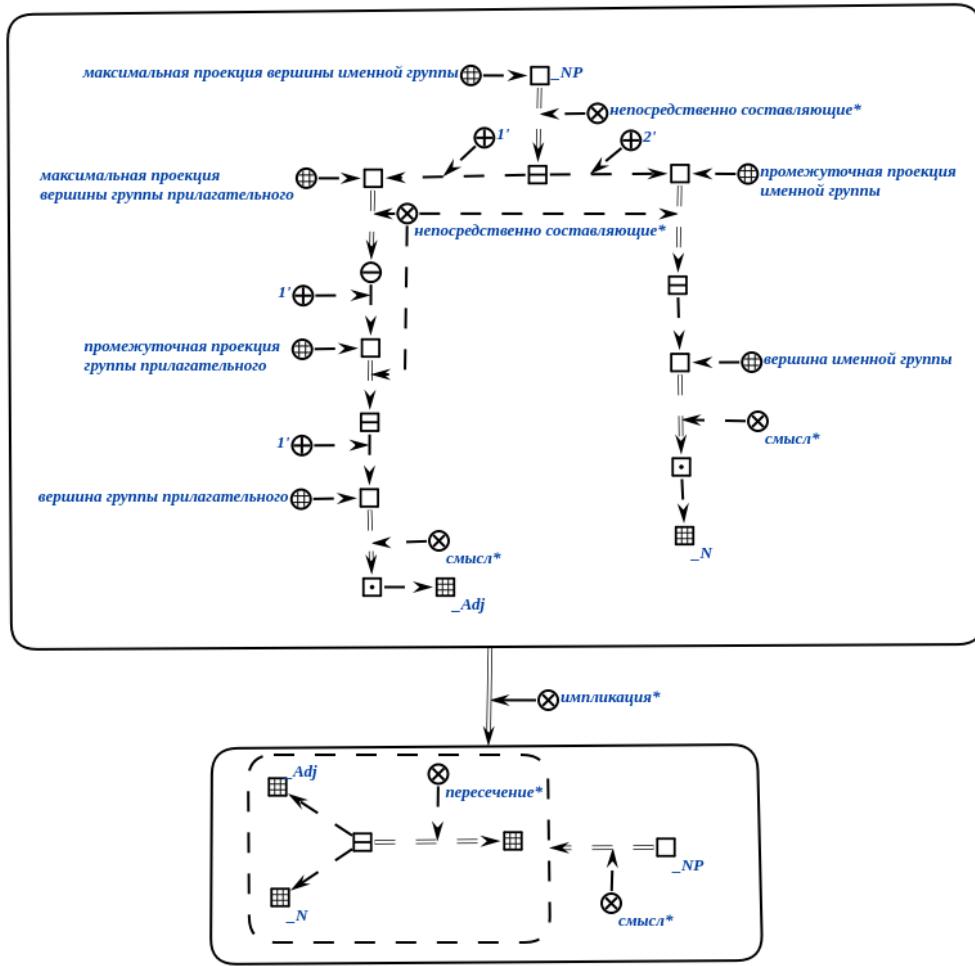
= Правило интерпретации вершины группы прилагательного и вершины именной группы

На рисунке [Правило интерпретации максимальной проекции вершины именной группы](#) приведено правило, по которому происходит интерпретация именной группы, максимальная проекция которой включается в себя также группу прилагательного. Как говорилось выше, для применения данного правила необходимо предварительное применение правила, представленного на рисунке [Правило интерпретации вершины группы прилагательного и вершины именной группы](#). Смыслом таких конструкций является класс, являющийся результатом пересечения классов, полученных в результате интерпретации вершин групп прилагательного и именной группы по отдельности. Например: "черный кот" – множество черных котов, пересечение множества котов и черных объектов.

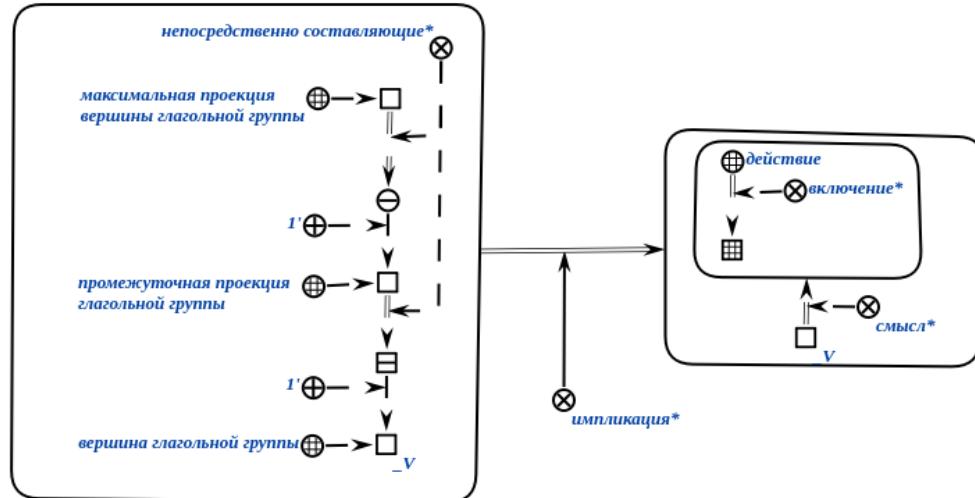
На рисунке [Правило интерпретации максимальной проекции вершины глагольной группы, содержащей непереходный глагол](#) приведено правило, по которому происходит интерпретация глагольной группы. Необходимость включения в посылку правила всей ветки глагольной группы объясняется ее необходимостью для определения типа глагола – данное правило предназначено для интерпретации непереходных глаголов. Смыслом такой конструкции является класс действий.

На рисунке [Правило интерпретации максимальной проекции вершины группы детерминанта](#) приведено правило, по которому происходит интерпретация группы детерминанта с неопределенным артиклем. Смыслом такой конструкции является существование элемента класса, являющегося смыслом входящей в состав данной группы детерминанта именной группы.

На рисунке [Правило интерпретации промежуточной проекции вершины временной группы](#) приведено правило, по которому происходит интерпретация промежуточной проекции вершины временной группы, состоящей из вспомогательного глагола и полнозначного глагола. Вспомогательный глагол в данном случае задает класс действий по времени (является ли оно запланированным, выполняемым, уже выполненным и т.д.).

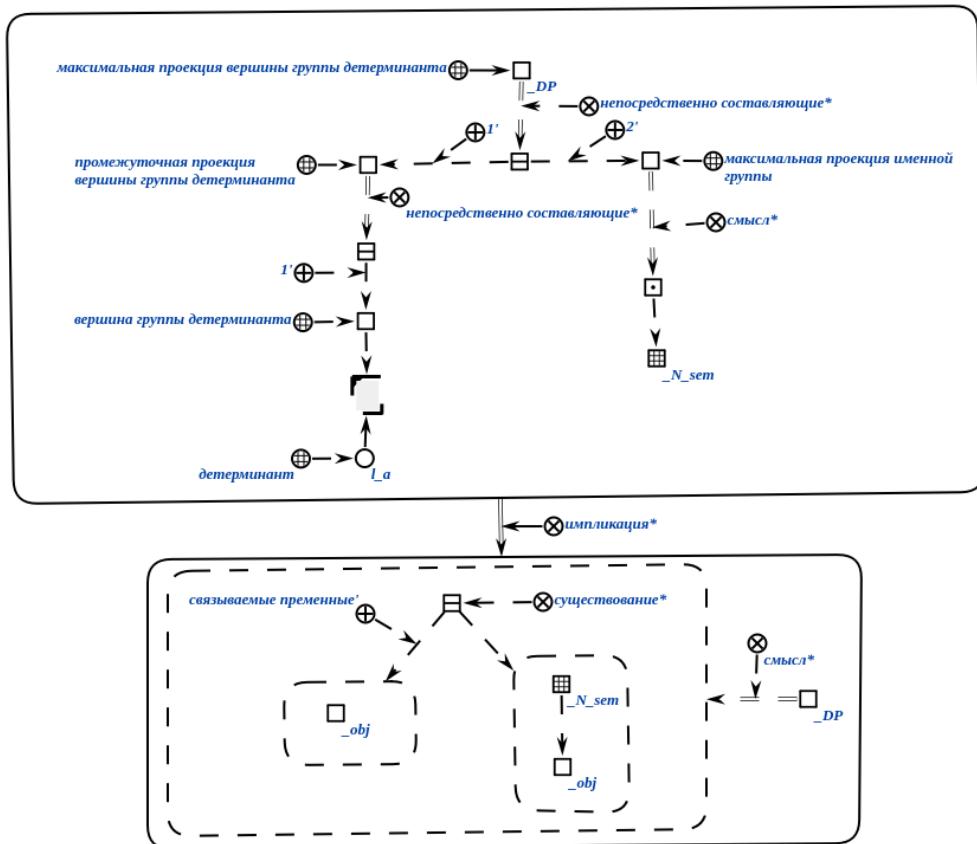


= Правило интерпретации максимальной проекции вершины именной группы



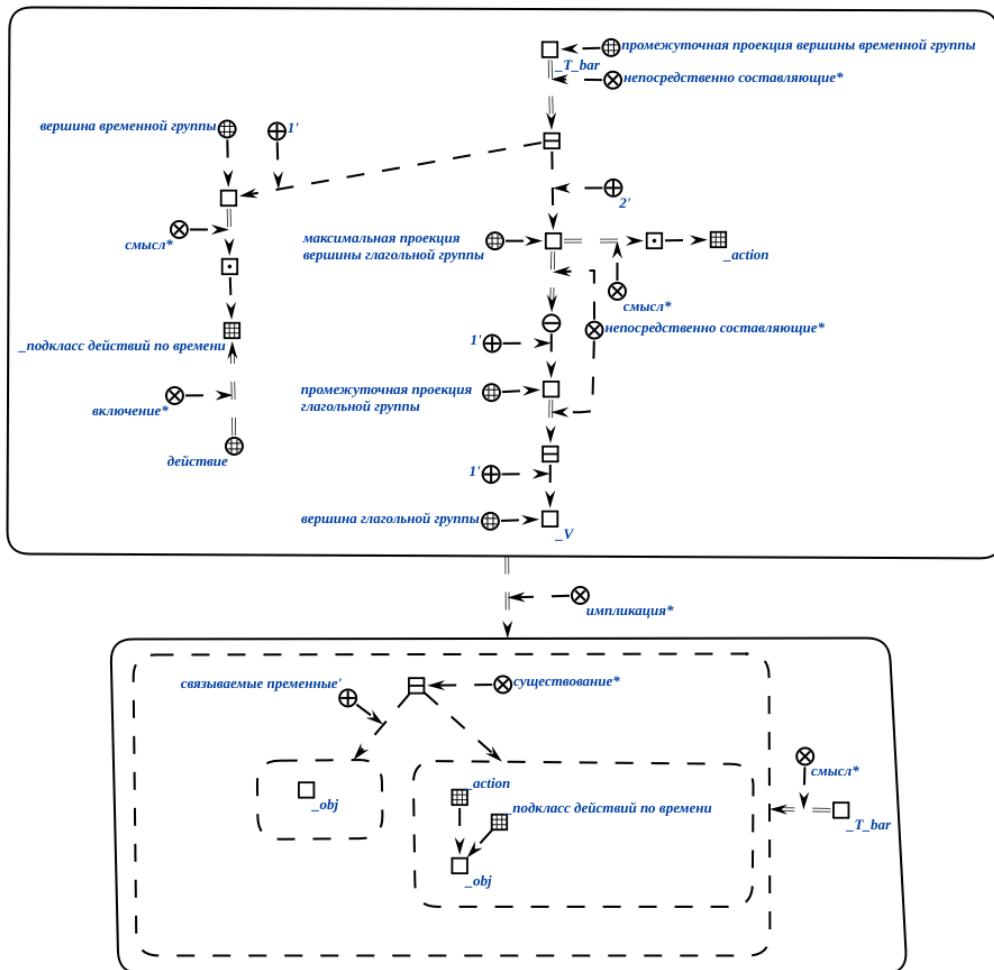
= Правило интерпретации максимальной проекции вершины глагольной группы, содержащей непереходный глагол

На рисунке *Правило интерпретации максимальной проекции вершины временной группы* приведено правило, по которому происходит интерпретация максимальной проекции вершины временной группы на основе полученного на предыдущем шаге смысла промежуточной проекции вершины временной группы и смысла максимальной проекции группы детерминанта.

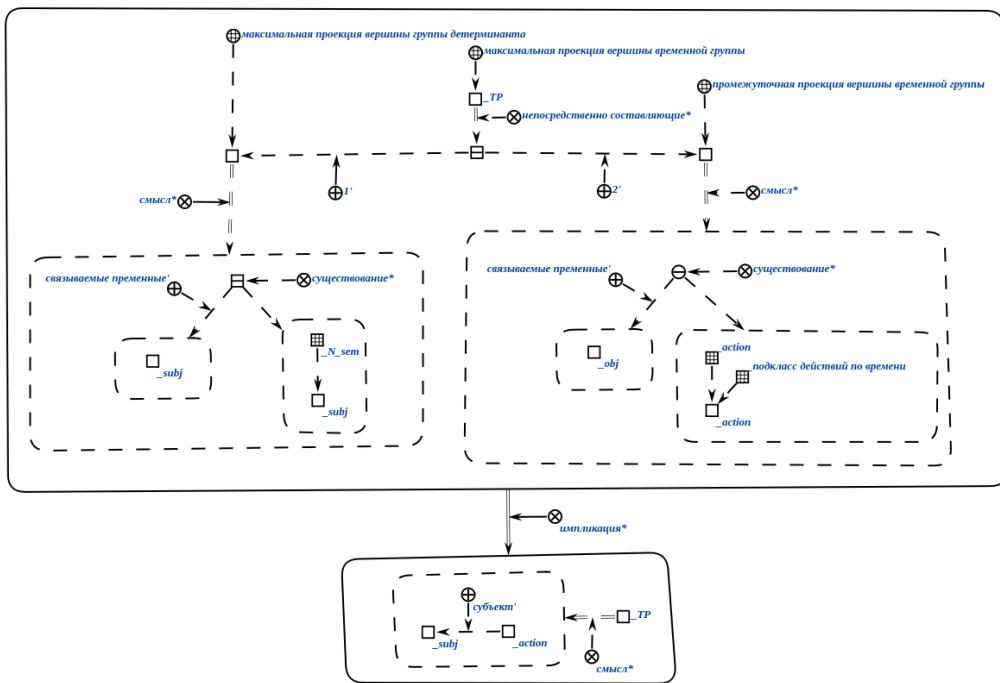


= Правило интерпретации максимальной проекции вершины группы детерминанта

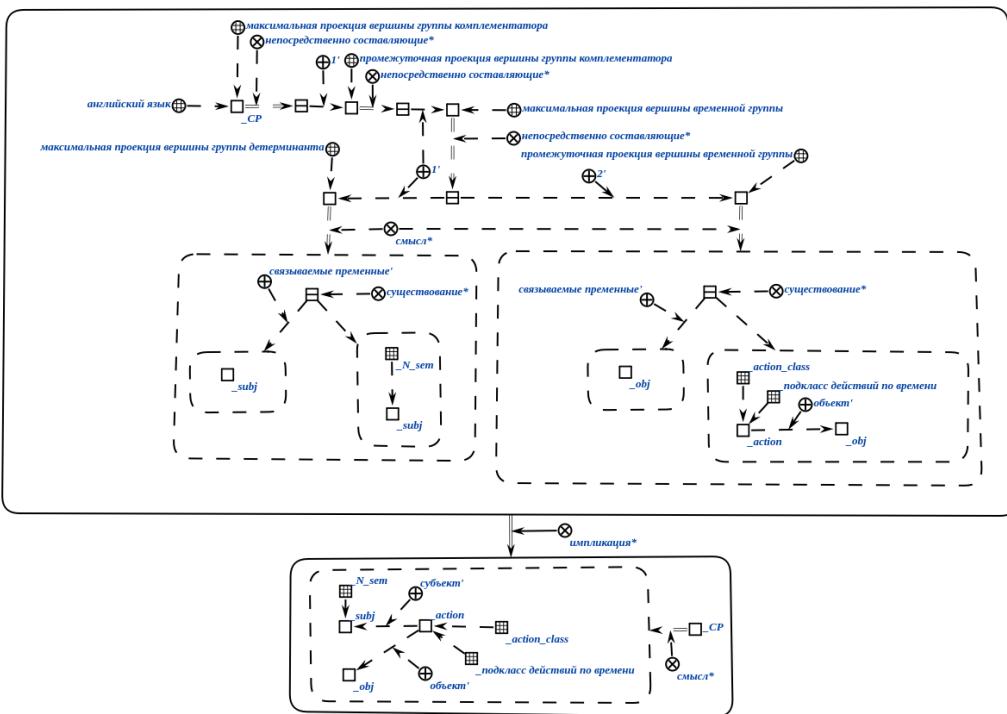
На рисунке *Правило интерпретации предложения с переходным глаголом* приведено правило, по которому происходит интерпретация максимальной проекции вершины группы комплементатора на основе полученных на предыдущих шагах смыслов более частных конструкций. Данным правилом задается интерпретация предложения с переходным глаголом.



= Правило интерпретации промежуточной проекции вершины временной группы



= Правило интерпретации максимальной проекции вершины временной группы



= Правило интерпретации предложения с переходным глаголом

Часть 3.

Многоагентные модели решателей задач интеллектуальных компьютерных систем нового поколения

Описание к главе

Глава 3.1.

Формализация понятий действия, задачи, метода, средства, навыка и технологии

⇒ *автор**:

- Шункевич Д.В.
- Ковалёв М.В.
- Никифоров С.Ф.

⇒ *аннотация**:

[В главе уточнена формальная трактовка таких понятий, как действие, задача, класс действий, класс задач, метод, навык, что в совокупности позволило определить на их основе понятие модели решения задач.]

⇒ *подраздел**:

- § 3.1.1. Глобальная предметная область и онтология действий, действий, методов, средств и технологий
- § 3.1.2. Локальные предметные области и онтологии действий

⇒ *библиографическая ссылка**:

- Standard2021
- Tuzov1986

Введение в Главу 3.1.

Возможности решателя задач интеллектуальной системы в значительной степени определяются качеством ее базы знаний. Другими словами, при разработке решателей задач необходимо описывать не только *операционную семантику* решателя, то есть семейство интерпретаторов соответствующих моделей решения задач, но и *декларативную семантику* модели решения задач, то есть собственно тексты программ (не программ низкоуровневых агентов, а программ более высокого уровня, интерпретируемых соответствующим набором агентов), логические утверждения, конкретные конфигурации искусственных нейронных сетей и т.д.

В рамках данной главы формально уточняются в рамках соответствующего набора онтологий такие понятия, как *действие, задача, модель решения задач, метод, навык* и другие, на основе которых в Главе 3.2. *Агентно-ориентированные модели гибридных решателей задач ostis-систем* уточняется собственно модель гибридного решателя задач *ostis-системы*.

Разработка указанного семейства онтологий позволяет:

- явно связать *класс задач* и способ (метод) решения задач данного класса;
- это, в свою очередь, позволит накапливать более сложные компоненты решателей задач и еще больше упростить их интеграцию, поскольку вместе с коллективом sc-агентов в соответствующий компонент также будут входить необходимые фрагменты базы знаний, априори согласованные с указанным коллективом sc-агентов;
- это, в свою очередь, позволит сделать средства автоматизации разработки решателей задач более интеллектуальными, в частности, позволит автоматизировать процесс подбора компонентов решателя на основе спецификации классов задач, которые должна уметь решать проектируемая интеллектуальная система;
- в дальнейшем это позволит интеллектуальной системе самостоятельно обращаться в библиотеку компонентов решателей задач и подбирать компоненты, исходя из новых классов задач, с которыми столкнулась система, то есть позволит интеллектуальной системе самостоятельно изучать новые *навыки*;
- с другой стороны, такой подход позволит интеллектуальной системе самостоятельно подобрать комбинацию моделей решения задач для решения задач определенного класса (точнее говоря, поскольку в основу решателя положен многоагентный подход, то коллектив sc-агентов, интерпретирующих различные модели решения задач, сможет лучше определить, какие именно из sc-агентов и в каком порядке должны работать при решении конкретной комплексной задачи).

§ 3.1.1. Глобальная предметная область и онтология воздействий, действий, методов, средств и технологий

⇒ *ключевое понятие**:

- *воздействие*
- *действие*
- *задача*
- *класс действий*
- *класс задач*
- *метод*
- *язык представления методов*
- *модель решения задач*
- *навык*

⇒ *ключевое отношение**:

- *субъект'*
- *объект'*
- *цель**

⇒ *подраздел**:

- *Пункт 3.1.1.1. Понятие действия и классификация действий*
- *Пункт 3.1.1.2. Понятие задачи и классификация задач*
- *Пункт 3.1.1.3. Понятие класса действий и класса задач*
- *Пункт 3.1.1.4. Понятие метода*
- *Пункт 3.1.1.5. Спецификация методов и понятие навыка*
- *Пункт 3.1.1.6. Понятие класса методов и языка представления методов*
- *Пункт 3.1.1.7. Общая классификация языков представления методов*
- *Пункт 3.1.1.8. Понятие модели решения задач*

Пункт 3.1.1.1. Понятие действия и классификация действий

Прежде чем говорить о моделях решения задач и решателя задач, необходимо формально уточнить понятие задачи и понятие действия, направленного на решение той или иной задачи или ее подзадач.

В рамках *Технологии OSTIS* задачу будем трактовать как формальную спецификацию некоторого действия, поэтому целесообразно вначале уточнить понятие *действия*, которое определяется через понятие *воздействия*. Рассмотрим спецификацию понятия *воздействие* в SCn-коде.

воздействие

- := [процесс воздействия одной сущности (или некоторого множества *сущностей*) на другую *сущность* (или на некоторое множество других *сущностей*)]
- := [процесс, в котором могут быть явно выделены хотя бы одна воздействующая сущность (*субъект'*) и хотя бы одна *сущность*, на которую осуществляется воздействие (*объект'*)]
- := [акция]
- ⊂ процесс
 - := [динамическая структура]

субъект'

- := [воздействующая сущность']
- := [сущность, создающая причину изменений другой сущности (объекта)']

объект'

- := [воздействуемая сущность']
- := [сущность, являющаяся в рамках заданного воздействия исходным условием (аргументом), необходимым для выполнения этого воздействия']

Каждому *воздействию* может быть поставлен в соответствие (1) некоторый *субъект'*, т.е. сущность, осуществляющая *воздействие* (в частности, это может быть некоторое физическое поле), и (2) некоторый *объект'*, т.е. сущность, на которую воздействие направлено. Если *воздействие* связано с *материальной сущностью*, то его объектом является либо сама эта *материальная сущность*, либо некоторая ее пространственная часть.

Поскольку *воздействия* являются частным видом *процессов*, воздействиями наследуются все свойства *процессов*. В частности, используются все *параметры*, заданные на множестве *процессов*, например, *длительность**, *момент начала процесса**, *момент завершения процесса^Δ*. Подробнее эти отношения описываются в Главе 2.4. *Представление формальных онтологий базовых классов сущностей в ostis-системах*.

Так же, так как воздействие является процессом и, соответственно, представляет собой *динамическую структуру*, то и знак *субъекта воздействия'*, и знак *объекта воздействия'* являются элементами данной структуры. В связи с этим можно рассматривать отношения *субъект воздействия'* и *объект воздействия'* как *ролевые отношения*. Данный факт не запрещает вводить аналогичные *неролевые отношения*, однако это нецелесообразно.

Рассмотрим спецификацию понятия *действие* в SCn-коде.

действие

- := [воздействие, в котором *субъект'* осуществляет *воздействие* целенаправленно, т.е. в соответствии с некоторой *целью**]
- := [целенаправленное воздействие, выполняемое одним или несколькими субъектами (кибернетическими системами) с возможным применением некоторых инструментов]
- := [акт]
- := [операция]
- := [осознанное воздействие]
- := [активное воздействие]
- ⊂ *воздействие*
 - := [процесс, в котором могут быть явно выделены хотя бы одна воздействующая сущность (*субъект воздействия'*) и хотя бы одна сущность, на которую осуществляется воздействие (*объект воздействия'*)]
 - ⊂ *процесс*
 - := [целенаправленный ("осознанный") процесс, выполняемый (управляемый, реализуемый) неким субъектом]
 - := [работа]
 - := [процесс решения некоторой задачи]
 - := [процесс достижения некоторой цели]
 - := [целостный фрагмент некоторой деятельности]
 - := [целенаправленный процесс, управляемый некоторым субъектом]
 - := [процесс выполнения некоторого действия некоторым субъектом (исполнителем) над некоторыми объектами]
- ⇒ *разбиение*:*
 - Разбиение класса действий по отношению к памяти кибернетической системы
 - = {• информационное действие
 - ▷ действие в sc-памяти
 - поведенческое действие
 - ▷ действие во внешней среде ostis-системы
 - эффекторное действие
 - ▷ эффекторное действие ostis-системы
 - рецепторное действие
 - ▷ рецепторное действие ostis-системы
 - }
- ▷ элементарное действие
 - := [действие, выполнение которого не требует его декомпозиции на множество поддействий (частных действий, действий более низкого уровня)]
 - ⇒ *пояснение*:*
 - [Элементарное действие выполняется одним индивидуальным субъектом и является либо элементарным действием, выполняемым в памяти этого субъекта (элементарным действием его "процессора"), либо элементарным действием одного из его эффекторов.]
- ▷ сложное действие
 - ⇒ *разбиение*:*
 - {• действие, выполняемое кибернетической системой в собственной памяти
 - действие, выполняемое кибернетической системой в своей внешней среде
 - действие, выполняемое кибернетической системой над своей физической оболочкой

*цель**

- := [целевая ситуация*]
- ⊂ *спецификация**
- := [описание того, что требуется получить (какая ситуация должна быть достигнута) в результате выполнения заданного (специфицируемого) действия*]

Каждое *действие*, выполняемое тем или иным *субъектом*, трактуется как процесс решения некоторой задачи, т.е. процесс достижения заданной *цели** в заданных условиях, и, следовательно, выполняется целенаправленно. При этом явное указание *действия* и его связи с конкретной задачей может не всегда присутствовать в памяти. Некоторые *задачи* могут решаться определенными субъектами перманентно, например, оптимизация базы знаний, поиск некорректностей и т.д. и для подобных задач не всегда есть необходимость явно вводить *структуру*, являющуюся формулировкой *задачи*.

Каждое *действие* может обозначать сколь угодно малое преобразование, осуществляющееся во внешней среде либо в памяти некоторой *кибернетической системы*, однако в памяти явно вводятся знаки только тех *действий*, для которых есть необходимость явно хранить в памяти их спецификацию в течение некоторого времени. При выполнении *действия* можно выделить этапы:

- построение *плана действия*, декомпозиция (детализация) действия в виде системы его *поддействий*;
- выполнение построенного плана *действия*

Результатом выполнения *информационного действия* в общем случае является некоторое новое состояние памяти информационной системы (не обязательно *sc-памяти*), достигнутое исключительно путем преобразования информации, хранящейся в памяти системы, то есть либо посредством генерации новых знаний на основе уже имеющихся, либо посредством удаления знаний, по каким-либо причинам ставших ненужными. Следует отметить, что если речь идет об изменении состояния *sc-памяти*, то любое преобразование информации можно свести к ряду элементарных действий по генерации, удалению или изменению инцидентности *sc-элементов* относительно друг друга.

В случае *поведенческого действия* результатом его выполнения будет новое состояние внешней среды. Очень важно отметить, что под внешней средой в данном случае понимаются также и компоненты системы, внешние с точки зрения памяти, то есть не являющиеся хранимыми в ней информационными конструкциями. К таким компонентам можно отнести, например, различные манипуляторы и прочие средства воздействия системы на внешний мир, то есть к поведенческим задачам можно отнести изменение состояния механической конечности робота или непосредственно вывод некоторой информации на экран для восприятия пользователем.

С точки зрения решения проблем, сформулированных в данной работе, наибольший интерес представляют информационные действия, выполняемые в памяти *ostis*-системы, то есть *действия в sc-памяти*. Классификация *действий в sc-памяти* представлена в базе знаний *Метасистемы IMS.ostis*, описывающей документацию текущего состояния *Технологии OSTIS MetacOSTIS-2022*эл.

На множестве действий задан ряд отношений, таких как *субъект действия*' (*исполнитель*'), *заказчик**, *объект действия*', *контекст действия**, *поддействие** *последовательность действий**, *результат** и других *Shunkevich.D.V.AgentOMMTCPSDIS-2018*ст, *MetacOSTIS-2022*эл.

Пункт 3.1.1.2. Понятие задачи и классификация задач

В свою очередь, *задачу* будем трактовать как спецификацию некоторого действия, в рамках которой, в зависимости от ситуации, при помощи перечисленных выше отношений может быть заранее указан контекст выполнения действия, способ его выполнения, исполнитель, заказчик, планируемый результат и т.д.

Рассмотрим спецификацию понятия *задача* в SCn-коде.

задача

- := [описание некоторого желаемого состояния или события либо в базе знаний, либо во внешней среде]
- := [формулировка задачи]
- := [задание на выполнение некоторого действия]
- := [постановка задачи]
- := [задачная ситуация]
- := [спецификация некоторого действия, обладающая достаточной полнотой для выполнения этого действия]

Каждая *задача* представляет собой спецификацию действия, которое либо уже выполнено, либо выполняется в текущий момент (в настоящее время), либо планируется (должно) быть выполненным, либо может быть выполнено (но не обязательно). В зависимости от конкретного класса задач, описываться может как внутреннее состояние самой интеллектуальной системы, так и требуемое состояние внешней среды.

Классификация задач может осуществляться по дидактическому признаку в рамках каждой предметной области, например, задачи на треугольники, задачи на системы уравнений и т.п.

Каждая *задача* может включать:

- факт принадлежности *действия* какому-либо частному классу *действий* (например, *действие. сформировать полную семантическую окрестность указываемой сущности*), в том числе состояние *действия* с точки зрения жизненного цикла (инициированное, выполняемое и т.д.);
- описание *цели** (*результата**) *действия*, если она точно известна;
- указание *заказчика** *действия*;
- указание *исполнителя** *действия* (в том числе коллективного);
- указание *аргумента(-ов) действия'*;
- указание инструмента или посредника *действия*;
- описание *декомпозиции действия**;
- указание *последовательности действий** в рамках *декомпозиции действия**, т.е. построение процедурного плана решения задачи. Другими словами, построение плана решения представляет собой декомпозицию соответствующего *действия* на систему взаимосвязанных между собой поддействий;
- указание области *действия*;
- указание условия инициирования *действия*;
- момент начала и завершения *действия*, в том числе планируемый и фактический, предполагаемая и/или фактическая длительность выполнения.

Некоторые задачи могут быть дополнительно уточнены контекстом – дополнительной информацией о сущностях, рассматриваемых в формулировке *задачи*, т.е. описанием того, что дано, что известно об указанных сущностях.

Кроме этого, *задача* может включать любую дополнительную информацию о *действии*, например:

- перечень ресурсов и средств, которые предполагается использовать при решении задачи, например, список доступных исполнителей, временные сроки, объем имеющихся финансов и т.д.;
- ограничение области, в которой выполняется *действие*, например, необходимо заменить одну *sc-конструкцию* на другую по некоторому правилу, но только в пределах некоторого *раздела базы знаний*;
- ограничение знаний, которые можно использовать для решения той или иной задачи, например, необходимо решить задачу по алгебре, используя только те утверждения, которые входят в курс школьной программы до седьмого класса включительно, и не используя утверждения, изучаемые в старших классах;
- и прочее.

Как и в случае с *действиями*, решаемыми системой, можно классифицировать *информационные задачи и поведенческие задачи*.

С другой стороны, с точки зрения формулировки поставленной задачи, можно выделить *декларативные формулировки задачи и процедурные формулировки задачи*. Следует отметить, что данные классы задач не противопоставляются друг другу, и могут существовать формулировки задач, использующие оба подхода.

задача

- ▷ *процедурная формулировка задачи*
- ▷ *декларативная формулировка задачи*
- ▷ *вопрос*
- ▷ *команда*
- ▷ *знание*
- ▷ *инициированная задача*
 - := [формулировка задачи, которая подлежит выполнению]
- ▷ *декларативная формулировка задачи*
- ▷ *процедурная формулировка задачи*
- ▷ *декларативно-процедурная формулировка задачи*
 - := [задача, в формулировке которой присутствуют как декларативные (целевые), так и процедурные аспекты]
- ▷ *задача, решаемая в памяти кибернетической системы*
 - ▷ *задача, решаемая в памяти индивидуальной кибернетической системы*
 - ▷ *задача, решаемая в общей памяти многоагентной системы*
 - := [информационная задача]
 - := [задача, направленная либо на генерацию или поиск информации, удовлетворяющей заданным требованиям, либо на некоторое преобразование заданной информации]
- ▷ *математическая задача*

Формулировка *задачи* может не содержать указания контекста (области решения) *задачи* (в этом случае областью решения *задачи* считается либо вся *база знаний*, либо ее согласованная часть), а также может не содержать либо описания исходной ситуации, либо описания целевой ситуации. Так, например, описание целевой ситуации для явно специфицированного противоречия, обнаруженног в *базе знаний*, не требуется.

декларативная формулировка задачи представляет собой описание исходной (начальной) ситуации, являющейся условием выполнения соответствующего *действия*, и целевой (конечной) ситуации, являющейся результатом выполнения этого *действия*, то есть описание ситуации (состояния), которое должно быть достигнуто в результате

выполнения планируемого действия. Другими словами, такая формулировка задачи включает явное или неявное описание:

- того, что дано, – исходные данные, условия выполнения специфицируемого действия;
- того, что требуется, – формулировка цели, результата выполнения указанного действия.

В случае ***процедурной формулировки задачи*** явно указывается характеристика действия, специфицируемого этой задачей, а именно, например, указывается:

- субъект или субъекты, выполняющие это действие;
- объекты, над которыми действие выполняется, – аргументы действия;
- инструменты, с помощью которых выполняется действие;
- момент и, возможно, дополнительные условия начала и завершения выполнения действия;
- явно указывается класс или классы, которым принадлежит каждое *действие* (включая поддействия).

При этом явно не уточняется, что должно быть результатом выполнения соответствующего действия.

Заметим, что, при необходимости, *процедурная формулировка задачи* может быть сведена к *декларативной формулировке задачи* путем трансляции на основе некоторого правила, например, определения класса действия через более общий класс.

Частными видами задач являются *вопрос* и *команда*.

вопрос

- \coloneqq [запрос]
- \subset задача, решаемая в памяти кибернетической системы
- \coloneqq [непроцедурная формулировка задачи на поиск (в текущем состоянии базы знаний) или на генерацию знания, удовлетворяющего заданным требованиям]
- \supset вопрос – что это такое
- \supset вопрос – почему
- \supset вопрос – зачем
- \supset вопрос – как
 - \coloneqq [каким способом]
 - \coloneqq [запрос метода (способа) решения заданного (указываемого) вида задач или класса задач либо плана решения конкретной указываемой задачи]
- \coloneqq [задача, направленная на удовлетворение информационной потребности некоторого субъекта-заказчика]

команда

- \coloneqq [инициированная задача]
- \coloneqq [спецификация инициированного действия]

Следует отметить, что, наряду в приведенной предельно общей классификацией задач, по сути отражающей классы задач с точки зрения их формулировки, должна существовать классификация задач с точки зрения их семантики, то есть с точки зрения сути специфицируемого действия. За основу такой классификация можно взять классификацию, представленную в работе **Fayans2020**.

В рамках же данной работы, как уже было сказано, наибольший интерес представляют задачи, решаемые в памяти.

Пункт 3.1.1.3. Понятие класса действий и класса задач

С точки зрения организации процесса решения задач более важными являются не столько понятия *действия* и *задачи*, сколько понятия *класса действий* и *класса задач*, поскольку именно для них разрабатываются соответствующие алгоритмы выполнения и способы решения.

Класс действий определим как максимальное множество аналогичных (похожих в определенном смысле) действий, для которого существует (но не обязательно известный в текущий момент) по крайней мере один **метод** (или средство), обеспечивающий выполнение любого действия из указанного множества действий.

класс действий

- \Leftarrow семейство подклассов*:
 - действие
- \coloneqq [множество однотипных действий]
- \supset класс элементарных действий
- \supset класс легковыполнимых сложных действий

Каждому выделяемому *классу действий* соответствует по крайней мере один общий для них *метод* выполнения этих *действий*. Это означает то, что речь идет о семантической "кластеризации" множества *действий*, т.е. о выделении *классов действий* по признаку семантической близости (сходства) *действий*, входящих в состав выделяемого *класса действий*. При этом прежде всего учитывается аналогичность (сходство) *исходных ситуаций* и *целевых ситуаций* рассматриваемых *действий*, т.е. аналогичность *задач*, решаемых в результате выполнения соответствующих *действий*. Поскольку одна и та же *задача* может быть решена в результате выполнения нескольких разных действий, принадлежащих разным *классам действий*, следует говорить не только о *классах действий* (множествах аналогичных действий), но и о *классах задач* (о множествах аналогичных задач), решаемых этими *действиями*. Так, например, на множестве *классов действий* заданы следующие *отношения*:

- *отношение*, каждая связка которого связывает два разных (непересекающихся) *класса действий*, осуществляющих решение одного и того же *класса задач*;
- *отношение*, каждая связка которого связывает два разных *класса действий*, осуществляющих решение разных *классов задач*, один из которых является надмножеством другого.

Кроме *класса действий* также выделяется понятие *класса элементарных действий*, то есть множества элементарных действий, указание принадлежности которому является необходимым и достаточным условием для выполнения этого действия. Множество всевозможных элементарных действий, выполняемых каждым субъектом, должно быть разбито на классы элементарных действий.

Принадлежность некоторого *класса действий* множеству *класса элементарных действий* фиксирует факт того, что при указании всех необходимых аргументов принадлежности *действия* данному классу достаточно для того, чтобы некоторый субъект мог приступить к выполнению этого действия.

При этом, даже если *класс действий* принадлежит множеству *класса элементарных действий*, не запрещается вводить более частные *классы действий*, для которых, например, заранее фиксируется один из аргументов.

Если конкретный *класс элементарных действий* является более частным по отношению к *действиям в sc-памяти*, то это говорит о наличии в текущей версии системы как минимум одного *sc-агента*, ориентированного на выполнение действий данного класса.

Кроме того, целесообразно также ввести понятие *класса легковыполнимых сложных действий*, то есть множества *сложных действий*, для которых известен и доступен по крайней мере один *метод*, интерпретация которого позволяет осуществить полную (окончательную, завершающуюся элементарными действиями) декомпозицию на поддействия каждого сложного действия из указанного выше множества.

Принадлежность некоторого *класса действий* множеству *класса легковыполнимых сложных действий* фиксирует факт того, что даже при указании всех необходимых аргументов принадлежности *действия* данному классу недостаточно для того, чтобы некоторый *субъект* приступил к выполнению этого действия, и требуются дополнительные уточнения.

В свою очередь, под *классом задач* будем понимать множество задач, для которого можно построить обобщенную формулировку задач, соответствующую всему этому множеству задач. Каждая *обобщенная формулировка задач* *соответствующего класса*, по сути, есть не что иное, как строгое логическое определение указанного класса задач.

класс задач

⇐ семейство подмножеств*:
задача

Конкретный класс действий может быть определен как минимум двумя способами.

класс действий

⇒ разбиение*:
{• *класс действий, однозначно задаваемый классом задач*
:= [класс действий, обеспечивающих решение соответствующего класса задач и использующих при этом любые, самые разные методы решения задач этого класса]
• *класс действий, однозначно задаваемый используемым методом решения задач*
}

Далее более подробно рассмотрим формальную трактовку понятия *метода*.

Пункт 3.1.1.4. Понятие метода

Под методом будем понимать описание того, как может быть выполнено любое или почти любое (с явным указанием исключений) действие, принадлежащее соответствующему классу действий.

Формально, *метод* – это спецификация решения задачи какого-то класса **Standard2021, Tuzov1986**. В состав спецификации каждого класса задач входит описание способа "привязки" метода к исходным данным конкретной задачи, решаемой с помощью этого метода.

метод

- := [метод решения соответствующего класса задач, обеспечивающий решение любой или большинства задач указанного класса]
- := [обобщенная спецификация выполнения действий соответствующего класса]
- := [обобщенная спецификация решения задач соответствующего класса]
- := [программа решения задач соответствующего класса, которая может быть как процедурной, так и декларативной (непроцедурной)]
- := [знание о том, как можно решать задачи соответствующего класса]
- С *знание*
- Є *вид знаний*
- := [способ]
- := [знание о том, как надо решать задачи соответствующего класса задач (множества эквивалентных (однотипных, похожих) задач)]
- := [метод (способ) решения некоторого (соответствующего) класса задач]
- := [информация (знание), достаточная для того, чтобы решить любую задачу, принадлежащую соответствующему классу задач, с помощью соответствующей модели решения задач]

В состав спецификации каждого *класса задач* входит описание способа "привязки" *метода* к исходным данным конкретной *задачи*, решаемой с помощью этого *метода*. Описание такого способа "привязки" включает в себя:

- набор переменных, которые входят как в состав *метода*, так и в состав *обобщенной формулировки задач соответствующего класса*, и значениями которых являются соответствующие элементы исходных данных каждой конкретной решаемой задачи;
- часть *обобщенной формулировки задач* того класса, которому соответствует рассматриваемый *метод*, являющихся описанием условия применения этого *метода*.

Сама рассматриваемая "привязка" *метода* к конкретной *задаче*, решаемой с помощью этого *метода*, осуществляется путем поиска в базе знаний такого фрагмента, который удовлетворяет условиям применения указанного *метода*. Одним из результатов такого поиска и является установление соответствия между указанными выше переменными используемого *метода* и значениями этих переменных в рамках конкретной решаемой *задачи*.

Другим вариантом установления рассматриваемого соответствия является явное обращение (вызов, call) соответствующего *метода* (программы) с явной передачей соответствующих параметров. Но такое не всегда возможно, т.к. при выполнении процесса решения конкретной *задачи* на основе декларативной спецификации выполнения этого действия нет возможности установить:

- когда необходимо инициировать вызов (использование) требуемого *метода*;
- какой конкретно *метод* необходимо использовать;
- какие параметры, соответствующие конкретной инициируемой *задаче*, необходимо передать для "привязки" используемого *метода* к этой *задаче*.

Процесс "привязки" *метода* решения задач к конкретной *задаче*, решаемой с помощью этого *метода*, можно также представить как процесс, состоящий из следующих этапов:

- построение копии используемого *метода*;
- склеивание основных (ключевых) переменных используемого *метода* с основными параметрами конкретной решаемой *задачи*.

В результате этого, на основе рассматриваемого *метода*, используемого в качестве образца (шаблона), строится спецификация процесса решения конкретной задачи – процедурная спецификация (*план*) или декларативная.

Заметим, что *методы* могут использоваться даже при построении *планов* решения конкретных *задач* в случае, когда возникает необходимость многократного повторения неких цепочек *действий* при априори неизвестном количестве таких повторений. Речь идет о различного вида *циклах*, которые являются простейшим видом процедурных *методов* решения задач, многократно используемых (повторяемых) при реализации *планов* решения некоторых *задач*.

Очевидно также, что одному *классу действий* может соответствовать несколько *методов*.

Термин “метод”, таким образом, будем считать синонимичным термину “программа” в обобщенном понимании этого термина.

метод

- := [программа]
- := [программа выполнения действий некоторого класса]
- ⊇ *процедурная программа*
 - := [обобщенный план]
 - := [обобщенный план выполнения некоторого класса действий]
 - := [обобщенный план решения некоторого класса задач]
 - := [обобщенная спецификация декомпозиции любого действия, принадлежащего заданному классу действий]
- ⊆ алгоритм

Рассмотрим более подробно понятие процедурной программы (процедурного метода). Каждая *процедурная программа* представляет собой обобщенный план выполнения *действий*, принадлежащих некоторому классу, то есть *семантическую окрестность*; *ключевым sc-элементом*¹ является *класс действий*, для элементов которого дополнительно детализируется процесс их выполнения.

Входным параметрам *процедурной программы* в традиционном понимании соответствуют аргументы, соответствующие каждому *действию из класса действий*, описываемого данной *процедурной программой*. При генерации на основе данной программы *плана выполнения конкретного действия* из данного класса эти аргументы принимают конкретные значения.

Каждая *процедурная программа* представляет собой систему описанных действий с дополнительным указанием для действия:

- либо *последовательности выполнения действий** (передачи инициирования), когда условием выполнения (инициирования) действий является завершение выполнения одного из указанных или всех указанных действий;
- либо события в базе знаний или внешней среде, являющегося условием его инициирования;
- либо ситуации в базе знаний или внешней среде, являющейся условием его инициирования.

Понятие метода позволяет определить отношение *эквивалентность задач** на множестве задач. Задачи являются эквивалентными в том и только в том случае, если они могут быть решены путем интерпретации одного и того же *метода* (способа), хранимого в памяти кибернетической системы. Некоторые *задачи* могут быть решены разными *методами*, один из которых, например, является обобщением другого. Таким образом, на множестве методов можно также задать ряд отношений.

Отметим, что понятие *метода* фактически позволяет локализовать область решения задач соответствующего класса, то есть ограничить множество знаний, которых достаточно для решения задач данного класса определенным способом. Это, в свою очередь, позволяет повысить эффективность работы системы в целом, исключая число лишних действий.

отношение, заданное на множестве методов

- ⊇ подметод*
- := [подпрограмма*]
- := [быть методом, использование которого (обращение к которому) предполагается при реализации заданного метода*]
- ↔ следует отличать*:
 - частный метод*
- := [быть методом, обеспечивающим решение класса задач, который является подклассом задач, решаемых с помощью заданного метода*]

В литературе, посвященной построению решателей задач, встречается понятие *стратегии решения задач*. Определим его как метаметод решения задач, обеспечивающий либо поиск одного релевантного известного метода, либо синтез целенаправленной последовательности акций применения в общем случае различных известных методов.

стратегия решения задач

- ⊆ метод

Можно говорить об универсальном метаметоде (универсальной стратегии) решения задач, объясняющем всевозможные частные стратегии. В частности, можно говорить о нескольких глобальных *стратегиях решения информационных задач* в базах знаний. Пусть в базе знаний появился знак инициированного действия с формулировкой, соответствующей информационной цели, т.е. цели, направленной только на изменение состояния базы знаний.

И пусть текущее состояние базы знаний не содержит контекст (исходные данные), достаточный для достижения указанной выше цели, т.е. такой контекст, для которого в доступном пакете (наборе) методов (программ) имеется метод (программа), использование которого позволяет достичь указанной выше цели. Для достижения такой цели, контекст (исходные данные) которой недостаточен, существует три подхода (три стратегии):

- декомпозиция (сведение изначальной цели к иерархической системе и/или подцелей (и/или подзадач) на основе анализа текущего состояния базы знаний и анализа того, чего не хватает в базе знаний для использования того или иного метода).

При этом наибольшее внимание уделяется методам, для создания условий использования которых требуется меньше усилий. В конечном счете мы должны дойти (на самом нижнем уровне иерархии) до подцелей, контекст которых достаточен для применения одного из имеющихся методов (программ) решения задач;

- генерация новых знаний в семантической окрестности формулировки изначальной цели с помощью любых доступных методов в надежде получить такое состояние базы знаний, которое будет содержать нужный контекст (достаточные исходные данные) для достижения изначальной цели с помощью какого-либо имеющегося метода решения задач;
- комбинация первого и второго подходов.

Аналогичные стратегии существуют и для поиска пути решения задач, решаемых во внешней среде.

Пункт 3.1.1.5. Спецификация методов и понятие навыка

Каждый конкретный *метод* рассматривается нами не только как важный вид спецификации соответствующего класса задач, но также и как объект, который и сам нуждается в спецификации, обеспечивающей непосредственное применение этого метода. Другими словами, метод является не только спецификацией (спецификацией соответствующего класса задач), но и объектом спецификации. Важнейшим видом такой спецификации является указание *операционной семантики метода*.

операционная семантика метода*

Спецификация*

:= [семейство методов, обеспечивающих интерпретацию заданного метода*]

:= [формальное описание интерпретатора заданного метода*]

⇒ второй домен*:

операционная семантика метода

полное представление операционной семантики метода

:= [представление *операционной семантики метода*, доведенное (детализированное) до уровня всех спецификаций элементарных действий, выполняемых в процессе интерпретации соответствующего метода]

декларативная семантика метода*

Спецификация*

:= [описание системы понятий, которые используются в рамках данного метода*]

Отношение *декларативная семантика метода** связывает *метод* и формальное описание системы понятий (фрагмент логической онтологии соответствующей *предметной области*), которые используются (упоминаются) в рамках данного метода. Это необходимо для того, чтобы гарантировать однозначность трактовки одного и того же понятия в рамках метода и остальной части базы знаний, что особенно актуально при заимствовании метода из библиотеки многократно используемых компонентов решателей задач. Важно отметить, что тот факт, что какие-либо понятия используются в рамках метода, не означает, что формальная запись их определений является частью данного метода. Например, в состав метода, позволяющего решать задачи на вычисление площади треугольника, будут входить различные формулы расчета площади треугольника, но не будут входить сами определения понятий “площадь”, “треугольник” и т.д., поскольку при наличии априори верных формул эти определения не будут непосредственно использоваться в процессе решения задачи. В то же время формальные определения указанных понятий будут входить в состав декларативной семантики данного метода.

Объединение *метода* и его операционной семантики, то есть информации о том, каким образом должен интерпретироваться данный *метод*, будем называть **навыком**.

навык

:= [умение]

:= [объединение *метода* с его исчерпывающей спецификацией – полным представлением *операционной семантики метода*]

- := [метод, интерпретация (выполнение, использование) которого полностью может быть осуществлено данной кибернетической системой, в памяти которой хранится указанный метод]
- := [метод, который данная кибернетическая система умеет (может) применять]
- := [метод + метод его интерпретации]
- := [умение решать соответствующий класс эквивалентных задач]
- := [метод плюс его операционная семантика, описывающая то, как интерпретируется (выполняется, реализуется) этот метод, и являющаяся одновременно операционной семантикой соответствующей модели решения задач]
- ⇒ разбиение*:
 - {• активный навык
 - := [самоинициирующийся навык]
 - пассивный навык

Таким образом, понятие *навыка* является важнейшим понятием с точки зрения построения решателей задач, поскольку объединяет в себе не только декларативную часть описания способа решения класса задач, но и операционную.

Навыки могут быть *пассивными навыками*, то есть такими *навыками*, применение которых должно явно инициироваться каким-либо агентом, либо *активными навыками*, которые инициируются самостоятельно при возникновении соответствующей ситуации в базе знаний. Для этого в состав *активного навыка*, помимо *метода* и его операционной семантики, включается также *sc-агент*, который реагирует на появление соответствующей ситуации в базе знаний и инициирует интерпретацию *метода* данного *навыка*. Такое разделение позволяет реализовывать и комбинировать различные подходы к решению задач, в частности, *пассивные навыки* можно рассматривать в качестве способа реализации концепции интеллектуального пакета программ.

Пункт 3.1.1.6. Понятие класса методов и языка представления методов

Как действия и задачи, методы могут быть классифицированы по различным классам. Будем называть **классом методов** множество методов, для которых можно унифицировать представление (спецификацию) этих методов.

класс методов

- ⇐ семейство подклассов*:
 - метод
- := [множество методов, для которых можно унифицировать представление (спецификацию) этих методов]
- := [множество всевозможных методов решения задач, имеющих общий язык представления этих методов]
- := [множество методов, для которых задан язык представления этих методов]
- Э процедурный метод решения задач
 - ▷ алгоритмический метод решения задач
- Э непроцедурный метод решения задач
 - ▷ логический метод решения задач
 - ▷ продукционный метод решения задач
 - ▷ функциональный метод решения задач
 - ▷ продукционный метод решения задач
 - ▷ функциональный метод решения задач
 - ▷ искусственная нейронная сеть
 - := [класс методов решения задач на основе искусственных нейронных сетей]
 - ▷ генетический "алгоритм"
- := [множество методов, основанных на общей онтологии]
- := [множество методов, представленных на одинаковом языке]
- := [множество методов решений задач, которому соответствует специальный язык (например, sc-язык), обеспечивающий представление методов из этого множества]
- := [множество методов, которому ставится в соответствие отдельная модель решения задач]

Каждому конкретному *классу методов* взаимно однозначно соответствует *язык представления методов*, принадлежащий этому (специфицируемому) *классу методов*. Таким образом, спецификация каждого *класса методов* сводится к спецификации соответствующего языка *представления методов*, т.е. к описанию его синтаксической, денотационной и операционной семантики. Примерами *языков представления методов* являются все языки *программирования*, которые, в основном, относятся к подклассу *языков представления методов* – к *языкам представления методов обработки информации*. Но сейчас все большую актуальность приобретает необходимость создания эффективных формальных языков представления методов выполнения действий во внешней среде ки-

бернетических систем. Без этого комплексная автоматизация **Pospelov2021**, в частности, в промышленной сфере, невозможна.

Таких специализированных языков может быть выделено целое множество, каждому из которых будет соответствовать своя модель решения задач (т.е. свой интерпретатор).

Под *языком представления методов* будем подразумевать формальный язык, (1) знаковыми конструкциями которого являются соответствующие методы, для которых существуют общие правила построения и (2) общие правила соотнесения с теми сущностями и связями между ними, которые описываются этими методами.

язык представления методов

- := [язык методов]
- := [язык представления методов, соответствующих определенному классу методов]
- := [язык (например, sc-язык) представления методов соответствующего класса методов]
- С языком
- С **формальный язык**
- := [формальный язык, (1) знаковыми конструкциями которого являются соответствующие методы, для которых существуют общие правила построения и (2) общие правила соотнесения с теми сущностями и связями между ними, которые описываются этими методами]
- := [язык программирования]
- ▷ **язык представления методов обработки информации**
 - := [язык программирования внутренних действий кибернетической системы, выполняемых в их памяти]
 - := [язык представления методов решения задач в памяти кибернетических систем]
- ▷ **язык представления методов решения задач во внешней среде кибернетических систем**
 - := [язык программирования внешних действий кибернетических систем]

Метод принадлежит языку представления методов, если он является синтаксически корректным, синтаксически целостным, семантически корректным и семантически целостным методом заданного языка представления методов (!).

отношение, заданное на множестве языков представления методов[^]

- := [отношение, область определения которого включает в себя множество всевозможных языков представления методов]
- Э **метод заданного языка представления методов***
- Э **синтаксически корректный метод для заданного языка представления методов***
 - := [метод, не содержащий синтаксических ошибок для заданного языка представления методов*]
 - С **синтаксически корректная знаковая конструкция для заданного языка***
- Э **синтаксически целостный метод для заданного языка представления методов***
 - С **синтаксически целостная знаковая конструкция для заданного языка***
- Э **семантически корректный метод для заданного языка представления методов***
 - := [метод, не содержащий семантических ошибок для заданного языка представления методов*]
 - С **семантически корректная знаковая конструкция для заданного языка***
- Э **семантически целостный метод для заданного языка представления методов***
 - С **семантически целостная знаковая конструкция для заданного языка***
 - := [метод заданного языка представления методов, содержащий достаточную информацию для установления его истинности*]

метод заданного языка представления методов*

- := [метод, принадлежащий заданному языку программирования*]
- С **текст заданного языка***
- ⇒ **второй домен*:**
 - метод**
 - ⇐ **объединение*:**
 - {• {}
 - ⇐ **объединение*:**
 - {• **синтаксически корректный метод для заданного языка представления методов***
 - **синтаксически целостный метод для заданного языка представления методов***
 - {}
 - ⇐ **объединение*:**
 - {• **семантически корректный метод для заданного языка представления методов***
 - **синтаксически целостный метод для заданного языка представления методов***

}

}

Пункт 3.1.1.7. Общая классификация языков представления методов

Языки представления методов в современном информационном обществе различают по их парадигмам: *процедурные, функциональные, логические, объектно-ориентированные* и т. д. Таким, например, в методах процедурного я.п.м. решение задачи компьютером формируется в виде последовательности операторов, в методах функционального я.п.м. — указанием других методов. В логическом я.п.м. применяются высказывания, а в объектно-ориентированном — объекты.

язык представления методов

- ▷ **язык представления методов общего назначения**
:= [язык программирования общего назначения]
- ▷ **предметно-ориентированный язык представления методов**
:= [предметно-ориентированный язык программирования]
- ⇒ **разбиение*:**
парадигма языка представления методов[^]
= {• процедурный язык представления методов
• непроцедурный язык представления методов
}

Процедурные языки представления методов задают вычисления как последовательность операторов (команд). Они ориентированы на компьютеры с архитектурой фон Неймана. Основные понятия процедурных я.п.м. тесно связаны с компонентами компьютера:

- переменными различных типов, которые моделируют ячейки памяти компьютера;
- операторами присваивания, которые моделируют пересылки данных между участками памяти;
- повторений действий в форме итерации, которые моделируют хранение информации в смежных ячейках памяти;
- и другое.

процедурный язык представления методов

- := [императивный язык представления методов]
- ▷ **структурный язык представления методов**
Э *пример'*:
• Fortran
• C
• Pascal
- ▷ **объектно-ориентированный язык представления методов**
Э *пример'*:
• Java
• Smalltalk
• HTML
- ▷ **аспектический язык представления методов**
- ▷ **скриптовый язык представления методов**
:= [склеивающий язык представления методов]

Непроцедурные языки представления методов, в отличие от процедурных, задают вычисления как последовательность связанных между собой объектов. Основные понятия непроцедурных я.п.м. обычно не связаны с компонентами компьютера.

непроцедурный язык представления методов

- := [декларативный язык представления методов]
- ▷ **логический язык представления методов**
Э *пример'*:
• Prolog
- ▷ **продукционный язык представления методов**
- ▷ **функциональный язык представления методов**
:= [аппликативный язык представления методов]

- Ǝ *пример*':
 - LISP

Пункт 3.1.1.8. Понятие модели решения задач

По аналогии с понятием стратегии решения задач введем понятие ***модели решения задач***, которое будем трактовать как метаметод интерпретации соответствующего класса методов.

модель решения задач

- С *метод*
- := [метаметод]
- := [абстрактная машина интерпретации соответствующего класса методов]
- := [иерархическая система "микропрограмм", обеспечивающих интерпретацию соответствующего класса методов]
- ▷ алгоритмическая модель решения задач
- ▷ процедурная параллельная синхронная модель решения задач
- ▷ процедурная параллельная асинхронная модель решения задач
- ▷ продукционная модель решения задач
- ▷ функциональная модель решения задач
- ▷ логическая модель решения задач
 - ▷ четкая логическая модель решения задач
 - ▷ нечеткая логическая модель решения задач
- ▷ "нейросетевая" модель решения задач
- ▷ "генетическая" модель решения задач

Каждая ***модель решения задач*** задается:

- соответствующим классом методов решения задач, т.е. языком представления методов этого класса;
- предметной областью этого класса методов;
- онтологией этого класса методов (т.е. денотационной семантикой языка представления этих методов);
- операционной семантикой указанного класса методов.

Важно отметить, что для интерпретации всех моделей решения задач может быть использован агентно-ориентированный подход, рассмотренный в работе [Shunkevich.D.V.AgentOMMTCPSDIS-2018ст.](#)

спецификация*

- ▷ ***модель решения задач****
 - = сужение отношения по первому домену (*спецификация**; класс методов)*
 - := [спецификация класса методов*]
 - := [спецификация языка представления методов*]
 - ⇒ разбиение*:
 - {• **синтаксис языка представления методов соответствующего класса***
 - **денотационная семантика языка представления методов соответствующего класса***
 - **операционная семантика языка представления методов соответствующего класса***

Модель решения задач ставит в соответствие некоторому классу методов синтаксис, денотационную и операционную семантику языка представления методов соответствующего класса.

денотационная семантика языка представления методов соответствующего класса

- := [онтология соответствующего класса методов]
- := [денотационная семантика соответствующего класса методов]
- := [денотационная семантика языка (sc-языка), обеспечивающего представление методов соответствующего класса]
- := [денотационная семантика соответствующей модели решения задач]
- ⇒ *примечание**:
 - [Если речь идет о языке, обеспечивающем внутреннее представление методов соответствующего класса в ostis-системе, то синтаксис этого языка совпадает с синтаксисом sc-кода]
- С онтология

операционная семантика языка представления методов соответствующего класса

- := [метод интерпретации соответствующего класса методов]
- := [семейство агентов, обеспечивающих интерпретацию (использование) любого метода, принадлежащего соответствующему классу методов]
- := [операционная семантика соответствующей модели решения задач]

Поскольку каждому *методу* соответствует *обобщенная формулировка задач*, решаемых с помощью этого *метода*, то каждому *классу методов* должен соответствовать не только определенный *язык представления методов*, принадлежащих указанному *классу методов*, но и определенный *язык представления обобщенных формулировок задач для различных классов задач*, решаемых с помощью *методов*, принадлежащих указанному *классу методов*.

§ 3.1.2. Локальные предметные области и онтологии действий

Заключение к Главе 3.1.

Дальнейшее развитие представленных в данной главе онтологий предполагает формализацию классификации задач, решаемых интеллектуальными системами, унификацию описания задач и классов задач, описания целей, хода и результата решения задачи, методов решения задач, связей между классами задач и методами решения задач данного класса. Это позволит обеспечить возможность глубокой интеграции всевозможных моделей решения задач различных классов и возможность облегчить процесс интеграции новых моделей решения задач в интеллектуальную систему, а также положит основу для решения ряда проблем в области разработки гибридных решателей задач, рассмотренных в Главе 3.2. *Агентно-ориентированные модели гибридных решателей задач ostis-систем*.

Глава 3.2.

Агентно-ориентированные модели гибридных решателей задач ostis-систем

⇒ *автор**:

- Шункевич Д.В.

⇒ *аннотация**:

[В главе сформулированы актуальные проблемы текущего состояния технологий разработки гибридных решателей задач, предложен подход к их решению на основе Технологии OSTIS. Сформулированы принципы построения решателя задач как иерархической системы навыков, основанной на многоагентном подходе, приведены онтологии агентов и выполняемых ими действий. Сформулированы принципы синхронизации деятельности агентов, а также разработана онтология базового языка программирования для реализации программ агентов и модель интерпретатора такого языка.]

⇒ *подраздел**:

- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению
- § 3.2.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой
- § 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты
- § 3.2.4. Принципы синхронизации деятельности sc-агентов
- § 3.2.5. Базовый язык программирования ostis-систем
- § 3.2.6. Решатели задач ostis-систем
- § 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

⇒ *ключевое понятие**:

- действие в sc-памяти
- действие в sc-памяти, инициируемое вопросом
- действие редактирования базы знаний
- задача, решаемая в sc-памяти
- класс логически атомарных действий
- sc-агент
- абстрактный sc-агент
- атомарный абстрактный sc-агент
- неатомарный абстрактный sc-агент
- абстрактный sc-агент, реализуемый на Языке SCP
- абстрактный sc-агент, не реализуемый на Языке SCP
- блокировка*
- тип блокировки
- планируемая блокировка*
- приоритет блокировки*
- удаляемые sc-элементы*
- транзакция в sc-памяти
- scp-оператор
- параметр scp-программы'
- scp-операнд'
- решатель задач ostis-системы
- машина обработки знаний

⇒ *ключевой знак**:

- Язык SCP
- Абстрактная scp-машина

⇒ *библиографическая ссылка**:

- Колесников А.В. ГибридИСТИР-2001кн
- Пратт Т. ЯзыкиПРИР-2002кн
- Гладков Л.А.. ГенетАУП-2006кн
- Емельянов В.В. Теори иПЭМ-2003кн
- Беркинблит М.Б. НейроСЭУП-1993кн

- Головко В.А.*НейроCOОиПУП-2001кн*
- Горбань А.Н.*НейроCнПК-1996кн*
- Вагин В.Н.*Досто иПВвИС-2008кн*
- Куллик Б.А.*ЛогикЕР-2001кн*
- Пойа Д.Матем иПР-1975кн
- Батыршин И.З.*ОсновыИииЛО-2001кн*
- Деменков Н.П.*НечетУвТСУП-2005кн*
- Поспелов Д.А.*МоделРОАМА-1989кн*
- Reiter R.*LogicDR-1980см*
- Еремеев А.П.*ПострРФиБТЛвСПРвУН-1997см*
- Кахро М.В..*ИнстрСПЕСЭВМ-1988кн*
- Ефимов Е.И..*РешатИЗ-1982кн*
- Раговский А.П.*ИнтелМСДВиОСО-2011см*
- Подколзин А.С.*КомпьюМЛПАРиЯРЗ-2008кн*
- Курбатов С.С..*ПрогрОдАРЗпП-2016см*
- Владимиров А.Н..*ПрогрКУПРАОЛВсЛВСнОМСП-2010см*
- Попов Э.В..*ИскусСИССОиЭС-1990кн*
- Jackson P.*IntroES-1998кн*
- WorldWWC-2023el
- RDFCAS-2023el
- OWLWOLDO-2023el
- SPARQLO-2023el
- Neo4j-2023el
- OWLI-2023el
- Грибова В.В..*БазовТРИСнОПРБЗиРЗ-2015см*
- Грибова В.В..*ПроектIACРaaSKИСОOB-2011см*
- Филипов А.А..*ЕдинаOPIАD-2016см*
- Борисов А.Н.*ПострИСнЗcПИК-2014см*
- Dutta S.*KnowlPaAAI-1993bk*
- Pau L.F.*KnowlBP-1990art*
- Wooldridge M.*Intro tMS-2009bk*
- Weyns D..*Envir aaFCAiMAS-2007art*
- FIPAACL MSS-2023el
- Finin T.*KQMLaaACL-1994art*
- KnowledgeIFS-2023el
- Harting R.L..*UsingMAtRwMO-2008art*
- Sims M..*AutoODfMAS-2008art*
- Excelente-Toledo C.B..*DynamSoCM-2004art*
- Nagendra Prasad M.V..*LearnSSCiCMAS-1999art*
- Vasconcelos W.W..*NormaCRiMAS-2009art*
- Rumbell T..*Emoti iAACAOMaF-2012art*
- Городецкий В.И..*БазовOKПIAueP-2015см*
- Bordini R.H..*ProgrMASiASUJ-2007bk*
- Castillo O..*RecenAoHIS-2014bk*
- Eve aWBAP-2023el
- GAMAP-2023el
- GOAL-2023el
- Everts R..*ImplеIMASUJACK-2004art*
- JAVAADF-2023el
- Boissier O..*MultiAOPwJCM-2013art*
- Omicini A..*Coord fIAD-1999art*
- Jagannathan V..*BlackAaA-1989bk*
- Поспелов Д.А.*СитуаУП-1986кн*
- Dijkstra E.W.*CoopeSP-2002bk*
- Hoare C.A.R.*CommuSP-1983art*
- Chatterjee B..*nBlockDUGwWCAB-2022art*
- Нариньян А.С.*НЕФакKB-2004см*
- Cao O.*IndepBUaUtBIA-2010art*
- Cao O..*BehavIaNP-2014art*
- Pavel M..*BehavIaCMiSoPHMaC-2015art*
- Альтишуллер Г.С..*НайтиИ:ВвТРИЗ-2010кн*
- Щедровицкий Г.П.*СхемаMCCCCиC-1995кн*

§ 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

- *Сапатый П.С. ЯВОЛНАкОНСдБЗ-1986см*
- *Moldovan D.I. SNAPaVLSIAfAIP-1985bk*
- *Летичевский А.А..ИнсерП-2003см*
- *Летичевский А.А..ИнсерМ-2012см*

Введение в Главу 3.2.

Одним из ключевых компонентов интеллектуальной системы, обеспечивающим возможность решать широкий круг задач, является решатель задач. Их особенностью по сравнению с другими современными программными системами является необходимость решать задачи в условиях, когда необходимые сведения не локализованы явно в базе знаний интеллектуальной системы и должны быть найдены в процессе решения задачи на основании каких-либо критериев.

Говоря другими словами, если в традиционных системах при решении задачи всегда подразумевается, что есть некоторые локализованные исходные данные ("дано") и некоторое описание желаемого результата ("что требуется"), то в интеллектуальной системе в качестве исходных данных при решении большого числа задач выступает вся имеющаяся на текущий момент в системе информация, то есть вся база знаний. Кроме того, при невозможности решения задачи в текущем состоянии базы знаний интеллектуальная система должна иметь возможность понять, чего именно не хватает для продолжения процесса решения и попытаться добыть недостающие сведения во внешней среде (например, запросить у пользователя).

К настоящему времени в рамках различных направлений искусственного интеллекта разработано большое количество различных *моделей решения задач*, каждая из которых позволяет решать задачи определенного класса. Расширение областей применения интеллектуальных систем требует от них возможности решать так называемые *комплексные задачи*, решение каждой из которых требует комбинирования нескольких моделей решения задач, при этом априори неизвестно, в каком порядке и сколько раз будет применяться так или иная модель. Решатели задач, в рамках которых комбинируются несколько моделей решения задач, получили название *гибридных решателей задач*, а интеллектуальные системы, в рамках которых комбинируются различные виды знаний и различные модели решения задач – *гибридных интеллектуальных систем* (см. *Колесников А.В. ГибриИСТиTP-2001кн*).

Повышение эффективности разработки и эксплуатации гибридных интеллектуальных систем требует унификации моделей представления различных видов знаний и моделей обработки знаний, которая бы позволила легко интегрировать на ее основе компоненты, соответствующие различным моделям решения задач.

§ 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

⇒ подраздел*:

- *Пункт 3.2.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*
- *Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Пункт 3.2.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач

Существующее многообразие подходов к решению задач в компьютерных системах можно разделить на два класса:

- **решение задач с использованием хранимых программ.** В данном случае предполагается, что в системе заранее присутствует программа решения задачи заданного класса и решение сводится к поиску такой программы и интерпретации ее на заданных входных данных. К системам, ориентированным на такой подход к решению задач, относятся в том числе системы, использующие:
 - программы, написанные на языках программирования, относящихся как к императивной, так и к дейкларативной парадигме, в том числе логических и функциональных (см. *Пратт Т. ЯзыкиПриР-2002кн*);
 - реализации генетических алгоритмов (см. *Гладков Л.А..ГенетАУП-2006кн*, *Емельянов В.В..Теори иПЭМ-2003кн*);

- нейросетевые модели обработки знаний (см. *Беркинблит М.Б.НейроСЭУП-1993кн, Головко В.А.Нейро-СООиПУП-2001кн, Горбань А.Н..НейроСнПК-1996кн*).
- Следует отметить, что даже в случае использования хранимой программы решение задачи далеко не всегда тривиально, поскольку, во-первых, требуется найти такую хранимую программу на основе некоторой спецификации, во-вторых, обеспечить ее интерпретацию.
- **решение задач в условиях, когда программа решения не известна.** В этом случае предполагается, что в системе необязательно присутствует готовая программа решения для класса задач, которому принадлежит некоторая сформулированная задача, подлежащая решению. В связи с этим необходимо применять дополнительные методы поиска путей решения задачи, не рассчитанные на какой-либо узкий класс задач (например, разбиение задачи на подзадачи, методы поиска решений в глубину и ширину, метод случайного поиска решения и метод проб и ошибок, метод деления пополам и др.), а также различные модели логического вывода: классические дедуктивные (см. *Вагин В.Н..Досто иПВвИС-2008кн*), индуктивные (см. *Кулик Б.А.ЛогикЕР-2001кн, Пойа Д.Матем иПР-1975кн*), абдуктивные (см. *Вагин В.Н..Досто иПВвИС-2008кн*); модели, основанные на нечетких логиках (см. *Батыршин И.З.ОсновоОНииЛО-2001кн, Деменков Н.П.НечетУвТСУП-2005кн, Поспелов Д.А.МоделРОАМА-1989кн*), логике умолчаний (см. *Reiter R.LogicDR-1980ст*), темпоральной логике (см. *Еремеев А.П.ПострРФнБТЛвСПРвУН-1997ст*), и многие другие.

Подробный обзор решателей задач, разработанных в период до 1982 г., таких как GPS, STRIPS, QA3, ПРИЗ (см. *Кахро М.В.ИнстрСПЕСЭВМ-1988кн*), ППР приведен в книге *Ефимов Е.И..РешатИЗ-1982кн*. Среди современных работ, исследующих вопросы применения моделей решения задач, не ориентированных на конкретную предметную область, можно выделить *Раговский А.П.ИнтелМСДВнОСО-2011ст*. Среди наиболее заметных представителей класса интеллектуальных решателей задач, разработанных в более поздний период, можно отметить Компьютерный решатель математических задач (см. *Подколзин А.С.КомпьюМЛПАРиЯРЗ-2008кн*), Решатель задач по планиметрии НИЦ ЭВТ (см. *Курбатов С.С..ПрогрОдАРЗпП-2016ст*), Программный комплекс “УДАВ” (см. *Владимиров А.Н..ПрогрКУПРАОЛВсЛВСнОМСП-2010ст*).

Отдельного внимания заслуживают популярные в настоящее время системы компьютерной алгебры, такие как Wolfram Mathematica, Maple, MathCAD и другие. Указанные программные комплексы обладают мощной функциональностью как для проведения различного рода вычислений и экспериментов, так и для построения на их основе систем различного назначения, например обучающих. Более подробно возможности применения систем данного семейства для решения задач в рамках Экосистемы OSTIS рассмотрены в ?? ??.

Однако при всем многообразии решаемых рассмотренными системами задач множество классов задач ограничивается имеющимся в системе набором жестко заданных приемов и алгоритмов решения задач, явно используемых при решении той или иной задачи. В то же время построение сложных систем, например, систем комплексной автоматизации, невозможно без обеспечения согласованного использования различных видов знаний и моделей решения задач в рамках одной системы при решении одной и той же *комплексной задачи*. Кроме того, становится актуальной задача поддержки такой системы в состоянии, соответствующем текущему уровню развития технологий, дополнения ее более совершенными моделями и методами решения задач. При этом очевидно, что подобная реконфигурация системы должна осуществляться непосредственно в процессе эксплуатации системы, а не требовать каждый раз, например, полной остановки всего производства или отдельных его частей.

Сказанное выше позволяет сформулировать требования *гибридному решателю задач*:

- в каждый момент времени решатель должен обеспечивать решение задач из оговоренного класса за оговоренное время, при этом результат решения задачи должен удовлетворять некоторым известным требованиям. Другими словами, как и в случае современных компьютерных систем, корректность результатов решения задач на этапе разработки системы должна верифицироваться специальными методами, в том числе для этого могут быть использованы такие современные подходы, как unit-тестирование, тестирование методом «черного ящика» и др. Более детально рассмотрим положения, уточняющие сформулированное требование:
 - для явно сформулированных задач система всегда должна давать какой-либо ответ за оговоренное время, при этом ответ может быть отрицательным (система не смогла решить поставленную задачу), возможно, с объяснением причин, по которым решение в текущий момент оказалось невозможным. Одним из факторов безуспешности решения является выход за рамки установленного промежутка времени;
 - если явно сформулированная задача решена, то все информационные процессы, направленные на ее решение, должны быть уничтожены. Особенно актуальным данное требование становится в ситуации, когда для решения одной и той же задачи параллельно используются сразу несколько подходов и заранее неизвестно, какой из них приведет к результату раньше других;
 - после решения задачи вся временная информация, сгенерированная в процессе решения этой задачи и имеющая ценность только в контексте решения указанной задачи, должна быть удалена из памяти;
- **гибридный решатель** должен обеспечивать возможность **согласованного использования различных моделей решения задач** при решении одной и той же *комплексной задачи* в случае необходимости;
- решатель должен быть легко **модифицируемым**, т. е. трудоемкость внесения изменений в уже разработанный решатель должна быть минимальна. Путями повышения модифицируемости решателя являются обеспечение

локальности вносимых изменений, в том числе – за счет стратификации решателя на независимые уровни и обеспечение максимальной независимости компонентов решателя друг от друга, а также наличие готовых компонентов, которые могут быть встроены в решатель при необходимости. При этом внесение изменений должно осуществляться непосредственно в процессе эксплуатации системы;

- для того чтобы интеллектуальная система имела возможность анализировать и оптимизировать имеющийся решатель задач, интегрировать в его состав новые компоненты (в том числе самостоятельно), оценивать важность тех или иных компонентов и применимость их для решения той или иной задачи, спецификация решателя должна быть описана языком, понятным системе, например, при помощи тех же средств, что и обрабатываемые знания. Другими словами, интеллектуальная система и, соответственно, решатель должны обладать *рефлексивностью*.

Несмотря на то что в настоящее время существует большое число моделей решения задач, многие из которых реализованы и успешно используются на практике в различных системах, остается актуальной проблема низкой согласованности принципов, лежащих в основе реализации таких моделей, и отсутствия единой унифицированной основы для реализации и интеграции различных моделей, что приводит к тому, что:

- затруднена возможность одновременного использования различных моделей решения задач в рамках одной системы при решении одной и той же комплексной задачи; практически невозможно комбинировать различные модели с целью решения задачи, для которой априори отсутствует алгоритм ее решения;
- практически невозможно использовать технические решения, реализованные в одной системе, в других системах, т. е. возможности использования компонентного подхода при построении решателей задач сильно ограничены. Как следствие, велико количество дублирований аналогичных решений в разных системах;
- фактически отсутствуют комплексные методики и средства построения решателей задач, которые бы обеспечивали возможность проектирования, реализации и отладки решателей различного вида.

Следствиями указанных проблем являются:

- высокая трудоемкость разработки каждого решателя, увеличение сроков их разработки, а значит, и увеличение затрат на разработку и поддержку соответствующих интеллектуальных систем;
- высокая трудоемкость внесения изменений в уже разработанные решатели, т. е. отсутствует или сильно затруднена возможность дополнения уже разработанного решателя новыми компонентами и внесения изменений в уже существующие компоненты в процессе эксплуатации системы. Таким образом, высока трудоемкость поддержки разработанных решателей;
- высокий уровень профессиональных требований к разработчикам решателей задач, что обусловлено, в частности:
 - высокой сложностью существующих формализмов в области решения задач, рассчитанных на их интерпретацию компьютерной системой, а не человеком;
 - отсутствием возможности рассматривать разрабатываемые решатели на разных уровнях детализации, выделения на каждом уровне достаточно независимых компонентов, что затрудняет процесс проектирования, тестирования и отладки таких решателей, а также снижает эффективность попыток объединения разработчиков решателей в коллективы по причине увеличения накладных расходов на согласование их деятельности;
 - низким уровнем информационной поддержки разработчиков и автоматизации их деятельности.

Для решения перечисленных проблем необходимо разработать комплекс моделей, методики и средств разработки гибридных решателей задач, удовлетворяющих перечисленным ранее требованиям.

Исторически сложились два основных подхода к построению решателей задач *интеллектуальных компьютерных систем*.

Первый подход предполагает наличие в системе фиксированного решателя (например, машины логического вывода), к которому впоследствии добавляется база знаний, наполнение которой определяется предметной областью, в которой должна работать система. Такие системы получили название "пустых" экспертных систем (см. [Попов Э.В. Искусство ИССОиЭС-1990гн](#)) или "оболочек" (expert system shells, см. [Jackson P. IntroES-1998гн](#)). Данный подход, как правило, использовался для разработки относительно несложных систем и в настоящее время не имеет широкого применения.

Второй подход, широко используемый в настоящее время, предполагает наличие программных средств доступа к информации, хранящейся в некоторой базе, совместимых с различными популярными языками программирования. Данный подход широко используется, например, в системах, построенных на основе стандартов W3C (см. [WorldWWC-2023ел](#)), таких как RDF (см. [RDFCAS-2023ел](#)), OWL (см. [OWLWOLDO-2023ел](#)), SPARQL (см. [SPARQLO-2023ел](#)), а также графовых с.у.б.д., таких как Neo4j (см. [Neo4j-2023ел](#)). Структура решателя, построенного на базе таких средств, определяется разработчиком в каждом конкретном случае и не фиксируется какими-либо стандартами. Такой подход обладает большой гибкостью, но отсутствие унификации в структуре и процессе разработки решателей приводит к отсутствию совместимости компонентов решателей, созданных разными разработчиками, большому количеству дублирований одних и тех же решений, повышению накладных расходов в процессе разработки и поддержки решателя. Также существует большое количество реализаций так называемых ризонеров (semantic

reasoners), осуществляющих логический вывод на онтологиях, представленных в формате OWL 2, а также средств разработки и редактирования таких онтологий. Полный список таких средств, признанных консорциумом W3C, можно найти на сайте [OWLI-2023el](#). Как видно из приведенной на нем таблицы, подавляющее большинство средств способно осуществлять только прямой логический вывод на основе отношений, описанных в онтологии.

Среди комплексных подходов к построению решателей задач, разрабатываемых русскоязычными авторами, можно выделить проект IACPaaS (см. [Грибова В.В..БазовТРИСиОПРБЗиРЗ-2015см](#), [Грибова В.В..ПроектIACPaaSSKИС-OOB-2011см](#)), активно развивающийся в настоящее время. Целью данного проекта является разработка облачной платформы для построения на ее основе интеллектуальных сервисов различного назначения. В данном проекте активно используются библиотеки многократно используемых компонентов интеллектуальных систем. Конкретно для построения решателей задач, а также пользовательских интерфейсов таких систем используется многоагентный подход. Несмотря на близость некоторых технологических решений, реализуемых в проекте IACPaaS и в рамках Технологии OSTIS, основной целью указанного проекта является предоставление пользователю большого числа разнородных сервисов, выбор которых осуществляется самим пользователем, в то время как одним из ключевых принципов Технологии OSTIS является разработка общей формальной основы для интеграции различных моделей решения задач с целью их комбинирования при решении одной и той же комплексной задачи.

Задачи интеграции различных подходов, в том числе связанных с решением задач, исследуются также в работе [Филипов А.А..ЕдинаОПИАД-2016см](#) и других работах тех же авторов.

Компонентному проектированию интеллектуальных систем, основанных на знаниях, посвящена работа [Борисов А.Н.ПострИСOnЗспИК-2014см](#), в которой обосновывается необходимость накопления и повторного использования различных компонентов интеллектуальных систем, предлагаются возможные решения данной проблемы с использованием онтологий.

Состояние работ англоязычных авторов, посвященных вопросам решения задач в системах, основанных на знаниях, и актуальных на момент начала 1990-х гг., отражено в обзорных публикациях [Dutta S.KnowlPaAAI-1993bk](#), [Pau L.F.KnowlBP-1990art](#). Более поздние англоязычные работы в данной области в основном ориентированы на решение конкретных частных задач в системах, построенных на основе стандартов W3C, о которых более подробно было сказано выше.

Таким образом, можно сказать, что существует ряд конкретных разработок в направлении построения комплексных технологий разработки интеллектуальных систем различных классов, в том числе с использованием библиотек многократно используемых компонентов, однако проблема разработки комплексной технологии построения гибридных решателей задач в рамках рассмотренных подходов не решена. Во многом это обусловлено отсутствием унифицированной формальной основы для представления любых видов знаний, в том числе различного рода программ, отсутствием строгих принципов, регламентирующих процесс построения решателей задач, а также средств поддержки разработчиков таких решателей и их компонентов.

Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Рассмотрим принципы, лежащие в основе предлагаемого подхода к разработке гибридных решателей задач:

- в качестве основы для построения модели гибридного решателя задач предлагается использовать многоагентный подход. Данный подход позволяет обеспечить основу для построения параллельных асинхронных систем, имеющих распределенную архитектуру, повысить модифицируемость и производительность разработанных решателей;
- процесс решения любой задачи предлагается декомпозировать на *логически атомарные действия*, что также позволит обеспечить совместимость и модифицируемость решателей;
- решатель (как объединенный, так и решатель частного вида) предлагается рассматривать как иерархическую систему, состоящую из нескольких взаимосвязанных уровней. Такой подход позволяет обеспечить возможность проектирования, отладки и верификации компонентов на разных уровнях независимо от других уровней, что существенно упрощает задачу построения решателя за счет снижения накладных расходов;
- предлагается записывать всю информацию о решателе и решаемых им задачах при помощи *SC-кода* в той же базе знаний, что и собственно предметные знания системы. В общем случае такая информация включает:
 - спецификацию агентов, входящих в состав решателя;
 - спецификацию методов, интерпретируемых агентами решателя;
 - спецификацию всех информационных процессов, выполняемых агентами в семантической памяти, в том числе – конструкций, обеспечивающие синхронизацию выполнения параллельных процессов;
 - спецификацию всех задач, на решение которых направлены указанные информационные процессы.

Описание всей указанной информации в единой семантической памяти позволит, во-первых, обеспечить независимость разрабатываемых решателей от *ostis-платформы*, во-вторых, обеспечить возможность систе-

§ 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

мы анализировать происходящие в ней процессы, оптимизировать и синхронизировать их выполнение, т. е. обеспечить *рефлексивность* проектируемых интеллектуальных систем.

Ориентация на многоагентный подход как основу для построения *гибридных решателей задач* обусловлена следующими основными преимуществами такого подхода (см. *Wooldridge M. An Intro to MS-2009bk*):

- автономность (независимость) агентов в рамках такой системы, что позволяет локализовать изменения, вносимые в решатель при его эволюции, и снизить соответствующие трудозатраты, а также обеспечить устойчивость такой системы к отказам некоторых агентов;
- децентрализация обработки, т. е. отсутствие единого контролирующего центра, что также позволяет локализовать вносимые в решатель изменения;
- возможность параллельной работы разных информационных процессов, соответствующих как одному агенту, так и разным агентам, как следствие, – возможность распределенного решения задач. Однако возможность параллельного выполнения информационных процессов подразумевает наличие средств синхронизации такого выполнения, разработка которых является отдельной задачей и подробно рассматривается ниже;
- активность агентов и многоагентной системы в целом, дающая возможность при общении с такой системой не указывать явно способ решения поставленной задачи, а формулировать задачу в **декларативном ключе**.

Построение модели многоагентной системы требует уточнения модели каждого компонента, входящего в ее состав, а именно:

- **модель собственно агента**, входящего в состав такой системы, включая классификацию таких агентов и набор понятий, характеризующих каждый агент в рамках системы. В настоящее время наиболее популярной является модель BDI (belief-desire-intention), в рамках которой предполагается описывать на соответствующих языках "убеждения", "желания" и "намерения" каждого агента системы;
- **модель среды**, в рамках которой находятся агенты, на события в которой они реагируют и в рамках которой могут осуществлять некоторые преобразования. Обзор разновидностей сред для многоагентных систем приводится в работе *Weijns D. Envir aaFCAiMAS-2007art*;
- **модель коммуникации агентов**, в рамках которой уточняется язык взаимодействия агентов (структура и классификация сообщений) и способ передачи сообщений между агентами.

В свою очередь, для разработки модели коммуникации агентов необходимо отдельно уточнить каждый из ее компонентов:

- **Принципы обмена сообщениями между агентами**, т. е. то, каким образом эти сообщения передаются от агента к агенту;
- **Классификацию, семантику и прагматику таких сообщений**, т. е. смысл передаваемой информации и цель такого взаимодействия. В настоящее время стандартами, описывающими структуру передаваемых агентами сообщений, являются Agent Communication Language (ACL) (см. *FIPA ACL MSS-2023el*), разработанный сообществом FIPA, язык KQML (см. *Finin T. KQMLaaACL-1994art*). Указанные стандарты уточняют базовые компоненты каждого сообщения (кодировка, язык сообщения, используемую онтологию понятий, получателя, отправителя и т. д.), не ограничивая при этом смысл сообщения в целом. Также для коммуникации между агентами используется язык KIF, предназначенный для обмена знаниями между любыми программными компонентами (см. *KnowledgeIFS-2023el*);
- **Принципы координации деятельности агентов** В литературе рассматривается большое число вариантов координации деятельности агентов. В работе *Harting R.L. Using MAtRwMO-2008art* предлагается выделить агенты более высокого уровня (метаагенты), задачей которых является сбор информации от агентов нижнего уровня и их координация, схожие идеи высказываются в работе *Sims M.. AutoODfMAS-2008art*. В работах *Excelente-Toledo C.B.. DynamSoCM-2004art*, *Nagendra Prasad M.V.. LearnSSCiCMAS-1999art* предлагаются варианты автоматического выбора оптимального механизма координации агентов для достижения общей цели. Предлагаются также социально-психологические модели координации деятельности агентов, например, на основе некоторых общих "законов" (см. *Vasconcelos W.W.. Norma CRiMAS-2009art*) или эмоций (см. *Rumbell T.. Emoti iAAC AoMaF-2012art*). В работе *Городецкий В.И.. Базов ОКПАиР-2015ст* предложен вариант онтологии коллективного поведения автономных агентов.

К основным недостаткам большинства популярных современных средств построения многагентных систем (см. *Bordini R.H.. ProgMASiASUJ-2007bk*, *Castillo O.. RecenAoHIS-2014bk*, *Eve aWBAP-2023el*, *GAMAP-2023el*, *GOAL-2023el*, *Everts R.. ImpleIMASUJACK-2004art*, *JAVAADF-2023el*, *Boissier O.. MultiAOPwJCM-2013art*) можно отнести следующие:

- жесткая ориентация большинства средств на модель BDI приводит к существенным накладным расходам, связанным с необходимостью выражения конкретной практической задачи в системе понятий BDI. В то же время ориентация на модель BDI неявно провоцирует искусственное разделение языков, описывающих собственно компоненты BDI и знания агента о внешней среде, что приводит к отсутствию унификации представления и, соответственно, дополнительным накладным расходам;
- большинство современных средств построения многоагентных систем ориентированы на представление знаний агента при помощи узкоспециализированных языков, зачастую не предназначенных для представления знаний в широком смысле. Речь при этом идет как о знаниях агента о себе самом, так и знаниях о внешней

среде. В некоторых подходах вначале строится онтология, для создания которой, однако, часто используются средства с низкой выразительной способностью, не предназначенные для построения онтологий (см. *Everts R..ImpléM_ASUJACK-2004art*, *JAVAADF-2023el*). В конечном итоге такой подход приводит к сильной ограниченности возможностей построенных многоагентных систем и их несовместимости;

- абсолютное большинство современных средств предполагает, что взаимодействие агентов осуществляется путем обмена сообщениями непосредственно от агента к агенту или посредством специальных коммуникационных центров (см. *Omicini A..Coord fIAD-1999art*), например, в случае взаимодействия агентов в глобальной сети. Такой подход обладает существенным недостатком, связанным с тем, что в этом случае каждый агент системы должен иметь достаточно полную информацию о других агентах в системе, что приводит к дополнительным затратам ресурсов, кроме того, добавление или удаление одного или нескольких агентов приводит к необходимости оповещения об этом других агентов. Данная проблема решается путем организации общения агентов по принципу "доски объявлений" (см. *Jagannathan V..BlackAaA-1989bk*), предполагающему, что сообщения помещаются в некоторую общую для всех агентов область, при этом каждый агент в общем случае может не знать, какому из агентов адресовано сообщение и от какого из агентов получено то или иное сообщение. Кроме того, в построенной таким образом системе легче обеспечивается параллельное решение несвязанных друг с другом задач, поскольку сообщения, относящиеся к одной задаче, будут игнорироваться агентами, решающими другую задачу. Однако данный подход не исключает проблему, связанную с необходимостью разработки специализированного языка взаимодействия агентов, который в общем случае не связан с языком, на котором описываются знания агента о решаемых задачах и окружающей среде;
- многие средства построения многоагентных систем построены таким образом, что логический уровень взаимодействия агентов жестко привязан к физическому уровню реализации многоагентной системы. Например, при передаче сообщений от агента к агенту разработчику многоагентной системы необходимо помимо семантически значимой информации указывать ip-адрес компьютера, на котором расположен агент-получатель, кодировку, с помощью которой закодирован текст сообщения, и другую техническую информацию, обусловленную исключительно особенностями текущей реализации средств;
- в большинстве подходов среда, с которой взаимодействуют агенты, уточняется отдельно разработчиком для каждой многоагентной системы, что с одной стороны, расширяет возможности применения соответствующих средств, но, с другой стороны, приводит к существенным накладным расходам и несовместимости таких многоагентных систем. Кроме того, в ряде случаев разработчик также обязан учитывать особенности технической реализации средств разработки в плане их стыковки с предполагаемой средой, в роли которой может выступать, например, локальная или глобальная сеть.

Перечисленные недостатки предполагается устранять за счет использования следующих принципов:

- коммуникацию агентов предлагается осуществлять по принципу "доски объявлений", однако в отличие от классического подхода в роли сообщений выступают спецификации в общей семантической памяти выполняемых агентами *действий* (процессов), направленных на решение каких-либо задач, а в роли среды коммуникации выступает сама семантическая память. Такой подход позволяет:
 - исключить необходимость разработки специализированного языка для обмена сообщениями;
 - обеспечить "обезличенность" общения, т. е. каждый из агентов в общем случае не знает, какие еще агенты есть в системе, кем сформулирован и кому адресован тот или иной запрос. Таким образом, добавление или удаление агентов в систему не приводит к изменениям в других агентах, что обеспечивает модифицируемость всей системы;
 - агентам, в том числе конечному пользователю, формулировать задачи в *декларативном ключе*, т. е. не указывать для каждой задачи способ ее решения. Таким образом, агенту заранее не нужно знать, каким образом система решит ту или иную задачу, достаточно лишь специфицировать конечный результат;
 - сделать средства коммуникации агентов и синхронизации их деятельности более понятными разработчику и пользователю системы, не требующими изучения специальных низкоуровневых типов данных и форматов сообщений. Таким образом повышается доступность предлагаемых решений широкому кругу разработчиков.

Следует отметить, что такой подход позволяет при необходимости организовать обмен сообщениями между агентами напрямую и, таким образом, может являться основой для моделирования многоагентных систем, предполагающих другие способы взаимодействия между агентами.

- в роли внешней среды для агентов выступает та же семантическая память, в которой формулируются задачи и посредством которой осуществляется взаимодействие агентов. Такой подход обеспечивает унификацию среды для различных систем агентов, что, в свою очередь, обеспечивает их совместимость;
- спецификация каждого агента описывается средствами *SC-кода* в базе знаний, что позволяет:
 - минимизировать число специализированных средств, необходимых для спецификации агентов, как языковых, так и инструментальных;
 - с одной стороны – минимизировать необходимую в общем случае спецификацию агента, которая включает условие его инициирования и программу, описывающую алгоритм работы агента, с другой стороны – обеспечить возможность произвольного расширения спецификации для каждого конкретного случая, в том числе возможность реализации модели BDI и других;

§ 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

- синхронизацию деятельности агентов предполагается осуществлять на уровне выполняемых ими процессов, направленных на решений тех или иных задач в семантической памяти. Таким образом, каждый агент трактуется как некий абстрактный процессор, способный решать задачи определенного класса. При таком подходе необходимо решить задачу обеспечения взаимодействия параллельных асинхронных процессов в общей семантической памяти, для решения которой можно заимствовать и адаптировать решения, применяемые в традиционной линейной памяти. При этом вводится дополнительный класс агентов – метаагенты, задачей которых является решение возникающих проблемных ситуаций, таких как взаимоблокировки;
- каждый информационный процесс в любой момент времени имеет ассоциативный доступ к необходимым фрагментам базы знаний, хранящейся в семантической памяти, за исключением фрагментов, заблокированных другими процессами в соответствии с рассмотренным ниже механизмом синхронизации. Таким образом, с одной стороны, исключается необходимость хранения каждым агентом информации о внешней среде, с другой стороны, каждый агент, как и в классических многоагентных системах, обладает только частью всей информации, необходимой для решения задачи.

Важно отметить, что в общем случае невозможно априори предсказать, какие именно знания, модели и способы решения задач понадобятся системе для решения конкретной задачи. В связи с этим необходимо обеспечить, с одной стороны, возможность доступа ко всем необходимым фрагментам базы знаний (в пределе – ко всей базе знаний), с другой стороны – иметь возможность локализовать область поиска путем решения задачи, например, рамками одной *предметной области*.

Каждый из агентов обладает набором ключевых элементов (как правило, понятий), которые он использует в качестве отправных точек при ассоциативном поиске в рамках базы знаний. Набор таких элементов для каждого агента уточняется на этапах проектирования решателя задач. Уменьшение числа ключевых элементов агента делает его более универсальным, однако снижает эффективность его работы за счет необходимости выполнения дополнительных операций.

Кроме многоагентного подхода, в основу принципов решения задачи в рамках *Технологии OSTIS* предлагается положить ряд идей, связанных с концепцией ситуационного управления, рассмотренной в работе Д.А. Поспелова *Поспелов Д.А. СитуаУП-1986кн.* До настоящего времени попытки реализации указанной концепции, несмотря на ее актуальность и востребованность, сводились к частным решениям для конкретных классов задач и, к сожалению, не получили широкого распространения. В значительной степени это обусловлено отсутствием универсальной унифицированной основы, которая бы позволила на ее базе создавать языки ситуационного управления в применении к конкретным предметным областям и, что еще важно, повторно использовать фрагменты описаний на таких языках.

Данную проблему можно решить используя предлагаемый в рамках *Технологии OSTIS SC-код* и семейство онтологий верхнего уровня, разработанных на его основе. В частности, реализации идей ситуационного управления способствуют такие принципы, как:

- *SC-код* как базовый язык для описания любой информации в базе знаний и, соответственно, для построения языков ситуационного управления на его основе;
- Базовая теоретико-множественная семантика *SC-кода*, которая позволяет обеспечить возможность формального уточнения всех используемых понятий в виде формального набора онтологий, что позволяет обеспечить совместимость разрабатываемых систем и возможность повторного использования их компонентов;
- Агентно-ориентированный подход к обработке информации, предполагающий реакцию агентов на возникновение в базе знаний определенных ситуаций и событий.

Учитывая рассмотренные выше принципы реализации многоагентного подхода и ситуационного управления в рамках *Технологии OSTIS*, сформулируем более детально основные принципы, лежащие в основе подхода к обработке информации, предлагаемого в *Технологии OSTIS*:

- В основе решателя задач каждой *ostis-системы* лежит многоагентная система, агенты которой взаимодействуют между собой только(!) через общую для них *sc-память* посредством спецификации в этой памяти выполняемых ими *действий в sc-памяти*. При этом пользователи *ostis-системы* также считаются агентами этой системы. Кроме того, *sc-агенты* делятся на внутренние, рецепторные и эффекторные. Взаимодействие между агентами через общую *sc-память* сводится к следующим видам действий:
 - К использованию общедоступной для соответствующей группы *sc-агентов* части хранимой базы знаний;
 - К формированию (генерации) новых фрагментов базы знаний и/или к корректировке (редактированию) каких-либо фрагментов доступной части базы знаний;
 - К интеграции (погружению) новых (обновленных) фрагментов в состав доступной части базы знаний;Подчеркнем, что *sc-агенты* не общаются между собой напрямую путем отправки сообщений, как это делается в большинстве современных подходов к построению многоагентных систем. Кроме того, *sc-агенты* имеют доступ к общей для них базе знаний за счет чего гарантируется семантическая совместимость (взаимопонимание) между агентами, включая и пользователей *ostis-систем*.
- Пользователь *ostis-системы* не может сам непосредственно выполнить какое-либо действие в *sc-памяти*, но он может средствами пользовательского интерфейса инициировать построение (генерацию, формирование в

sc-памяти) sc-текста, являющегося спецификацией *действия в sc-памяти*, выполняемого либо одним *атомарным sc-агентом* за один акт, либо одним *атомарным sc-агентом* за несколько актов, либо коллективом *sc-агентов (неатомарным sc-агентом)*. В спецификации каждого такого *действия в sc-памяти*, инициированного пользователем, этот пользователь указывается как заказчик этого действия. Таким образом, пользователь *ostis-системы* дает поручения (задания, команды) *sc-агентам* этой системы на выполнение различных специфицируемых им действий в *sc-памяти*.

- Каждый *sc-агент*, выполняя некоторое *действие в sc-памяти*, должен "помнить", что *sc-память*, над которой он работает, является общим ресурсом не только для него, но и для всех остальных *sc-агентов*, работающих над этой же *sc-памятью*. Поэтому *sc-агент* должен соблюдать определенную этику поведения в коллективе таких *sc-агентов*, которая должна минимизировать помехи, которые он создает другим *sc-агентам*.
- Деятельность каждого агента *ostis-системы* дискретна и представляет собой множество элементарных действий (актов). При этом при выполнении каждого акта агент может устанавливать блокировки нескольких типов на фрагменты базы знаний. Указанные блокировки позволяют запретить другим агентам изменять указанный фрагмент базы знаний или вообще сделать его "невидимым" для других агентов. Блокировки устанавливаются самим агентом при выполнении соответствующего акта и снимаются им же на последнем этапе выполнения этого акта или раньше, если это возможно.
- Если некий *sc-агент* выполняет некоторое *действие в sc-памяти*, то он на время выполнения этого действия может:
 - Запретить другим *sc-агентам* изменять состояние некоторых *sc-элементов*, хранимых в *sc-памяти* – удалять их, изменять тип;
 - Запретить другим *sc-агентам* добавлять или удалять элементы некоторых множеств, обозначаемых соответствующими *sc-узлами*;
 - Запретить другим *sc-агентам* доступ на просмотр некоторых *sc-элементов*, то есть эти *sc-элементы* становятся полностью "невидимыми" (полностью заблокированными) для других *sc-агентов* но только на время выполнения соответствующего действия.

Указанные блокировки должны быть полностью сняты до завершения выполнения соответствующего действия. Подчеркнем, что число *sc-элементов*, блокируемых на время выполнения некоторого действия, в основном входят атомарные и неатомарные связки, и не должны входить *sc-узлы*, обозначающие бесконечные классы каких-либо сущностей, и, тем более, *sc-узлы*, обозначающие различные понятия (ключевые классы различных предметных областей). Этичное (неэгоистичное) поведение *sc-агента*, касающееся блокировки *sc-элементов*

(то есть ограничения к ним доступа другим *sc-агентам*) предполагает соблюдение следующих правил:

- Не следует блокировать больше *sc-элементов*, чем это необходимо для решения задачи;
- Как только для какого-либо *sc-элемента* необходимость его блокировки отпадает до завершения выполнения соответствующего действия, этот *sc-элемент* желательно сразу деблокировать;

Для того, чтобы *sc-агент* имел возможность работы с каким-либо произвольным *sc-элементом*, он должен либо убедиться в том, что этот *sc-элемент* не входит во фрагмент базы знаний, входящий в *полную блокировку*, либо убедиться в том, что эта блокировка не установлена самим этим агентом. Особой группой полностью заблокированных *sc-элементов* (на время выполнения действия *sc-агентом*) являются вспомогательные *sc-элементы* ("леса"), создаваемые только на время выполнения этого действия. Эти *sc-элементы* в конце выполнения действия должны не деблокироваться, а удаляться.

- Если *действие в sc-памяти*, выполняемое *sc-агентом*, завершилось (т.е. стало прошлой сущностью), то *sc-агент* оформляет результат этого *действия*, указывая (1) удаленные *sc-элементы* и (2) сгенерированные *sc-элементы*. Это необходимо, если по каким-либо причинам придется сделать откат этого *действия*, т.е. возвратиться к состоянию базы знаний до выполнения указанного *действия*.

Перечислим некоторые достоинства предлагаемого подхода к организации обработки знаний в *ostis-системах*:

- поскольку обработка осуществляется агентами, которые обмениваются сообщениями только через общую память, добавление нового агента или исключение (деактивация) одного или нескольких существующих агентов, как правило, не приводит к изменениям в других агентах, поскольку агенты не обмениваются сообщениями напрямую;
- иницирование агентов осуществляется децентрализованно и чаще всего независимо друг от друга, таким образом, даже существенное расширение числа агентов в рамках одной системы не приводит к ухудшению ее производительности;
- спецификации агентов и, как будет показано ниже, их программы могут быть записаны на том же языке, что и обрабатываемые знания, что существенно сокращает перечень специализированных средств, предназначенных для проектирования таких агентов и их коллективов, а также их анализа, верификации и оптимизации, и упрощает разработку системы за счет использования более универсальных компонентов.

§ 3.2.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой

⇒ *ключевое понятие**:

- *действие в sc-памяти*
- *действие в sc-памяти, инициируемое вопросом*
- *действие редактирования базы знаний*
- *задача, решаемая в sc-памяти*
- *класс логически атомарных действий*

Прежде, чем детально рассматривать предлагаемую архитектуру гибридных решателей задач, необходимо уточнить общее понятие действия, выполняемого в sc-памяти и соответствующих задач, классов действий и классов задач.

действие в sc-памяти

- := [внутреннее действие ostis-системы]
- := [действие, выполняемое в sc-памяти]
- := [действие, выполняемое в абстрактной унифицированной семантической памяти]
- := [действие, выполняемое машиной обработки знаний ostis-системы]
- := [действие, выполняемое агентом или коллективом агентов ostis-системы]
- := [информационный процесс над базой знаний, хранимой в sc-памяти]
- := [процесс решения информационной задачи в sc-памяти]
- ⊂ *процесс в sc-памяти*

Каждое **действие в sc-памяти** обозначает некоторое преобразование, выполняемое некоторым *sc-агентом* (или коллективом *sc-агентов*) и ориентированное на преобразование *sc-памяти*. Спецификация действия после его выполнения может быть включена в протокол решения некоторой задачи.

Преобразование состояния базы знаний включает, в том числе и информационный поиск, предполагающий (1) локализацию в базе знаний ответа на запрос, явное выделение структуры ответа и (2) трансляцию ответа на некоторый внешний язык.

Во множество **действий в sc-памяти** входят знаки действий самого различного рода, семантика каждого из которых зависит от конкретного контекста, т.е. ориентации действия на какие-либо конкретные объекты и принадлежности действия какому-либо конкретному классу действий.

Следует четко отличать:

- Каждое конкретное **действие в sc-памяти**, представляющее собой некоторый переходный процесс, переводящий sc-память из одного состояния в другое;
- Каждый тип **действий в sc-памяти**, представляющий собой некоторый класс однотипных (в том или ином смысле) действий;
- sc-узел, обозначающий некоторое конкретное **действие в sc-памяти**;
- sc-узел, обозначающий структуру, которая является описанием, спецификацией, заданием, постановкой соответствующего действия.

Рассмотрим более детально классификацию действий в sc-памяти:

действие в sc-памяти

- ▷ *действие в sc-памяти, инициируемое вопросом*
- ▷ *действие редактирования базы знаний ostis-системы*
- ▷ *действие установки режима ostis-системы*
- ▷ *действие редактирования файла, хранимого в sc-памяти*
- ▷ *действие интерпретации программы, хранимой в sc-памяти*
- ▷ *действие интерпретации scr-программы*

действие в sc-памяти, инициируемое вопросом

- := [действие, направленное на формирование ответа на поставленный вопрос]
- ▷ *действие. сформировать заданный файл*
- ▷ *действие. сформировать заданную структуру*
 - ▷ *действие. верифицировать заданную структуру*
 - ▷ *действие. установить истинность или ложность указываемого логического высказывания*
 - ▷ *действие. установить корректность или некорректность указанной структуры*
 - ▷ *действие. сформировать структуру, описывающую некорректности, имеющиеся в указываемой структуре*
 - ▷ *действие. уточнить тип заданного sc-элемента*

- ▷ действие. установить позитивность/негативность указываемой *sc*-дуги принадлежности или непринадлежности
- ▷ действие. сформировать семантическую окрестность
 - ▷ действие. сформировать полную семантическую окрестность указываемой сущности
 - ▷ действие. сформировать базовую семантическую окрестность указываемой сущности
 - ▷ действие. сформировать частную семантическую окрестность указываемой сущности
- ▷ действие. сформировать структуру, описывающую связи между указываемыми сущностями
 - ▷ действие. сформировать структуру, описывающую сходства указываемых сущностей
 - ▷ действие. сформировать структуру, описывающую различия указываемых сущностей
- ▷ действие. сформировать структуру, описывающую способ решения указанной задачи
- ▷ действие. сформировать план генерации ответа на указанный вопрос
- ▷ действие. сформировать протокол выполнения указанного действия
- ▷ действие. сформировать обоснование корректности указанного решения
- ▷ действие. верифицировать обоснование корректности указанного решения
- ▷ действие, направленное на установление темпоральных характеристик указанной сущности
- ▷ действие, направленное на установление пространственных характеристик указанной сущности

действие редактирования базы знаний

- ▷ действие. изменить направление указанной *sc*-дуги
- ▷ действие. исправить ошибки в заданной структуре
- ▷ действие. преобразовать указанную структуру в соответствии с указанным правилом
- ▷ действие. отождествить два указанных *sc*-элемента
- ▷ действие. включить множество
 - := [сделать все элементы множества *Si* явно принадлежащими множеству *Sj*, то есть сгенерировать соответствующие *sc*-дуги принадлежности]
- ▷ действие генерации *sc*-элементов
 - ▷ действие генерации, одним из аргументов которого является некоторая обобщенная структура
 - ▷ действие. сгенерировать структуру, изоморфную указываемому образцу
 - ▷ действие. сгенерировать *sc*-элемент указанного типа
 - ▷ действие. сгенерировать *sc*-коннектор указанного типа
 - ▷ действие. сгенерировать *sc*-узел указанного типа
 - ▷ действие. сгенерировать файл с заданным содержимым
 - ▷ действие. установить указанный файл в качестве основного идентификатора указанного *sc*-элемента для указанного внешнего языка
- ▷ действие. обновить понятия
 - := [действие. заменить неосновные понятия на их определения через основные понятия]
 - := [действие. заменить некоторое множество понятий на другое множество понятий]
- ▷ действие. интегрировать информационную конструкцию в текущее состояние базы знаний
 - ▷ действие. интегрировать содержимое указанного файла в текущее состояние базы знаний
 - ▷ действие. протранслировать содержимое указанного файла в *sc*-память
 - ▷ действие. интегрировать указанную структуру в текущее состояние базы знаний
- ▷ действие. дополнить описание прошлого состояния *ostis*-системы
 - ▷ действие. дополнить структуру, описывающую историю эволюции *ostis*-системы
 - ▷ действие. дополнить структуру, описывающую историю эксплуатации *ostis*-системы
- ▷ действие удаления *sc*-элементов
 - ▷ действие. удалить указанные *sc*-элементы
 - ▷ действие. удалить *sc*-элементы, входящие в состав указанной структуры и не являющиеся ключевыми узлами каких-либо *sc*-агентов

действие. отождествить два указанных *sc*-элемента

- := [действие. совместить два указанных *sc*-элемента]
- := [действие. склеить два указанных *sc*-элемента]
- ⇒ разбиение*:
 - {• действие. физически отождествить два указанных *sc*-элемента
 - действие. логически отождествить два указанных *sc*-элемента

Каждое **действие. отождествить два указанных *sc*-элемента** может быть выполнено как **действие. физически отождествить два указанных *sc*-элемента** или **действие. логически отождествить два указанных *sc*-элемента**. В случае логического отождествления в протоколе деятельности агентов сохраняется само действие с его спецификацией, включающей обязательное указание того, какие элементы были сгенерированы, а какие удалены. В случае физического отождествления протокол действия не сохраняется.

Каждое **действие. обновить понятия** обозначает переход от какой-то группы понятий, использовавшихся ранее, к другой группе понятий, которые будут использоваться вместо первых, и станут *основными понятиями*. В общем случае **действие. обновить понятия** состоит из следующих этапов:

- Определить заменяемые понятия на основе заменяющих;
- Внести соответствующие изменения в программы sc-агентов, ключевыми узлами которых являются обновляемые понятия;
- Заменить все конструкции в базе знаний, содержащие заменяемые понятия, в соответствии с определениями этих понятий через заменяющие их понятия;
- При необходимости, *sc-элементы*, обозначающие замененные таким образом понятия, могут быть полностью выведены из текущего состояния базы знаний.

Первым аргументом (входящим в знак *действия* под атрибутом 1') **действия. обновить понятия** является знак множества *sc-узлов*, обозначающих заменяемые понятия, вторым (входящим в знак *действия* под атрибутом 2') - знак множества *sc-узлов*, обозначающих заменяющие понятия. В общем случае любое или оба этих множества могут быть *синглетонами*.

действие. удалить указанные sc-элементы

⇒ разбиение*:

- {• *действие. физически удалить указанные sc-элементы*
- *действие. логически удалить указанные sc-элементы*

}

Каждое **действие. удалить указанные sc-элементы** может быть выполнено как *действие. физически удалить указанные sc-элементы* или *действие. логически удалить указанные sc-элементы*. В случае логического удаления в протоколе деятельности агентов сохраняется само действие с его спецификацией, включающей обязательное указание того, какие элементы были удалены, т.е. по сути, элементы просто исключаются из текущего состояния базы знаний. В случае физического удаления протокол действия не сохраняется.

В случае удаления какого-либо *sc-элемента*, инцидентные ему *связки*, в том числе *sc-коннекторы*, также удаляются.

Для того, чтобы выполнить **действие. интегрировать указанную структуру в текущее состояние базы знаний**, необходимо склеить *sc-элементы*, входящие в интегрируемую *структурную* с синонимичными им *sc-элементами*, входящими в текущее состояние базы знаний, заменить неиспользуемые (например, устаревшие) понятия, входящие в интегрируемую *структурную*, на используемые (т.е. заменить неиспользуемые понятия на их определения через используемые), явно включить все элементы интегрируемой *структурной* в число элементов утвержденной части базы знаний и явно включить все элементы интегрируемой *структурной* в число элементов одного из атомарных разделов утвержденной части базы знаний.

Более подробно соотношение между понятиями “*действие*”, “*задача*”, “*класс действий*” и “*класс задач*” рассмотрено в Главе 3.1. *Формализация понятий действия, задачи, метода, средства, навыка и технологии*. Рассмотрим указанные соотношения в контексте решения задач в *sc-памяти*.

задача, решаемая в sc-памяти

⊂ задача

:= [спецификация действия, выполняемого в *sc-памяти*]

:= [структура, являющаяся таким описанием (постановкой, заданием) соответствующего действия в *sc-памяти*, которое обладает достаточной полнотой для выполнения указанного действия]

:= [семантическая окрестность некоторого действия в *sc-памяти*, обеспечивающая достаточно полное задание этого действия]

класс действий

▷ класс действий в *sc-памяти*

⇐ семейство подмножеств*:

действие в *sc-памяти*

⇒ разбиение*:

- {• класс логически атомарных действий

:= [класс автономных действий]

▷ класс логически атомарных действий в *sc-памяти*

- класс логически неатомарных действий

:= [класс неавтономных действий]

}

Каждое *действие*, принадлежащее некоторому *классу логически атомарных действий*, обладает двумя необходимыми свойствами:

- выполнение действия не зависит от того, является ли указанное действие частью декомпозиции более общего действия. При выполнении данного действия также не должен учитываться тот факт, что данное действие предшествует каким-либо другим действиям или следует за ними (что явно указывается при помощи отношения *последовательность действий**);
- указанное действие должно представлять собой логически целостный акт преобразования, например, в семантической памяти. Такое действие по сути является транзакцией, т. е. результатом такого преобразования становится новое состояние преобразуемой системы, а выполняемое действие должно быть либо выполнено полностью, либо не выполнено совсем, частичное выполнение не допускается.

В то же время логическая атомарность не запрещает декомпозировать выполняемое действие на более частные, каждое из которых, в свою очередь, также будет являться логически атомарным.

На логически атомарные действия предлагается делить всю деятельность, направленную на решение каких-либо задач *ostis*-системой. Соответственно *решатель задач ostis-системы* предлагается делить на компоненты, соответствующие таким *классам логически атомарных действий в sc-памяти*, что является основой для обеспечения его *модифицируемости*. Такие компоненты решателя называются *sc-агентами*.

В свою очередь под *методом* понимается описание того, как может быть выполнено любое или почти любое (с явным указанием исключений) действие, принадлежащее соответствующему *классу действий*. Поскольку конкретному *классу действий* ставится в соответствие некоторый конкретный *класс задач*, то можно сказать, что метод описывает способ решения любых задач принадлежащих заданному классу. Понятие метода можно считать обобщением понятия "программа", в связи с чем в рамках Технологии *OSTIS* термины "метод" и "программа" являются синонимичными (см. Главу 3.1. *Формализация понятий действия, задачи, метода, средства, навыка и технологии*).

Примером конкретного метода может быть процедурная программа на конкретном языке программирования, или множество логических высказываний, составляющих формальную теорию заданной предметной области (аналог логической программы).

Частным случаем метода является программа атомарного компонента *решателя задач ostis-системы (атомарного sc-агента*, см. § 3.2.3. *Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*), в этом случае в качестве операционной семантики метода выступает коллектив агентов более низкого уровня, интерпретирующий соответствующую программу (в предельном случае это будут агенты, являющиеся частью платформы интерпретации моделей компьютерных систем, в том числе аппаратной).

В свою очередь, навык трактуется как объединение некоторого *метода* и его операционной семантики, то есть информации о том, каким образом должен интерпретироваться данный *метод*.

Таким образом, можно говорить об *иерархии методов* и о *методах* интерпретации других *методов*, а также, соответственно, об *иерархии навыков*, которыми обладает *ostis-система* на данный момент времени.

§ 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты

⇒ *ключевое понятие**:

- *sc-агент*
- *абстрактный sc-агент*
- *атомарный абстрактный sc-агент*
- *неатомарный абстрактный sc-агент*
- *абстрактный sc-агент, реализуемый на Языке SCP*
- *абстрактный sc-агент, не реализуемый на Языке SCP*

Как было сказано выше, *решатель задач ostis-системы* предлагается делить на компоненты, соответствующие *классам логически атомарных действий в sc-памяти*, называемые *sc-агентами*.

sc-агент

:= [единственный вид *субъектов*, выполняющих преобразования в *sc-памяти*]

:= [*субъект*, способный выполнять *действия в sc-памяти*, принадлежащие некоторому определенному *классу логически атомарных действий*]

Логическая атомарность выполняемых *sc-агентом* действий предполагает, что каждый *sc-агент* реагирует на соответствующий ему класс ситуаций и/или событий, происходящих в *sc-памяти*, и осуществляет определенное преобразование *sc-текста*, находящегося в семантической окрестности обрабатываемой ситуации и/или события.

При этом каждый sc-агент в общем случае не имеет информацию о том, какие еще sc-агенты в данный момент присутствуют в системе и осуществляет взаимодействие в других sc-агентами исключительно посредством формирования некоторых конструкций (как правило – спецификаций действий) в общей sc-памяти. Таким сообщением может быть, например, вопрос, адресованный другим sc-агентам в системе (заранее не известно, каким конкретно), или ответ на поставленный другими sc-агентами вопрос (заранее не известно, каким конкретно). Таким образом, каждый sc-агент в каждый момент времени контролирует только фрагмент базы знаний в контексте решаемой данным агентом задачи, состояние всей остальной базы знаний в общем случае непредсказуемо для sc-агента.

Поскольку предполагается, что копии одного и того же *sc-агента* или функционально эквивалентные *sc-агенты* могут работать в разных *ostis*-системах, будучи при этом физически разными sc-агентами, то целесообразно рассматривать свойства и классификацию не sc-агентов, а классов функционально эквивалентных sc-агентов, которые будем называть *абстрактными sc-агентами*. Под *абстрактным sc-агентом* понимается некоторый класс функционально эквивалентных sc-агентов, разные экземпляры (т.е. представители) которого могут быть реализованы по-разному.

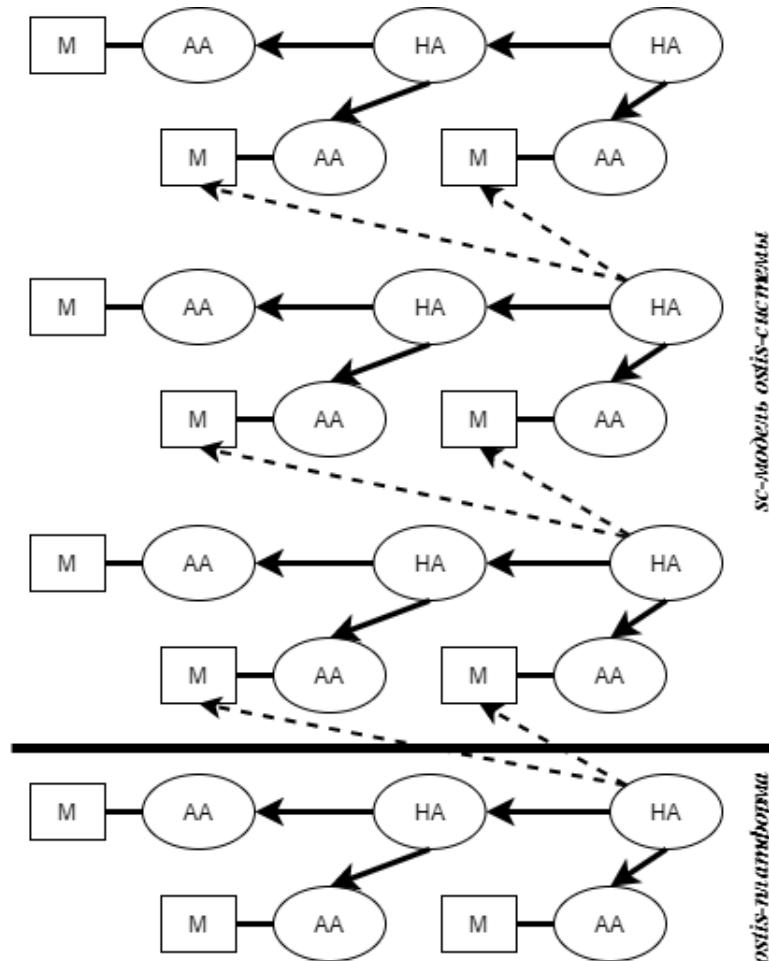
Каждый *абстрактный sc-агент* имеет соответствующую ему спецификацию. В спецификацию каждого *абстрактного sc-агента* входит:

- указание ключевых sc-элементов этого sc-агента, т.е. тех sc-элементов, хранимых в sc-памяти, которые для данного sc-агента являются «точками опоры»;
- формальное описание условий инициирования данного sc-агента, т.е. тех *ситуация* в sc-памяти, которые инициируют деятельность данного sc-агента;
- формальное описание первичного условия инициирования данного sc-агента, т.е. такой ситуации в sc-памяти, которая побуждает sc-агента перейти в активное состояние и начать проверку наличия своего полного условия инициирования (для внутренних абстрактных sc-агентов);
- строгое, полное, однозначно понимаемое описание деятельности данного sc-агента, оформленное при помощи каких-либо понятных, общепринятых средств, не требующих специального изучения, например на естественном языке.
- описание результатов выполнения данного sc-агента.

Sc-агенты можно классифицировать по различным признакам. Поскольку можно говорить об иерархии методов (методах интерпретации других методов) и, соответственно, иерархии навыков, то есть необходимость говорить и об иерархии sc-агентов, обеспечивающих интерпретацию того или иного метода. В данном контексте можно говорить об иерархии sc-агентов в двух аспектах:

- *абстрактному sc-агенту* (и соответственно, *sc-агенту*) может однозначно соответствовать *метод* (программа sc-агента), описывающий деятельность данного sc-агента. Такие агенты будем называть *атомарными абстрактными sc-агентами*;
- *абстрактные sc-агенты* иногда целесообразно объединять в коллектизы таких агентов, которые можно рассматривать как один целостный *абстрактный sc-агент*, с логической точки зрения работающий по тем же принципам, что и *атомарные абстрактные sc-агенты*, то есть реагирующий на события в sc-памяти и описы-вающий свою деятельность в рамках этой памяти. Такому *абстрактному sc-агенту* не будет соответствовать какой-то конкретный *метод*, хранимый в sc-памяти, но остальная часть спецификации *абстрактного sc-агента* (условие инициирования, описание начальной ситуации и результата работы sc-агента и т.д.) остается такой же, как у *атомарного абстрактного sc-агента*. Таким образом, можно сказать, что понятие атомарности/неатомарности абстрактного sc-агента указывает на то, каким образом уточняется реализация данного *абстрактного sc-агента* – посредством указания конкретного метода (программы sc-агента) или посредством декомпозиции *абстрактного sc-агента* на более простые. Важно отметить, что *неатомарные абстрактные sc-агенты* тоже могут входить в состав других, более сложных *неатомарных абстрактных sc-агентов*. Таким образом формируется иерархическая система абстрактных sc-агентов, в общем случае имеющая произвольное количество уровней.
- В свою очередь, соответствующий sc-агенту метод должен интерпретироваться каким-либо другим sc-агентом более низкого уровня, а чаще всего – коллективом таких агентов, каждому из которых ставится в соответствие свой метод, описывающий поведение данного агента, но уже на более низком уровне. Таким образом, можно сказать, что понятие атомарности/неатомарности абстрактных sc-агентов применимо в рамках одного языка *описания методов*. В свою очередь можно говорить об иерархии *абстрактных sc-агентов* с точки зрения уровня языка описания соответствующих таким агентам методов. В общем случае такая иерархия тоже может иметь неограниченное число уровней, однако очевидно, что при понижении уровня языка описания методов мы рано или поздно должны подойти к языку описания методов, который будет интерпретироваться агентами, реализуемыми на уровне *ostis-платформы*, а спускаясь еще ниже – на уровень языка описания методов, интерпретируемых на аппаратном уровне. Таким образом, для обеспечения платформенной независимости *ostis-систем* целесообразно выделить такой язык описания методов, который бы интерпретировался на уровне *ostis-платформы* и являлся основой для разработки интерпретаторов более высокоуровневых языков. В качестве такого языка предлагается *Язык SCP* (*Semantic Code Programming*), который рассматривается в качестве ассемблера для *ассоциативного семантического компьютера*.

На рисунке *Иерархия sc-агентов* проиллюстрирована иерархия *абстрактных sc-агентов* и соответствующим *атомарным абстрактным sc-агентам методов*. Буквой “M” на рисунке условно обозначены *методы*, буквами “AA” и “HA” – *атомарные абстрактные sc-агенты* и *неатомарные абстрактные sc-агенты* соответственно, сплошными стрелками показана декомпозиция *неатомарных sc-агентов* на более простые, а пунктирными стрелками – связь между *методами* и их операционной семантикой, то есть *абстрактными sc-агентами*, обеспечивающими интерпретацию этих *методов*. Как показано на приведенной иллюстрации, должна существовать четкая граница между методами, которые описываются на уровне ostis-платформы, и методами, которые могут быть описаны и на платформенно-независимом уровне. Более подробно этот вопрос рассматривается в Главе 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем.



= Иерархия sc-агентов

Иерархический подход к описанию систем sc-агентов, определяющих операционную семантику *решателей задач ostis-систем*, и соответственно, самих *решателей задач ostis-систем* обладает рядом важных преимуществ, таких как обеспечение модифицируемости решателей и удобство их проектирования и отладки на разных уровнях.

Сказанное выше позволяет описать классификацию *абстрактных sc-агентов* по признаку атомарности:

абстрактный sc-агент

⇒ разбиение*:

- {• неатомарный *абстрактный sc-агент*
 - атомарный *абстрактный sc-агент*
- }

Под *неатомарным абстрактным sc-агентом* понимается *абстрактный sc-агент*, который декомпозируется на коллектив более простых *абстрактных sc-агентов*, каждый из которых в свою очередь может быть как *атомарным абстрактным sc-агентом*, так и *неатомарным абстрактным sc-агентом*. При этом в каком либо варианте декомпозиции *абстрактного sc-агента** дочерний *неатомарный абстрактный sc-агент* может стать *атомарным абстрактным sc-агентом*, и реализовываться соответствующим образом.

Под *автомарным абстрактным sc-агентом* понимается *абстрактный sc-агент*, для которого уточняется способ его реализации, т.е. существует соответствующая связка отношения *программа sc-агента**. Подчеркнем при этом, что в рамках конкретной *ostis*-системы для каждого языка *представления методов* может существовать своя иерархия *абстрактных sc-агентов*.

В свою очередь, Язык *SCP* позволяет установить границу между логико-семантической моделью *ostis*-системы и *ostis*-платформой. В связи с этим будем считать платформенно-независимыми *абстрактные sc-агенты*, реализованные на языке *SCP* или более высокого уровня языках на его основе, а платформенно-зависимыми *абстрактные sc-агенты*, которые реализованы на уровне платформы (например, с целью повышения их производительности). В то же время существует ряд абстрактных sc-агентов, которые принципиально не могут быть реализованы на Языке *SCP*. Сказанное отражено в следующей иерархии:

абстрактный sc-агент

- ⇒ разбиение*:
 - {• внутренний абстрактный sc-агент
 - эффекторный абстрактный sc-агент
 - рецепторный абстрактный sc-агент
- ⇒ разбиение*:
 - {• абстрактный sc-агент, не реализуемый на Языке *SCP*
 - абстрактный sc-агент, реализуемый на Языке *SCP*
- ⇒ разбиение*:
 - {• абстрактный sc-агент интерпретации *scp*-программ
 - абстрактный программный sc-агент
 - абстрактный sc-метаагент
- ⇒ разбиение*:
 - {• платформенно-зависимый абстрактный sc-агент
 - ▷ абстрактный sc-агент, не реализуемый на Языке *SCP*
 - платформенно-независимый абстрактный sc-агент

абстрактный sc-агент

- ⇒ разбиение*:
 - {• абстрактный sc-агент, не реализуемый на Языке *SCP*
 - абстрактный sc-агент, реализуемый на Языке *SCP*
- ⇒ разбиение*:
 - {• платформенно-зависимый абстрактный sc-агент
 - ▷ абстрактный sc-агент, не реализуемый на Языке *SCP*
 - платформенно-независимый абстрактный sc-агент

*абстрактный sc-агент, не реализуемый на Языке *SCP**

- := [абстрактный sc-агент, который не может быть реализован на платформенно-независимом уровне]
- ⇒ разбиение*:
 - {• эффекторный абстрактный sc-агент
 - рецепторный абстрактный sc-агент
 - абстрактный sc-агент интерпретации *scp*-программ

*абстрактный sc-агент, реализуемый на Языке *SCP**

- := [абстрактный sc-агент, который может быть реализован на платформенно-независимом уровне]
- ⇒ разбиение*:
 - {• абстрактный sc-метаагент
 - абстрактный программный sc-агент, реализуемый на Языке *SCP*

абстрактный программный sc-агент

- ⇒ разбиение*:
 - {• эффекторный абстрактный sc-агент

- рецепторный абстрактный sc-агент
 - абстрактный программный sc-агент, реализуемый на Языке SCP
- }

атомарный абстрактный sc-агент

⇒ разбиение*:

- {• платформенно-независимый абстрактный sc-агент
 - платформенно-зависимый абстрактный sc-агент
- }

К **платформенно-независимым абстрактным sc-агентам** относят *атомарные абстрактные sc-агенты*, реализованные на базовом языке программирования Технологии OSTIS, т.е. на **Языке SCP**.

При описании **платформенно-независимых абстрактных sc-агентов** под платформенной независимостью понимается платформенная независимость с точки зрения Технологии OSTIS, т.е реализация на специализированном языке программирования, ориентированном на обработку семантических сетей (**Языке SCP**), поскольку *атомарные sc-агенты*, реализованные на указанном языке могут свободно переноситься с одной ostis-платформы на другую. При этом языки программирования, традиционно считающиеся платформенно-независимыми, в данном случае не могут считаться таковыми.

Существуют **sc-агенты**, которые принципиально не могут быть реализованы на платформенно-независимом уровне, например, собственно **sc-агенты** интерпретации **sc-моделей** или рецепторные и эффекторные **sc-агенты**, обеспечивающие взаимодействие с внешней средой.

К **платформенно- зависимым абстрактным sc-агентам** относят *атомарные абстрактные sc-агенты*, реализованные ниже уровня **sc-моделей**, т.е. не на **Языке SCP**, а на каком-либо другом языке описания программ.

Каждый **внутренний абстрактный sc-агент** обозначает класс **sc-агентов**, которые реагируют на события в **sc-памяти** и осуществляют преобразования исключительно в рамках этой же **sc-памяти**.

Каждый **эффекторный абстрактный sc-агент** обозначает класс **sc-агентов**, которые реагируют на события в **sc-памяти** и осуществляют преобразования во внешней относительно данной **ostis-системы** среде.

Каждый **рецепторный абстрактный sc-агент** обозначает класс **sc-агентов**, которые реагируют на события во внешней относительно данной **ostis-системы** среде и осуществляют преобразования в памяти данной системы.

Каждый **абстрактный sc-агент, не реализуемый на Языке SCP** должен быть реализован на уровне **ostis-платформы**, в том числе, аппаратной. К таким **абстрактным sc-агентам** относятся абстрактные **sc-агенты** интерпретации **scp-программ**, а также эффекторные и рецепторные абстрактные **sc-агенты**.

Каждый **абстрактный sc-агент, реализуемый на Языке SCP** может быть реализован на **Языке SCP**, то есть платформенно-независимом уровне, но при необходимости, может реализовываться и на уровне платформы, например, с целью повышения производительности.

К **абстрактным sc-агентам интерпретации scp-программ** относятся не реализуемые на платформенно-независимом уровне **абстрактные sc-агенты**, обеспечивающие интерпретацию **scp-программ** и **scp-метапрограмм**, в том числе создание **scp-процессов**, собственно интерпретацию **scp-операторов**, а также другие вспомогательные действия. По сути, агенты данного класса обеспечивают работу **sc-агентов** более высоких уровней (программных **sc-агентов** и **sc-метаагентов**), реализованных на **Языке SCP**, в частности, обеспечивают соблюдение указанными агентами общих принципов синхронизации.

К **абстрактным программным sc-агентам** относятся все **абстрактные sc-агенты**, обеспечивающие основной функционал системы, то есть ее возможность решать те или иные задачи. Агенты данного класса должны работать в соответствии с общими принципами синхронизации деятельности субъектов в **sc-памяти**.

Задачей **абстрактных sc-метаагентов** является координация деятельности **абстрактных программных sc-агентов**, в частности, решение проблемы взаимоблокировок. Агенты данного класса могут быть реализованы на **Языке SCP**, однако для синхронизации их деятельности используются другие принципы, соответственно, для реализации таких агентов требуется Язык SCP другого уровня, типология операторов которого полностью аналогична типологии **scp-операторов**, однако эти операторы имеют другую операционную семантику, учитывающую отличия в принципах синхронизации (работы с **блокировками***). Программы такого языка будем называть **scp-метапрограммами**, соответствующие им **процессы в sc-памяти – scp-метапроцессами**, операторы – **scp-метаоператорами**.

*декомпозиция абстрактного sc-агента**

∈ отношение декомпозиции

⇒ первый домен*:

неатомарный абстрактный sc-агент

Отношение **декомпозиции абстрактного sc-агента*** трактует неатомарные абстрактные sc-агенты как коллектизы более простых абстрактных sc-агентов, взаимодействующих через sc-память.

Другими словами, **декомпозиция абстрактного sc-агента*** на абстрактные sc-агенты более низкого уровня уточняет один из возможных подходов к реализации этого *абстрактного sc-агента* путем построения коллектиза более простых *абстрактных sc-агентов*.

sc-агент

:= [агент над sc-памятью]
 ⊂ субъект
 ⇒ семейство подмножеств*:
 абстрактный sc-агент

Под **sc-агентом** понимается конкретный экземпляр (с теоретико-множественной точки зрения - элемент) некоторого *атомарного абстрактного sc-агента*, работающий в какой-либо конкретной интеллектуальной системе.

Таким образом, каждый **sc-агент** - это субъект, способный выполнять некоторый класс однотипных действий либо только над *sc-памятью*, либо над *sc-памятью* и внешней средой (для эффекторных *sc-агентов*). Каждое такое действие инициируется либо состоянием или ситуацией в *sc-памяти*, либо состоянием или ситуацией во внешней среде (для рецепторных *sc-агентов*-датчиков), соответствующей условию инициирования *атомарного абстрактного sc-агента*, экземпляром которого является заданный *sc-агент*. В данном случае можно провести аналогию между принципами объектно-ориентированного программирования, рассматривая *атомарный абстрактный sc-агент* как класс, а конкретный *sc-агент* – как экземпляр, конкретную имплементацию этого класса.

Взаимодействие *sc-агентов* осуществляется только через *sc-память*. Как следствие, результатом работы любого *sc-агента* является некоторое изменение состояния *sc-памяти*, т.е. удаление либо генерация каких-либо *sc-элементов*.

В общем случае один *sc-агент* может явно передать управление другому *sc-агенту*, если этот *sc-агент* априори известен. Для этого каждый *sc-агент* в *sc-памяти* имеет обозначающий его *sc-узел*, с которым можно связать конкретную ситуацию в текущем состоянии базы знаний, которую инициируемый *sc-агент* должен обработать.

Однако далеко не всегда легко определить того *sc-агента*, который должен принять управление от заданного *sc-агента*, в связи с чем описанная выше ситуация возникает крайне редко. Более того, иногда условие инициирования *sc-агента* является результатом деятельности непредсказуемой группы *sc-агентов*, равно как и одна и та же конструкция может являться условием инициирования целой группы *sc-агентов*.

При этом общаются через *sc-память* не программы *sc-агентов**, а сами описываемые данными программами *sc-агенты*.

В процессе работы *sc-агент* может сам для себя порождать вспомогательные *sc-элементы*, которые сам же удаляет после завершения акта своей деятельности (это вспомогательные *структуры*, которые используются в качестве "информационных лесов" только в ходе выполнения соответствующего акта деятельности и после завершения этого акта удаляются).

sc-агент

▷ активный sc-агент
 ⇒ первый домен*:
 • ключевые sc-элементы sc-агента*
 • программа sc-агента*
 • первичное условие инициирования*
 • условие инициирования и результат*

Под **активным sc-агентом** понимается *sc-агент ostis*-системы, который реагирует на события, соответствующие его условию инициирования, и, как следствие, его *первичному условию инициирования**. Не входящие во множество **активных sc-агентов** *sc-агенты* не реагируют ни на какие события в *sc-памяти*.

Связки отношения **ключевые sc-элементы sc-агента*** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и *sc-узел*, обозначающий множество *sc-элементов*, которые являются ключевыми для данного *абстрактного sc-агента*, то данные *sc-элементы* явно упоминаются в рамках программ, реализующих данный *абстрактный sc-агент*.

Связки отношения **программа sc-агента*** связывают между собой *sc-узел*, обозначающий *атомарный абстрактный sc-агент* и *sc-узел*, обозначающий множество программ, реализующих указанный *атомарный абстрактный sc-агент*. В случае *платформенно-независимого абстрактного sc-агента* каждая связка отношения *программа sc-агента** связывает *sc-узел*, обозначающий указанный *абстрактный sc-агент* с множеством *scr-программ*, описывающих деятельность данного *абстрактного sc-агента*. Данное множество содержит одну *агентную scr-*

программу, и произвольное количество (может быть, и ни одной) *scp-программы*, которые необходимы для выполнения указанной *агентной scp-программы*.

В случае *платформенно-зависимого абстрактного sc-агента* каждая связка отношения *программа sc-агента** связывает *sc-узел*, обозначающий указанный *абстрактный sc-агент* с множеством файлов, содержащих исходные тексты программы на некотором внешнем языке программирования, реализующей деятельность данного *абстрактного sc-агента*.

Связки отношения *первичное условие инициирования** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и бинарную ориентированную пару, описывающую первичное условие инициирования данного *абстрактного sc-агента*, т.е. такой спецификацию *ситуации в sc-памяти*, возникновение которой побуждает *sc-агента* перейти в активное состояние и начать проверку наличия своего полного условия инициирования.

Первым компонентом данной ориентированной пары является знак некоторого класса *элементарных событий в sc-памяти**, например, *событие добавления sc-дуги, выходящей из заданного sc-элемента**.

Вторым компонентом данной ориентированной пары является произвольный в общем случае *sc-элемент*, с которым непосредственно связан указанный тип события в *sc-памяти*, т.е., например, *sc-элемент*, из которого выходит либо в который входит генерируемая либо удаляемая *sc-дуга*, либо *файл*, содержимое которого было изменено.

После того, как в *sc-памяти* происходит некоторое событие, активизируются все *активные sc-агенты*, *первичное условие инициирования** которых соответствует произошедшему событию.

Связки отношения *условие инициирования и результат** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и бинарную ориентированную пару, связывающую условие инициирования данного *абстрактного sc-агента* и результаты выполнения данного экземпляров данного *sc-агента* в какой-либо конкретной системе.

Указанную ориентированную пару можно рассматривать как логическую связку импликации, при этом на *sc-переменные*, присутствующие в обеих частях связки, неявно накладывается квантор всеобщности, на *sc-переменные*, присутствующие либо только в посылке, либо только в заключении неявно накладывается квантор существования.

Первым компонентом указанной ориентированной пары является логическая формула, описывающая условие инициирования описываемого *абстрактного sc-агента*, то есть конструкции, наличие которой в *sc-памяти* побуждает *sc-агента* начать работу по изменению состояния *sc-памяти*. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

Вторым компонентом указанной ориентированной пары является логическая формула, описывающая возможные результаты выполнения описанного *абстрактного sc-агента*, то есть описание произведенных им изменений состояния *sc-памяти*. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

описание поведения sc-агента
 \subseteq семантическая окрестность

Описание поведения sc-агента представляет собой *семантическую окрестность*, описывающую деятельность *sc-агента* до какой-либо степени детализации, однако такое описание должно быть строгим, полным и однозначно понимаемым. Как любая другая *семантическая окрестность*, *описание поведения sc-агента* может быть про-транслировано на какие-либо понятные, общепринятые средства, не требующие специального изучения, например на естественный язык.

Описываемый *абстрактный sc-агент* входит в соответствующее *описание поведения sc-агента* под атрибутом *ключевой sc-элемент'*.

§ 3.2.4. Принципы синхронизации деятельности sc-агентов

\Rightarrow *ключевое понятие*:*

- *действие в sc-памяти*
- *блокировка**
- *тип блокировки*
- *планируемая блокировка**
- *приоритет блокировки**
- *удаляемые sc-элементы**
- *транзакция в sc-памяти*

Одной из важных особенностей многоагентного подхода к решению задач является возможность параллельного решения различных задач, что в свою очередь, предполагает параллельность выполнения соответствующих информационных процессов.

Понятия *действие в sc-памяти*, и *процесс в sc-памяти* (информационный процесс, выполняемый агентом в семантической памяти), являются синонимичными, поскольку все процессы, протекающие в sc-памяти, являются осознанными и выполняются каким-либо sc-агентами. Тем не менее, когда идет речь о синхронизации выполнения каких-либо преобразований в памяти компьютерной системы, в литературе принято использовать именно термины “процесс”, “взаимодействие процессов” (см. *Dijkstra E.W.CoopeSP-2002bk*, *Hoare C.A.R.CommuSP-1983art*), в связи с чем будем использовать этот термин при описании принципов синхронизации деятельности sc-агентов при выполнении ими параллельных процессов в sc-памяти.

действие в sc-памяти

\coloneqq часто используемый sc-идентификатор*:
[процесс в sc-памяти]

процесс в sc-памяти

\Rightarrow разбиение*:
 {• процесс в sc-памяти, соответствующий платформенно-зависимому sc-агенту
 • scr-процесс
 \Rightarrow разбиение*:
 {• scr-процесс, не являющийся scr-метапроцессом
 • scr-метапроцесс
 }
 }

процесс в sc-памяти, соответствующий платформенно-зависимому sc-агенту

\Rightarrow разбиение*:
 {• процесс в sc-памяти, соответствующий платформенно-зависимому sc-агенту и не являющийся действием абстрактной scr-машины
 • действие абстрактной scr-машины
 \supset действие интерпретации scr-программы
 }

Для синхронизации выполнения *процессов в sc-памяти* предлагается использовать механизм блокировок, построенный на основе существующих алгоритмов синхронизации информационных процессов в традиционных системах (см. *Dijkstra E.W.CoopeSP-2002bk*, *Hoare C.A.R.CommuSP-1983art*). В качестве возможного направления развития данного подхода можно указать набирающие популярность идеи lock-free алгоритмов (см. *Chatterjee B..nBlockDUGwWCAB-2022art*).

Отношение *блокировка** связывает знаки *действий в sc-памяти* со знаками *структур* (ситуативных), которые содержат элементы, заблокированные на время выполнения данного действия или на какую-то часть этого периода. Каждая такая *структура* принадлежит какому-либо из *типов блокировки*.

Первым компонентом связок отношения *блокировка** является знак *действия в sc-памяти*, вторым – знак заблокированной *структурой*.

*блокировка**

\in бинарное отношение

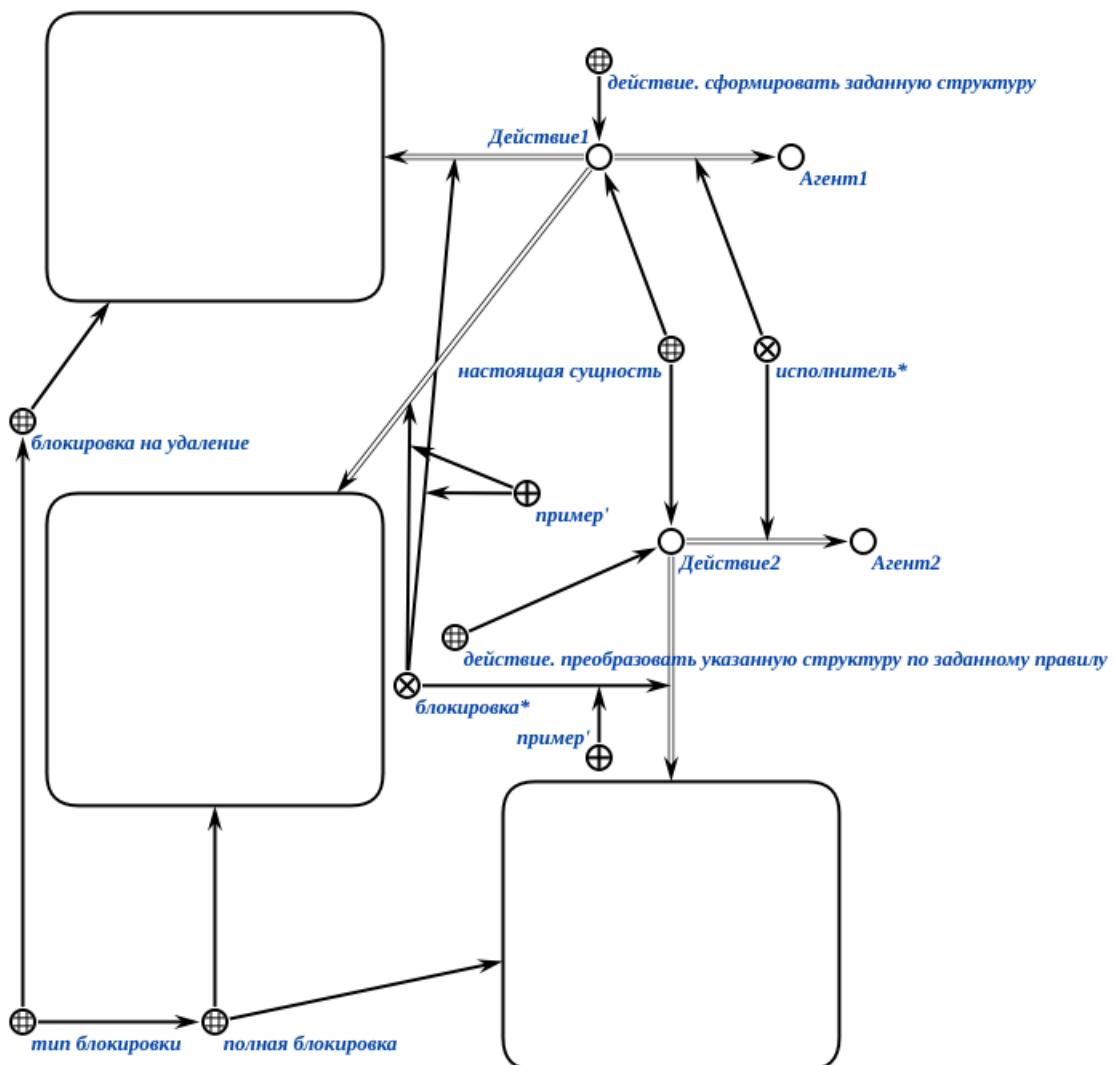
тип блокировки

\exists полная блокировка
 \exists блокировка на любое изменение
 \exists блокировка на удаление

Множество *тип блокировки* содержит все возможные классы блокировок, т.е. *структуры*, содержащие *sc-элементы*, заблокированные каким-либо *sc-агентом* на время выполнения им некоторого *действия в sc-памяти*.

Каждая *структура*, принадлежащая множеству *полная блокировка* содержит *sc-элементы*, просмотр и изменение (удаление, добавление инцидентных *sc-коннекторов*, удаление самих *sc-элементов*, изменение содержимого в случае файла) которых запрещены всем *sc-агентам*, кроме собственно *sc-агента*, выполняющего соответствующее данной структуре *действие в sc-памяти*, связанное с ней отношением *блокировка**.

Для того, чтобы исключить возможность реализации *sc-агентов*, которые могут внести изменения в конструкции, описывающие блокировки других *sc-агентов*, все элементы этих конструкций, в том числе, сам знак *структуры*,



= Пример использования блокировок

содержащей заблокированные sc-элементы (принадлежащей как множеству **полная блокировка**, так и любому другому типу блокировки) и связи отношения **блокировка***, связывающие эту структуру и конкретное действие в sc-памяти, добавляются в **полную блокировку**, соответствующую данному действию в sc-памяти. Таким образом, каждой **полной блокировке** соответствует петля принадлежности, связывающая ее знак с самим собой.

Каждая структура, принадлежащая множеству **блокировка на любое изменение** содержит sc-элементы, изменение (физическое удаление, добавление инцидентных sc-коннекторов, физическое удаление самих sc-элементов, изменение содержимого в случае файла) которых запрещено всем sc-агентам, кроме собственно sc-агента, выполняющего соответствующее данной структуре действие в sc-памяти, связанное с ней отношением **блокировка***. Однако не запрещен просмотр (чтение) этих sc-элементов любым sc-агентом.

Каждая структура, принадлежащая множеству **блокировка на удаление** содержит sc-элементы, удаление которых запрещено всем sc-агентам, кроме собственно sc-агента, выполняющего соответствующее данной структуре действие в sc-памяти, связанное с ней отношением **блокировка***. Однако не запрещен просмотр (чтение) этих sc-элементов любым sc-агентом, добавление инцидентных sc-коннекторов.

Рассмотрим принципы работы с блокировками:

- в каждый момент времени одному процессу в sc-памяти может соответствовать только одна блокировка каждого типа;
- в каждый момент времени одному процессу в sc-памяти может соответствовать только одна блокировка, установленная на некоторый конкретный sc-элемент;
- при завершении выполнения любого процесса в sc-памяти все установленные им блокировки автоматически снимаются;

- для повышения эффективности работы системы в целом каждый процесс должен в каждый момент времени блокировать минимально необходимое множество sc-элементов, снимая блокировку с каждого sc-элемента сразу же, как это становится возможным (безопасным);
- В случае когда в рамках *процесса в sc-памяти* явно выделяются более частные подпроцессы (при помощи отношений *temporalная часть**, *поддействие**, *декомпозиция действия** и т. д.), то каждый такой подпроцесс с точки зрения синхронизации выполнения рассматривается как самостоятельный процесс, которому в соответствие могут быть поставлены все необходимые блокировки.
 - все дочерние процессы в sc-памяти имеют доступ к блокировкам родительского процесса так же, как если бы эти были блокировки соответствующие каждому из таких дочерних процессов;
 - в свою очередь, родительский процесс не имеет какого-либо привилегированного доступа к sc-элементам, заблокированным дочерними процессами, и работает с ними так же, как любой другой процесс в sc-памяти. Исключение составляют sc-элементы, обозначающие сами дочерние процессы, поскольку родительский процесс должен иметь возможность управления дочерним, например, приостановки или прекращения их выполнения;
 - все дочерние процессы по отношению друг к другу работают так же, как и по отношению к любым другим процессам;
 - в случае, когда родительский процесс приостанавливает выполнение (становится *отложенным действием*), все его дочерние процессы также приостанавливают выполнение. В свою очередь, приостановка одного из дочерних процессов в общем случае не инициирует явно остановку всего родительского процесса и соответственно других дочерних.

Рассмотрим принципы работы с *полными блокировками*:

- если sc-элемент, инцидентный некоторому sc-коннектору, попадает в какую-либо полную блокировку, то сам этот sc-коннектор по умолчанию также считается заблокированным этой же блокировкой. Обратное в общем случае неверно, т. к. часть sc-коннекторов, инцидентных некоторому sc-элементу, может быть полностью заблокирована, при этом сам этот элемент заблокирован не будет. Такая ситуация типична, например, для sc-узлов, обозначающих классы понятий;
- каждый процесс в sc-памяти может свободно изменять или удалять любые sc-элементы, попадающие в полную блокировку, соответствующую этому процессу.

Принципы работы с *полными блокировками*, с одной стороны, наиболее просты, поскольку все процессы, кроме установившего такую блокировку, не имеют доступа к заблокированным sc-элементам и конфликты возникнуть не могут. С другой стороны, частое использование блокировок такого типа может привести к тому, что система не сможет использовать в полной мере имеющиеся у нее знания и давать неполные или даже некорректные ответы на поставленные вопросы.

Рассмотрим принципы работы с *блокировками на любое изменение и блокировками на удаление*:

- на один и тот же sc-элемент в один момент времени может быть установлена только одна блокировка одного типа, но разные процессы могут одновременно установить на один и тот же элемент блокировки двух разных типов. Это касается случая, когда первый процесс установил на некоторый sc-элемент блокировку на удаление, а второй процесс затем устанавливает блокировку на любое изменение. В других случаях возникает конфликт блокировок;
- установка блокировки любого типа также считается изменением, таким образом, если на некоторый sc-элемент была установлена блокировка на любое изменение, то другой процесс не сможет установить на этот же sc-элемент блокировку любого типа, пока первый процесс не снимет свою;
- если блокировка на удаление устанавливается на некоторый sc-коннектор, то по умолчанию та же блокировка устанавливается на инцидентные этому sc-коннектору sc-элементы, поскольку удаление этих элементов приведет к удалению этого коннектора.

процесс в sc-памяти

⇒ разбиение*:

Классификация процессов в sc-памяти с точки зрения синхронизации их выполнения

$$= \{ \begin{array}{l} \bullet \text{ действие поиска sc-элементов} \\ \bullet \text{ действие генерации sc-элементов} \\ \bullet \text{ действие удаления sc-элементов} \\ \bullet \text{ действие установки блокировки некоторого типа на некоторый sc-элемент} \\ \bullet \text{ действие снятия блокировки с некоторого sc-элемента} \end{array} \}$$

В некоторых случаях для того, чтобы обеспечить синхронизацию, необходимо объединять несколько элементарных действий над sc-памятью в одно неделимое действие (*транзакцию в sc-памяти*), для которого гарантируется, что ни один сторонний процесс не сможет прочитать или изменить участвующие в этом действии sc-элементы, пока действие не завершится. При этом, в отличие от ситуации с полной блокировкой, процесс, пытающийся получить

доступ к таким элементам, не продолжает выполнение так, как если бы этих элементов просто не было в sc-памяти, а ожидает завершения транзакции, после чего может выполнять с данными элементами любые действия согласно общим принципам синхронизации процессов. Проблема обеспечения транзакций не может быть решена на уровне SC-кода и требует реализации таких неделимых действий на уровне *ostis-платформы*.

В случае выполнения *действия поиска sc-элементов* все найденные и сохраненные в рамках какого-либо процесса sc-элементы попадают в соответствующую данному процессу *блокировку на любое изменение*. Таким образом, гарантируется целостность фрагмента базы знаний, с которым работает некоторый процесс в sc-памяти. При этом поиск и автоматическая установка такой блокировки должны быть реализованы как *транзакция в sc-памяти*.

Такой подход также позволяет избежать ситуации, когда один процесс заблокировал некоторый sc-элемент на любое изменение, а второй процесс пытается сгенерировать или удалить *sc-коннектор*, инцидентный данному *sc-элементу*. В таком случае второй процесс должен будет предварительно найти и заблокировать указанный *sc-элемент* на любое изменение, что вызовет конфликт блокировок (*взаимоблокировку**).

В случае генерации любого sc-элемента в рамках некоторого процесса он автоматически попадает в полную блокировку, соответствующую данному процессу. При этом генерация и автоматическая установка такой блокировки должны быть реализованы как *транзакция в sc-памяти*. При необходимости сгенерированные элементы могут быть удалены (т. е. их временное существование вообще никак не отразится на деятельности других процессов) или разблокированы в случае, когда сгенерирована информация, которая может иметь некоторую ценность в дальнейшем.

В случае если какой-либо процесс пытается установить блокировку любого типа на какой-либо sc-элемент, уже заблокированный каким-либо другим процессом, то, с одной стороны, блокировка не может быть установлена, пока другой процесс не разблокирует указанный sc-элемент; с другой стороны, для того чтобы обеспечить возможность поиска и устранения *взаимоблокировок*, необходимо явно указывать тот факт, что какой-либо процесс хочет получить доступ к какому-либо заблокированному другим процессом sc-элементу. Для того чтобы иметь возможность указать, какие процессы пытаются заблокировать уже заблокированный *sc-элемент*, предлагается наряду с отношением *блокировка** использовать отношение *планируемая блокировка**, полностью аналогичное отношению *блокировка**.

Описанный механизм регулирует также и процессы поиска, поскольку поиск и сохранение некоторого sc-элемента предполагает установку *блокировки на любое изменение*. Кроме того, следует учитывать, что на один sc-элемент *блокировка на любое изменение* может быть установлена после *блокировки на удаление*, соответствующей другому процессу. В этом случае использовать отношение *планируемые блокировки** нет необходимости.

Действие проверки наличия на некотором sc-элементе блокировки и в зависимости от результата проверки, установки блокировки или планируемой блокировки (с указанием приоритета при необходимости) должно быть реализовано как транзакция.

планируемая блокировка*
 \subseteq *блокировка**

Процесс, которому в соответствие поставлена *планируемая блокировка**, приостанавливает выполнение до тех пор, пока уже установленные блокировки не будут сняты, после чего *планируемая блокировка** становится реальной *блокировкой** и процесс продолжает выполнение в соответствии с общими правилами.

приоритет блокировки*
 \Rightarrow *область определения*:*
*планируемая блокировка**

В случае, когда на один и тот же sc-элемент планируют установить блокировку сразу несколько процессов, используется отношение *приоритет блокировки**, связывающее между собой пары отношения *планируемая блокировка**. Как правило, приоритет блокировки определяется тем, какой из процессов раньше попытался установить блокировку на рассматриваемый sc-элемент, хотя в общем случае приоритет может устанавливаться или меняться в зависимости от дополнительных критериев.

В случае попытки удаления некоторого sc-элемента некоторым процессом удаление может быть осуществлено только в случае, когда на данный sc-элемент не установлена (и не планируется) ни одна блокировка каким-либо другим процессом.

В других случаях необходимо обеспечить корректное завершение выполнения всех процессов, работающих с данным sc-элементом, и только потом удалить его физически.

Для реализации такой возможности каждому процессу в соответствие может быть поставлено множество удаляемых данным процессом sc-элементов.

Действие проверки наличия блокировок или планируемых блокировок на удаляемый sc-элемент и собственно его удаление или добавление во множество удаляемых sc-элементов для соответствующего процесса должно быть реализовано как транзакция.

удаляемые sc-элементы*

⇒ *первый домен*:*
процесс в sc-памяти

Sc-элементы, попавшие во множество удаляемых sc-элементов некоторого процесса в sc-памяти, доступны процессам, уже установившим (или планирующим установить) на эти sc-элементы блокировки ранее (до попытки его удаления), а для всех остальных процессов эти sc-элементы уже считаются удаленными. Процесс, пытающийся удалить sc-элемент, приостанавливает свое выполнение до того момента, пока все заблокировавшие и планирующие заблокировать данный sc-элемент процессы не разблокируют его. В общем случае один sc-элемент может входить во множества удаляемых элементов одновременно для нескольких процессов, в этом случае все такие процессы одновременно продолжат выполнение после снятия с этого sc-элемента всех блокировок. Если удаление пытается осуществить один из процессов, уже установивший на указанный sc-элемент блокировку, то алгоритм действий остается прежним – sc-элемент добавляется во множество удаляемых данным процессом sc-элементов, и будет физически удален, как только все остальные процессы, установившие на данный sc-элемент блокировки, снимут их.

Рассмотрим алгоритм снятия блокировки с некоторого sc-элемента:

- если на данный sc-элемент установлена одна или несколько *планируемых блокировок**, то первая из них по приоритету (или единственная) становится *блокировкой**, соответствующий ей процесс продолжает выполнение (становится настоящей сущностью); связка отношения приоритет выполнения, соответствовавшая удаленной связке отношения *планируемая блокировка** также удаляется, т. е. приоритет смещается на одну позицию;
- если *планируемых блокировок**, установленных на данный sc-элемент, нет, но он попадает во множество удаляемых sc-элементов для одного или нескольких процессов, то рассматриваемый sc-элемент физически удаляется, а приостановленные до его удаления процессы продолжают свое выполнение (становятся настоящими сущностями);
- если на данный sc-элемент не установлены планируемые блокировки и он не входит во множество удаляемых для какого-либо процесса, то блокировка просто снимается без каких-либо дополнительных изменений.

транзакция в sc-памяти

⇒ *разбиение*:*

- {
- поиск некоторой конструкции в sc-памяти и автоматическая установка блокировки на любое изменение на найденные sc-элементы
 - генерация некоторого sc-элемента и автоматическая установка на него полной блокировки
 - проверка наличия на некотором sc-элементе блокировки и в зависимости от результата проверки установка блокировки или планируемой блокировки
 - проверка наличия блокировок или планируемых блокировок на удаляемый sc-элемент и собственно его удаление или добавление во множество удаляемых sc-элементов для соответствующего процесса
 - снятие блокировки с заданного sc-элемента и при необходимости установка первой по приоритету планируемой блокировки или удаление данного sc-элемента, если он входит во множество удаляемых sc-элементов для некоторого процесса
 - поиск подпроцессов процесса и добавление их во множество отложенных действий в случае добавления самого процесса в данное множество
 - поиск подпроцессов процесса и удаление их из множества отложенных действий в случае удаления самого процесса из данного множества
- }

При реализации *абстрактных программных sc-агентов* на языке *SCP*, соблюдение всех принципов синхронизации соответствующих этим sc-агентам процессов обеспечивается на уровне *sc-агентов интерпретации scp-программ*, т. е. средствами *ostis-платформы*. При реализации *абстрактных программных sc-агентов* на уровне платформы, соблюдение всех принципов синхронизации возлагается, во-первых, непосредственно на разработчика агентов, во-вторых, – на разработчика платформы. Так, например, платформа может предоставлять доступ к хранимым в sc-памяти элементам через некоторый программный интерфейс, уже учитывающий принципы работы с блокировками, что избавит разработчика агентов от необходимости учитывать все эти принципы вручную.

Кроме того, выделяется ряд специфичных принципов работы *абстрактных программных sc-агентов*, реализованных на языке *SCP*:

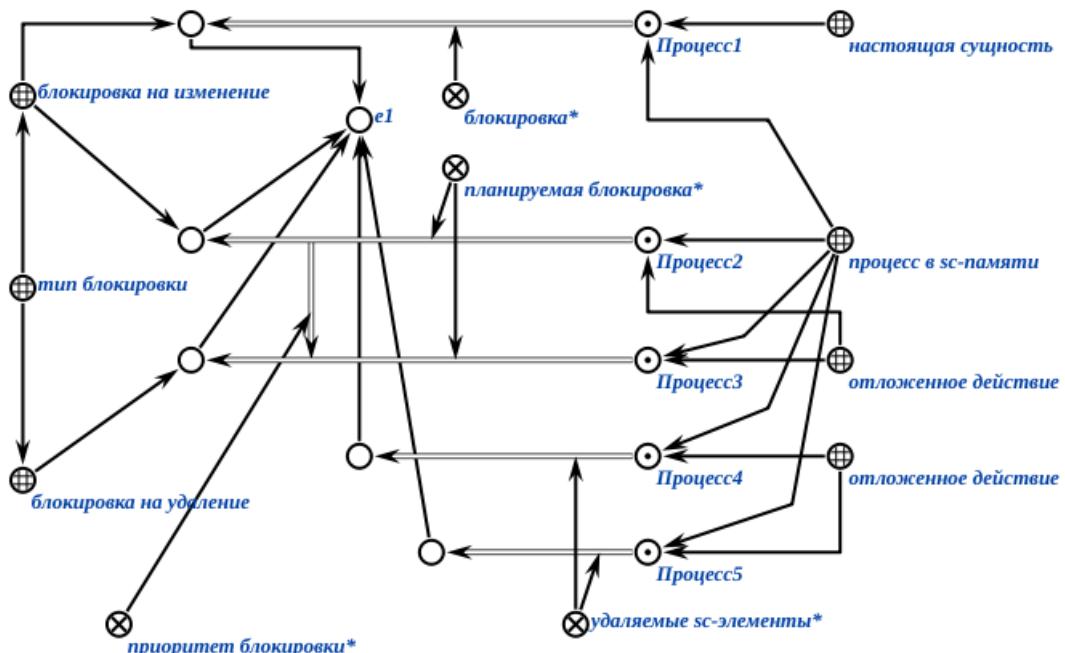
- в результате появления в sc-памяти некоторой конструкции, удовлетворяющей условию инициирования какого-либо *абстрактного sc-агента*, реализованного при помощи Языка *SCP*, в sc-памяти генерируется

и инициируется *scp-процесс*. В качестве шаблона для генерации используется *агентная scp-программа*, соответствующая данному *абстрактному sc-агенту*.

- каждый такой *scp-процесс*, соответствующий некоторой *агентной scp-программе*, может быть связан с набором структур, описывающих блокировки различных типов. Таким образом, синхронизация взаимодействия параллельно выполняемых *scp-процессов* осуществляется так же, как и в случае любых других *действий в sc-памяти*.
- несмотря на то что каждый *scp-оператор* представляет собой атомарное действие в sc-памяти, являющееся под действием в рамках всего *scp-процесса*, блокировки, соответствующие одному оператору, не вводятся, чтобы избежать громоздкости и избытка дополнительных системных конструкций, создаваемых при выполнении некоторого *scp-процесса*. Вместо этого используются блокировки, общие для всего *scp-процесса*. Таким образом, *агенты интерпретации scp-программ* работают только с учетом блокировок, общих для всего интерпретируемого *scp-процесса*.
- процессы, описывающие деятельность агентов интерпретации *scp-программ*, как правило, не создаются, следовательно, и не вводятся соответствующие им блокировки. Поскольку такие агенты работают с уникальным *scp-процессом* и их число ограничено и известно, то использование блокировок для их синхронизации не требуется.
- в случае приостановки *scp-процесса* (добавления его во множество *отложенных действий*) в соответствии с общими правилами синхронизации все его дочерние процессы также должны быть приостановлены. В связи с этим все *scp-операторы*, которые в этот момент являются *настоящими сущностями*, становятся *отложенными действиями*.
- во избежание нежелательных изменений в самом теле *scp-процесса*, вся конструкция, сгенерированная на основе некоторой *scp-программы* (весь *sc-текст*, описывающий декомпозицию *scp-процесса* на *scp-операторы*), должна быть добавлена в *полную блокировку*, соответствующую данному *scp-процессу*.
- при необходимости разблокировать или заблокировать некоторую конструкцию каким-либо типом блокировки используются соответствующие *scp-операторы* класса *scp-оператор управления блокировками*.
- после завершения выполнения некоторого *scp-процесса* его текст, как правило, удаляется из *sc-памяти*, а все заблокированные конструкции освобождаются (разрушаются знаки структур, обозначавших блокировки).
- как правило, частный класс *действий*, соответствующий конкретной *scp-программе*, явно не вводится, а используется более общий класс *scp-процесс*, за исключением тех случаев, когда введение специального класса *действий* необходимо по каким-либо другим соображениям.

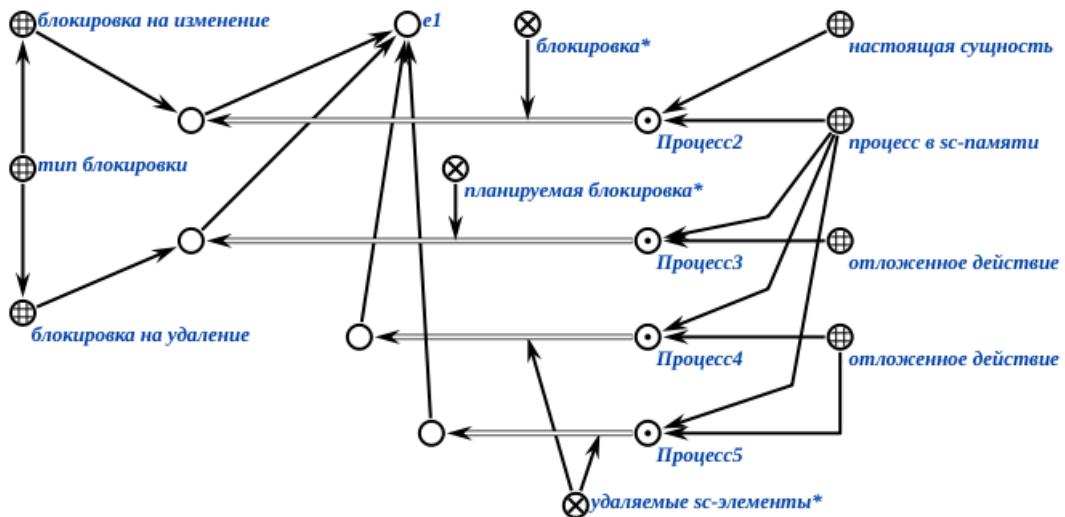
В общем случае весь механизм блокировок может описываться как на уровне SC-кода (для повышения уровня платформенной независимости), так и при необходимости может быть реализован на уровне *ostis-платформы*, например для повышения производительности. Для этого каждому выполняемому в sc-памяти процессу на нижнем уровне может быть поставлена в соответствие некая уникальная таблица, в каждый момент времени содержащая перечень заблокированных элементов с указанием типа блокировки.

Рассмотрим пример применения описанного механизма.



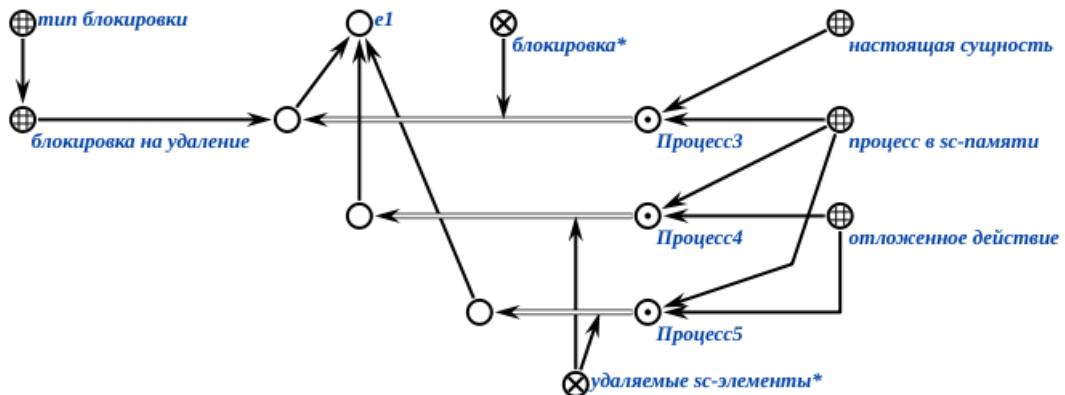
= Пример использования планируемых блокировок

В данном примере *Процесс1* непосредственно работает с sc-элементом *e1*. *Процесс2* и *Процесс3* планируют установить блокировку на любое изменение и блокировку на удаление соответственно, причем *Процесс2* попытался установить свою блокировку раньше, чем *Процесс3*, поэтому согласно направлению связки отношения *приоритет блокировки**, его блокировка будет установлена раньше. *Процесс4* и *Процесс5* ожидают снятия всех блокировок и планируемых блокировок, после чего *e1* будет удален и *Процесс1* и *Процесс2* продолжат свое выполнение. Никакие другие планируемые блокировки установлены быть уже не могут, поскольку *e1* попал во множество удаляемых sc-элементов как минимум одного процесса и, в соответствии с изложенными выше правилами, все остальные процессы кроме *Процесс1-Процесс5*, уже не смогут получить доступ к этому sc-элементу. Выполняемый процесс принадлежит множеству настоящая сущность, приостановленные – множеству отложенное действие.



= Пример использования планируемых блокировок (продолжение)

После того как *Процесс1* разблокировал sc-элемент *e1*, этот элемент будет заблокирован *Процессом2*, и *Процесс2* продолжит выполнение. *Планируемая блокировка**, установленная *Процессом2*, становится обычной блокировкой*.

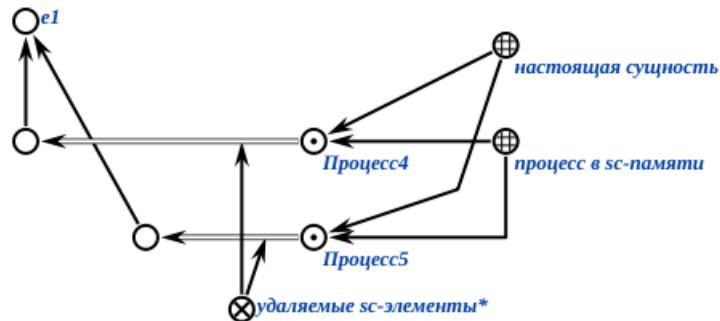


= Пример использования блокировки на удаление

После того как *Процесс2* разблокировал sc-элемент *e1*, этот элемент будет заблокирован *Процессом3*, и *Процесс3* продолжит выполнение.

Когда все процессы снимут блокировки с sc-элемента *e1*, он может быть физически удален и *Процесс4* и *Процесс5* продолжат выполнение.

В зависимости от конкретных типов блокировок установленных параллельно выполняемыми процессами на некоторые sc-элементы и того, какие конкретно действия с этими sc-элементами предполагается выполнить далее в рамках выполнения этих процессов, возможны ситуации взаимоблокировки, когда каждый из указанных процессов будет ожидать снятия блокировки вторым процессом с нужного sc-элемента, не снимая при этом установленной им самим блокировки с sc-элемента, доступ к которому необходим второму процессу.



= Удаляемые sc-элементы

В случае когда хотя бы одна из блокировок является *полной блокировкой*, ситуация взаимоблокировки возникнуть не может, поскольку *sc-элементы*, попавшие в *полную блокировку* некоторого *scр-процесса*, не доступны другим *scр-процессам* даже для чтения и, таким образом, остальные *scр-процессы* будут работать так, как будто заблокированные *sc-элементы* просто отсутствуют в текущем состоянии *sc-памяти*.

В случаях, когда ни одна из установленных блокировок не является *полной блокировкой*, возможно появление взаимоблокировок.

Устранение взаимоблокировки невозможно без вмешательства специализированного *sc-метаагента*, который имеет право игнорировать блокировки, установленные другими процессами.

В общем случае проблема конкретной взаимоблокировки может быть решена путем выполнения специализированным *sc-метаагентом* следующих шагов:

- откат нескольких операций, выполненных одним из участвующих в взаимоблокировке процессов настолько шагов назад, насколько это необходимо для того, чтобы второй процесс получил доступ к необходимым *sc-элементам* и смог продолжить выполнение;
- ожидание выполнения второго процесса вплоть до завершения или до снятия им всех блокировок с *sc-элементов*, доступ к которым необходимо получить первому процессу;
- повторное выполнение в рамках первого процесса отмененных операций и продолжение его выполнения, но уже с учетом изменений в памяти, внесенных вторым процессом.

Для *sc-метаагентов* все *sc-элементы*, в том числе описывающие блокировки, планируемые блокировки и т. д. полностью эквивалентны между собой с точки зрения доступа к ним, т. е. любой *sc-метаагент* имеет доступ к любым *sc-элементам*, даже попавшим в полную блокировку для какого-либо другого процесса. Это необходимо для того, чтобы *sc-метаагенты* смогли выявлять и устранять различные проблемы, например, описанную выше проблему взаимоблокировки.

Таким образом, проблема синхронизации деятельности *sc-метаагентов* требует введения дополнительных правил.

Указанную проблему разделим на две более частные:

- обеспечение синхронизации деятельности *sc-метаагентов* между собой;
- обеспечение синхронизации деятельности *sc-метаагентов* и *программных sc-агентов*.

Первую проблему предлагается решить за счет запрета параллельного выполнения *sc-метаагентов*. Таким образом, в каждый момент времени в рамках одной *ostis-системы* может существовать только один процесс, соответствующий *sc-метаагенту* и являющийся *настоящей сущностью*.

Вторую проблему предлагается решить за счет введения дополнительных привилегий для *sc-метаагентов* при обращении к какому-либо *sc-элементу*. Для этого достаточно одного правила:

Если некоторый *sc-элемент* стал использоваться в рамках процесса, соответствующего *sc-метаагенту* (например, стал элементом хотя бы одного *scр-оператора*, входящего в данный процесс), то все процессы, в блокировке соответствующие которым попадает указанный *sc-элемент*, становятся отложенными действиями (приостанавливают выполнение). Как только указанный *sc-элемент* перестает использоваться в рамках процесса, соответствующего *sc-метаагенту*, все приостановленные по этой причине процессы продолжают выполнение.

Рассмотренные ограничения не ухудшают производительность *ostis-системы* существенно, поскольку *sc-метаагенты* предназначены для решения достаточно узкого класса задач, которые, как показал опыт практической разработки прототипов различных *ostis-систем*, возникают достаточно редко.

Стоит отметить, что возможна ситуация, при которой выполнение некоторого процесса в *sc-памяти* прервано по причине возникновения какой-либо ошибки. В таком случае существует вероятность того, что блокировка, установленная данным процессом не будет снята до тех пор, пока этого не сделает *sc-метаагент*, обнаруживший подобную

ситуацию. Однако указанная проблема на уровне sc-модели может быть решена лишь частично, для случаев, когда ошибка возникает при интерпретации scp-программы, отслеживается scp-интерпретатором и в памяти формируется соответствующая конструкция, сообщающая о проблеме sc-метаагенту. Случай, когда возникла ошибка на уровне scp-интерпретатора или sc-хранилища, должны рассматриваться на уровне *ostis*-платформы.

§ 3.2.5. Базовый язык программирования *ostis*-систем

⇒ подраздел*:

- Пункт 3.2.5.1. Денотационная семантика Базового языка программирования *ostis*-систем
- Пункт 3.2.5.2. Операционная семантика Базового языка программирования *ostis*-систем

⇒ ключевой знак*:

- Язык SCP
- Абстрактная scp-машина

⇒ ключевое понятие*:

- scp-оператор
- параметр scp-программы'

Как было показано в § 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты, выделение Базового языка программирования для *ostis*-систем позволяет обеспечить четкое разделение уровня методов и соответственно, навыков *ostis*-системы, которые могут быть полностью описаны на уровне базы знаний и более низкоуровневых навыков, обеспечивающих интерпретацию указанных навыков более высокого уровня. Другими словами, выделение такого языка позволяет обеспечить платформенную независимость *ostis*-систем, как в случае программной реализации *ostis*-платформы, так и в случае ассоциативного семантического компьютера.

В качестве базового языка для описания программ обработки текстов SC-кода предлагается Язык SCP. Язык SCP – это графовый язык процедурного программирования, предназначенный для эффективной обработки sc-текстов. Язык SCP является языком параллельного асинхронного программирования.

Язык SCP

:= часто используемый sc-идентификатор*:
[scp-программа]

Языком представления данных для текстов Языка SCP (scp-программ) является SC-код и, соответственно, любые варианты его внешнего представления. Язык SCP сам построен на основе SC-кода, вследствие чего scp-программы сами по себе могут входить в состав обрабатываемых данных для scp-программ, в т.ч. по отношению к самим себе. Таким образом, язык SCP предоставляет возможность построения реконфигурируемых программ. Однако для обеспечения возможности реконфигурирования программы непосредственно в процессе ее интерпретации необходимо на уровне интерпретатора Языка SCP (Абстрактной scp-машины) обеспечить уникальность каждой исполняемой копии исходной программы. Такую исполняемую копию, сгенерированную на основе scp-программы, будем называть scp-процессом. Включение знака некоторого действия в sc-памяти во множество scp-процессов гарантирует тот факт, что в декомпозиции данного действия будут присутствовать только знаки элементарных действий (scp-операторов), которые может интерпретировать реализация Абстрактной scp-машины (интерпретатора scp-программ).

Язык SCP рассматривается как ассемблер для семантического компьютера.

Абстрактная scp-машина

∈ scp-машина
 ⇐ обобщенная модель*:
 scp-интерпретатор

Базовая модель обработки sc-текстов включает в себя Предметную область Базового языка программирования *ostis*-систем, то есть описание синтаксиса и денотационной семантики Языка SCP, а также описание Абстрактной scp-машины, которая является моделью scp-интерпретатора, который должен являться частью *ostis*-платформы (хотя в общем случае могут существовать варианты платформы, не содержащие такого интерпретатора, что, однако, не позволит использовать достоинства предлагаемой базовой модели).

Рассмотрим ключевые особенности и достоинства Базовой модели обработки sc-текстов:

- Тексты программ Языка SCP записываются при помощи тех же унифицированных семантических сетей, что и обрабатываемая информация, таким образом, можно сказать, что Синтаксис языка SCP на базовом уровне совпадает с Синтаксисом SC-кода.

- Подход к интерпретации *scp-программы* предполагает создание при каждом вызове *scp-программы* уникального *scp-процесса*.
- Одновременно в общей памяти могут выполняться несколько независимых *sc-агентов*, при этом разные копии *sc-агентов* могут выполняться на разных серверах, за счет распределенной реализации *ostis-платформы*. Более того, Язык *SCP* позволяет осуществлять параллельные асинхронные вызовы подпрограмм с последующей синхронизацией, и даже параллельно выполнять операторы в рамках одной *scp-программы*.
- Перенос *sc-агента* из одной системы в другую заключается в простом переносе фрагмента базы знаний, без каких-либо дополнительных операций, зависящих от платформы интерпретации.
- Тот факт, что спецификации *sc-агентов* и их программы могут быть записаны на том же языке, что и обрабатываемые знания, существенно сокращает перечень специализированных средств, предназначенных для проектирования машин обработки знаний, и упрощает их разработку за счет использования более универсальных компонентов.
- Тот факт, что для интерпретации *scp-программы* создается соответствующий ей уникальный *scp-процесс*, позволяет по возможности оптимизировать план выполнения перед его реализацией и даже непосредственно в процессе выполнения без потенциальной опасности испортить общий универсальный алгоритм всей программы. Более того, такой подход к проектированию и интерпретации программ позволяет говорить о возможности создания самореконфигурируемых программ.

scp-программа

С *программа в sc-памяти*
 Д *агентная scp-программа*

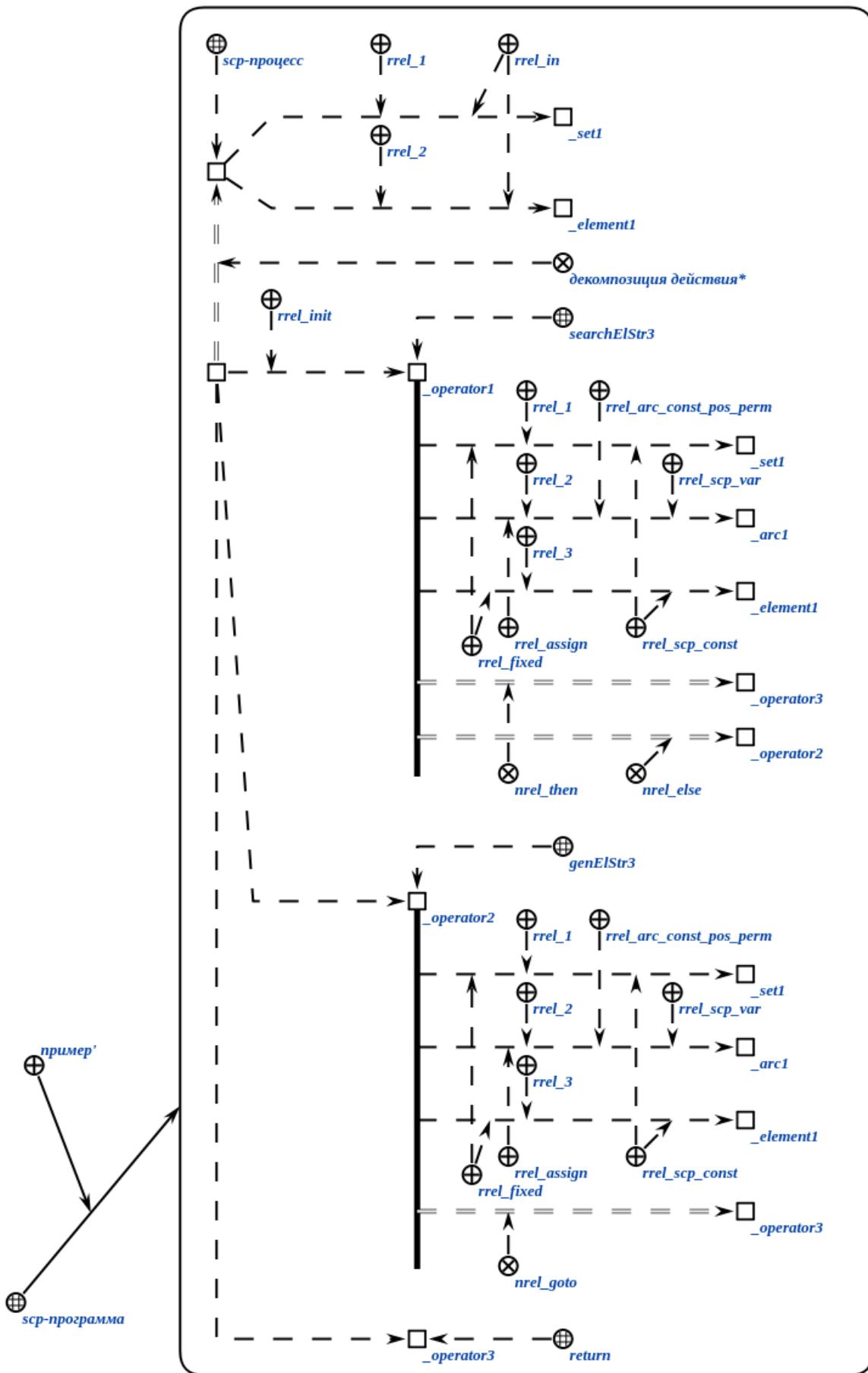
Каждая *scp-программа* представляет собой обобщенную *sc-структуру*, описывающую один из вариантов декомпозиции действий некоторого класса, выполняемых в *sc-памяти*. Знак *sc-переменной*, соответствующей конкретному декомпозируемому действию является в рамках *scp-программы* *ключевым sc-элементом*¹. Также явно указывается принадлежность данного знака множеству *scp-процессов*.

Принадлежность некоторого действия множеству *scp-процессов* гарантирует тот факт, что в декомпозиции данного действия будут присутствовать только знаки элементарных действий (*scp-операторов*), которые может интерпретировать реализация абстрактной *scp-машины*.

Таким образом, каждая *scp-программа* описывает в обобщенном виде декомпозицию некоторого *scp-процесса* на взаимосвязанные *scp-операторы*, с указанием, при их наличии, аргументов для данного *scp-процесса*.

По сути каждая *scp-программа* представляет собой описание последовательности элементарных операций, которые необходимо выполнить над семантической сетью, чтобы выполнить более сложное действие некоторого класса.

На рисунке [Пример scp-программы](#) приведен пример простой *scp-программы*. В приведенном примере показана *scp-программа*, состоящая из трех *scp-операторов*. Данная программа проверяет, содержится ли в заданном множестве (первый параметр) заданный элемент (второй параметр), и, если нет, то добавляет его в это множество.



= Пример scp-программы

Агентные scp-программы представляют собой частный случай *scp-программ* вообще, однако заслуживают отдельного рассмотрения, поскольку используются наиболее часто. *Scp-программы* данного класса представляют собой реализации программ агентов обработки знаний, и имеют жестко фиксированный набор параметров. Каждая такая программа имеет ровно два *in-параметра*¹. Значение первого параметра является знаком бинарной ориентированной пары, являющейся вторым компонентом связки отношения *первичное условие инициирования** для абстрактного *sc-агента*, в множество *программ sc-агента** которого входит рассматриваемая *агентная scp-программа*, и, по сути, описывает класс событий, на которые реагирует указанный *sc-агент*.

Значением второго параметра является *sc-элемент*, с которым непосредственно связано событие, в результате возникновения которого был инициирован соответствующий *sc-агент*, т.е., например, сгенерированная либо удаляемая *sc-дуга* или *sc-ребро*.

Рассмотрим принципы реализации *абстрактных sc-агентов, реализуемых на Языке SCP*:

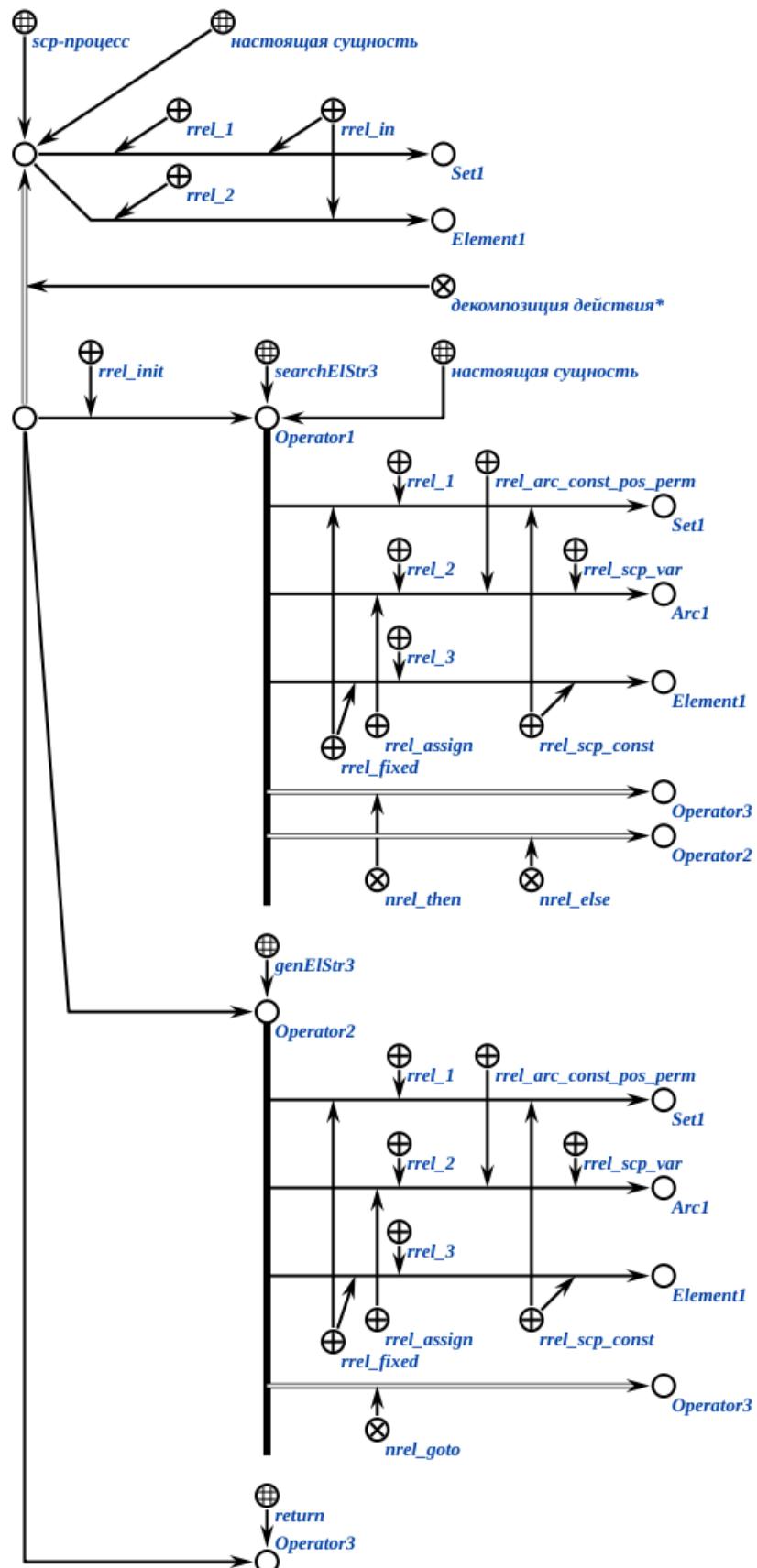
- общие принципы организации взаимодействия *sc-агентов* и пользователей *ostis-системы* через общую *sc-память*;
- в результате появления в *sc-памяти* некоторой конструкции, удовлетворяющей условию инициирования какого-либо *абстрактного sc-агента*, реализованного при помощи *Языка SCP*, в *sc-памяти* генерируется и инициируется *scp-процесс*. В качестве шаблона для генерации используется *агентная scp-программа*, указанная во множестве программ соответствующего *абстрактного sc-агента*;
- каждый такой *scp-процесс*, соответствующий некоторой *агентной scp-программе*, может быть связан с набором структур, описывающих блокировки различных типов. Таким образом, синхронизация взаимодействия параллельно выполняемых *scp-процессов* осуществляется так же, как и в случае любых других *действий в sc-памяти*;
- В рамках *scp-процесса* могут создаваться дочерние *scp-процессы*, однако синхронизация между ними при необходимости осуществляется посредством введения дополнительных внутренних блокировок. Таким образом, каждый *scp-процесс* с точки зрения *процессов в sc-памяти* является атомарным и законченным актом деятельности некоторого *sc-агента*;
- во избежание нежелательных изменений в самом теле *scp-процесса*, вся конструкция, сгенерированная на основе некоторой *scp-программы* (весь текст *scp-процесса*), должна быть добавлена в *полную блокировку*, соответствующую данному *scp-процессу*;
- все конструкции, сгенерированные в процессе выполнения *scp-процесса*, автоматически попадают в *полную блокировку*, соответствующую данному *scp-процессу*. Дополнительно следует отметить, что знак самой этой структуры и вся метаинформация о ней также включаются в эту структуру;
- при необходимости можно вручную разблокировать или заблокировать некоторую конструкцию каким-либо типом блокировки, используя соответствующие *scp-операторы* класса *scp-оператор управления блокировками*;
- после завершения выполнения некоторого *scp-процесса* его текст как правило, удаляется из *sc-памяти*, а все заблокированные конструкции освобождаются (разрушаются знаки структур, обозначавших блокировки);
- несмотря на то, что каждый *scp-оператор* представляет собой атомарное *действие в sc-памяти*, дополнительные блокировки, соответствующие одному оператору не вводятся, чтобы избежать громоздкости и избытка дополнительных системных конструкций, создаваемых при выполнении некоторого *scp-процесса*. Вместо этого используются блокировки, общие для всего *scp-процесса*. Таким образом, агенты *Абстрактной scp-машины* при интерпретации *scp-операторов* работают только с учетом блокировок, общих для всего интерпретируемого *scp-процесса*;
- как правило, частный класс *действий*, соответствующий конкретной *scp-программе* явно не вводится, а используется более общий класс *scp-процесс*, за исключением тех случаев, когда введение специального класса *действий* необходимо по каким-либо другим соображениям.

Под *scp-процессом* понимается некоторое *действие в sc-памяти*, однозначно описывающее конкретный акт выполнения некоторой *scp-программы* для заданных исходных данных. Если *scp-программа* описывает алгоритм решения какой-либо задачи в общем виде, то *scp-процесс* обозначает конкретное действие, реализующее данный алгоритм для заданных входных параметров.

По сути, *scp-процесс* представляет собой уникальную копию, созданную на основе *scp-программы*, в которой каждой *sc-переменной*, за исключением *scp-переменных*¹, соответствует сгенерированная *sc-константа*.

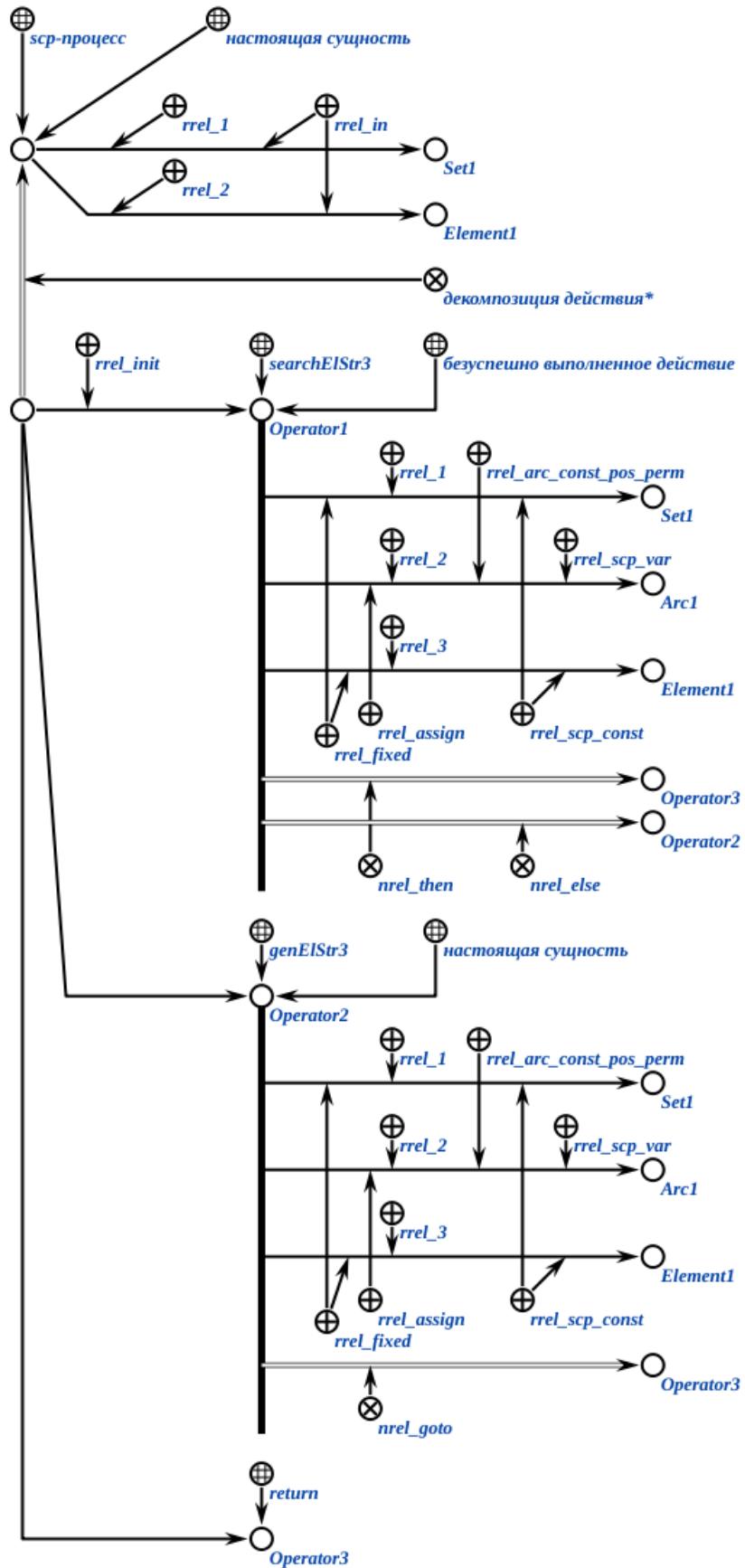
Принадлежность некоторого действия множеству *scp-процессов* гарантирует тот факт, что в декомпозиции данного действия будут присутствовать только знаки элементарных действий (*scp-операторов*), которые может интерпретировать реализация *Абстрактной scp-машины*.

Рассмотрим пример поэтапного выполнения *scp-процесса* (Рис. *Пример scp-процесса на начальной стадии выполнения – Пример scp-процесса: выполнение завершено*), соответствующего ранее рассмотренному примеру *scp-программы*. В приведенном примере последовательно показаны состояния *scp-процесса*, соответствующего *scp-программе*, добавляющей заданный элемент в заданное множество, если он там ранее не содержался. В примере предполагается, что рассматриваемый элемент (*Element1*) изначально не содержится во множестве (*Set1*).



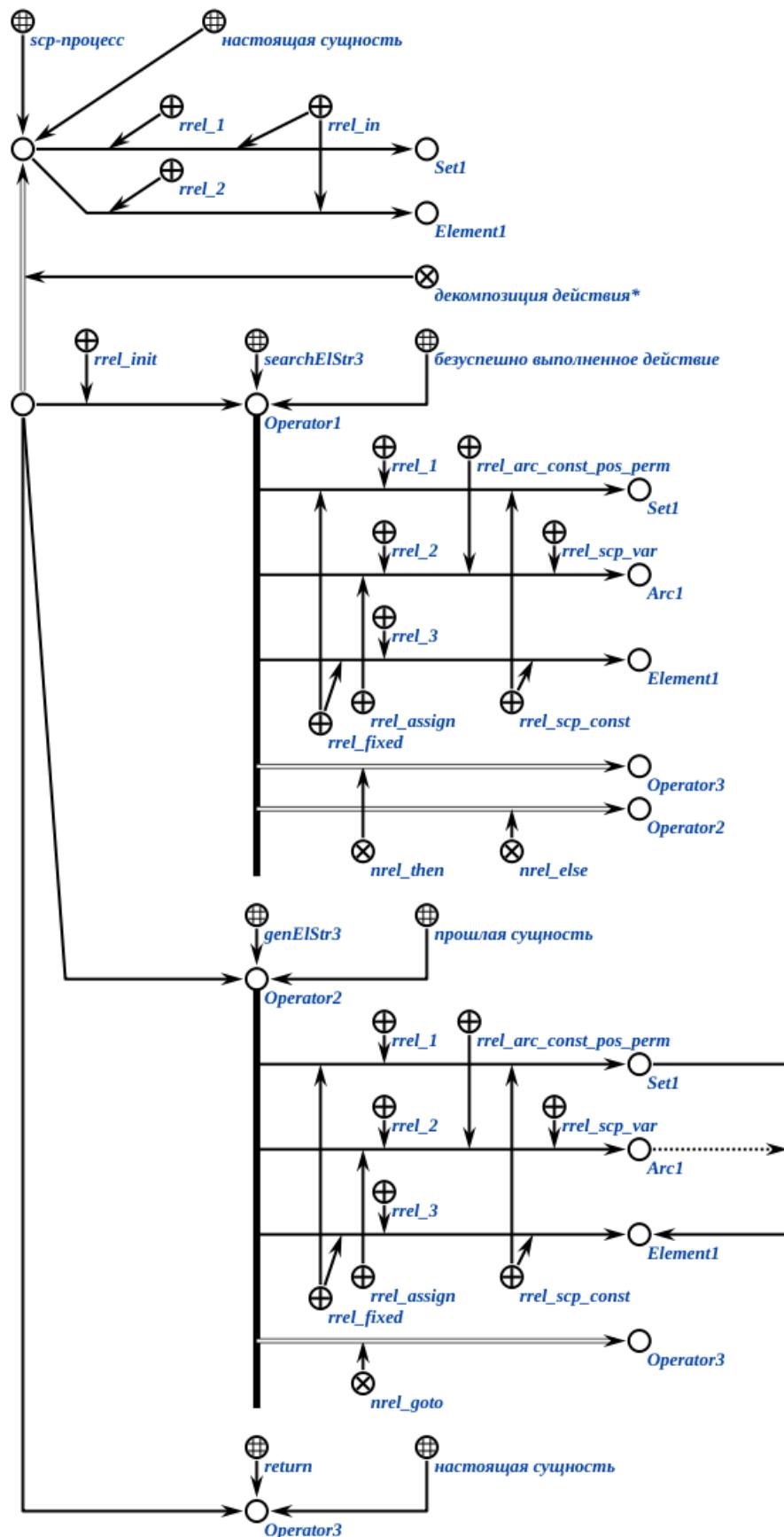
= Пример *scp-процесса на начальной стадии выполнения*

Осуществляется вызов *scp-программы*. Генерируется соответствующий *scp-процесс*. Происходит инициирование начального оператора *scp-процесса Operator1*.



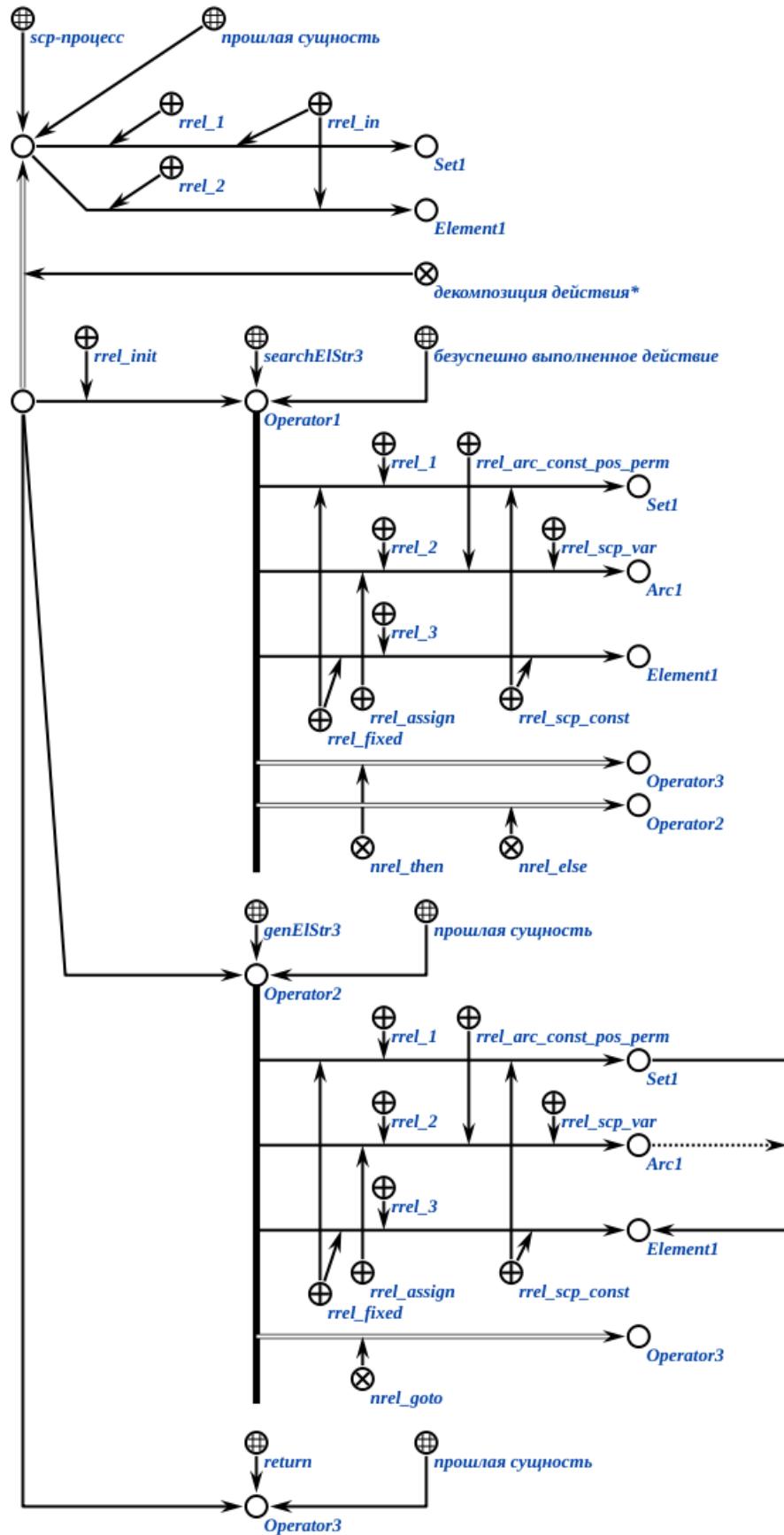
= Пример scp-процесса: безуспешно выполнен оператор поиска

Оператор *Operator1* оказался безуспешно выполненным. Производится инициализация scp-оператора генерации трёхэлементной конструкции *Operator2*.



= Пример *scp-процесса*: выполнен оператор генерации, элемент добавлен во множество

Оператор *Operator2* выполнился. Производится инициирование *scp-оператора завершения выполнения программы Operator3*.



= Пример *scp-процесса: выполнение завершено*

Оператор *Operator3* выполнился. Выполнение *scp-процесса* завершается.

scp-оператор

\subset действие в sc-памяти
 \Leftarrow семейство подмножеств*: атомарный тип scp-оператора

Каждый **scp-оператор** представляет собой некоторое элементарное действие в sc-памяти. Аргументы **scp-оператора** будем называть операндами. Порядок операндов указывается при помощи соответствующих ролевых отношений ($1'$, $2'$, $3'$ и так далее). Операнд, помеченный ролевым отношением $1'$, будем называть первым операндом, помеченный ролевым отношением $2'$ – вторым операндом, и т.д. Тип и смысл каждого операнда также уточняется при помощи различных подклассов отношения **scp-операнд**'. В общем случае операндом может быть любой sc-элемент, в том числе, знак какой-либо scp-программы, в том числе самой программы, содержащей данный оператор.

Каждый **scp-оператор** должен иметь один и более операнд, а также указание того **scp-оператора** (или нескольких), который должен быть выполнен следующим. Исключение их данного правила составляет **scp-оператор завершения выполнения программы**, который не содержит ни одного операнда и после выполнения которого никакие **scp-операторы** в рамках данной программы выполняться не могут.

Каждый **атомарный тип scp-оператора** представляет собой класс **scp-операторов**, который не разбивается на более частные, и, соответственно, интерпретируется реализацией *Абстрактной scp-машины*.

начальный оператор'
 $\subset 1'$

Ролевое отношение **начальный оператор'** указывает в рамках декомпозиции соответствующего **scp-программе scp-процесса** те **scp-операторы**, которые должны быть выполнены в первую очередь, т.е. те, с которых собственно начинается выполнение **scp-процесса**.

параметр scp-программы'
 \subset аргумент действия'
 \Rightarrow разбиение*:
{• *in-параметр'*
• *out-параметр'*
}

Ролевое отношение **параметр scp-программы'** связывает знак соответствующего **scp-программе scp-процесса** с его аргументами.

Параметры типа *in-параметр'* хоть и соответствуют *переменным scp-программы'*, не могут менять значение в процессе ее интерпретации. Фиксированное значение переменной устанавливается при создании уникальной копии **scp-программы** (**scp-процесса**) для ее интерпретации, и, таким образом, соответствующая **scp-переменная'** на момент начала ее интерпретации становится **scp-константой'** в рамках каждого **scp-оператора**, в котором встречалась данная **scp-переменная'**. Использование *in-параметров* можно рассматривать по аналогии с использованием варианта механизма передачи по значению в традиционных языках программирования, с тем условием, что значение локальной переменной в рамках дочерней программы не может быть изменено.

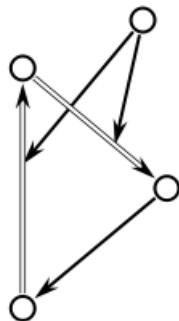
Параметры типа *out-параметр'* соответствуют *переменным scp-программы'* и обладают всеми теми же соответствующими свойствами. Чаще всего предполагается, что значение данного параметра необходимо родительской **scp-программе**, содержащей оператор вызова текущей **scp-программы**. При этом на момент начала интерпретации в качестве параметра дочернему процессу передается непосредственно узел, обозначающий переменную (а точнее, ее уникальную копию в рамках процесса) родительского процесса. Указанная переменная может при необходимости иметь значение, либо не иметь. После завершения и во время интерпретации дочернего процесса родительский процесс по-прежнему может работать с переменной, переданной в качестве *out-параметра'*, при необходимости просматривая или изменяя ее значение. Использование *out-параметра* можно рассматривать по аналогии с использованием механизма передачи по ссылке в традиционных языках программирования.

Рассмотрим классификацию sc-конструкций с точки зрения *Базовой модели обработки sc-текстов*.

sc-конструкция
 \Rightarrow разбиение*:
Классификация sc-конструкций с точки зрения Базовой модели обработки sc-текстов
= {• sc-конструкция нестандартного вида
• sc-конструкция стандартного вида
}
 \Rightarrow разбиение*:
{• одноЗлементная sc-конструкция

- трехэлементная sc-конструкция
 - пятиэлементная sc-конструкция
- }

Каждая sc-конструкция нестандартного вида состоит из произвольного количества sc-элементов произвольного типа (Рис. [Пример пятиэлементной sc-конструкции в SCg-коде](#)).



= Пример sc-конструкции нестандартного вида

В свою очередь, каждый элемент sc-конструкции стандартного вида имеет свою условную строго фиксированную позицию в рамках этой sc-конструкции (первый элемент, второй элемент и т. д.). В зависимости от указанной позиции вводятся дополнительные ограничения на тип соответствующего sc-элемента.

Каждая одноэлементная sc-конструкция состоит из одного sc-элемента произвольного типа (Рис. [Пример одноэлементных sc-конструкций в SCg-коде](#)).



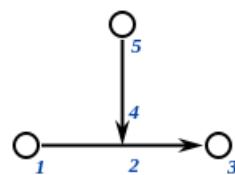
= Пример одноэлементных sc-конструкций в SCg-коде

Каждая трехэлементная sc-конструкция состоит из трех sc-элементов (Рис. [Пример трехэлементной sc-конструкции в SCg-коде](#)). Второй элемент всегда является sc-коннектором, остальные элементы могут быть произвольного типа.



= Пример трехэлементной sc-конструкции в SCg-коде

Каждая пятиэлементная sc-конструкция состоит из пяти sc-элементов (Рис. [Пример пятиэлементной sc-конструкции в SCg-коде](#)). Второй и четвертый элементы обязательно являются sc-коннекторами, остальные элементы могут быть произвольного типа.



= Пример пятиэлементной sc-конструкции в SCg-коде

Пункт 3.2.5.1. Денотационная семантика Базового языка программирования ostis-систем

⇒ *ключевое понятие**:

- *scp-оператор*
- *scp-операнд'*

scp-оператор

⊂ *действие в sc-памяти*

⇐ *семейство подмножеств**:

автомарный тип scp-оператора

⇒ *разбиение**:

- *scp-оператор генерации конструкций*

 ⇒ *разбиение*:

- {• *scp-оператор генерации конструкции по произвольному образцу*
- *scp-оператор генерации пятиэлементной конструкции*
- *scp-оператор генерации трехэлементной конструкции*
- *scp-оператор генерации одноэлементной конструкции*

 }

- *scp-оператор ассоциативного поиска конструкций*

 ⇒ *разбиение*:

- {• *scp-оператор поиска конструкции по произвольному образцу*
- *scp-оператор поиска пятиэлементной конструкции с формированием множеств*
- *scp-оператор поиска трехэлементной конструкции с формированием множеств*
- *scp-оператор поиска пятиэлементной конструкции*
- *scp-оператор поиска трехэлементной конструкции*

 }

- *scp-оператор удаления конструкций*

 ⇒ *разбиение*:

- {• *scp-оператор удаления множества элементов трехэлементной конструкции*
- *scp-оператор удаления одноэлементной конструкции*
- *scp-оператор удаления пятиэлементной конструкции*
- *scp-оператор удаления трехэлементной конструкции*

 }

- *scp-оператор проверки условий*

 ⇒ *разбиение*:

- {• *scp-оператор сравнения числовых содержимых файлов*
- *scp-оператор проверки равенства числовых содержимых файлов*
- *scp-оператор проверки совпадения значений operandов*
- *scp-оператор проверки наличия содержимого у файла*
- *scp-оператор проверки наличия значения у переменной*
- *scp-оператор проверки типа sc-элемента*

 }

- *scp-оператор управления значениями operandов*

 ⇒ *разбиение*:

- {• *scp-оператор удаления значения переменной*
- *scp-оператор присваивания значения переменной*

 }

- *scp-оператор управления scp-процессами*

 ⇒ *разбиение*:

- {• *scp-оператор удаления значения переменной*
- *scp-оператор завершения выполнения программы*
- *конъюнкция предшествующих scp-операторов*
- *scp-оператор ожидания завершения выполнения множества scp-программ*
- *scp-оператор ожидания завершения выполнения scp-программы*
- *scp-оператор асинхронного вызова подпрограммы*

 }

- *scp-оператор управления событиями*

 ▷ *scp-оператор ожидания события*

- *scp-оператор обработки содержимых файлов*

 ⇒ *разбиение*:

- {• *scp-оператор вычисления арксинуса числового содержимого файла*

- *scp-оператор вычисления арккосинуса числового содержимого файла*
- *scp-оператор деления числовых содержимых файлов*
- *scp-оператор умножения числовых содержимых файлов*
- *scp-оператор вычитания числовых содержимых файлов*
- *scp-оператор сложения числовых содержимых файлов*
- *scp-оператор вычисления тангенса числового содержимого файла*
- *scp-оператор вычисления косинуса числового содержимого файла*
- *scp-оператор вычисления синуса числового содержимого файла*
- *scp-оператор вычисления логарифма числового содержимого файла*
- *scp-оператор возвведения числового содержимого файла в степень*
- *scp-оператор удаления содержимого файла*
- *scp-оператор копирования содержимого файла*
- *scp-оператор нахождения остатка от деления числовых содержимых файлов*
- *scp-оператор нахождения целой части от деления числовых содержимых файлов*
- *scp-оператор вычисления арктангенса числового содержимого файла*
- *scp-оператор перевода в верхний регистр строкового содержимого файла*
- *scp-оператор перевода в верхний регистр строкового содержимого файла*
- *scp-оператор замены определенной части строкового содержимого файла на содержимое указанной файла*
- *scp-оператор проверки совпадения конца строкового содержимого файла со строковым содержимым другого файла*
- *scp-оператор проверки совпадения начальной части строкового содержимого файла со строковом содержимым другого файла*
- *scp-оператор получения части строкового содержимого файла по индексам*
- *scp-оператор поиска строкового содержимого файла в строковом содержимом другого файла*
- *scp-оператор вычисления длины строкового содержимого файла*
- *scp-оператор разбиения строки на подстроки*
- *scp-оператор лексикографического сравнения строковых содержимых файлов*
- *scp-оператор проверки равенства строковых содержимых файлов*
- }
- *scp-оператор управления блокировками*
 - ⇒ *разбиение*:*
 - {• *scp-оператор снятия всех блокировок данного scp-процесса*
 - *scp-оператор снятия блокировки с sc-элемента*
 - *scp-оператор установки полной блокировки на sc-элемент*
 - *scp-оператор установки блокировки на изменение sc-элемента*
 - *scp-оператор установки блокировки на удаление sc-элемента*
 - *scp-оператор снятия блокировки со структуры*
 - *scp-оператор установки полной блокировки на структуру*
 - *scp-оператор установки блокировки на изменение структуры*
 - *scp-оператор установки блокировки на удаление структуры*
 - }
- }

scp-операнд'

С аргумент действия'

Е неосновное понятие

Е ролевое отношение

⇒ разбиение*:

- {• *scp-константа'*
- *scp-переменная'*

}

⇒ разбиение*:

- {• *scp-операнд с заданным значением'*
- *scp-операнд со свободным значением'*

}

⇒ разбиение*:

- {• *константный sc-элемент'*
- *переменный sc-элемент'*

}

⇒ включение*:

- *формируемое множество'*
 - ⇒ *разбиение*:*
 - {• *формируемое множество 1'*
 - *формируемое множество 2'*
 - *формируемое множество 3'*
 - *формируемое множество 4'*
 - *формируемое множество 5'*
- *удаляемый sc-элемент'*
- *тun sc-элемента'*
 - ⇒ *разбиение*:*
 - {• *sc-узел'*
 - ⇒ *разбиение*:*
 - {• *структура'*
 - *отношение'*
 - ▷ *ролевое отношение'*
 - *класс'*
- *sc-дуга'*
 - ⇒ *разбиение*:*
 - {• *sc-дуга общего вида'*
 - *sc-дуга принадлежности'*
 - ▷ *sc-дуга основного вида'*
 - = (*константный sc-элемент'* ∩ *позитивная sc-дуга принадлежности'* ∩ *постоянная sc-дуга принадлежности'*)
 - ⇒ *разбиение*:*
 - {• *позитивная sc-дуга принадлежности'*
 - *негативная sc-дуга принадлежности'*
 - *нечеткая sc-дуга принадлежности'*
 - ⇒ *разбиение*:*
 - {• *временная sc-дуга принадлежности'*
 - *постоянная sc-дуга принадлежности'*
- *sc-ребро'*
- *файл'*

Ролевое отношение *scp-операнд'* является неосновным понятием и указывает на принадлежность аргументов *scp-оператору*. Помимо указания какого-либо класса *scp-операндов'* порядок аргументов *scp-оператора* дополнительно уточняется *ролевыми отношениями 1', 2'* и т. д.

В рамках *scp-программы scp-константы'* явно участвуют в *scp-операторах* в качестве элементов (в теоретико-множественном смысле) и напрямую обрабатываются при интерпретации *scp-программы*. Константами в рамках *scp-программы* могут быть *sc-элементы* любого типа, как *sc-константы*, так и *sc-переменные*. Константа в рамках *scp-программы* остается неизменной в течение всего срока интерпретации. Константа *scp-программы* может быть рассмотрена как переменная, значение которой совпадает с самой переменной в каждый момент времени, и изменено быть не может. Таким образом, далее будем считать, что *scp-константа'* и ее значение это одно и то же. Каждый *in-параметр'* при интерпретации каждой конкретной копии *scp-программы* становится *scp-константой'* в рамках всех ее операторов, хотя в исходном теле данной программы в каждом из этих операторов он является *scp-переменной'*.

В рамках *scp-программы scp-переменные'* не обрабатываются явно при интерпретации, обрабатываются значения переменных. Каждая переменная *scp-программы* может иметь одно значение в каждый момент времени, т. е. представляет собой ситуативный синглтон, элементом которого является текущее значение *scp-переменной'*. Значение каждой *scp-переменной'* может меняться в ходе интерпретации *scp-программы*. При этом интерпретатор при обработке *scp-оператора* работает непосредственно со значениями *scp-переменных'*, а не самими *scp-переменными'* (которые также являются узлами той же семантической сети).

Значение operandов, помеченных ролевым отношением *scp-операнд с заданным значением'*, считается заданным в рамках текущего *scp-оператора*. Данное значение учитывается при выполнении *scp-оператора* и остается неизменным после окончания выполнения *scp-оператора*. Каждая *scp-константа'* по умолчанию рассматривается

как *scp-операнд с заданным значением'*, в связи с чем явное использование данного ролевого отношения в таком случае является избыточным. В таком случае в качестве значения рассматривается непосредственно сам операнд. В случае если отношением *scp-операнд с заданным значением'* помечена *scp-переменная'*, то осуществляется попытка поиска значения для данной *scp-переменной'* (ее элемента). Если попытка оказалась безуспешной, то возникает ошибка времени выполнения, которая должна быть обработана соответствующим образом.

Любой *scp-операнд с заданным значением'* независимо от конкретного типа *scp-оператора* может быть *scp-переменной'*.

Значение operandов, помеченных ролевым отношением *scp-операнд со свободным значением'*, считается свободным (не заданным заранее) в рамках текущего *scp-оператора*. В начале выполнения *scp-оператора* связь между *scp-переменной'*, помеченной данным ролевым отношением, и ее элементом (значением) всегда удаляется. В результате выполнения данного оператора может быть либо сгенерировано новое значение *scp-переменной'*, либо не сгенерировано, тогда *scp-переменная'* будет считаться не имеющей значения. Ни одна *scp-константа'* не может быть помечена как *scp-операнд со свободным значением'*, поскольку константа не может изменять свое значение в ходе интерпретации *scp-программы*.

Ролевое отношение *тип sc-элемента'* используется для уточнения типа *sc-элемента*, выступающего в роли значения некоторого операнда. *тип sc-элемента'* имеет смысл указывать только для operandов, помеченных как *scp-операнд со свободным значением'*, тогда данное уточнение типа *sc-элемента* будет использовано для сужения области поиска либо уточнения параметров генерации каких-либо конструкций. Значением *scp-operandов с заданным значением'* является конкретный, известный на момент начала выполнения *scp-оператора* *sc-элемент* с конкретным типом, не зависящим от указания *типа sc-элемента'*, в связи с чем использование ролевого отношения *тип sc-элемента'* в данном случае является некорректным.

Допускается использование комбинаций семантически непротиворечащих друг другу подмножеств указанного отношения. Например, допускается комбинация *константный sc-элемент'* и *sc-дуга общего вида'*, но не допускается комбинация *sc-узел'* и *sc-дуга'*.

Ролевое отношение *формируемое множество'* используется для указания того факта, что в результате выполнения *scp-оператора* должно быть сформировано либо дополнено некоторое множество *sc-элементов*, являющееся значением одного из operandов данного *scp-оператора*. При этом если данный operand помечен как *scp-операнд со свободным значением'*, то множество будет сформировано с нуля (сгенерирован новый *sc-элемент*, обозначающий данное множество), в противном случае уже существующее множество может быть дополнено. Использование данного ролевого отношения предполагает, что при его отсутствии множество бы не формировалось, а значением указанного операнда стал бы произвольный *sc-элемент* из данного множества.

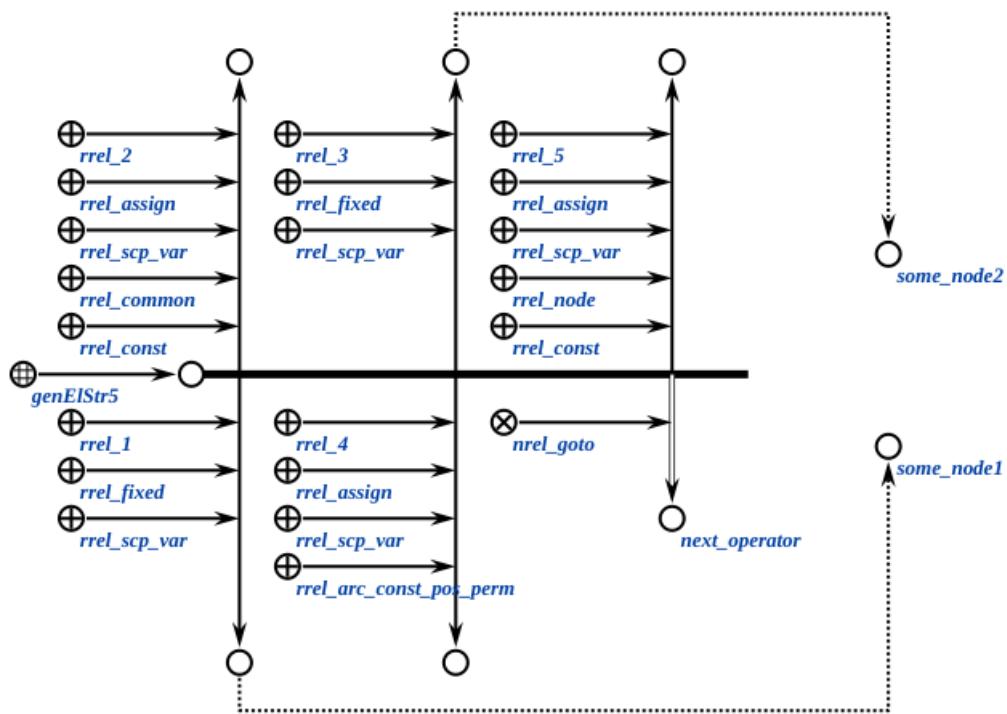
Ролевое отношение *формируемое множество'* без уточнения порядкового номера используется только в *scp-операторах обработки произвольных конструкций*. Для явного указания номера operandса, которому соответствует *формируемое множество'*, используются подмножества данного ролевого отношения, аналогичные ролевым отношениям, задающим порядок элементов в кортеже (1', 2', 3' и т. д.), например *формируемое множество 1'*, *формируемое множество 2'* и т. д. Указанные ролевые отношения используются только в *scp-операторах поиска конструкций с формированием множеств*.

Ролевое отношение *удаляемый sc-элемент'* используется для указания тех operandов, значение которых должно быть удалено в процессе выполнения *scp-операторов удаления*. Данным ролевым отношением может быть помечен как *scp-операнд с заданным значением'*, так и *scp-операнд со свободным значением'*. При этом удаляемым *sc-элементом* может быть как *scp-константа'*, так и *scp-переменная'* (в случае *scp-переменной'* удаляется не только связка принадлежности между этой *scp-переменной'* и ее значением, но и непосредственно сам *sc-элемент*, являющийся значением).

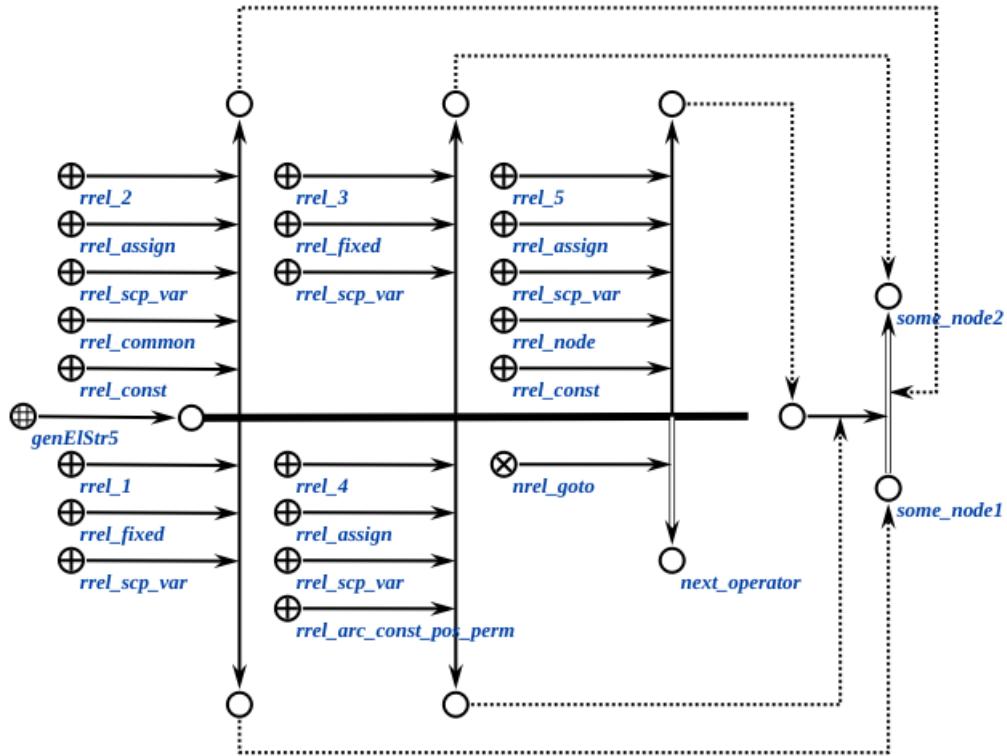
следует отличать*

- ∅ {• *scp-переменная'*
 - *sc-переменная*
}
- ∅ {• *scp-константа'*
 - *sc-константа*
}

На рисунках [Пример выполнения scp-оператора генерации пятиэлементной конструкции \(вызов scp-оператора\)](#) – [Пример выполнения scp-оператора генерации пятиэлементной конструкции \(результат выполнения scp-оператора\)](#) показан пример работы *scp-оператора генерации пятиэлементной конструкции*. В приведённом примере выполняется генерация пятиэлементной конструкции, которая имеет два *scp-операнда* с заданным значением. В примере предполагается, что рассматриваемые элементы (*some_node1* и *some_node2*) изначально никак не связаны между собой.



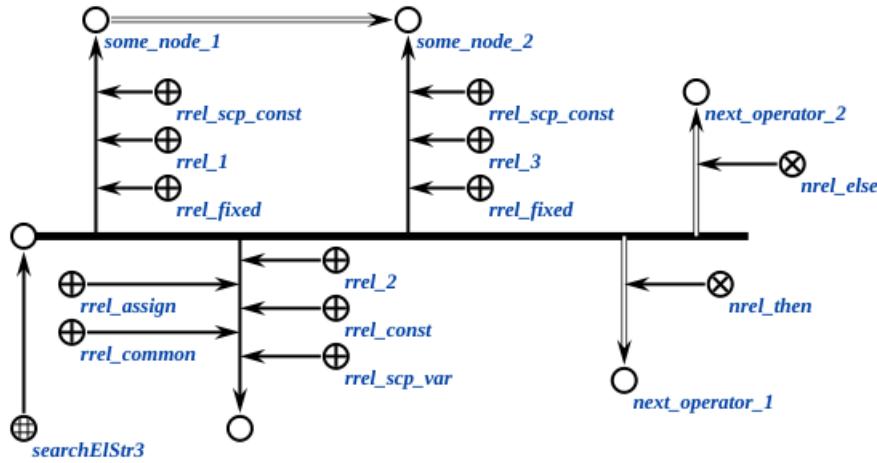
= Пример выполнения *scp*-оператора генерации пятиэлементной конструкции (вызов *scp*-оператора)



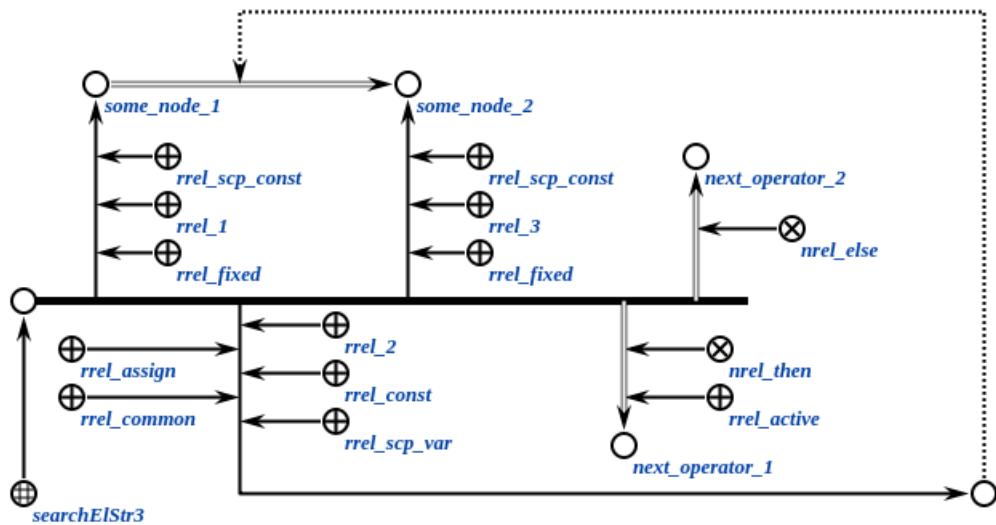
= Пример выполнения *scp*-оператора генерации пятиэлементной конструкции (результат выполнения *scp*-оператора)

На рисунках *Пример выполнения *scp*-оператора поиска трехэлементной конструкции (вызов *scp*-оператора)* – *Пример выполнения *scp*-оператора поиска трехэлементной конструкции (результат выполнения *scp*-оператора)*

приведён пример scp-оператора поиска трехэлементной конструкции, которая имеет два scp-операнда с заданным значением. В примере предполагается, что рассматриваемые элементы (`some_node1` и `some_node2`) изначально связаны между собой константной постоянной sc-дугой.

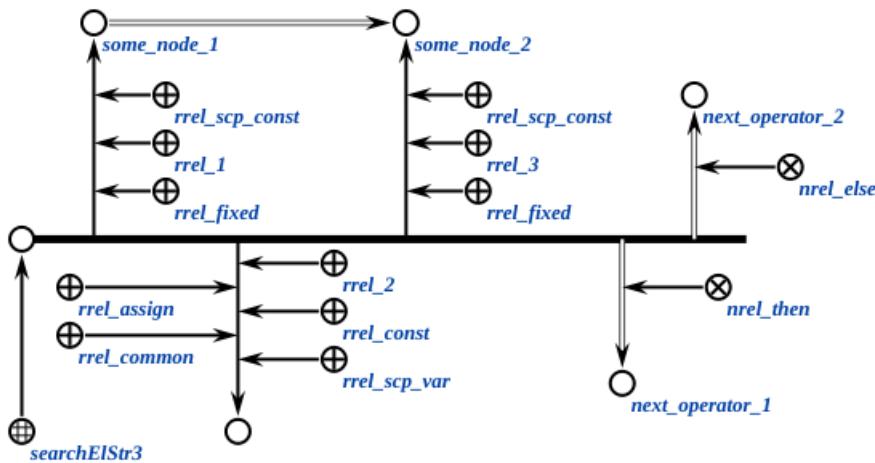


= Пример выполнения scp-оператора поиска трехэлементной конструкции (вызов scp-оператора)

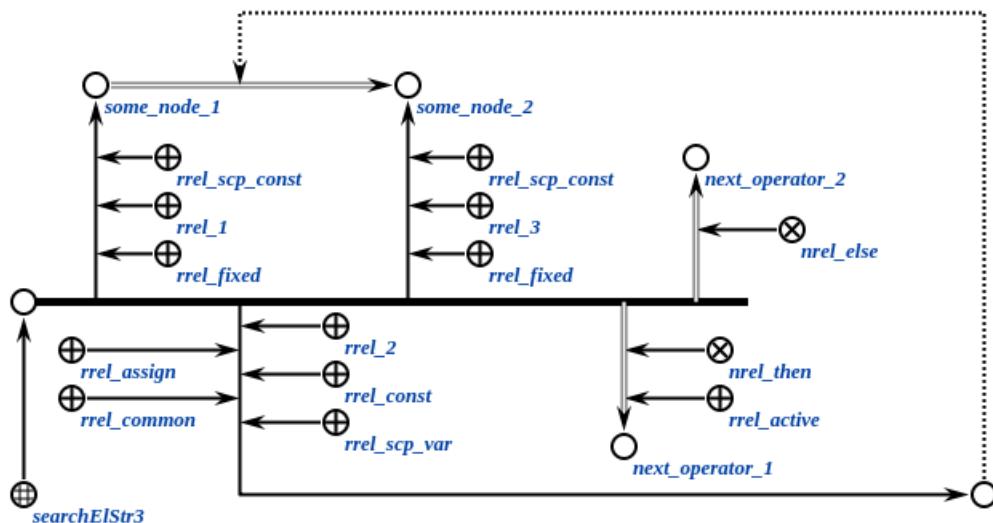


= Пример выполнения scp-оператора поиска трехэлементной конструкции (результат выполнения scp-оператора)

На рисунках [Пример выполнения scp-оператора удаления одноэлементной конструкции \(вызов scp-оператора\)](#) – [Пример выполнения scp-оператора удаления одноэлементной конструкции \(результат выполнения scp-оператора\)](#) показан пример scp-оператора удаления одноэлементной конструкции. В примере предполагается, что рассматриваемые элементы (`node1` и `node2`) изначально связаны между собой базовой sc-дугой принадлежности.



= Пример выполнения scpr-оператора удаления одноэлементной конструкции (вызов scpr-оператора)



= Пример выполнения scpr-оператора удаления одноэлементной конструкции (результат выполнения scpr-оператора)

Пункт 3.2.5.2. Операционная семантика Базового языка программирования ostis-систем

⇒ ключевой знак*:

- Абстрактная scpr-машина

Преимущества предложенного многоагентного подхода к обработке информации могут работать не только на платформенно-независимом уровне, но и на более низких уровнях. Так, в частности, интерпретатор Базового языка программирования ostis-систем также предлагается строить как *неатомарный абстрактный sc-агент*, обеспечивающий интерпретацию методов, описанных на Языке SCP. Таким образом, такой интерпретатор входит в общую иерархию агентов ostis-системы и является *абстрактным sc-агентом, не реализуемым на Языке SCP*.

В общем случае вариантов реализации таких интерпретаторов может быть много. В рамках Стандарта OSTIS один из них предлагается в качестве стандартного и называется *Абстрактной scpr-машиной*.

Абстрактная scpr-машина

⇒ декомпозиция абстрактного sc-агента*:

- {• Абстрактный sc-агент создания scpr-процессов
- Абстрактный sc-агент интерпретации scpr-операторов
- Абстрактный sc-агент синхронизации процесса интерпретации scpr-программ

- Абстрактный sc-агент уничтожения scp-процессов
 - Абстрактный sc-агент синхронизации событий в sc-памяти и ее реализации
 - ⇒ декомпозиция абстрактного sc-агента*:
 - {• Абстрактный sc-агент трансляции сформированной спецификации события в sc-памяти во внутреннее представление
 - Абстрактный sc-агент обработки события в sc-памяти, инициирующего агентную scp-программу
- }
- }

Задачей *Абстрактного sc-агента создания scp-процессов* является создание *scp-процессов*, соответствующих данной *scp-программе*. Данный *sc-агент* активируется при появлении в *sc-памяти* инициированного действия, принадлежащего классу *действие интерпретации scp-программы*. После проверки *sc-агентом* условия инициирования выполняется создание *scp-процесса* с учетом конкретных параметров интерпретации *scp-программы*, после чего осуществляется поиск *начального оператора' scp-процесса* и добавление его во множество *настоящих сущностей*.

Задачей *Абстрактного sc-агента интерпретации scp-операторов* является собственно интерпретация операторов *scp-программы*, то есть выполнение в *sc-памяти* действий, описываемых конкретным *scp-оператором*. Данный *sc-агент* активируется при появлении в *sc-памяти* *scp-оператора*, принадлежащего классу *настоящих сущностей*. После выполнения действия, описанного *scp-оператором*, *scp-оператор* добавляется во множество *прошлых сущностей*. В случае когда семантика действия, описанного *scp-оператором*, предполагает возможность ветвления *scp-программы* после выполнения данного *scp-оператора*, то используется одно из подмножеств класса *выполненных действий – успешно выполненное действие или успешно выполненное действие*.

Задачей *Абстрактного sc-агента синхронизации процесса интерпретации scp-программ* является обеспечение переходов между *scp-операторами* в рамках одного *scp-процесса*. Данный *sc-агент* активизируется при добавлении некоторого *scp-оператора* во множество *прошлых сущностей*. Далее осуществляется переход по *sc-дуге*, принадлежащей отношению *последовательность действий** (или более частным отношениям, в случае, если *scp-оператор* был добавлен во множество *успешно выполненных действий или успешно выполненных действий*). При этом очередной *scp-оператор* становится *настоящей сущностью* (активным *scp-оператором*) в том случае, если хотя бы один *scp-оператор*, связанный с ним входящими *sc-дугами*, принадлежащими отношению *последовательность действий** (или более частным отношениям), стал *прошлой сущностью* (или, соответственно, подмножеством прошлых сущностей). В случае, когда необходимо дождаться завершения выполнения всех предыдущих операторов, для синхронизации используется оператор класса *конъюнкция предшествующих операторов*.

Задачей *Абстрактного sc-агента уничтожения scp-процессов* является уничтожение *scp-процесса*, т. е. удаление из *sc-памяти* всех *sc-элементов*, его составляющих. Данный *sc-агент* активируется при появлении в *sc-памяти* *scp-процесса*, принадлежащего множеству *прошлых сущностей*. При этом уничтожаемый *scp-процесс* необязательно должен быть полностью сформирован. Необходимость уничтожения не до конца сформированного *scp-процесса* может возникнуть в случае, если при создании *scp-процесса* возникли проблемы, не позволяющие продолжить создание *scp-процесса* и его выполнение.

Задачей *Абстрактного sc-агента синхронизации событий в sc-памяти и ее реализации* является обеспечение работы *неатомарных sc-агентов*, реализованных на языке *SCP*.

Задачей *Абстрактного sc-агента трансляции сформированной спецификации события в sc-памяти во внутреннее представление* является трансляция ориентированных пар, описывающих *первичное условие инициирования** некоторого *sc-агента* во внутреннее представление элементарных событий на уровне *sc-хранилища*, при условии, что этот *sc-агент* реализован на платформенно-независимом уровне (с использованием языка *SCP*). Условием инициирования данного *sc-агента* является появление в *sc-памяти* нового элемента множества *активных sc-агентов*, для которого будет найдена и протранслирована соответствующая ориентированная пара.

Задачей *Абстрактного sc-агента обработки события в sc-памяти, инициирующего агентную scp-программу*, является поиск *агентной scp-программы*, входящей во множество *программ sc-агента** для каждого *sc-агента*, первичное условие инициирования которого соответствует событию, произошедшему в *sc-памяти*, а также генерация и инициирование действия, направленного на интерпретацию этой программы. В результате работы данного *sc-агента* в *sc-памяти* появляется *инициированное действие*, принадлежащее классу *действие интерпретации scp-программы*.

§ 3.2.6. Решатели задач ostis-систем

- ⇒ *ключевое понятие*:*
- решатель задач ostis-системы

- *машина обработки знаний*

С учетом того тезиса, что существуют *методы* интерпретации других *методов* и, следовательно, иерархия *методов*, а также, соответственно, иерархия *навыков*, можно уточнить и понятие решателя задач, как иерархической системы навыков. Таким образом, определим *решатель задач ostis-системы* определяется как совокупность всех *навыков*, которыми обладает *ostis*-система на текущий момент времени.

Такой подход к уточнению архитектуры *решателей задач ostis-систем* позволяет обеспечить их модифицируемость, что, в свою очередь, позволяет *ostis-системе* при необходимости легко приобретать новые *навыки*, модифицировать (совершенствовать) уже имеющиеся, и даже избавляться от некоторых навыков с целью повышения производительности системы. Таким образом, имеет смысл говорить не о жестко фиксированном решателе задач, который разрабатывается один раз при создании первой версии системы и далее не меняется, а о совокупности навыков, фиксированной в каждый текущий момент времени, но постоянно эволюционирующей.

решатель задач ostis-системы

- ⇐ семейство подмножеств*:
 - навык
 - := [иерархическая система навыков, которыми обладает *ostis*-система]
 - ▷ гибридный решатель задач *ostis*-системы
 - := [решатель задач *ostis*-системы, реализующий две и более модели решения задач]
 - ▷ объединенный решатель задач *ostis*-системы
 - := [полный решатель задач *ostis*-системы]
 - := [интегрированный решатель задач *ostis*-системы]
 - := [решатель задач *ostis*-системы, реализующий все ее функциональные возможности, как основные, так и вспомогательные]

В общем случае *объединенный решатель задач ostis-системы*, решает задачи, связанные с:

- обеспечением основных функциональных возможностей системы (например, решение явно сформулированных задач по требованию пользователя);
- обеспечением корректности и оптимизацией работы самой *ostis*-системы (перманентно на протяжении всего жизненного цикла *ostis*-системы);
- обеспечением повышения квалификации конечных пользователей и разработчиков *ostis*-системы;
- обеспечением автоматизации развития и управления развитием *ostis*-системы.

В свою очередь, под *машиной обработки знаний* будем понимать совокупность интерпретаторов всех *навыков*, составляющих некоторый *решатель задач*. С учетом многоагентного подхода к обработке информации, используемого в рамках Технологии *OSTIS*, *машина обработки знаний* представляет собой *sc-агент* (чаще всего – *неатомарный sc-агент*), в состав которого входят более простые *sc-агенты*, обеспечивающие интерпретацию соответствующего множества *методов*. Таким образом, *машина обработки знаний* в общем случае представляет собой иерархическую систему *sc-агентов*.

машина обработки знаний

- ⊂ *sc-агент*

Рассмотрим классификацию решателей задач *ostis*-систем по различным признакам.

Классификация решателей задач *ostis*-систем по типу соответствующей *ostis*-системы:

решатель задач ostis-системы

- Ξ Решатель задач Метасистемы *OSTIS*
- ▷ решатель задач вспомогательной *ostis*-системы
 - ▷ решатель задач интерфейса компьютерной системы
 - ⇒ разбиение*:
 - {• решатель задач пользовательского интерфейса компьютерной системы
 - решатель задач интерфейса компьютерной системы с другими компьютерными системами
 - решатель задач интерфейса компьютерной системы с окружающей средой
 - ▷ решатель задач *ostis*-подсистемы поддержки проектирования компонентов определенного класса
 - ▷ решатель задач *ostis*-подсистемы поддержки проектирования баз знаний
 - ▷ решатель задач повышения качества базы знаний
 - ▷ решатель задач верификации базы знаний
 - ▷ решатель задач поиска и устранения некорректностей в базе знаний
 - ▷ решатель задач поиска и устранения неполноты
 - ▷ решатель задач оптимизации структуры базы знаний

- ▷ решатель задач выявления и устранения информационного мусора
- ▷ решатель задач ostis-подсистемы поддержки проектирования решателей задач ostis-систем
 - ⇒ разбиение*:
 - решатель задач ostis-подсистемы поддержки проектирования программ обработки знаний
 - решатель задач ostis-подсистемы поддержки проектирования агентов обработки знаний
- }
- ▷ решатель задач подсистемы управления проектирования компьютерных систем и их компонентов
- ▷ решатель задач самостоятельной ostis-системы

Классификация решателей задач ostis-систем по типу интерпретируемой модели решения задач:

решатель задач ostis-системы

- ▷ решатель задач с использованием хранимых методов
 - := [решатель, способный решать задачи тех классов, для которых на данный момент времени известен соответствующий метод решения]
- ▷ решатель задач на основе нейросетевых моделей
- ▷ решатель задач на основе генетических алгоритмов
- ▷ решатель задач на основе императивных программ
 - ▷ решатель задач на основе процедурных программ
 - ▷ решатель задач на основе объектно-ориентированных программ
- ▷ решатель задач на основе декларативных программ
 - ▷ решатель задач на основе логических программ
 - ▷ решатель задач на основе функциональных программ
- ▷ решатель задач в условиях, когда метод решения задач данного класса в текущий момент времени не известен
 - := [решатель, реализующий стратегии решения задач, позволяющие породить метод решения задачи, который в текущий момент времени не известен ostis-системе]
 - := [решатель, использующий для решения задач метаметоды, соответствующие более общим классам задач по отношению к заданной]
 - := [решатель задач, позволяющий породить метод, который является частным по отношению какому-либо известному ostis-системе методу и интерпретируется соответствующей машиной обработки знаний]
- ▷ решатель, реализующий стратегию поиска путей решения задачи в глубину
- ▷ решатель, реализующий стратегию поиска путей решения задачи в ширину
- ▷ решатель, реализующий стратегию проб и ошибок
- ▷ решатель, реализующий стратегию разбиения задачи на подзадачи
- ▷ решатель, реализующий стратегию решения задач по аналогии
- ▷ решатель, реализующий концепцию интеллектуального пакета программ

Отдельно выделим классификацию машин обработки знаний, которые в общем случае могут соответствовать одним и тем же фрагментам базы знаний, но при этом в совокупности с ними образовывать разные навыки и соответственно разные решатели задач:

машина обработки знаний

- ▷ машина логического вывода
- ▷ машина дедуктивного вывода
 - ▷ машина прямого дедуктивного вывода
 - ▷ машина обратного дедуктивного вывода
- ▷ машина индуктивного вывода
- ▷ машина абдуктивного вывода
- ▷ машина нечеткого вывода
- ▷ машина вывода на основе логики умолчаний
- ▷ машина логического вывода с учетом фактора времени

Классификация решателей задач ostis-систем по типу решаемой задачи (цели решения задачи):

решатель задач ostis-системы

- ▷ решатель задач информационного поиска
 - ⇒ разбиение*:
 - решатель задач поиска информации, удовлетворяющей заданным критериям
 - решатель задач поиска информации, не удовлетворяющей заданным критериям

- }
- ▷ решатель явно сформулированных задач
 - := [решатель задач, для которых явно сформулирована цель]
 - ▷ решатель задач поиска или вычисления значений заданного множества величин
 - ▷ решатель задач установления истинности заданного логического высказывания в рамках заданной формальной теории
 - ▷ решатель задач формирования доказательства заданного высказывания в рамках заданной формальной теории
 - ▷ машина верификации ответа на указанную задачу
 - ▷ машина верификации решения указанной задачи
 - ▷ машина верификации доказательства заданного высказывания в рамках заданной формальной теории
 - ▷ решатель задач классификации сущностей
 - ▷ машина соотнесения сущности с одним из заданного множества классов
 - ▷ машина разделения множества сущностей на классы по заданному множеству признаков
 - ▷ решатель задач синтеза информационных конструкций
 - ▷ решатель задач синтеза естественно-языковых текстов
 - ▷ решатель задач синтеза изображений
 - ▷ решатель задач синтеза сигналов
 - ▷ решатель задач синтеза речи
 - ▷ решатель задач анализа информационных конструкций
 - ▷ решатель задач анализа естественно-языковых текстов
 - ▷ решатель задач понимания естественно-языковых текстов
 - ▷ решатель задач верификации естественно-языковых текстов
 - ▷ решатель задач анализа изображений
 - ▷ решатель задач сегментации изображений
 - ▷ решатель задач понимания изображений
 - ▷ решатель задач анализа сигналов
 - ▷ решатель задач анализа речи
 - ▷ решатель задач понимания речи

§ 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

В данной главе был детально рассмотрен подход к построению решателей задач, позволяющий решить ряд фундаментальных проблем в области построения решателей задач, таких как обеспечение совместимости различных решателей задач и их компонентов, а также обеспечение обучаемости (модифицируемости и рефлексивности) самих решателей. В то же время существует ряд проблем, остающихся актуальными и требующих решений.

Первая проблема связана с отсутствием достаточно строгой формализованной классификации задач, решаемых интеллектуальными системами, отсутствием унификации описания задач и классов задач, описания целей, хода и результата решения задачи, методов решения задач, связей между классами задач и методами решения задач данного класса. Решение данной проблемы, с одной стороны, позволит обеспечить возможность глубокой интеграции всевозможных моделей решения задач различных классов и возможность облегчить процесс интеграции новых моделей решения задач в интеллектуальную систему, а с другой стороны, станет предпосылкой для решения других проблем, описанных ниже.

Вторая проблема заключается в том, что на настоящий момент основное внимание в области разработки гибридных решателей задач уделено снижению трудоемкости интеграции различных компонентов решателя задач в интеллектуальную систему и реализации возможности накопления многократно используемых компонентов решателей, однако в общем случае не говорится о том, как конкретно интеллектуальная система будет применять те или иные компоненты при решении задач конкретных классов. Таким образом, построение общего плана решения задачи, т.е. выбор методов решения задач, определение порядка их применения и выбор исходных данных (аргументов) для применения того или иного метода, фактически определяется разработчиком на этапе проектирования системы или на этапе ее эволюции в процессе эксплуатации. Предпосылкой для решения данной проблемы является решение ранее рассмотренной проблемы унификации представления задач различных классов и методов их решения. Решение же рассматриваемой проблемы предполагает разработку комплекса *стратегий решения задач* (или метаметодов решения задач), которые позволят интеллектуальной системе самостоятельно формировать план решения задачи с учетом имеющихся в системе методов решения задач и, при необходимости, даже запрашивать

недостающие для решения задачи компоненты в соответствующих библиотеках. Следует отметить, что попытки разработки универсальных высокоуровневых подходов к решению задач предпринимались еще на заре развития искусственного интеллекта, в 1950-60ые гг, однако не увенчались успехом и вскоре прекратились. Во многом это связано с отсутствием на тот момент унифицированных моделей представления и обработки знаний, которые в настоящий моментлагаются в рамках *Технологии OSTIS*.

Еще одна актуальная проблема, тесно связанная с рассмотренными выше, заключается в том, что интеллектуальные системы часто вынуждены решать задачи в условиях так называемых не-факторов, то есть неполноты описания задачи и возможных путей ее решения, нечеткости и некорректности имеющихся знаний, отсутствия критерии для оценки оптимальности полученного решения и т.д. (см. *Нариньни А.С.НЕФакКВ-2004ст*). В особенности это актуально при решении поведенческих задач, связанных с изменением состояния объектов среды, внешней по отношению к интеллектуальной системе. Для решения задач в подобных условиях интеллектуальная система должна не только обладать достаточным набором компонентов решателя задач, реализующих модели решения задач в условиях наличия не-факторов (нечеткие логические модели, модели машинного обучения, генетические алгоритмы и т.д.), но и реализовывать *стратегии решения задач*, которые бы позволили принимать решения и формировать план решения задачи в такого рода условиях.

Рассмотренные проблемы связаны в первую очередь с процессом решения конкретной задачи интеллектуальной системой. В то же время очевидно, что в каждый момент времени интеллектуальная система вынуждена параллельно решать несколько задач, которые могут быть связаны как с непосредственным функциональным назначением системы, так и с обеспечением жизнедеятельности и эволюции самой системы. Во втором случае имеются в виду, в частности, задачи, связанные с актуализацией имеющихся у нее сведений о внешнем мире, поиском и устранением ошибок в базе знаний, оптимизацией структуры базы знаний и решателя системы, поиском и устранением информационного мусора и многие другие. При этом разные задачи могут иметь разный приоритет, который может меняться в зависимости от ситуации даже в процессе её решения. В то же время, в ситуации, когда априори не известно, какой из возможных способов решения задачи окажется наиболее эффективным, может оказаться целесообразным параллельное использование нескольких подходов к решению одной и той же задачи. Таким образом, актуальной является проблема организации управления информационными процессами решения задач в интеллектуальной системе и взаимодействия параллельно выполняемых информационных процессов с учетом приоритетности процессов, возможности отслеживать текущее состояние информационных процессов, порождать, приостанавливать и уничтожать информационные процессы. Для решения данной проблемы целесообразно заимствовать решения, широко используемые в традиционных компьютерных системах, в частности, реализуемые в современных операционных системах, и адаптировать их к специфике решения задач в интеллектуальных системах. Важно отметить, что реализация модели управления информационными процессами на основе общих унифицированных моделей обработки информации, предлагаемых в рамках Технологии OSTIS, позволит сделать одни информационные процессы объектом анализа других информационных процессов, что, в свою очередь, даст возможность анализировать ход решения задачи непосредственно в процессе решения, оценивать эффективность тех или иных методов решения задач, накапливать наиболее удачные решения для применения в дальнейшем для решения аналогичных задач и многое другое.

Решение перечисленных проблем позволит разработать принципиально новую иерархическую модель *гибридного решателя задач*, обладающую рядом существенных преимуществ, которая, в свою очередь, должна будет интерпретироваться на каких-либо платформах. Без унификации требований к платформе интерпретации моделей интеллектуальных систем и четкого разделения платформенно-независимой модели системы (и в частности решателя) и платформы невозможно говорить о реализации модели решателя, реализующей рассмотренные выше идеи. Это приведет к необходимости дублирования одних и тех же компонентов модели для разных платформ, значительно усложнит интеграцию компонентов решателя, поскольку потребует учета при такой интеграции особенностей каждой платформы. Кроме того, четкое разделение уровня модели системы и уровня платформы даст возможность независимо друг от друга развивать различные платформы и модели интеллектуальных систем. Таким образом, предлагается сформулировать унифицированные требования к платформе интерпретации семантических моделей интеллектуальных систем, а также построить общую модель такой платформы, удовлетворяющую указанным требованиям. Более подробно такая модель платформы рассмотрена в Главе 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем.

С другой стороны, как уже было сказано, решатель задач представляет собой сложную систему, ориентированную на работу со знаниями, а не с данными, в отличие от современных программных систем, в которых изначально известно, где конкретно локализованы нужные данные и в какой форме они представлены. В связи с этим, применение для разработки интеллектуальных систем современных программно-аппаратных платформ, ориентированных на адресный доступ к хранящимся в памяти данным, не всегда оказывается эффективным, поскольку при разработке интеллектуальных систем фактически приходится моделировать нелинейную память на базе линейной. Повышение эффективности решения задач интеллектуальными системами требует разработки специализированных платформ, в том числе аппаратных, ориентированных на унифицированные семантические модели представления и обработки информации. В качестве основы для таких разработок предлагается использовать предложенную в рамках Технологии OSTIS общую концепцию семантического компьютера, семантической памяти и базового языка

программирования, ориентированного на обработку информации в такой памяти, и дополнить их идеями волновых языков программирования, инсерционного программирования и других подходов, направленными на повышение эффективности обработки знаний, в том числе на аппаратном уровне. Более подробно концепция семантического компьютера рассматривается в Главе 6.2. *Ассоциативные семантические компьютеры для ostis-систем*.

Разработка решателей задач, включая рассмотренные выше проблемы разработки гибридных решателей задач, на настоящий момент рассматриваются в контексте одиночных (самостоятельных) интеллектуальных систем, функционирующих в некоторой среде (частью которой является и пользователь, если он есть). В то же время очевидна тенденция современных информационных технологий к переходу от одиночных систем к коллективам распределенных взаимодействующих компьютерных систем, в частности, к распределенному хранению данных и распределенным вычислениям. В случае интеллектуальных компьютерных систем важнейшим свойством систем, входящих в такие коллективы, становится *интероперабельность*, то есть способность системы к согласованному взаимодействию с другими подобными системами с целью решения каких-либо задач. Таким образом, особо актуальным является переход от разработки решателей задач отдельно взятых интеллектуальных систем к решателям задач взаимодействующих интероперабельных интеллектуальных систем, включая разработку принципов решения задач в таких распределенных коллективах с учетом решения всех обозначенных выше проблем. Для решения данной проблемы предлагается применить идеи, предлагаемые в рамках теории многоагентных систем, и переосмысленные в контексте взаимодействия гибридных интеллектуальных систем.

Кроме того, важнейшей проблемой в случае распределенного коллектива интеллектуальных систем является не просто обеспечение возможности решения задач таким коллективом в текущий момент времени, а перманентная поддержка семантической совместимости и, как следствие, интероперабельности систем, входящих в такой коллектив на протяжении всего их жизненного цикла. Очевидно, что каждая из систем, входящих в такой коллектив, и, соответственно, ее решатель задач может эволюционировать независимо от других систем, но при этом всегда должна сохраняться интероперабельность между системами, в противном случае решение задач в таком коллективе станет невозможным. Решение данной проблемы предполагает разработку методов перманентного анализа семантической совместимости распределенного коллектива взаимодействующих интеллектуальных систем, выявления и устранения проблем.

Для решения указанных проблем предлагается на основе подхода к построению гибридных решателей задач, рассмотренного в данной главе, разработать:

- Комплексную онтологию действий, задач и методов их решения, а также онтологию *гибридных решателей задач* на основе которой уточнить понятие решателя и его архитектуру. В Главе 3.1. *Формализация понятий действия, задачи, метода, средства, навыка и технологии* представлена первая версия Глобальной предметной области действий и задач и соответствующая ей онтология методов и технологий, на ее основе предлагается разработать комплексную онтологию действий и задач, решаемых ostis-системами;
- Комплекс унифицированных обобщенных стратегий (метаметодов) решения задач в интеллектуальных системах, позволяющий интеллектуальной системе самостоятельно формировать план решения задачи с учетом имеющихся в системе методов решения задач. В основу разрабатываемых стратегий кроме опыта аналогичных работ предлагается внести также некоторые общеметодологические идеи, связанные с теорией бихевиоризма и набирающими популярность идеями ее применения в информатике (см. *Cao O.IndepBUaUtBIA-2010art*, *Cao O.BehavLaNP-2014art*, *Pavel M..BehavLaCMiSoPHMaC-2015art*), ТРИЗ (см. *Альтшулер Г.С. Найти: ВвТРИЗ-2010кн*), а также СМД-методологией, предложенной школой Г. П. Щедровицкого (см. *Щедровицкий Г.П. СхемаМСССиС-1995кн*);
- Онтологическую модель формирования плана решения задачи и управления процессом решения задач в гибридных решателях задач в условиях различных не-факторов и отсутствия четких критериев оценки оптимальности полученного решения. Для разработки данной модели предлагается адаптировать теорию ситуационного управления (см. *Поспелов Д.А. СитуаУТП-1986кн*), и реализовать ее в контексте семантической теории решателей задач, разрабатываемой в рамках Технологии OSTIS;
- Комплексную онтологическую модель управления информационными процессами решения задач в интеллектуальных системах, построенных на базе унифицированных семантических моделей представления и обработки информации;
- Онтологическую модель платформы интерпретации унифицированных семантических моделей представления и обработки информации (*ostis-платформы*). Первая версия данной модели рассмотрена в Главе 6.1. *Универсальная модель интерпретации логико-семантических моделей ostis-систем*;
- Комплексную иерархическую модель гибридного решателя задач, основанную на многоагентном подходе и учитывающую необходимость решения задач как в рамках одиночных интеллектуальных систем, так и в рамках распределенных коллективов интероперабельных интеллектуальных систем;
- Комплекс методов анализа качества гибридных решателей задач и их компонентов;
- Комплекс методик и средств поддержки проектирования гибридных решателей задач. Первая версия такой методики и средств рассмотрена в Главе 5.3. *Методика и средства компонентного проектирования решателей задач ostis-систем*.

Заключение к Главе 3.2.

В данной главе рассмотрены актуальные на сегодняшний день проблемы в области разработки гибридных решателей задач и предложен общий подход к построению гибридных решателей задач, который решает такие проблемы, как обеспечение совместимости и модифицируемости решателей задач, а также создает предпосылки к решению других актуальных проблем, подробнее рассмотренных в § 3.2.7. *Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач.*

Сформулируем ряд конкретных направлений развития предложенных в данной главе подходов:

- Более тесно и полно интегрировать идеи ситуационного управления в предлагаемый подход;
- Доработать предложенный механизм блокировок, в частности, минимизировать число классов блокировок, учесть и реализовать идеи реализации lock-free алгоритмов;
- Исключить необходимость введения sc-метаагентов и scp-метапрограмм.
- Доработать Язык SCP до того, чтобы иметь возможность описывать в рамках scp-программ рецепторное и эффекторное взаимодействие ostis-систем.
- При разработке Абстрактной scp-машины учесть принципы построения волновых языков программирования (см. *Сапатый П.С. ЯВОЛНАкОНСдБ3-1986см, Moldovan D.I. SNAPaVLSIAfAIP-1985bk*) и идеи инсерционного программирования и моделирования (см. *Летичевский А.А..ИнсерП-2003см, Летичевский А.А..ИнсерМ-2012см*).

Глава 3.3.

Семантическая теория программ для ostis-систем

⇒ *автор**:

- Зотов Н.В.
- Шункевич Д.В.

⇒ *аннотация**:

[Несмотря на активное развитие и использование языков программирования, общей теории программ, на основе которой можно было бы проектировать и разрабатывать прикладные системы на данный момент не существует. В данной главе предлагается единая онтология языков программирования и представления программ на разных языках программирования в ostis-системах. Работа показывает особенности представления и ключевые моменты процесса интерпретации программ в ostis-системах.]

⇒ *подраздел**:

- § 3.3.1. Проблемы текущего состояния в области разработки и применения языков программирования
- § 3.3.2. Существующие онтологии языков программирования
- § 3.3.3. Предлагаемый подход к разработке технологий программирования для ostis-систем
- § 3.3.4. Синтаксис и семантика программ в ostis-системах
- § 3.3.5. Синтаксис и семантика языков программирования в ostis-системах
- § 3.3.6. Help-система поддержки проектирования и разработки программ в ostis-системах
- § 3.3.7. Критерии эффективности (качества) программ в ostis-системах

⇒ *ключевое понятие**:

- ***

⇒ *библиографическая ссылка**:

- ***

Введение в Главу 3.3. Семантическая теория программ для ostis-систем

За долгий период развития компьютерных систем (к.с.) практически сняты аппаратные ограничения на решение различных задач. Оставшиеся ограничения отводятся на долю программного обеспечения. Прежде всего эти ограничения связаны с текущими проблемами развития программного обеспечения:

- аппаратная сложность опережает умение человечества строить программные к.с., использующее потенциальные возможности аппаратуры;
- навыки и технологии разработки программ отстают от требований, предъявляемых к разработке программ нового поколения;
- возможностям эксплуатировать существующие программы угрожает низкое качество их разработки.

Ключом к решению этих проблем является глубокое понимание и грамотное использование существующих языков программирования как основного инструмента для массового создания программных к.с. нового поколения.

В данной главе акцент делается на достижение следующих результатов:

- (1) изложить классические основы, отражающие накопленный мировой опыт в области разработки и применения современных языков программирования;
- и (2) систематизировать основные результаты в этой области и представить их в виде единой унифицированной семантической теории программ.

В данной главе подробно описываются проблемы текущего состояния в области программ и языков программирования. Она посвящена базовым понятиям теории языков программирования, дается обзорная характеристика областей применения языков программирования, достаточно востребованных современным человеческим обществом, рассматриваются способы представления и интерпретации программ различных языков программирования, подробно описываются формы и содержание критериев для оценки эффективности языков.

§ 3.3.1. Проблемы текущего состояния в области разработки и применения языков программирования

В современную эру развития информационных технологий существует огромное количество языков программирования, каждый из которых имеет свою важную назначение в области проектирования программных систем. Каждый язык демонстрирует не только свою специфику, но имеет свои достоинства и недостатки. Многообразие языков программирования **Sebesta2012** и решений, созданных на них, настолько велико, что очень легко потеряться в море информации о всех аспектах применения и проектирования языков программирования. Кроме этого, основная проблема заключается не в количестве существующих решений в области разработки и применения современных языков программирования, а количестве форм (!), на которых представляются конкретные языки программирования. Так, декларативные знания, то есть знания, являющиеся, например, спецификацией какой-то программы, и процедурные знания, то есть знания, которые являются программами, принадлежащими какому-то языку программирования, представляются совершенно различными способами, методами и средствами.

В связи со сказанным можно выделить следующие ключевые проблемы в области разработки и применения современных языков программирования:

- Поскольку количество языков программирования растёт с увеличением потребности в них, то растут и потребности в описании этих языков программирования для дальнейшего использования и проектирования прикладных систем. Это в свою очередь требует высокого уровня качества спецификации конкретного языка: и описания синтаксиса и семантики конструкций этого языка, и описания средств и методов реновации инструментальных средств, обеспечивающих интерпретацию или трансляцию этого языка. То есть, с увеличением количества языков программирования растёт не только многообразие форм представления знаний (языков программирования), но и количество программных систем на различных формах представления знаний **Zapata J.Ontol iSEATGKA**.
- Большое многообразие форм представления знаний, как говорилось выше, предоставляет большой спектр возможностей проектирования программных к.с. на каждой из них. Получается, чтобы произвести интеграцию нескольких программных систем, реализованных на разных языках программирования, необходимо сделать так, чтобы системы могли коммуницировать между собой на каждом из тех языков, на котором они реализованы **Golenkov.V.V..PrinciplesOASCSD-2019art**. Так, стремление к использованию существующих программных компонентов затрудняется реализацией самих компонентов, поскольку чтобы объединить эти компоненты необходимо изменить их программный код **Penta2020, ScalabriNo2016**. Наличие многообразие форм затрудняет реализацию совместимых интероперабельных к.с. **Голенков.В.В..ГрафМПОЗПРП-2012ст**.
- С ростом сложности программного кода, уменьшается количество способных понять его смысл. Современные разработчики создают программные к.с., не учитывая полный её жизненный цикл **Brooks2021**. Системы должны постоянно обновляться и совершенствоваться с развитием технологий, на которых она основана **Sellitto2022**. Это должно обеспечиваться хорошей документацией реализации компонентов этих систем – это снижает не только потребности в привлечении новых ресурсов и кадров, но и способствует снижению реинжиниринга программных к.с. **Penta2020, ScalabriNo2016**.
- Полная автоматизация проектирования программных к.с. невозможна, поскольку современные языки, на которых они проектируются не имеют свойства рефлексивности – системы не могут познавать и понимать себя и развиваться почти в полной мере самостоятельно. Таким образом, существующие интеллектуальные к.с. не являются как таковыми интеллектуальными, потому что не имеют необходимых им свойств **GolenkovPrinciples2021**.
- Ключом к легкому и глубокому освоению конкретного языка как основного профессионального инструмента программиста является понимание общих принципов построения и применения языков программирования **Turner2007, Constanta2022**, описываемых их общей теорией. До сегодняшнего дня, общей теории языков программирования до сих пор не существует, что затрудняет разработку, верификацию и использование новых и существующих языков программирования. Без общей теории языков программирования каждый может разрабатывать принципиально общие методы и средства так, как хочется, а не так, как требуется **Голенков.В.В..ГрафМПОЗПРП-2012ст**.
- Достижение максимума услуг и средств при минимуме затрат возможно только путём глубокого понимания принципов построения языков программирования за счёт простоты средств и методов представления знаний. Сложное нужно сводить к простому и изъяснять простыми понятиями, не создавая дополнительной иллюзии важности **Sellitto2022, Chaраго2014, Posnett2011**.

Все эти проблемы связаны и являются проблемами текущего состояния направлений развития в области Искусственного интеллекта **Skeeter2020, Constanta2022**.

Итак, для решения перечисленных проблем необходимо создавать комфортные условия для реализации компьютерных систем, семантически совместимых и интероперабельных между собой. В контексте языков программирования необходима общая теория проектирования программ интеллектуальных к.с. нового поколения, которая:

- позволит без больших усилий и затрат интегрировать имеющиеся решения в области проектирования программ компьютерных систем **Golenkov.V.V..MethodsTECCS-2019ст**;

- объединит формы представления знаний декларативного и процедурного вида;
- будет иметь широкий спектр средств не только для описания синтаксиса и семантики существующих языков программирования, но и для проектирования новых аналогов;
- будет понятна не только человеку, но и машине Zapata J..Ontol iSEATGKA;
- обозначит принципы, по которым необходимо проектировать языки программирования нового поколения.

К проектированию таких общих теорий, строго говоря, нужно подходить с высокой степенью важности. Проектируемые к.с. должны всегда иметь возможности использовать те свойства, которые им начертаны. Для того, чтобы и эта теория могла быть использована как некоторая система знаний о том, как надо проектировать и использовать языки программирования и программы в программных к.с., и том, как интерпретировать их программы, необходимо, чтобы эта теория была описана средствами и методами, которыми проектируются эти программные к.с. Речь идёт о том, что принципиально важным подходом к проектированию общей теории программ является онтологический подход **Zapata J..Ontol iSEATGKA, Golenkov.V.V.MethodsTECCS-2019ст, Sales2022, Samaa2020**.

Для воплощения данных идей необходимо изучить и интегрировать опыт, накопленный в области разработки и применения современных языков программирования. Поэтому далее будут рассмотрены результаты других исследований в области проектирования общей теории языков программирования и программ.

§ 3.3.2. Существующие онтологии языков программирования

В большинстве, идеи, предлагаемые в научных работах по исследованию языков программирования, безусловно являются востребованными и полезными для проектирования программных к.с. Так, идея о том, что языки программирования и программы, реализуемых на них, должны быть организованы в общую таксономию понятий, является основополагающей, поскольку обеспечивает наиболее качественную среду для проектирования и реализации к.с. Общая теория программ нужна не только для того чтобы описывать термины и понятия как некоторую спецификацию, используемую для проектирования программных к.с. (что тоже не мало важно), но и для того, чтобы определять качество языков программирования и программ по таким вопросам, как: "Является ли данный язык языком программирования", "Является ли данное знание программой", "Насколько эффективна данная программа", "Какова степень интеллекта данной программной системы" и т. д. Данные идеи предложены и рассмотрены в работах Raymond Turner **Eden2007, Turner2012**.

До сегодняшнего дня существует большое количество аналогов онтологий языков программирования и программ. Примеры можно найти в работах **Lando2007, Lando2009, Turner2007**. Также стоит отметить разработанные онтологии программ **Turner2012, Turner2014, Jacobs2022**, система понятий в которых определяется строго и однозначно на формальных языках: языках логики и языках описания грамматик формальных языков. Однако ни одна из них не является таким результатом, который можно было бы использовать при проектировании программных к.с. без существенных проблем. Разработанные онтологии сосредотачивают в себе лишь краткое описание связанных между собой понятий, но общей картины того, как данные онтологии можно использовать в конкретных задачах, почти не видно.

Сегодня встречаются и вовсе противоположные суждения о назначении программ и языков программирования **Rapaport2020**, противоречащие формальным основам Искусственного интеллекта **Grimmelmann2022**. Программные к.с. должны быть не только понятны человеку, но и сами должны понимать себя, свои возможности, намерения, действия и цели, и понимать себе подобные кибернетические системы. Только таким образом человечество и результаты его деятельности в виде каких-то конкретных систем смогут работать сообща, дополняя друг друга и преумножая свои результаты **Голенков.В.В..ГрафМПОЗПРП-2012ст**.

В результате анализа приведенных работ можно сделать вывод о том, что:

- общей теории программ и языков программирования, которая могла быть задействована при решении любой прикладной задачи и представлении и реализации средств проектирования компьютерных систем до сих пор не сложилась;
- унификация представления средств описания и реализации по этим описаниям как главный аргумент к оперированию смысловому представлению знаний, к полному взаимопониманию между компьютерными системами вовсе не рассматривается;
- программы и совокупности этих программ в виде программных к.с. реализуются в большинстве случаев в индивидуальном порядке и плохо документируются, что усложняет их использование, интеграцию с другими программами и программными к.с., тестирование и совершенствование.

Ключом к решению всех этих проблем является общая технология проектирования компьютерных систем нового поколения, на базе которой можно построить общую теорию программ (дисциплину программирования) **Deikstra1978**, которая будет рассмотрена дальше.

§ 3.3.3. Предлагаемый подход к разработке технологий программирования для ostis-систем

Несмотря на обширное разнообразие используемых человечеством классических технологий, общего решения, позволяющего решить поставленные проблемы в комплексе, не существует. Накопленный опыт в сфере проектирования и реализации компьютерных систем показывает, что для более качественной и эффективной реализации различных методов существующие технологии не приспособлены для решения комплексных задач. Для повышения качества разработки сложных систем необходимо разрабатывать и использовать принципиально новые технологии программирования.

Почему Технология OSTIS является ключом к решению описанных проблем в области проектирования и применения языков программирования?

- Стандарт Технологии OSTIS **Standard2021** уже реализует базовые средства, необходимые для проектирования и разработки интероперабельных к.с., в основе которых лежит смысловое представление знаний. Это устраняет не только необходимость создания онтологий верхнего уровня, которые должны быть использованы в общей теории программ как базовые для описания понятий этой теории, но и помогает проектировать решения согласованно с другими онтологиями. В результате формируется общая слаженная картина мира, которая (1) непротиворечива, то есть согласована, (2) однозначно трактуема, (3) универсальная и, (4) самое главное, понятна для каждого.
- Технология OSTIS проектируется одним языком унифицированного представления знаний, называемым SC-кодом. Смысл программ и языков программирования понятен и однозначен тогда и только тогда, когда этот смысл описывается на одном общем языке, понятному любой кибернетической системе.
- SC-код синтаксически минимален. Для описания объектов и связей между ними используется минимальное количество знаков. В то же время многообразие этих связей сводится к многообразию знаковых конструкций. Всё это обеспечивается за счёт представления информации в виде графовых структур **Kasyanov2003, Petrov1978**.
- SC-код не просто удобен для описания и проектирования каких-то сложных объектов – с его помощью можно проектировать и реализовывать любые языки представления знаний, в том числе программы, компьютерные системы и, вообще, реальный мир.
- Онтологический и компонентный подходы **Sales2022, Samaa2020** к проектированию любых сложных объектов обеспечивают выполнение главных принципов, по которым должны проектироваться современные системы. То, что реализовано и можно использовать, нужно переиспользовать везде **O4IS2007, Molorodov2019**.

Проектирование и реализация программы на каком-либо языке программирования должна сводиться к описанию её синтаксиса и денотационной семантики в базе знаний ostis-системы с помощью некоторой библиотеки предметных областей и онтологий программ, описываемой в рамках этой базы знаний. Для этого нужна онтология программ, которые позволили бы в достаточном объёме описывать программы на любых языках программирования в ostis-системах. Такой подход позволяет не только описывать сложноструктурные объекты простым и понятным языком, но и позволяет унифицировать представление различных видов знаний. Тем самым, информация о программах и сами программы представляются на одном и том же языке (имеют один синтаксис), но содержательно описываются при помощи разных онтологий. Таким образом, решением всех проблем будет являться общая теория программ, однозначно соответствующей некоторой онтологии программ, с помощью которых можно было бы описывать синтаксис и денотационную семантику любых программ в ostis-системах.

Таким образом, результатом данной главы является *Предметная область и онтология программ* (далее - Предметная область и онтология методов), с помощью которой можно описывать синтаксис, денотационную и операционную семантику различных методов в ostis-системах. *Предметная область и онтология методов* является дочерней предметной областью по отношению к *Предметной области и онтологии информационных конструкций и языков*. Это означает, что она наследует все свойства исследуемых в ней понятий и отношений.

Предметная область и онтология информационных конструкций и языков

⇒ дочерняя предметная область*:

- *Предметная область и онтология языков*
 - ⇒ дочерняя предметная область*:
 - *Предметная область и онтология естественных языков*
 - *Предметная область и онтология формальных языков*

Предметная область и онтология формальных языков

⇒ дочерняя предметная область*:

- *Предметная область и онтология языков представления знаний*
 - ⇒ дочерняя предметная область*:
 - *Предметная область и онтология методов*

Предметная область и онтология методов

⇒ *дочерняя предметная область**:

- *Предметная область и онтология методов ostis-систем*
⇒ *дочерняя предметная область**:
 - *Предметная область и онтология процедурных методов ostis-систем*

Каждая теория должна быть согласована понятийно. Несмотря на то, что в литературе сложилась разное трактование понятия языка программирования, должно быть одно универсальное. Для этого вместо языков программирования далее будем говорить о языках представления методов, а вместо программ этих языков программирования – о методах как знаковых конструкциях языков представления методов (я.п.м.). Такое решение обосновывается тем, что обычно язык выступает в роли инструмента какого-то знания определенного вида, а термин языка программирования является вырожденным, поскольку стоит говорить не о языках, на которых что-то можно программировать, а о языках, на которых можно представлять знания определённого вида, в данном случае – знания процедурного типа. Сами термины “языка программирования” и “программы” будем считать неосновными идентификаторами понятий “языка представления методов” и “метода”, соответственно. Также это правило применяется на все понятия, используемые в данной главе и содержащие термин “метод”.

Общая теория программ в ostis-системах не отрицает весь накопленный опыт в сфере разработки современных технологий программирования. Наоборот, предлагаемая в данной главе идея позволяет переиспользовать те проверенные инструменты и методы для наиболее быстрой и качественной реализации программ в сложных программных системах.

§ 3.3.4. Синтаксис и семантика программ в ostis-системах

⇒ *подраздел**:

- *Пункт 3.3.4.1. Синтаксис программ в ostis-системах*
- *Пункт 3.3.4.2. Денотационная семантика программ в ostis-системах*
- *Пункт 3.3.4.3. Операционная семантика программ в ostis-системах*

⇒ *ключевое понятие**:

- *спецификация метода**
- *денотационная семантика метода**
- *операционная семантика метода**

⇒ *библиографическая ссылка**:

- ***

Синтаксис и семантика метода представляют его *спецификацию*. Семантику метода можно рассматривать с двух ракурсов: как множество знание, связанных между собой, что определяется денотационной семантикой данного метода, и как знание, которое может быть интерпретировано другим методом, что определяется операционной семантикой данного метода.

спецификация метода*

⇒ *разбиение**:

- {• *синтаксис метода**
- *денотационная семантика метода**
- := [обобщенная формулировка класса задач, решаемых с помощью данного метода*]
 \Leftrightarrow *семантически близкий знак**:
 обобщенная формулировка задач соответствующего класса методов*
- *операционная семантика метода**
- := [перечень обобщенных агентов, обеспечивающих интерпретацию метода*]
 := [семейство методов интерпретации данного метода*]
 := [формальное описание интерпретатора заданного метода*]

}

Пункт 3.3.4.1. Синтаксис программ в ostis-системах

Любый метод состоит из атомарных информационных конструкций, которые задают порядок действий в базе знаний, с помощью которых нужно перейти от исходного состояния к целевому, решив таким образом какую-то

конкретную задачу. Так, например, в процедурном методе любой такой оператор представляет собой некоторую математическую функцию. Для композиции этих функций в более крупные фрагменты используются выражения и операторы. В свою очередь, линейные последовательности операторов и условные ветвления также могут быть представлены функциями, составленными из функций отдельных компонентов этих конструкций. Цикл легко описывается рекурсивной функцией, составленной из компонентов, входящих в его тело.

*Синтаксис метода** определяет множество его допустимых конструкций. Внешний вид элементов метода задают с помощью конкретного синтаксиса. Он описывает такие лексические детали, как размещение ключевых слов и знаков пунктуации. Для спецификации конкретного синтаксиса применяют грамматики.

Синтаксис я.п.м. в ostis-системах может быть формально описан различными способами. Так, например, можно использовать метаязык Бэкуса-Наура для описания синтаксиса какого-то методов конкретного я.п.м. Другими не менее известными формами представления методов являются контекстно-свободные грамматики, расширенная форма Бэкуса-Наура, синтаксические графы Sebesta2012, Scott2006, Scott1972.

Однако значительно более логично и целесообразно описывать синтаксис других языков на универсальном языке представления знаний – *SC-коде*. Такой подход позволит ostis-системам самостоятельно понимать, анализировать и генерировать тексты указанных языков на основе принципов, общих для любых форм внешнего представления информации, в том числе нелинейных Petrov1978. Таким образом, языки, написанные на SC-код, имеют такой же синтаксис как и сам SC-код.

Пункт 3.3.4.2. Денотационная семантика программ в ostis-системах

Семантика метода разъясняет смысл синтаксических конструкций метода. Наиболее распространенными методами описания семантики языков программирования являются: денотационной, операционный, аксиоматический, алгебраический Orlov2013. На базе принципов Технологии OSTIS, под семантикой метода будем подразумевать объединение денотационной и операционной семантики метода.

Описание способа "привязки" метода к какому-то классу задач включает в себя:

- набор переменных, которые входят как в состав метода, так и в состав обобщенной формулировки задач соответствующего класса, и значениями которых являются соответствующие элементы исходных данных каждой конкретной решаемой задачи;
- часть обобщенной формулировки задач того класса, которому соответствует рассматриваемый метод, являющихся описанием условия применения этого метода;
- описание условия инициирования метода и его результата;
- описание начальной и целевой ситуаций в sc-памяти.

"Привязка" метода к конкретной задаче, решаемой с помощью этого метода осуществляется путем поиска в базе знаний такого фрагмента, который удовлетворяет условиям применения указанного метода. Одним из результатов такого поиска является установление соответствия между указанными выше переменными используемого метода и значениями этих переменных в рамках конкретной решаемой задачи. Другим вариантом установления рассматриваемого соответствия является явное обращение (вызов, call) соответствующего метода (программы) с явной передачей соответствующих параметров. Но такое не всегда возможно, т.к. при выполнении процесса решения конкретной задачи на основе декларативной спецификации выполнения этого действия нет возможности установить:

- когда необходимо инициировать вызов (использование) требуемого метода;
- какой конкретно метод необходимо использовать;
- какие параметры, соответствующие конкретной инициируемой задаче, необходимо передать для "привязки" используемого метода к этой задаче.

Под *процессом* понимается некоторое действие в sc-памяти, однозначно описывающее конкретный акт выполнения некоторого метода для заданных исходных данных Deikstra1978. Если метод описывает алгоритм решения какой-либо задачи в общем виде, то процесс обозначает конкретное действие, реализующее данный алгоритм для заданных входных параметров. По сути, процесс представляет собой уникальную копию, созданную на основе метода, в котором каждой sc-переменной, соответствует созданная sc-константа.

отношение, заданное на множестве (процесс)^λ

:= [отношение, область определения которого включает в себя множество всевозможных процессов]

Э параметр'

⇒ разбиение*:

- {• in-параметр'
- out-параметр'

}

- Э *in-параметр'*
- Э *out-параметр'*
- Э *начальная информационная конструкция'*
- Э *подпроцесс**

Процесс "привязки" метода решения задач к конкретной задаче, решаемой с помощью этого метода, можно также представить как процесс, состоящий из следующих этапов:

- построение копии используемого метода;
- склеивание основных (ключевых) переменных используемого метода с основными параметрами конкретной решаемой задачи.

В результате этого на основе рассматриваемого метода используемого в качестве образца (шаблона) строится спецификация процесса решения конкретной задачи. Описание процесса "привязки" метода решения к конкретной задаче, а также описание элементов метода является *денотационной семантикой данного метода*.

денотационная семантика метода

- Э *общая формулировка класса задач**
 - := [текстовая формулировка множества задач, решаемых данным методом]
 - С *пояснение**
- Э *первичное условие инициирования**
- Э *условие инициирования и результат**
 - ⇐ *декартово произведение*:*
 - {• *класс методов*
 - *импликация**
- Э *условие начальной и целевой ситуаций**
 - ⇐ *декартово произведение*:*
 - {• *класс методов*
 - *импликация**

Пример части спецификации, описывающей денотационную семантику Метода нахождения удвоенной суммы двух чисел, приведён на рисунке [Спецификация метода решения задачи вычисления удвоенной суммы двух чисел](#).

Отношение *общая формулировка класса задач** представляет собой класс sc-связок между sc-связкой, обозначающей множество методов, и файлом ostis-системы, являющийся пояснением того, какие классы задач можно решать при помощи данного множества методов. В некоторых редких случаях наличие такой sc-связки в спецификации метода может и не быть, поскольку нет необходимости уточнять, какие классы задач можно решать при помощи данного метода.

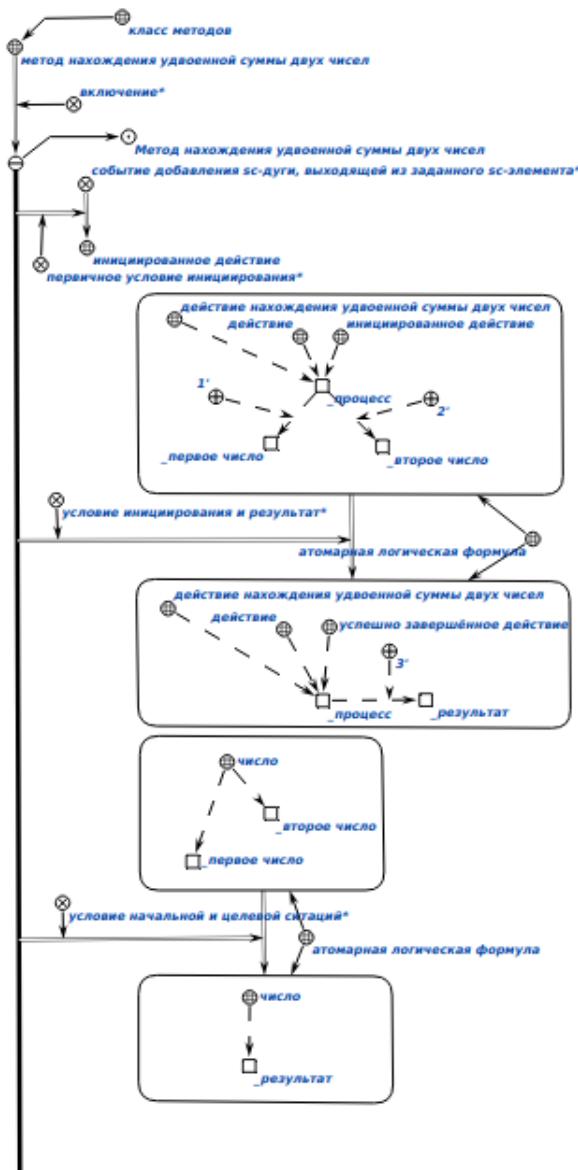
Связки отношения *первичное условие инициирования** связывают между собой sc-связку, обозначающей множество методов, и бинарную ориентированную пару, описывающую первичное условие инициирования данного метода, т.е. такой спецификацию ситуации в sc-памяти, возникновение которой побуждает метаметод-исполнитель перевести заданное множество методов в активное состояние и начать проверку наличия их полного условия инициирования.

Первым компонентом данной ориентированной пары является знак некоторого класса элементарных событий в sc-памяти*, например, событие добавления sc-дуги, выходящей из заданного sc-элемента*.

Вторым компонентом данной ориентированной пары является произвольный в общем случае sc-элемент, с которым непосредственно связан указанный тип события в sc-памяти, т.е., например, sc-элемент, из которого выходит либо в который входит генерируемая либо удаляемая sc-дуга, либо файл, содержимое которого было изменено.

Связки отношения *условие инициирования и результат** связывают между собой sc-связку, обозначающей множество методов, и бинарную ориентированную пару, связывающую условие инициирования данного множества методов и результаты выполнения этого множества методов в какой-либо конкретной системе. Указанную ориентированную пару можно рассматривать как логическую связку импликации, при этом на sc-переменные, присутствующие в обеих частях связки, неявно накладывается квантор всеобщности, на sc-переменные, присутствующие либо только в посылке, либо только в заключении неявно накладывается квантор существования.

Первым компонентом указанной ориентированной пары является логическая формула, описывающая условие инициирования описываемого метода, то есть конструкции, наличие которой в sc-памяти вызывает множество методов для начала работы по изменению состояния в sc-памяти. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.



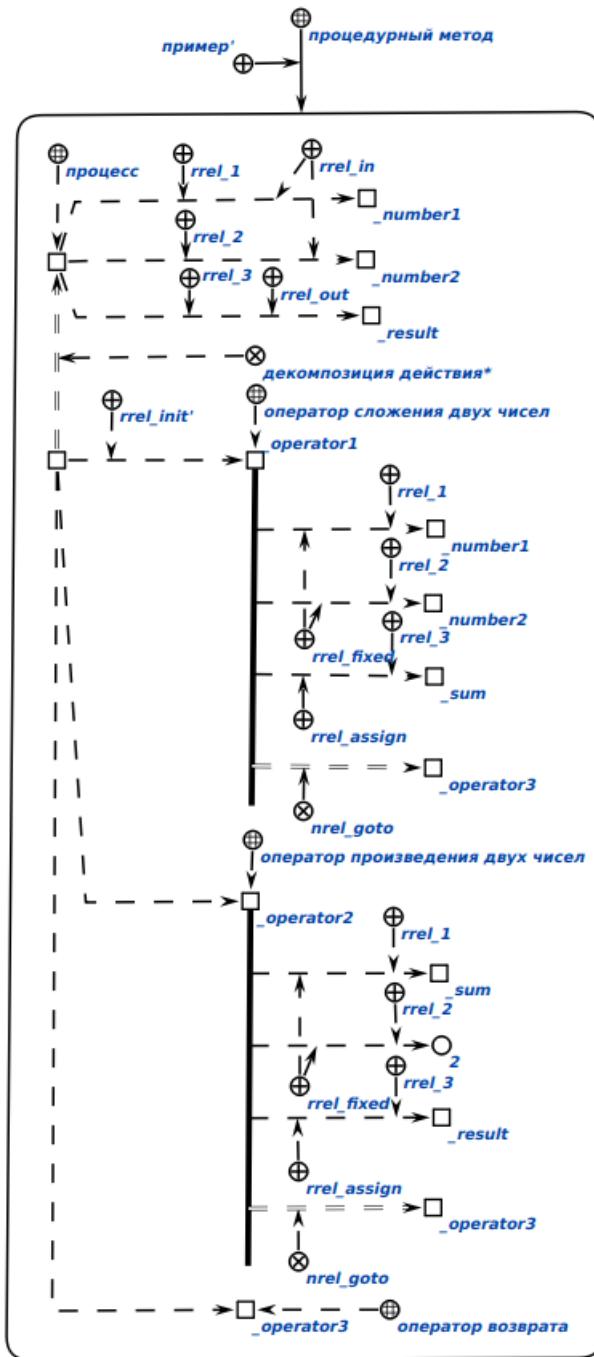
= Спецификация метода решения задачи вычисления удвоенной суммы двух чисел

Вторым компонентом указанной ориентированной пары является логическая формула, описывающая возможные результаты выполнения описываемого множества методов, то есть описание произведенных им изменений состояния sc-памяти. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

Связки отношения *условие начальной и целевой ситуации** связывают между собой sc-связку, обозначающей множество методов, и бинарную ориентированную пару, связывающую начальную и целевую ситуацию в sc-памяти, то есть, кратко говоря, ситуацию до применения метода и желаемую ситуацию после применения метода. Указанную ориентированную пару можно также рассматривать как логическую связку импликации, при этом на sc-переменные, присутствующие в обеих частях связки, неявно накладывается квантор всеобщности, на sc-переменные, присутствующие либо только в посылке, либо только в заключении неявно накладывается квантор существования. Для первой и второй компоненты указанной ориентированной пары накладывается те же ограничения и свойства, что для компонент ориентированной пары, являющейся вторым компонентом отношения *условие инициирования и результат**.

Стоит отметить, что связки отношения *условие инициирования и результат** и отношения *условие начальной и целевой ситуации** могут быть представлены иначе. Иногда может и не быть необходимости создавать и проверять второе условие метода, по которому проверяется наличие начальной ситуации в sc-памяти и проверка достижения целевой ситуации в sc-памяти в результате применения метода. Если так, то условие начальной и целевой ситуации* может быть уточнено в логических формулах, являющихся компонентами второго компонента связки отношения *условие инициирования и результат**.

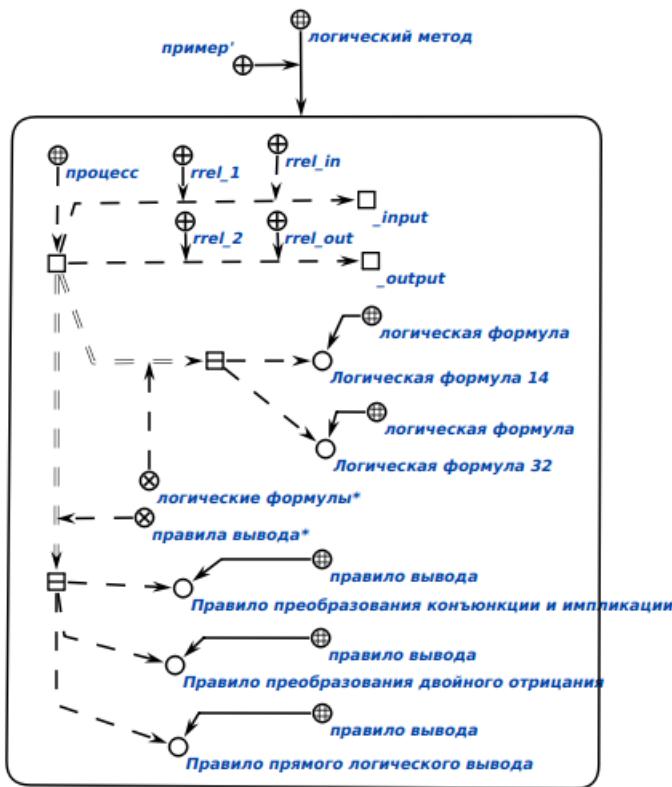
Программы в зависимости от его способа задания в конкретном языке представления методов будут различаться. В этом можно убедиться, сравнив примеры процедурного (рис. [Пример процедурного метода решения задачи вычисления удвоенной суммы двух чисел](#)) и логического (рис. [Пример логического метода решения задачи вычисления удвоенной суммы двух чисел](#)) методов решения одной и той же задачи.



= Пример процедурного метода решения задачи вычисления удвоенной суммы двух чисел

С помощью SC-кода можно представлять и те языки, которые не написаны на нём. Проблема будет в том, что форма и смысл языка и его методов будут разделены, то есть будут представлены по-разному. В данном случае SC-код выступает мощным инструментом для интеграции спецификаций различных языков внешнего представления знаний. Однако стоит отметить, что в представлении различных форм методов, принадлежащих разным языкам представления методов, в рамках Технологии OSTIS нет необходимости. Это объясняется тем, что:

- SC-код является достаточно универсальным языком для представления любых видов знаний. Это означает, что различные формы алгоритма решения одной и той же задачи можно свести к минимуму. В SC-коде фундаментом является формальная теория, что обеспечивает универсальное представление различных видов декларативных и процедурных знаний. Так, логические методы можно представлять в виде процедурных программ, в которых в качестве operandов операторов будут не только логические формулы и правила вывода,



= Пример логического метода решения задачи вычисления удвоенной суммы двух чисел

но и другие методы, обеспечивающие интерпретацию этих логических формул при помощи правил вывода. Таким образом, SC-код можно называть не только языком унифицированного представления знания, но и языком, на котором можно решать различные классы задач одним и тем же способом.

- Различные виды знаний в ostis-системах, проектируемые по принципам Технологии OSTIS глубоко интегрированы между собой. Это даёт не только простоту для создания этих систем на базе имеющихся языков, которые могут быть описаны на SC-коде, но большие возможности для создания базовых языков программирования для компьютерных систем нового поколения таких, как, например, базового языка представления процедурных методов SCP, базового языка представления продукционных методов и т. д. Современные языки представления методов создаются с целью упрощения описания какого-то алгоритма для быстрого и качественного решения определённого класса задач **Benri2000**. В свою очередь, предлагаемые методики и модели позволяют проектировать я.п.м. для компьютерных систем нового поколения с помощью базовых языков представления знаний таким образом, чтобы сама форма представления знаний не менялась. Методы разных я.п.м. должны иметь одну универсальную форму представления, то есть один и тот же синтаксис, но могут давать возможности описывать и представлять разными способами денотационную и операционную семантику своих методов с помощью одного и того же синтаксиса.
- Проектирование новых я.п.м. должно сводится к их полному описанию на минимальном семействе языков SC-кода: самого SC-кода, SCP и SCL. Речь идёт о том, чтобы спроектировать новый язык представления методов достаточно разработать (неатомарный) метаметод на языках SCP и SCL, который будет интерпретировать методы проектируемых языков, а также описать денотационную семантику этих методов. Метаметод интерпретации методов я.п.м. можно называть интерпретатором этих языков, то есть некоторой абстрактной sc-машиной, на которой возможно выполнение методов определённого языка представления этих методов.

Пункт 3.3.4.3. Операционная семантика программ в ostis-системах

Полная спецификация метода* кроме денотационной семантики этого метода* должна включать операционную семантику этого метода*, то есть формальное описание интерпретатора заданного метода. Операционная семантика я.п.м. описывает выполнение метода, составленного на данном языке, средствами виртуального компьютера. Виртуальный компьютер определяется как абстрактный автомат. Внутренние состояния этого автомата моделируют состояния вычислительного процесса при выполнении метода. Автомат транслирует исходный текст метода в набор формально определенных операций. Этот набор задает переходы автомата из исходного состояния

в последовательность промежуточных состояний, изменения значения переменных метода. Автомат завершает свою работу, переходя в некоторое конечное состояние. Таким образом, здесь идет речь о достаточно прямой абстракции возможного использования я.п.м. Операционная семантика описывает смысл метода путем выполнения его операторов на простой машине-автомате. Изменения, происходящие в состоянии машины при выполнении данного оператора, определяют смысл этого оператора.

Операционная семантика конкретного метода сводится к описанию *метаметода*, который его интерпретирует, верифицирует и т. д.

метаметод

С метод

:= [метод, значениями параметров которого являются другие методы]

операционная семантика метода

Ξ метаметод интерпретации*

⇐ декартово произведение*:
 {• класс методов
 • метод
 }

Ξ метаметод верификации и оценки качества*

⇐ декартово произведение*:
 {• класс методов
 • метод
 }

Отношение *метаметод интерпретации** представляет собой класс sc-связок между sc-связкой, обозначающей множество методов, и sc-узлом, обозначающим метод, который способен произвести интерпретацию заданного множества методов. Отношение *метаметод верификации и оценки качества** представляет собой класс sc-связок между sc-связкой, обозначающей множество методов, и sc-узлом, обозначающим метод, который способен произвести верификацию и оценку качества заданного множества методов.

В рамках Технологии OSTIS таких метаметодов может быть большое разнообразие. Каждый из них может состоять из множества атомарных и неатомарных подметодов. Это могут быть как метаметоды, интерпретирующие методы определённых я.п.м., так и метаметоды, верифицирующие и анализирующие качество этих методов. В том числе метаметоды могут производить операции над другими метаметодами.

метаметод интерпретации методов базовых языков представления методов

⇒ класс подметодов*:

- метаметод интерпретации методов языка представления логических методов SCP
- метаметод интерпретации методов языка представления логических методов SCL
- метаметод интерпретации методов языка представления продукционных методов
- метаметод интерпретации методов языка представления функциональных методов
- метаметод интерпретации методов языка представления нейросетей
- метаметод интерпретации методов языка представления генетических алгоритмов

метаметод верификации и оценки качества методов базовых языков представления методов

⇒ класс подметодов*:

- метаметод верификации и оценки качества методов языка представления логических методов SCP
- метаметод верификации и оценки качества методов языка представления логических методов SCL
- метаметод верификации и оценки качества методов языка представления продукционных методов
- метаметод верификации и оценки качества методов языка представления функциональных методов
- метаметод верификации и оценки качества методов языка представления нейросетей
- метаметод верификации и оценки качества методов языка представления генетических алгоритмов

Так, например, при реализации методов в ostis-системах метаметодами будут являться интерпретатор базового языка программирования SCP, а также интерпретаторы, реализованные непосредственно на языке SCP.

Понятие синтаксиса, денотационной и операционной семантики языков представления методов сводятся к понятию синтаксиса, денотационной и операционной семантики вообще любого языка.

§ 3.3.5. Синтаксис и семантика языков программирования в ostis-системах

Понятно, что для использования я.п.м. следует описать каждую конструкцию языка в отдельности, а также ее применение в совокупности с другими конструкциями. В языке существует множество различных конструкций, точное определение которых необходимо как программисту, применяющему язык, так и разработчику компилятора для этого языка. Программисту эти знания позволяют прогнозировать вычисления, производимые операторами метода. Разработчику описания конструкций необходимы для создания правильной реализации компилятора.

Описание формальной модели языка программирования можно задать его *спецификацией*. Спецификация содержит описание синтаксиса и семантики я.п.м.

спецификация языка представления методов*

⇒ *отношение, заданное на множестве (язык представления методов)**

⇒ *разбиение*:*

- {• *синтаксис языка представления методов**
 - ⊂ *синтаксис языка**
 - := [быть теорией правильно построенных информационных конструкций, принадлежащих заданному языку представления методов]
 - *денотационная семантика языка представления методов**
 - ⊂ *денотационная семантика языка**
 - := [обобщенная формулировка классов задач, решаемых с помощью данного языка представления методов*]
 - *операционная семантика языка представления методов**
 - ⊂ *операционная семантика языка**
 - := [перечень обобщенных агентов, обеспечивающих интерпретацию методов заданного языка представления методов*]
 - := [семейство методов интерпретации текстов данного языка представления методов*]
 - := [формальное описание интерпретатора заданного языка представления методов*]

}

Под *синтаксисом я.п.м.** подразумевается бинарное ориентированное отношение, каждая пара которого связывает знак некоторого языка с описанием синтаксически выделяемых классов фрагментов конструкций заданного я.п.м., с описанием отношений, заданных на этих классах и с конъюнкцией кванторных высказываний, являющихся синтаксическими правилами заданного языка, то есть правилами, которым должны удовлетворять все синтаксические правильные (правильно построенные) конструкции указанного я.п.м. В общем случае, отношение *синтаксиса я.п.м.** ничем не отличается от отношения *синтаксиса языка**, но всё-таки уточнение есть, поскольку я.п.м. являются языками вообще и синтаксис я.п.м. наследует все свойства синтаксиса любых языков. *Синтаксиса я.п.м.** объединяет синтаксисы всех методов, принадлежащих данному языку представления методов.

Под *денотационной семантикой я.п.м.** подразумевается бинарное ориентированное отношение, каждая пара которого связывает знак некоторого языка со знаком некоторой онтологии, с помощью которой можно описывать методы этого языка, а под *операционной семантикой я.п.м.** – описание метаметода интерпретации методов этого языка.

В контексте данной работы конкретные виды денотационной и операционной семантики рассматриваться далее не будут.

§ 3.3.6. Help-система поддержки проектирования и разработки программ в ostis-системах

Текущее состояние в области проектирования и разработки программного обеспечения говорит о том, что разработчики больше стремятся автоматизировать разработку методов на конкретных языках представления методов, чем обеспечить инструментальными обучающими средствами их проектирования, в том числе проектирования новых языков представления методов. Это приводит к следующим проблемам:

- В то время, как количество разработчиков, понимающих код какой-то сложной программной системы, уменьшается, требования к этой системе растут всё быстрее и быстрее. Зачастую, разработчики сложных программных систем сами не в состоянии объяснить логику работы этих систем. По этой причине необходимо создавать инструментальные средства, которые будут позволять автоматизировать документирование программных систем **lu2022rethinking**.
- Для обучения новых разработчиков навыкам работы с программными системами и их разработки необходимо привлекать ресурсы экспертов, понимающих принципы работы этих программных систем. Проблема решается разработкой справочной системы, которая будет позволять не только обучать пользователя тому, как проек-

- тировать методы решения задачи и программные системы на основе этих методов, но и указывать на пробелы в смежных дисциплинах, необходимых для достижения качественных результатов всей своей деятельности.
- В инженерии часто разработчики проектируют и разрабатывают решения, которые уже когда-то были созданные другими специалистами. Таким образом, получаются функционально эквивалентные методы решения задач, а то, и вовсе, программные системы, решающие схожие проблемы. Ключом к решению данной проблемы является проектирование семантически мощной библиотеки многократно используемых методов решения задач.

Таким образом, одной семантической теории программ недостаточно. Кроме неё, для перманетного и беспрепятственного проектирования и разработки методов различного класса необходимо разрабатывать:

- интеллектуальную help-систему поддержки проектирования и разработки методов, упомянутую в работе **Gulakina2012**, которая будет не только помогать разработчику верифицировать разрабатываемый метод, но и подсказывать способы его разработки;
- семантически мощную библиотеку многократно используемых компонентов **Ford B..CompoD-2019art** для быстрого поиска существующих методов решения задач и их применения для решения других более комплексных задач **sales2022explainable**.

Потенциальная help-система должна быть частью общего инструментального средства разработки интеллектуальны компьютерных систем нового поколения - ostis-платформы **Platform2021** - и может состоять из следующих компонентов:

- интеллектуальной help-системы по семантической теории программ;
- интеллектуальной help-системы по библиотеке многократно используемых методов решения задач,
- интеллектуальной help-системы по комплексу инструментальных средств проектирования методов решения задач,
- интеллектуальной help-системы по методике обучения проектированию различных методов решения задач.

Каждый компонент должен содержать:

- справочную подсистему,
- подсистему мониторинга и анализа деятельности разработчика методов решения задач,
- подсистему управления обучением.

Каждая из подсистем взаимодействует с другими подсистемами, а также может функционировать автономно.

Справочная подсистема является консультантом-экспертом в области семантической теории программ, который может ответить на любой вопрос новичка или опытного пользователя. Каждая из таких систем может становиться индивидуальным помощников в обучении новых специалистов - персональным ostis-ассистентом.

§ 3.3.7. Критерии эффективности (качества) программ в ostis-системах

Язык представления методов можно определить множеством показателей, характеризующих отдельные его свойства. Возникает задача введения меры для оценки степени приспособленности я.п.м. к выполнению возложенных на него функций — *критерии эффективности Orlov2013*. Критерии эффективности методов приводятся на основе частных показателей эффективности этих методов (показателей качества). Способ связи между частными показателями определяет вид критерия эффективности.

эффективность метода

⇒ *свойство-предпосылка**:

- легкость чтения и понимания метода
- легкость представления метода
- стоимость метода
- общий объем задач, решаемых при помощи данного класса методов
- многообразие видов задач, решаемых при помощи данного класса методов
- надежность метода

Лёгкость чтения метода должна способствовать легкому выделению основных понятий каждой части метода без обращения к его спецификации.

легкость чтения и понимания метода

⇒ *свойство-предпосылка**:

- простота синтаксиса я.п.м.
- ортогональность информационных конструкций я.п.м.
- структурированность потока управления в методе

Язык представления методов должен предоставить *простой* набор информационных конструкций, которые могут быть использованы в качестве базисных элементов при создании методов. Сильное воздействие на простоту оказывает синтаксис языка: он должен прозрачно отражать семантику конструкций, исключать двусмысленность и неоднозначность толкования.

Ортогональность означает, что любые возможные комбинации различных информационных конструкций будут осмысленными, без неожиданного поведения, возникающих в результате взаимодействия конструкций или контекста использования.

Порядок передач управления между операторами метода, то есть *поток управления*, должен быть удобен для чтения и понимания человеком.

Легкость создания метода отражает удобство языка для представления этого метода в конкретной предметной области.

легкость представления метода

⇒ *свойство-предпосылка**:

- *простота синтаксиса я.п.м.*
- *естественность синтаксиса я.п.м.*
- *ортогональность информационных конструкций я.п.м.*
- *полнота и точность спецификации я.п.м.*
- *согласованность и целостность спецификации я.п.м.*

Синтаксис метода должен способствовать легкому и прозрачному отображению в нем алгоритмических структур предметной области. Синтаксис я.п.м. должен быть не только *простым*, но и *естественнym*, и поддерживать *ортогональность* информационных конструкций языка.

Лёгкость представления нового метода обеспечивается *полной и точной, согласованной и целостной спецификацией* соответствующего языка. То есть необходимо достаточное количество информационных конструкций в этом языке для того чтобы представить конкретный метод. При этом спецификация языка должна быть согласованной и целостной чтобы представлять на ней непротиворечивые методы.

Стоимость метода я.п.м. складывается из нескольких составляющих.

общая стоимость метода

⇒ *свойство-предпосылка**:

- *стоимость применения метода*
- *стоимость интерпретации метода*
- *стоимость создания, тестирования и использования метода*
- *стоимость сопровождения метода*
- *согласованность и целостность спецификации языка представления методов*

Стоимость применения метода во многом зависит от структуры я.п.м. Язык, требующий многочисленных проверок синтаксических типов во время применения метода, будет препятствовать быстрой работе программы.

Размер стоимости интерпретации метода зависит от возможностей используемого метаметода интерпретации. Чем совершеннее методы оптимизации, тем дороже стоит интерпретация. Размер стоимости создания, тестирования и использования метода зависит от используемого метаметода верификации и оценки качества этого метода.

Многочисленные исследования показывают, что значительную часть стоимости используемого метода составляет не стоимость его разработки, а *стоимость его сопровождения Brooks2021*. Связывая сопровождение методов с другими их характеристиками, следует выделить, прежде всего, зависимость от читабельности, поскольку сопровождение обычно происходит следующим поколением разработчиков.

Общий объем задач и многообразие видов задач, решаемых при помощи данного класса методов, являются не менее важными характеристиками и показывают степень универсальности соответствующего я.п.м. Чем больше задач можно решить на я.п.м, тем больше он универсальнее.

Надёжность методов я.п.м. должно обеспечиваться минимумом ошибок при работе конкретного метода.

Все эти критерии можно применить и касательно самих языков представления методов.

Заключение к Главе 3.3. Семантическая теория программ для ostis-систем

Данная глава является началом семантической теории программ для к.с. нового поколения. Логичным развитием данной главы будут:

- уточнение и дополнение понятий *Предметной области и онтологии методов* для достижения полноты теории;
- описание дочерних предметных областей *Предметной области и онтологии методов* для конкретных видов методов, а также уточнение денотационной и операционной семантики спецификации этих методов;
- описание возможных путей реализации метаметодов интерпретации методов различных я.п.м;
- формализация математических моделей для подсчёт оценок эффективности методов.

Глава 3.4.

Язык вопросов для ostis-систем

**Самодумкин С.А.
Шункевич Д.В.**

⇒ *аннотация**:
[Аннотация к главе.]

§ 3.4.1. Синтаксис языка вопросов для ostis-систем

§ 3.4.2. Денотационная семантика языка вопросов для ostis-систем

§ 3.4.3. Операционная семантика языка вопросов для ostis-систем

Глава 3.5.

Логические, продукционные и функциональные модели решения задач в ostis-системах

Василевская А.П.

Орлов М.К.

Шункевич Д.В.

⇒ автор*:

- Орлов М.К.
- Василевская А.П.

⇒ аннотация*:

[Логические модели решения задач являются основой обработки знаний в интеллектуальных системах. В данной главе рассматривается интеграция различных моделей решения задач, в том числе принципы логического вывода, для решения задач на основе общей формальной модели.]

⇒ библиографическая ссылка*:

- Голенков В.В..БазовПТЯSCL-1996мо
- Averin A.I..UsingPiDI-2004art
- Голенков В.В..ГрафоАМиСПОИвСИИ-2011см
- Sethy S.S.MediaIS-2021art
- Norton J.D..aDemonstrationofIoCoII-2019art
- Yuxuan Z..MissiEAKGIItDGLaT-2022art
- Safawi A.R..tDecisPoAI-2015art
- Gungov A.tAmpliLiDtAoAiCR-2018art
- Geramian A..FuzzyISAfFAiAI-2017art
- Son L.H..PictuISaNFISoPFS-2017art
- Uehara K..FuzzyIIPaP-2017art
- Lupea M.aTheorPfCaRDL-2002art
- Weydert E.DefauLaPaTP-2022art
- Chen G..TempoLifFDoSSwGPD-2021art
- Рыбаков В.В.МультВНЛЛПД-2020см
- Гаврилова.Т.А..БазыЗИСУП-2000кн

⇒ подраздел*:

- § 3.5.1. Операционная семантика логических языков, используемых ostis-системами

§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами

Логика решает задачи доказательства истинности высказываний, аргументации того или иного высказывания, задачу генерации и опровержения гипотез. Некоторые гипотезы могут быть опровергнуты, однако извлекая причины того, почему гипотеза опровергнута, можно изменить посылку гипотезы так, чтобы создать новую гипотезу, которая впоследствии может стать теоремой.

Технология OSTIS позволяет интегрировать любые модели решения задач и принципы логического вывода для решения задач в интеллектуальных системах на основе общей формальной модели. Для того, чтобы использовать какую-либо новую или существующую модель, необходимо привести ее к предлагаемому формализму, что позволит интегрировать и синхронизировать ее с уже имеющимися в соответствующей библиотеке совместимых компонентов. Формализм SC-кода позволяет описывать отношения между понятиями любой формы и сложности, что делает его подходящим вариантом для использования логического вывода в интеллектуальных компьютерных системах нового поколения. А также воспользоваться техникой иерархии за счёт онтологического подхода, ле-

жащего в основе баз знаний ostis-систем. Исходя из этого предлагается следующая иерархия интегрированных предметных областей:

Предметная область логических формул, высказываний и формальных теорий

⇒ дочерняя предметная область*:

- Предметная область логических языков
- Предметная область логического вывода

Предметная область логических языков

⇒ дочерняя предметная область*:

Предметная область языка логики высказываний

⇒ дочерняя предметная область*:

Предметная область языка логики предикатов

Предметная область логических моделей решения задач

⇐ дочерняя предметная область*:

- Предметная область логических языков
- Предметная область логического вывода

Наследование предметных областей позволяет использовать описанные логики и их компоненты при описании любых логик. Базовые понятия позволяют разработчикам интеллектуальной системы добавлять новые логики. Для реализации конкретной логической модели решения задач необходимо создать предметную область, которая будет дочерней по отношению к *Предметной области логических моделей решения задач* и предметной области некоторого логического языка, например, языка логики высказываний, языка логики предикатов, языка нечёткой логики и других.

Предметная область логических формул, высказываний и формальных теорий задаёт денотационную семантику логических формул, высказываний и формальных теорий и содержит формальную спецификацию понятий, необходимых для формирования логических формул и высказываний любых логик, в том числе традиционных, нечётких, правдоподобных, темпоральных, логик умолчания и любых других. Логические формулы и высказывания интерпретируются с помощью понятий, описанных в *Предметной области логических моделей решения задач*, включающую модель и реализацию абстрактных агентов, необходимых для решения логических задач. Эта предметная область включает в себя спецификацию таких понятий, как логический вывод, правила вывода, равносильные преобразования и аксиомные схемы.

Современная логика изучает формальные языки, служащие для выражения логических рассуждений. Логический язык — формальный язык, предназначенный для воспроизведения логических форм контекстов естественного языка, а также выражения логических законов и способов правильных рассуждений в логических теориях, строящихся в данном языке. Логика не изучает то, как были получены знания, она позволяет представлять знания, а также из существующих знаний вывести новые (то есть из имеющихся формул логики вывести новые формулы этой же логики), установить правильность рассуждений.

Язык SCL — подъязык SC-кода для записи логических утверждений (см. [Голенков В.В. БазовПТЯSCL-1996мо](#)), он подробно рассматривается в главе [2.6. Смысловое представление логических формул и высказываний в различного вида логиках](#). Над высказываниями языка SCL можно проводить логический вывод.

Выводом в формальной системе называется любая последовательность формул такая, что любая формула либо аксиома этой формальной системы, либо непосредственное следствие каких-либо предыдущих формул по одному из правил вывода. Идея выводимости центральна в логике: в любой формальной аксиоматической теории ‘теорема’ – это формула, которая выводится из аксиом. Правильность умозаключений вводится и проверяется совершенно формально, без какой-либо связи с истинностью входящих в него посылок, т.е. исключительно с точки зрения структуры рассуждения. С практической точки зрения самое важное свойство такой формальной правильности рассуждений заключается в следующем: если нам удалось доказать, пользуясь методами формальной логики, правильность рассуждения, и нам известно из опыта, что все используемые посылки истинны, то мы можем быть уверены в истинности заключения (см. [Averin A.I.. UsingPiDI-2004art](#)). Истинность используемых посылок задаётся состоянием базы знаний.

Различные логические подходы позволяют проектировать решатели задач для интеллектуальных систем в разных предметных областях, учитывая их специфику. Решатель задач каждой конкретной системы во многом зависит от назначения данной системы, множества решаемых задач, предметной области и других факторов. Некоторые операции, необходимые в одной предметной области будут избыточными в другой. Например, в системе, решающей задачи по геометрии, химии и другим естественным наукам обоснованным будет использование дедуктивных методов вывода, поскольку решение задач в таких предметных областях основывается только на достоверных правилах. В системах же медицинской диагностики, к примеру, постоянно возникает ситуация, когда диагноз может быть поставлен только с некоторой долей уверенности и абсолютно достоверным ответ на поставленный вопрос

быть не может. В связи с этим возникает необходимость использования различных решателей задач в различных системах, при этом их состав и возможности в конкретной системе определяется не только непосредственно разработчиком, а требует консультаций с экспертами в данной предметной области. Тем не менее основой для всех видов логик является классическая логика и наиболее общие её методы распространяются на другие логики с некоторыми модификациями, уточнениями и ограничениями (см. [Голенков В.В..ГрафоАМиСПОИвСИИ-2011ст](#)).

Приведем краткую классификацию существующих логических методов решения задач:

- **Классический дедуктивный вывод.** Классический дедуктивный вывод является наиболее популярным при построении автоматических решателей задач, так как всегда дает достоверный результат. Дедуктивный вывод включает в себя прямой и обратный и логический вывод (принцип резолюции, процедуру Эрбрана и др.) (см. [Averin A.I..UsingPiDI-2004art](#)), все виды силлогизмов (см. [Sethy S.S.MediaIS-2021art](#)) и т.д. Основной проблемой дедуктивного вывода является невозможность его использования в ряде случаев, когда отсутствуют достоверные знания.
- **Индуктивный вывод.** Индуктивный вывод предоставляет возможность в процессе решения использовать различные предположения, что делает его удобным для использования в слабоформализованных и трудноформализуемых предметных областях, например при построении систем медицинской диагностики. Подробно принципы индуктивного вывода рассмотрены в работах [Norton J.D..aDemonstrationIoCoII-2019art](#), [Yuxuan Z..MissiEAKGIItDGLaT-2022art](#).
- **Абдуктивный вывод.** Под абдуктивным выводом в искусственном интеллекте, как правило, понимается вывод наилучшего абдуктивного объяснения, т.е. объяснения некоторого события, ставшего неожиданным для системы. Причем наилучшим считается такое объяснение, которое удовлетворяет специальным критериям, определяемым в зависимости от решаемой задачи и используемой формализации. Абдуктивный вывод подробно рассматривается в работах [Safawi A.R..tDecisPoAI-2015art](#), [Gungor A.tAmpliLiDtAoAIiCR-2018art](#).
- **Нечеткая логика.** Теория нечетких множеств и, соответственно, нечетких логик, также применяется в системах, связанных с трудноформализуемыми предметными областями (см. [Geramian A..FuzzyISAfFAiAI-2017art](#), [Son L.H..PictuIsaNFISoPFS-2017art](#)). Здесь импликативные высказывания могут рассматриваться как "если истинна посылка, то с некоторой вероятностью (часто или редко) истинно заключение", в отличие от классической логики, где зачастую используются статические предметные области и выражение "часто или редко" не применимо (корректно использовать только наречие "всегда"). Подробнее теория нечетких логик рассматривается в работе [Uehara K..FuzzyIPaP-2017art](#).
- **Логика умолчаний.** Логика умолчаний применяется, в том числе, для того, чтобы оптимизировать процесс рассуждений, дополняя процесс достоверного вывода вероятностными предположениями в тех случаях, когда вероятность ошибки крайне мала. Подробнее логика умолчаний рассмотрена в статьях [Lupea M.aTheorPfCaRDL-2002art](#), [Weydert E.DefauLaPaTP-2022art](#).
- **Темпоральная логика.** Применение темпоральной логики является очень актуальным для нестатичных предметных областей, в которых истинность того или иного утверждения меняется со временем, что существенно влияет на ход решения какой-либо задачи (см. [Chen G..TempoLifFDoSSwGPD-2021art](#), [Рыбаков В.В.МультВНЛПД-2020ст](#)). Следует отметить, что используемый в данной работе язык представления знаний предоставляет все необходимые возможности для описания таких динамических предметных областей.

База знаний интеллектуальной системы включает в себя как модель фактографических знаний о предметной области, для которой предназначена система, так и модель знаний, включающая в себя логические формулы об этой предметной области (аксиомы, теоремы и правила вывода).

Абстрактная scl-машина является машиной логического вывода и относится к классу абстрактных sc-машин (см. [Голенков В.В..БазовПТЯSCL-1996мо](#)). Внутренним языком scl-машины является указанный выше графовый логический язык SCL, её операции соответствуют правилам логического вывода. Семейство специализированных абстрактных графодинамических машин обработки знаний является формальным уточнением операционной семантики указанных выше специализированных графовых языков представления знаний, каждому из которых соответствует одна или несколько абстрактных машин. Эти абстрактные машины соответствуют различным моделям решения задач, различным логикам, различным моделям правдоподобных рассуждений. Агент из семейства агентов логического вывода может представлять собой какое-либо правило вывода, которое можно применять для решения логической задачи. Кроме того, необходимы агенты для выполнения равносильных преобразований логической формулы (например, записать формулу эквиваленции как конъюнкцию двух дизъюнкций) и другие агенты, помогающие применять правила вывода на множестве формул языка логики.

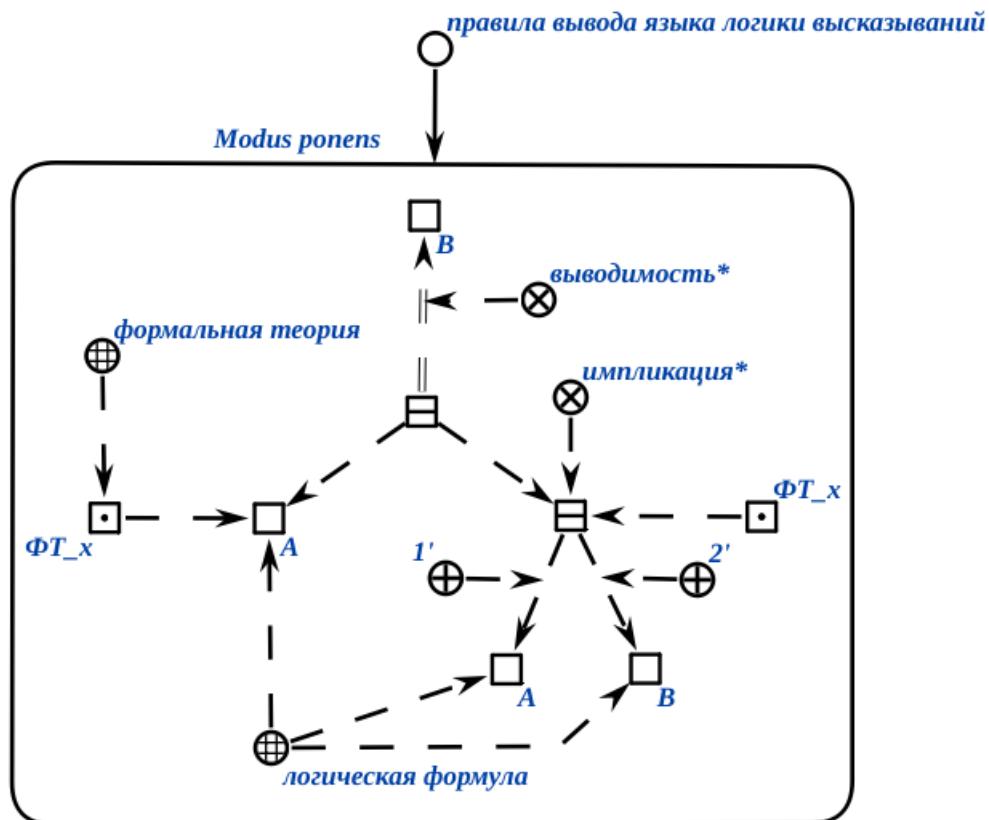
Абстрактная scl-машина

⇒ декомпозиция абстрактного sc-агента*:

- {• Абстрактный sc-агент применения правила вывода
 - Абстрактный sc-агент эквивалентных преобразований логической формулы
 - Абстрактный sc-агент прямого логического вывода
 - Абстрактный sc-агент обратного логического вывода

}

Задачей Абстрактного sc-агента применения правила вывода является применение заданного правила вывода с заданными логическими формулами. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие применение правила вывода*. После проверки sc-агентом условия инициирования выполняется процесс применения правила вывода, который заключается в проверке, существует ли в sc-памяти структуры, соответствующие условию применения данного правила и генерации sc-конструкций в соответствии с применяемым правилом. Агент применения правила вывода зачастую используется в процессе работы агентов прямого логического вывода, обратного логического вывода и других агентов. Примером правила вывода может быть правило Modus ponens, представленное на рисунке 3.5.1.

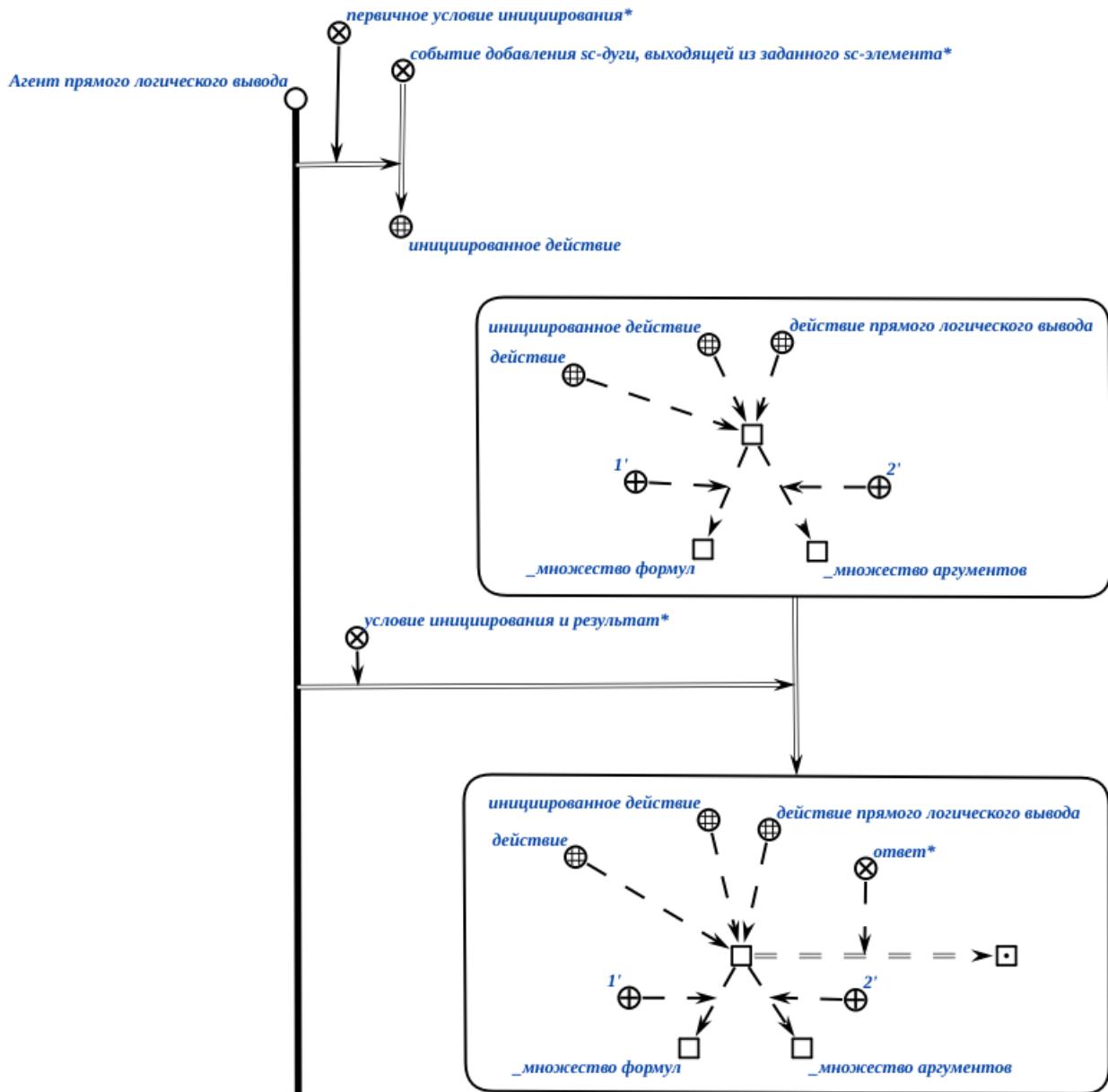


= Формализация правила вывода Modus ponens

Задачей Абстрактного sc-агента эквивалентных преобразований логической формулы является применение некоторых правил, которые приводят логическую формулу в определённый вид. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие эквивалентное преобразование логической формулы*. После проверки sc-агентом условия инициирования выполняется процесс преобразования формулы из одной формы в другую, при этом никакие новые знания в sc-памяти с точки зрения исследуемой предметной области не генерируются. Ответом данного агента является множество формул, эквивалентных по смыслу, но различных по форме представления. Такими формами могут быть, например, конъюнктивная нормальная форма или дизъюнктивная нормальная форма. Агент эквивалентных преобразований зачастую вызывается в процессе работы агента применения правила вывода, так как логические формулы не всегда находятся в той форме, которая доступна для применения того или иного правила вывода, однако может быть приведена к нужной форме.

Задачей Абстрактного sc-агента прямого логического вывода является генерации новых знаний на основе некоторых логических утверждений. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие прямого логического вывода*. После проверки sc-агентом условия инициирования выполняется процесс прямого логического вывода, который состоит из циклических операций применения правил вывода, генерации новых знаний в sc-памяти и проверки некоторого условия, например, появление в памяти sc-элементов из целевой sc-структурь (см. Гаврилова Т.А..Базы ЗИСУП-2000kn). Входными аргументами такого агента является целевая структура, множество формул, которые используются в ходе вывода агентом применения правил вывода, множество правил вывода, входная структура и выходная структура. В результате выполнения агентом логического вывода действия, в sc-памяти формируется sc-структурь, представляющая собой дерево решения. Это дерево состоит из последовательности узлов, представляющих собой применённые правила, которые привели к появлению в sc-памяти требуемых знаний. Такое дерево может быть пустым в случае, если требуемую структуру

не удалось сгенерировать в ходе логического вывода. На рисунке 3.5.2 приведён пример спецификации агента прямого логического вывода.



= Спецификация агента прямого логического вывода

Задачей Абстрактного sc-агента обратного логического вывода является проверка гипотез. Некоторые гипотезы могут быть опровергнуты, однако извлекая причины того, почему гипотеза опровергнута, можно изменить посылку гипотезы так, чтобы создать новую гипотезу, которая впоследствии может стать полезной теоремой. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие обратного логического вывода*. После проверки sc-агентом условия инициирования выполняется процесс обратного логического вывода, который схож с процессом прямого логического вывода за исключением того, что поиск правил основывается не на посылках формул, а на их следствиях (см. Гаврилова.Т.А..Базы ЗИСУП-2000кн.). Ответом данного агента будет также дерево вывода, которое показывает, с использованием каких правил можно доказать или опровергнуть выдвинутую гипотезу.

Абстрактный sc-агент эквивалентных преобразований логической формулы

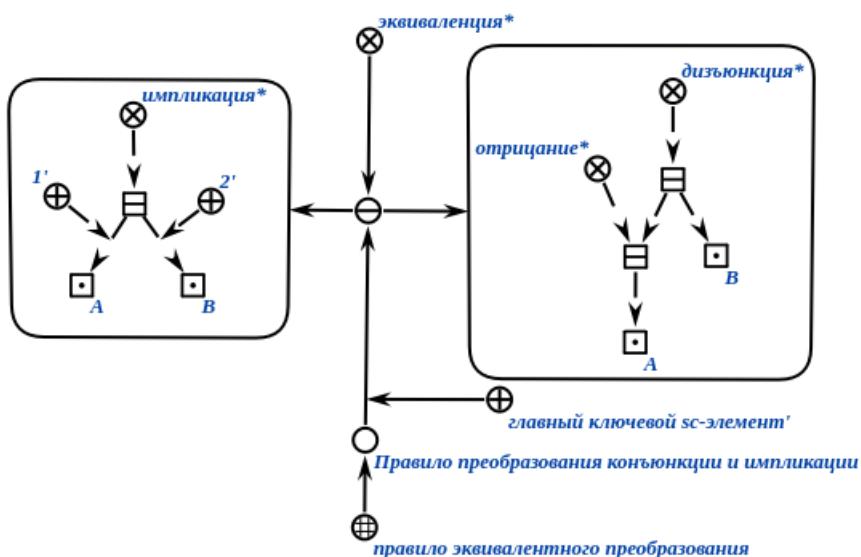
⇒ декомпозиция абстрактного sc-агента*:

- {• Абстрактный sc-агент преобразования формулы в конъюнктивную нормальную форму
- Абстрактный sc-агент преобразования формулы в дизъюнктивную нормальную форму

- Абстрактный sc-агент применения закона Де Моргана
 - Абстрактный sc-агент эквивалентных преобразований логической формулы по определению
 - Абстрактный sc-агент применения свойств отрицания логических формул
 - Абстрактный sc-агент применения закона идемпотентности логических формул
 - Абстрактный sc-агент применения закона коммутативности логических формул
 - Абстрактный sc-агент применения закона ассоциативности логических формул
 - Абстрактный sc-агент применения закона поглощения логических формул
 - Абстрактный sc-агент применения закона противоречия логических формул
 - Абстрактный sc-агент применения закона двойного отрицания логических формул
 - Абстрактный sc-агент применения закона расщепления логических формул
- }

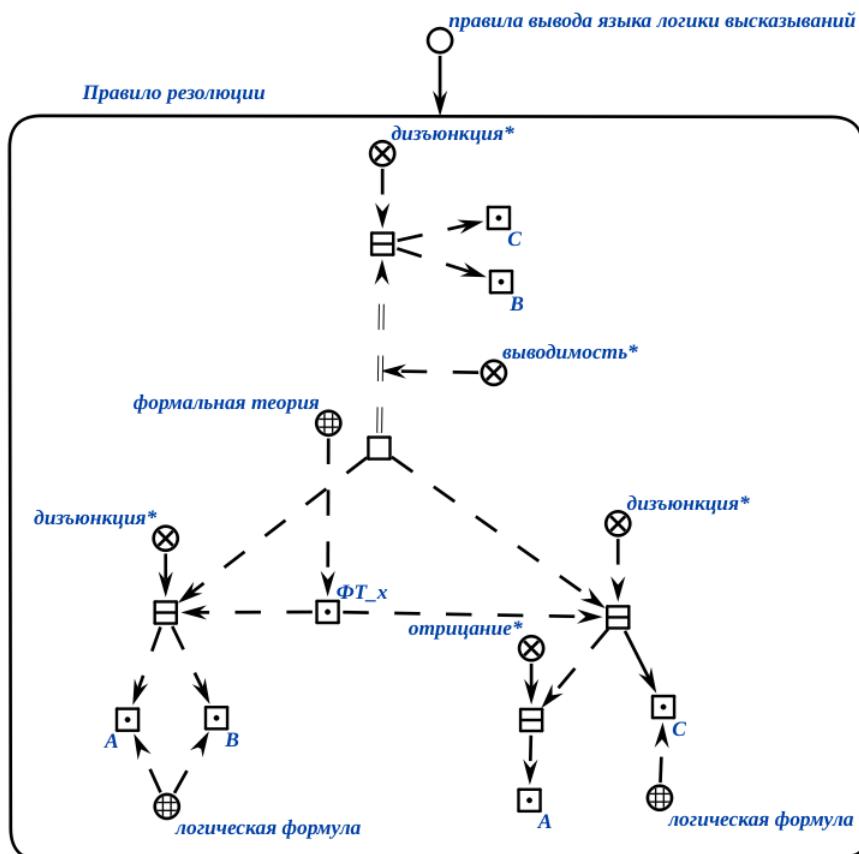
Любая формула эквивалентна некоторой формуле в конъюнктивной нормальной форме, в связи с этим иногда удобно применять правило резолюции. Используя законы Де Моргана можно также получить формулы, пригодные для использования правила резолюции. С помощью правила резолюции можно эффективно доказывать формулы языка логики высказываний.

Однако ничего принципиально нового правило резолюции не привносит, поскольку формула $A \Rightarrow B$ равносильно $\neg A \vee B$ и из выводимости A и $A \rightarrow B$ следует выводимость B .



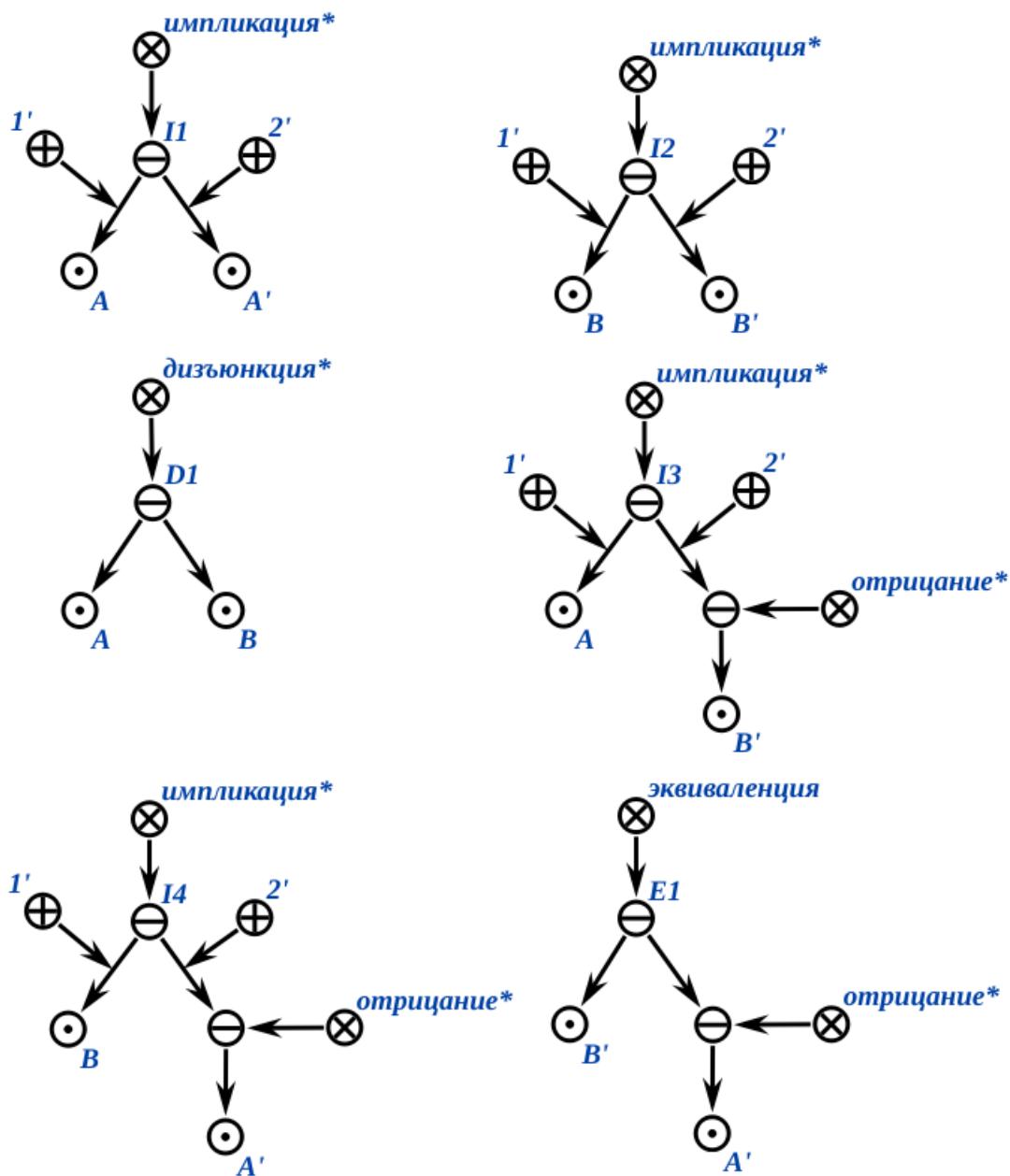
= *Формализация конъюнктивной нормальной формы для импликации*

Если в любых двух дизъюнктах C_1 и C_2 имеется пара формул A и $\neg A$, то можно сформировать новый дизъюнкт из оставшихся частей изначальных дизъюнктов.



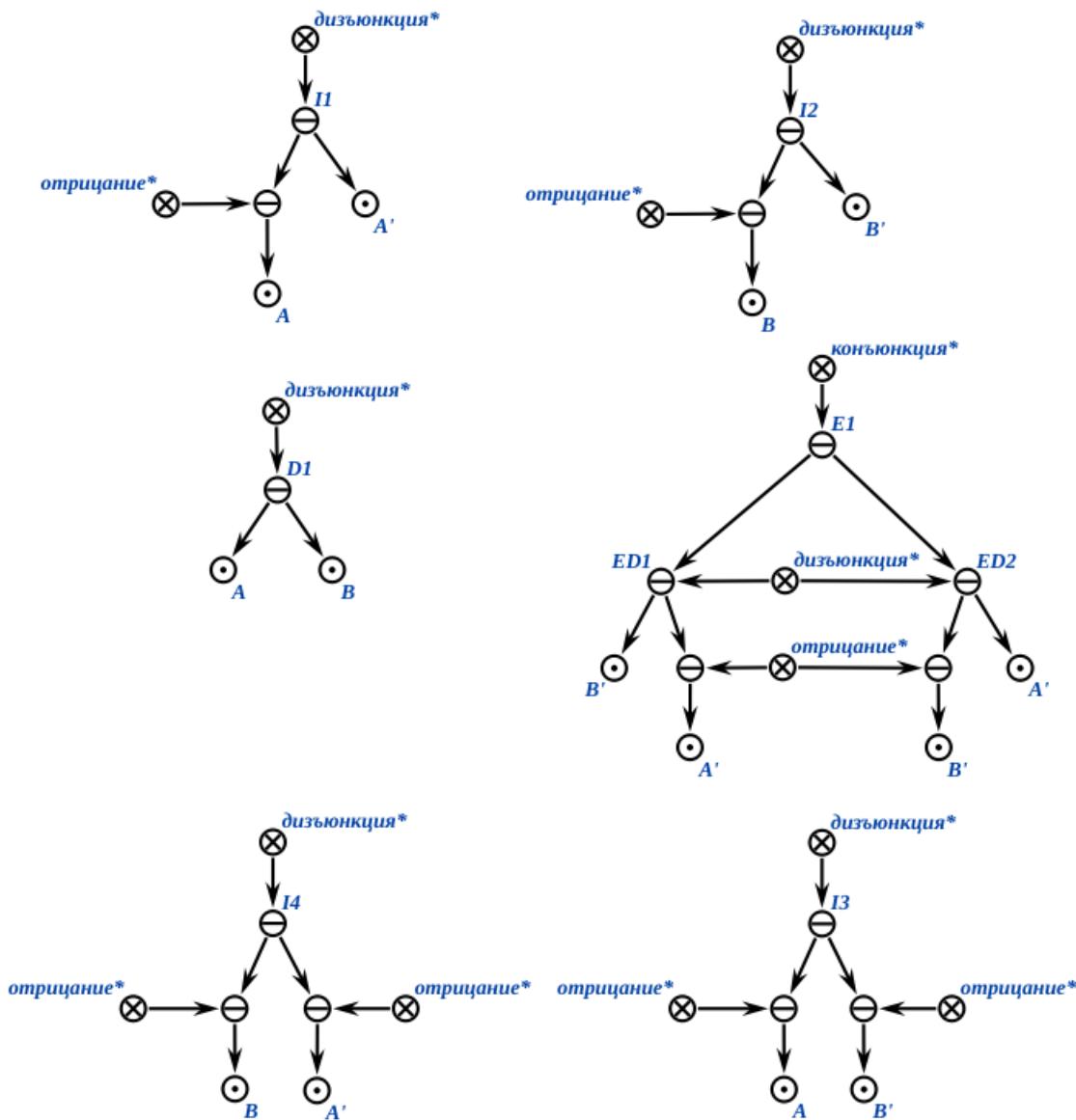
= Формализация правила резолюции

Приведём пример вывода формулы из множества посылок принципом резолюции. Если команда А выигрывает в футбол, то город А' торжествует, а если выигрывает команда В, то торжествовать будет город В'. Выиграть может или только город А', или только город В'. Однако, если выигрывает команда А, то город В' не торжествует, а если выигрывает команда В, то не торжествует город А'. Следовательно, город В' торжествует тогда и только тогда, когда не будет торжествовать город А'. Цель логического вывода - удостовериться, что город В' торжествует тогда и только тогда, когда не будет торжествовать город А'. Доказать вывода формулы равносильно доказательству противоречивости вывода отрицания этой формулы. При использовании правила резолюции это особенно удобно использовать. Формализация логических формул, соответствующих примеру приведена на рисунке ниже. Каждая неатомарная формула на рисунке принадлежит некоторой формальной теории, то есть считается истинной.



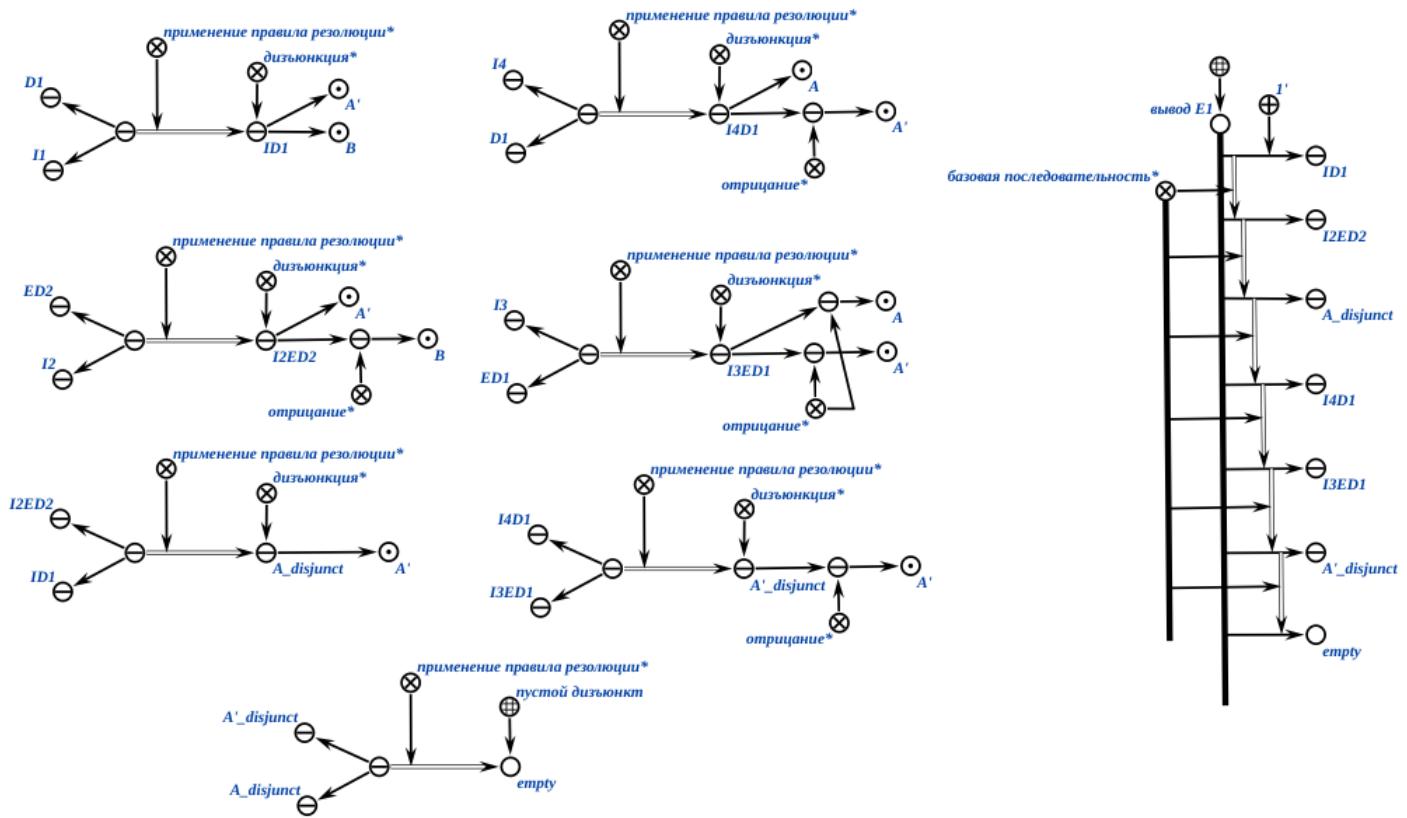
= Формализация правил для применения правила резолюции

Структура А представляет собой атомарную логическую формулу, которая обозначает победу команды А, структура А' представляет формулу, обозначающую торжество города А'. Соответственно, то же самое для структур В и В'. Прежде всего необходимо привести импликацию в конъюнктивную нормальную форму по формуле 3.5..3 и эквиваленцию по определению. А также применим отрицание к формуле, которую необходимо вывести (эквиваленция). В результате получим следующие формулы:



= Формализация правил для применения правила резолюции после преобразования в конъюнктивную нормальную форму

Далее применяя правило резолюции для преобразованных формул получаем пустой дизъюнкт, что говорит о противоречивости множества формул и доказывает формулу эквиваленции о том, что город В' торжествует тогда и только тогда, когда не будет торжествовать город А'.



= Применение принципа резолюции

Таким образом, можно использовать различные правила вывода, различные агенты логического вывода в различных видах логик в зависимости от специфики предметной области, в которой решается задача. Каждая модель является совместимой в рамках общей формальной модели решателей задач ostis-систем.

§ 3.5.2. Языки продукционного программирования, используемые ostis-системами

Пункт 3.5.2.1. Синтаксис языков продукционного программирования, используемых ostis-системами

Пункт 3.5.2.2. Денотационная семантика языков продукционного программирования, используемых ostis-системами

Пункт 3.5.2.3. Операционная семантика языков продукционного программирования, используемых ostis-системами

Глава 3.6.

Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах

⇒ *автор**:

- Ковалёв М.В.
- Крощенко А.А.
- Головко В.А.

⇒ *аннотация**:

[В главе рассмотрен подход к интеграции и конвергенции искусственных нейронных сетей с базами знаний в интеллектуальных компьютерных системах нового поколения с помощью представления и интерпретации искусственных нейронных сетей в базе знаний. Описаны синтаксис, денотационная и операционная семантика языка представления нейросетевых методов в базах знаний. Описаны этапы построения нейросетевых методов решения задач с помощью интеллектуальной среды проектирования искусственных нейронных сетей.]

⇒ *подраздел**:

- *Модели искусственных нейронных сетей, используемых в ostis-системах* *Модели искусственных нейронных сетей, используемых в ostis-системах*
- *Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем* *Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

⇒ *ключевые элементы**:

- *нейросетевой метод решения задач*
- *язык представления нейросетевых методов решения задач*
- *нейросетевая модель решения задач*
- *навык решения задач с помощью искусственных нейронных сетей*
- *действие по построению искусственных нейронных сетей*

⇒ *библиографическая ссылка**:

- *Castelvecchi.B. AI BlackBox-2016*
- *Головко В.А.. НейроТОД-2017кн*
- *Гудфеллоу.Я.. ГлубОбуч-2017кн*

Введение в Главу 3.6.

Современные решатели задач интеллектуальных систем все чаще сталкиваются с необходимостью решения комплексных задач с помощью различных традиционных и интеллектуальных методов решения задач в едином информационном ресурсе (в пределе – в единой базе знаний).

С другой стороны, интеллектуальные компьютерные системы нового поколения обладают, среди прочих, следующими способностями **Standard2021**:

- способность постоянно повышать качество решения задач;
- способность приобретать навыки решения принципиально новых задач;
- способность обосновывать свои решения;
- способность находить и устранять ошибки в своих решениях (способность к интроспекции).

Представление различных методов решения задач в единой базе знаний обеспечивает семантическую совместимость этих методов. Решая задачу с помощью таких методов, система не взаимодействует с ними по принципу "входов-выходов". Напротив, единая память позволяет отслеживать преобразование входных знаний в реальном времени с помощью любых имеющихся методов, что обеспечивает способность к интроспекции и способность объяснять решения системы.

Перспективными и активно развивающимися методами решения задач являются *искусственные нейронные сети* (и.н.с.), что обуславливается, с одной стороны, развитием теории и.н.с., а с другой – аппаратных возможностей машин, которые используются для их обучения.

Достоинствами и.н.с. можно назвать способность решения задач при неизвестных закономерностях, а так же способность решения задач без необходимости разработки проблемоориентированных подходов.

Большинство нейросетевых моделей работают как "черный ящик" *Castelvecchi.B.AIBlackBox-2016*, что является одним из основных недостатков этого метода решения задач. Большой объём обрабатываемых этими моделями данных создает необходимость мониторинга, объяснения и понимания механизмов их работы с целью вербализации оценки и оптимизации деятельности и.н.с.

Современные задачи все чаще требуют обоснования своего решения. Появилось целое направление Explainable AI, в рамках которого предпринимаются различные попытки объяснить решения и.н.с. (*Ribeiro, Singh u Guestrin 2016, Lundberg u Lee 2017*). Развиваются подходы, предлагающие интеграцию нейронных сетей с базами знаний *Tarek.R..NeuralSymbolic-2017; Garcez и др. 2015; Kroshchanka и др. 2022*.

Еще одним недостатком и.н.с. можно назвать эвристический характер процесса подбора архитектур моделей и параметров их обучения и высокие требования к объему знаний проектировщиков нейросетевых моделей.

Исходя из перечисленных способностей, наличие которых необходимо обеспечивать в интеллектуальных компьютерных системах нового поколения, встает проблема разработки подхода к интеграции и.н.с. в базу знаний интеллектуальной системы как в качестве метода решения задач, так и в качестве объекта автоматического проектирования новых методов. Решение этой проблемы позволит преодолеть указанные выше недостатки нейросетевого метода.

Можно выделить два основных направления интеграции и.н.с. с базами знаний:

- построение интеллектуальных систем, способных использовать нейросетевые методы наравне с другими имеющимися в системе методами для решения комплексных задач. Такие системы смогут учитывать семантику решаемых задач на более высоком уровне, что сделает решения более структурированными и прозрачными.
- построение интеллектуальной среды по разработке, обучению и интеграции различных и.н.с., совместимых с базами знаний через представление и.н.с. с помощью онтологических структур и их интерпретацию средствами представления знаний. Такая среда предоставит возможность интроспекции и.н.с., возможность сохранения состояний и.н.с. после обучения и реконфигурации сети, что позволит производить более глубокий анализ ее работы. Так же формальное описание знаний в рамках предметной области и.н.с. поможет уменьшить порог вхождения разработчиков в область методов решения задач с помощью и.н.с.

§ 3.6.1. Модели искусственных нейронных сетей, используемых в ostis-системах

Как уже было описано в главе [Формализация понятий действия, задачи, метода, средства, навыка и технологии](#), решатель задач занимается обработкой фрагментов базы знаний. На операционном уровне обработка сводится к добавлению, поиску, редактированию и удалению sc-узлов и sc-коннекторов базы знаний. На семантическом же уровне такая операция является *действием, выполняемым в памяти субъекта действия*, где, в общем случае, субъектом является ostis-система, а база знаний – ее памятью. *Действие* определяется как процесс воздействия одной сущности (или некоторого множества сущностей) на другую сущность (или на некоторое множество других сущностей) в соответствии с некоторой целью.

Действия выполняются в соответствии с поставленными задачами. *Задача* – это формальная спецификация некоторого действия, достаточная для выполнения данного действия каким-либо субъектом. В зависимости от конкретного класса задач, описываться может как внутреннее состояние самой интеллектуальной системы, так и требуемое состояние внешней среды.

Схожие задачи объединены в классы, для которых заданы обобщенные формулировки задач. Для и.н.с. выделены следующие классы задач:

- *задача классификации*. Задача построения классификатора, т.е. отображения $\tilde{c} : X \rightarrow C$, где $X \in \mathbb{R}^m$ – признаковое пространство входного образа, $C = C_1, C_2, \dots, C_k$ – конечное и обычно небольшое множество меток классов.
- *задача регрессии*. Задача построения оценочной функции по примерам $(x_i, f(x_i))$, где $f(x)$ – неизвестная функция. *Оценочная функция* – отображение вида $\hat{f} : X \rightarrow \mathbb{R}$, где $X \in \mathbb{R}^m$ – признаковое пространство входных данных.
- *задача кластеризации*. Задача построения функции $a : X \rightarrow Y$, которая любому объекту $x \in X$ ставит в соответствие номер кластера $y \in Y$ в соответствие с определенной метрикой расстояния $\rho(x, x')$, где X – множество объектов, Y – множество номеров (имен, меток) кластеров, $x, x' \in X$.
- *задача понижения размерности признакового пространства*. Задача построения функции $h : X \rightarrow Y$, сохраняющей заданные соотношения между точками множеств X и Y , где $X \subset \mathbb{R}^p$, $Y = h(X) \subset \mathbb{R}^q$, $q < p$;
- *задача управления*. Задача построения модели-регулятора состояния сложного динамического объекта;
- *задача фильтрации*. Задача построения модели, которая производит очистку исходного сигнала, содержащего некоторый шум и уменьшает влияние случайных ошибок в сигнале.

- *задача детекции.* Является частным случаем задачи классификации и задачи регрессии. Задача построения модели, осуществляющей обнаружение объектов определенных типов на фото- и видеоизображениях.
- *задача с ассоциативной памятью.* Задача построения модели, позволяющей выполнить реконструкцию исходного образа на основании сохраненных ранее образов.

Для классов задач формулируются классы методов их решения. *Метод решения задач* определяется как программа решения задач соответствующего класса, которая может быть как процедурной, так и декларативной. В свою очередь, *класс методов решения задач* определяется как множество всевозможных методов решения задач, имеющих общий язык представления этих методов. Язык представления методов позволяет описывать синтаксическую, денотационную и операционную семантику этого метода.

Предлагается рассматривать и.н.с. как класс методов решения задач со своим языком представления. Таким образом, искусственная нейронная сеть - это *нейросетевой метод решения задач*. В соответствии с технологией OSTIS, спецификация класса методов решения задач сводится к спецификации соответствующего языка представления методов, т.е. к описанию его синтаксической, денотационной и операционной семантики.

Для достижения семантической совместимости с другими методами решения задач технологии OSTIS, предлагается описывать нейросетевые методы внутри семантической памяти, соответственно, синтаксис языка представления нейросетевых методов решения задач является синтаксисом SC-кода, использующимся в технологии OSTIS для представления знаний.

Таким образом, чтобы добавить в арсенал технологии OSTIS нейросетевые методы решения задач и тем самым расширить круг задач, решаемых ostis-системами, необходимо описать денотационную и операционную семантику *языка представления нейросетевого метода решения задач*.

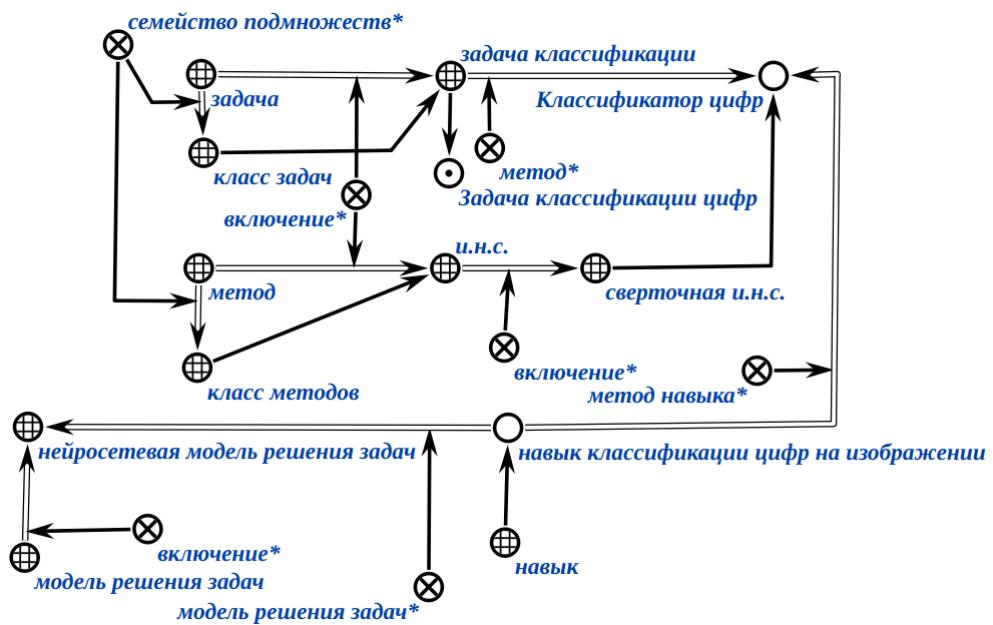
Денотационная семантика языка представления нейросетевого метода решения задач описывается в рамках предметной области и соответствующей ей онтологии нейросетевого метода.

Операционной семантикой любого языка представления методов решения задач является спецификация семейства агентов, обеспечивающих интерпретацию любого метода, принадлежащего соответствующему классу методов. Это семейство является интерпретатором соответствующего метода решения задач. В рамках технологии OSTIS такой интерпретатор называется *моделью решения задач*. Так как в рамках технологии OSTIS используется многоагентный подход, то разработка нейросетевой модели решения задач сводится к разработке агентно-ориентированной модели интерпретации и.н.с.

Навыком называется метод, интерпретация которого полностью может быть осуществлена данной кибернетической системой, в памяти которой хранится указанный метод. Таким образом, формируя спецификацию в ostis-системе для нейросетевого метода решения задач и нейросетевой модели решения задач можно говорить о наличии у такой системы *навыка решения задач с помощью и.н.с.*

На рисунке [Фрагмент теоретико-множественной онтологии и.н.с.](#) представлен фрагмент теоретико-множественной онтологии и.н.с., описывающий связь таких понятий и узлов, как:

- класс задач, решаемых с помощью и.н.с.(для примера, взят класс задач классификации);
- класс нейросетевых методов решения задач;
- нейросетевая модель решения задач;
- навык решения задач с помощью и.н.с.;
- конкретные задачи и методы их решения(для примера взята конкретная обученная сверточная и.н.с.).



= Фрагмент теоретико-множественной онтологии и.н.с.

Использование и.н.с. как метода решения задач подразумевает использование уже спроектированной и обученной и.н.с. Однако наличие языка описания нейросетевого метода решения задач в памяти ostis-системы открывает дорогу для автоматизации самих процессов проектирования и обучения и.н.с. Такая автоматизация представляется отдельными классами задач и соответствующими навыками их решения. Подход к такой автоматизации описан в § 3.6.2. *Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем.*

Пункт 3.6.1.1. Денотационная семантика моделей искусственных нейронных сетей, используемых в ostis-системах

Как уже было сказано, денотационная семантика языка представления нейросетевого метода описывается в рамках предметной области (ПрО) и соответствующей ей онтологии нейросетевого метода. ПрО нейросетевого метода является дочерней ПрО метода.

Максимальным классом объектов исследования предметной области искусственных нейронных сетей является *искусственная нейронная сеть*.

искусственная нейронная сеть

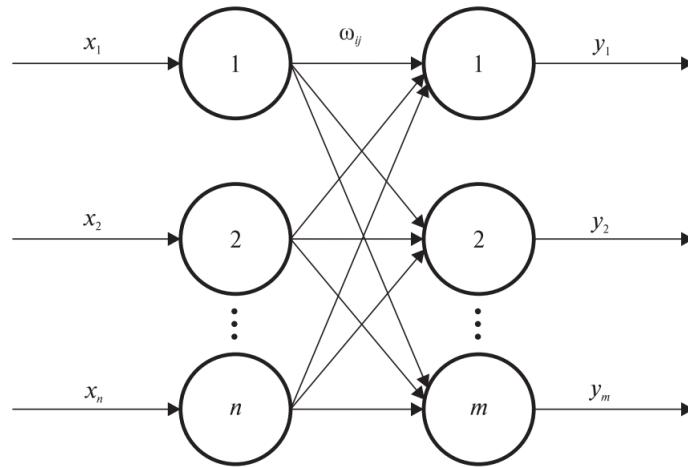
- := [и.н.с.]
- := [нейросетевой метод]
- := [множество искусственных нейронных сетей]
- := [нейронная сеть]
- ⇒ пояснение*:

[Совокупность нейронных элементов и связей между ними [\[Головко В.А..НейроТОД-2017кн.\]](#).]

Искусственная нейронная сеть состоит из *формальных нейронов*, которые связаны между собой посредством *синаптических связей*. Нейроны организованы в *слои*. Каждый нейрон слоя принимает сигналы со входящих в него синаптических связей, обрабатывает их единым образом с помощью заданной ему или всему слою *функции активации* и передает результат на выходящие из него синаптические связи.

Архитектурой и.н.с. будем называть совокупность информации о структуре ее слоев, формальных нейронов, синаптических связей и функций активаций. Т.е. то, что можно обучить и использовать для решения задач.

Пример архитектуры и.н.с. представлен на рисунке [Пример архитектуры и.н.с..](#)



= Пример архитектуры и.н.с.

В соответствии с тем, какая у и.н.с. архитектура, можно выделить следующую иерархию классов и.н.с.. Рассмотрим эту иерархию в SCn-коде.

искусственная нейронная сеть

:= [нейросетевой метод]

⇐ включение*:

метод

⇒ разбиение*:

Типология и.н.с. по признаку направленности связей[^]

= {• и.н.с. с прямыми связями

⇒ декомпозиция*:

{• персептрон

⇒ декомпозиция*:

{• персептрон Розенблатта

• автоэнкодерная и.н.с.

}

• машина опорных векторов

• и.н.с. сеть радиально-базисных функций

• сверточная и.н.с.

}

• и.н.с. с обратными связями

⇒ декомпозиция*:

{• и.н.с. Хопфилда

• и.н.с. Хэмминга

}

• рекуррентная искусственная нейронная сеть

⇒ декомпозиция*:

{• и.н.с. Джордана

• и.н.с. Элмана

• мультирекуррентная и.н.с.

• LSTM-элемент

• GRU-элемент

}

}

⇒ разбиение*:

Типология и.н.с. по признаку полноты связей[^]

= {• полносвязная и.н.с.

• слабосвязная и.н.с.

}

Рассмотрим архитектурные компоненты подробнее.

формальный нейрон

- \coloneqq [искусственный нейрон]
- \coloneqq [нейрон]
- \coloneqq [ф.н.]
- \coloneqq [нейронный элемент]
- \coloneqq [множество нейронов искусственных нейронных сетей]
- \coloneqq [математическая модель биологического нейрона]
- \subset *искусственная нейронная сеть*
- \Rightarrow *пояснение*:*
[Основной элемент *и.н.с.*, применяющий свою *функцию активации* [*Головко В.А..НейроТОД-2017кн*] к сумме произведений входных сигналов на весовые коэффициенты:

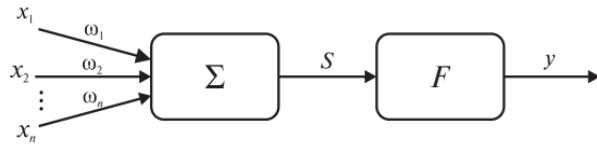
$$y = F \left(\sum_{i=1}^n w_i x_i - T \right) = F(WX - T)$$

где $X = (x_1, x_2, \dots, x_n)^T$ – вектор входного сигнала; $W = (w_1, w_2, \dots, w_n)$ – вектор весовых коэффициентов; T – пороговое значение; F – *функция активации*.]

Отдельный формальный нейрон является искусственной нейронной сетью с одним нейроном в единственном слое. Формальные нейроны могут быть классифицированы следующим образом:

- полносвязный формальный нейрон – нейрон, у которого есть полный набор связей с нейронами предшествующего слоя. Отдельный обрабатывающий элемент *и.н.с.*, выполняющий функциональное преобразование взвешенной суммы элементов вектора входных значений с помощью *функции активации*
- сверточный формальный нейрон – отдельный обрабатывающий элемент *и.н.с.*, выполняющий функциональное преобразование результата операции свертки матрицы входных значений с помощью *функции активации*. Сверточный *формальный нейрон* может быть представлен полносвязным *формальным нейроном*.
- рекуррентный *формальный нейрон* – нейрон, имеющий обратную связь с самим собой или с другими нейронами *и.н.с.*

Схематически формальный нейрон можно представить в виде следующей модели (рисунок [Формальный нейрон](#)).

**= Формальный нейрон**

Определим понятия *синаптической связи* и *слоя и.н.с.*:

синаптическая связь

- \coloneqq [синапс]
- \subset *ориентированная пара*
- \Rightarrow *пояснение*:*
[ориентированная пара, первым компонентом которой является нейрон, из которого исходит сигнал, а вторым компонентом – нейрон, который принимает этот сигнал]

слой и.н.с.

- \coloneqq [слой]
- \coloneqq [слой искусственной нейронной сети]
- \coloneqq [множество слоев искусственных нейронных сетей]
- \subset *искусственная нейронная сеть*
- \Rightarrow *пояснение*:*
[множество нейронных элементов, на которые в каждый тик времени параллельно поступает информация от других нейронных элементов сети *Головко В.А..НейроТОД-2017кн*]
- \Rightarrow *пояснение*:*
[множество формальных нейронов, осуществляющих параллельную независимую обработку вектора или матрицы входных значений]

Отдельный слой является искусственной нейронной сетью с одним слоем. Следует отметить принципиальную важность этого замечания. Один *слой и.н.с.* уже является нейронной сетью, поскольку над ним можно производить все основные операции, которые производятся над "большой" *и.н.с.* (его можно обучить и использовать для решения определенной задачи).

Слои и.н.с. могут быть классифицированы следующим образом (по признаку операции, осуществляющейся слоем):

- полносвязный слой и.н.с. – слой, в котором каждый нейрон является полносвязным
- сверточный слой и.н.с. – слой, в котором каждый нейрон является сверточным
- слой и.н.с. нелинейного преобразования – слой, осуществляющий нелинейное преобразование входных данных
- dropout слой и.н.с. – слой, реализующий технику регуляризации dropout
- pooling слой и.н.с. – подвыборочный слой
- слой и.н.с. батч-нормализации

Как правило, слой нелинейного преобразования выделяется в отдельный слой только в программных реализациях. Фактически он рассматривается как финальный этап расчета выходной активности любого нейрона – применение *функции активации*.

Dropout-слой функционирует только во время обучения *и.н.с.*. Поскольку полносвязные слои имеют большое количество настраиваемых параметров, они подвержены эффекту *переобучения*. Один из способов устраниć такой негативный эффект – выполнить частичный отсев результатов на выходе полносвязного слоя. На этапе обучения техника dropout позволяет отбросить выходную активность некоторых нейронов с определенной, заданной вероятностью. Выходная активность "отброшенных" нейронов полагается равной нулю.

Назначение подвыборочного слоя – в осуществлении уменьшения размерности входных данных.

Нужно отметить, что данный перечень неполный – разновидности *слоев и.н.с.* появляются практически в каждой заслуживающей внимания публикации по нейросетевым алгоритмам и на текущий момент их существует достаточно много, однако, как правило, при построении более традиционных архитектур ограничиваются только приведенными вариантами слоев.

Слои и.н.с. также могут быть классифицированы по исполняемой роли в рамках архитектуры (место в последовательности *слоев и.н.с.*).

Так, например, слой, расположенный первым, называется распределяющим. Слои, расположенные далее, за исключением последнего, называются обрабатывающими. Наконец, последний слой носит название выходного *слоя и.н.с.*

Последний архитектурный компонент *и.н.с.* – это функция активации:

*функция активации**

- := [функция активации нейрона*]
- Є *неролевое отношение*
- Є *бинарное отношение*
- ⇒ *пояснение*:*
 - [неролевое отношение, связывающее формальный нейрон с функцией, результат применения которой к *взвешенной сумме нейрона* определяет его *выходное значение*.]
- ⇒ *первый домен*:*
 - формальный нейрон*
- ⇒ *второй домен*:*
 - функция*

Перечислим некоторые, наиболее известные и применяемые типы функций активации:

- линейная функция
 - ⇒ *формула*:*
 - [

$$y = kS$$

где k – коэффициент наклона прямой, S – в.с.]

- пороговая функция
 - ⇒ *формула*:*
 - [

$$y = \text{sign}(S) = \begin{cases} 1, & S > 0, \\ 0, & S \leq 0 \end{cases}$$

]

- сигмоидная функция
 - ⇒ *формула*:*

[

$$y = \frac{1}{1 + e^{-cS}}$$

где $c > 0$ – коэффициент, характеризующий ширину сигмоидной функции по оси абсцисс, S – в.с.]

- функция гиперболического тангенса

 \Rightarrow формула*:

[

$$y = \frac{e^{cS} - e^{-cS}}{e^{cs} + e^{-cS}}$$

где $c > 0$ – коэффициент, характеризующий ширину сигмоидной функции по оси абсцисс, S – в.с.]

- функция softmax

 \Rightarrow формула*:

[

$$y_j = \text{softmax}(S_j) = \frac{e^{S_j}}{\sum_j e^{S_j}}$$

где S_j – в.с. j -го выходного нейрона]

- функция ReLU

 \Rightarrow формула*:

[

$$y = F(S) = \begin{cases} S, S > 0, \\ kS, S \leq 0 \end{cases}$$

где $k = 0$ или принимает небольшое значение, например, 0.01 или 0.001.]

В рамках предметной области formalizovana ierarkhia parametrov i.n.c..

параметр и.н.с. \subset параметр \Rightarrow разбиение*:

= {• настраиваемый параметр и.н.с.

 \Rightarrow декомпозиция*:

- {• весовой коэффициент синаптической связи
- пороговое значение
 - ядро свертки
- }

• архитектурный параметр и.н.с.

 \Rightarrow декомпозиция*:

- {• количество слоев
- количество нейронов
 - количество синаптических связей
- }

}

}

Так же в ПрО нейросетевых методов добавлены понятия для описания метрик эффективности нейросетевых методов. Данные метрики учитываются решателем задач при принятии решения об использовании того или иного нейросетевого метода.

Метрики могут быть классифицированы по типу решаемой задачи.

метрика оценки качества и.н.с. \Rightarrow разбиение*:Типология метрик по признаку решаемой задачи^Λ

= {• классификационные метрики

 \Rightarrow декомпозиция*:

- {• точность и.н.с.
- полнота и.н.с.
 - $F1$ -метрика
- }

• регрессионные метрики

 \Rightarrow декомпозиция*:

- {• MAE
- MAPE
- }

$$\left. \begin{array}{c} \bullet \quad RMSE \\ \} \\ \} \end{array} \right.$$

точность и.и.с.

- \coloneqq [precision]
- \coloneqq [доля верно идентифицированных положительных исходов в общем числе исходов, которые были идентифицированы как положительные]
- \Rightarrow формула*:

[

$$PRE = \frac{TP}{TP + FP}$$

где TP и FP – число истинно-положительных и ложно-положительных предсказаний нейронной сети соответственно]

полнота и.и.с.

- \coloneqq [recall]
- \coloneqq [доля верно идентифицированных положительных исходов в общем числе положительных исходов]
- \Rightarrow формула*:

[

$$REC = \frac{TP}{TP + FN}$$

где TP и FN – число истинно-положительных и ложно-отрицательных предсказаний нейронной сети соответственно]

F1-метрика

- \Rightarrow формула*:

[

$$F1 = 2 * \frac{PRE * REC}{PRE + REC}$$

где PRE и REC – точность и полнота и.и.с. соответственно]

MAE

- \coloneqq [mean absolute error]
- \Rightarrow формула*:

$$\left[\frac{1}{N} \sum_{i=1}^N |y_{\text{еталон}}^i - y_{\text{predicted}}^i|, \right.$$

$y_{\text{еталон}}^i$ – эталонное значение,
 $y_{\text{predicted}}^i$ – значение, полученное и.и.с.,
 N – объем обучающей выборки]

MAPE

- \coloneqq [mean absolute percentage error]
- \Rightarrow формула*:

$$\left[\frac{1}{N} \sum_{i=1}^N \frac{|y_{\text{еталон}}^i - y_{\text{predicted}}^i|}{y_{\text{еталон}}^i} * 100\%, \right.$$

$y_{\text{еталон}}^i$ – эталонное значение,
 $y_{\text{predicted}}^i$ – значение, полученное и.и.с.,
 N – объем обучающей выборки]

RMSE

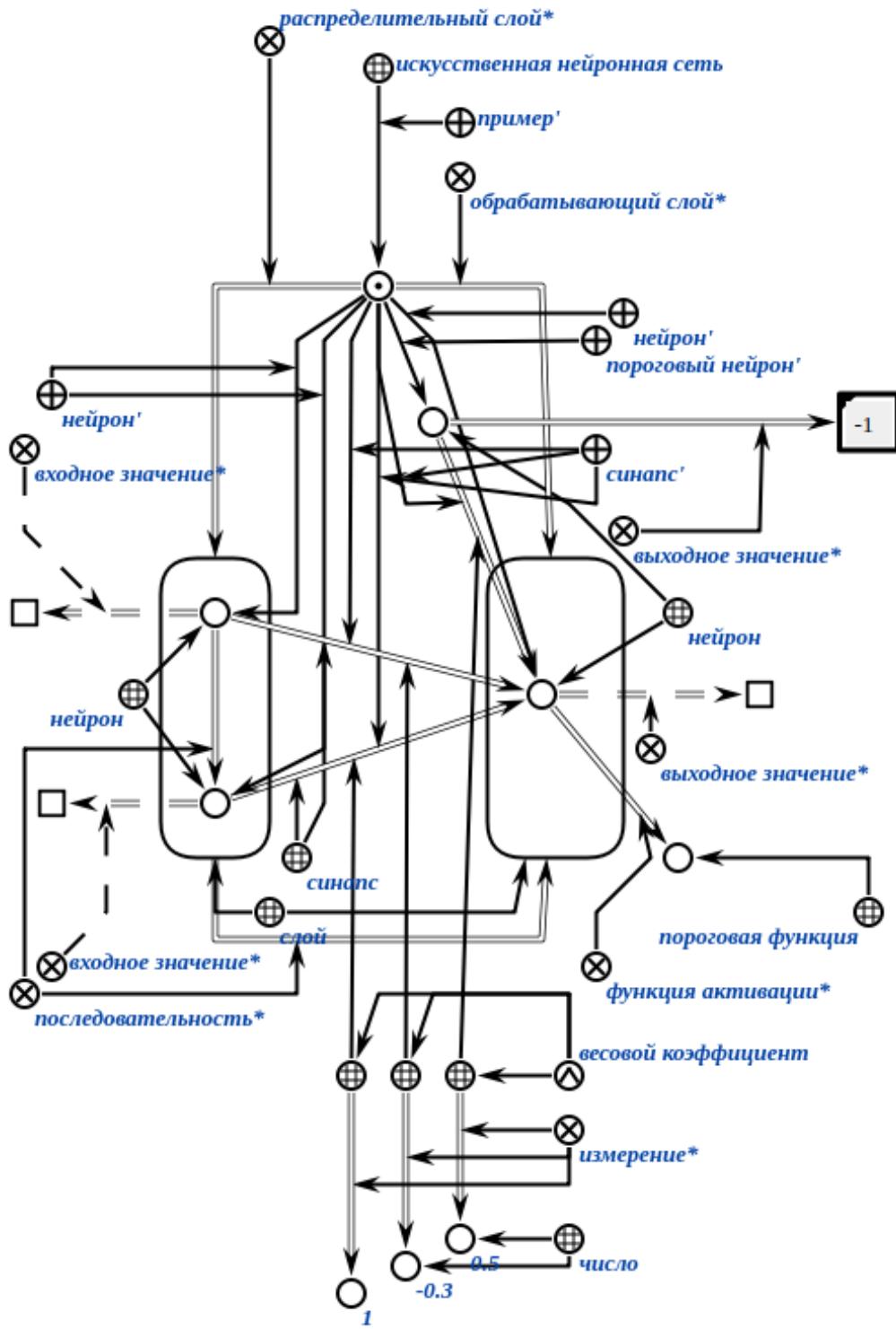
- \coloneqq [root mean squared error]
- \Rightarrow формула*:

$$\left[\sqrt{\frac{1}{N} \sum_{i=1}^N (y_{\text{еталон}}^i - y_{\text{predicted}}^i)^2}, \right.$$

$y_{\text{еталон}}^i$ – эталонное значение,
 $y_{\text{predicted}}^i$ – значение, полученное и.и.с.,
 N – объем обучающей выборки]

С помощью выделенных понятий становится возможна формализация в базе знаний архитектуры конкретных и.и.с.. В качестве примера, на рисунке [Пример формализации архитектуры искусственной нейронной сети в базе](#)

[знаний](#) представлен пример формализации полносвязной двухслойной и.н.с. с двумя нейронами на входном слое и одном нейроне на обрабатывающем слое.



= Пример формализации архитектуры искусственной нейронной сети в базе знаний

Следует отметить, что в практике авторов еще не было необходимости явно представлять и.н.с., как это показано на рисунке [Пример формализации архитектуры искусственной нейронной сети в базе знаний](#). Чаще всего, представление и.н.с. сводилось к представлению ее операционной семантики в виде SCP-программы, как это будет показано далее.

Пункт 3.6.1.2. Операционная семантика моделей искусственных нейронных сетей, используемых в ostis-системах

Операционная семантика языка представления нейросетевого метода задается агентно-ориентированной моделью интерпретации искусственных нейронных сетей и спецификацией соответствующих действий.

Нейросетевой метод описан в виде программы на некотором языке программирования, который может быть как внешним по отношению к ostis-системе, так и внутренним (на данный момент, SCP). Каждому такому языку программирования соответствует некоторая дочерняя предметная область ПрО нейросетевых методов.

Предметная область нейросетевых методов

:= [Предметная область искусственных нейронных сетей]
 ⇒ *дочерняя предметная область**:
 {• Предметная область нейросетевых методов SCP
 • Предметная область нейросетевых методов Python
 • Предметная область нейросетевых методов C++
 }

В случае описания нейросетевого метода на внешнем языке, такой метод описывается в соответствующей предметной области, в рамках которой также специфицируется действие интерпретации данного метода. Данному действию соответствует агент, реализованный на соответствующем языке программирования.

Однако для достижения конвергенции и интеграции необходимо описывать нейросетевые методы на внутреннем языке ostis-системы, которым является SCP.

SCP-программа является последовательностью обобщенных спецификаций (шаблонов) scp-операторов. Каждый scp-оператор является действием в памяти ostis-системы (sc-памяти). При интерпретации scp-программы, абстрактный sc-агент создания scp-процессов создает scp-процесс с учетом конкретных параметров интерпретации scp-программы, что на операционном уровне сводится к подстановке аргументов в обобщенные спецификации scp-операторов программы и генерации конкретных экземпляров этих программ(методов). Далее происходит добавление начального оператора во множество настоящих сущностей и начинается выполнение программы.

Таким образом, интерпретация scp-программы сводится к агентно-ориентированной обработке действий в sc-памяти. Этими действиями являются scp-операторы.

действие интерпретации слова и.н.с.

⇒ *декомпозиция**:
 {• действие вычисления взвешенной суммы всех нейронов слова
 • действие вычисления функции активации всех нейронов слова
 • действие интерпретации сверточного слова
 • действие интерпретации пулинг слова
 }

Для описания спецификации указанных действий необходимо ввести понятия *ориентированного множества чисел и матрицы*, с помощью которых задаются входные значения и.н.с., выходные значения и.н.с., матрицы весовых коэффициентов и прочее.

Каждый элемент ориентированного множества чисел является некоторым числом. Числа могут быть представлены в виде sc-узлов, либо с помощью строкового представления всего множества, для чего используется специальное отношение *строковое представление ормножества чисел**, которое введено в целях оптимизации некоторых вариантов реализации агента, интерпретирующего действие, использующее понятие ориентированного множества чисел.

ориентированное множество чисел

:= [ориентированное множество чисел]
 ⇐ *включение**:
 число
 ⇐ *включение**:
 ориентированное множество
 ⇐ *первый домен**:
 строковое представление ормножества чисел*

Матрица является ориентированным множеством ориентированных множеств чисел равной мощности.

1. Действие вычисления взвешенной суммы всех нейронов слова

Аргументы(*объекты'*) этого действия задаются следующими отношениями:

входной вектор'

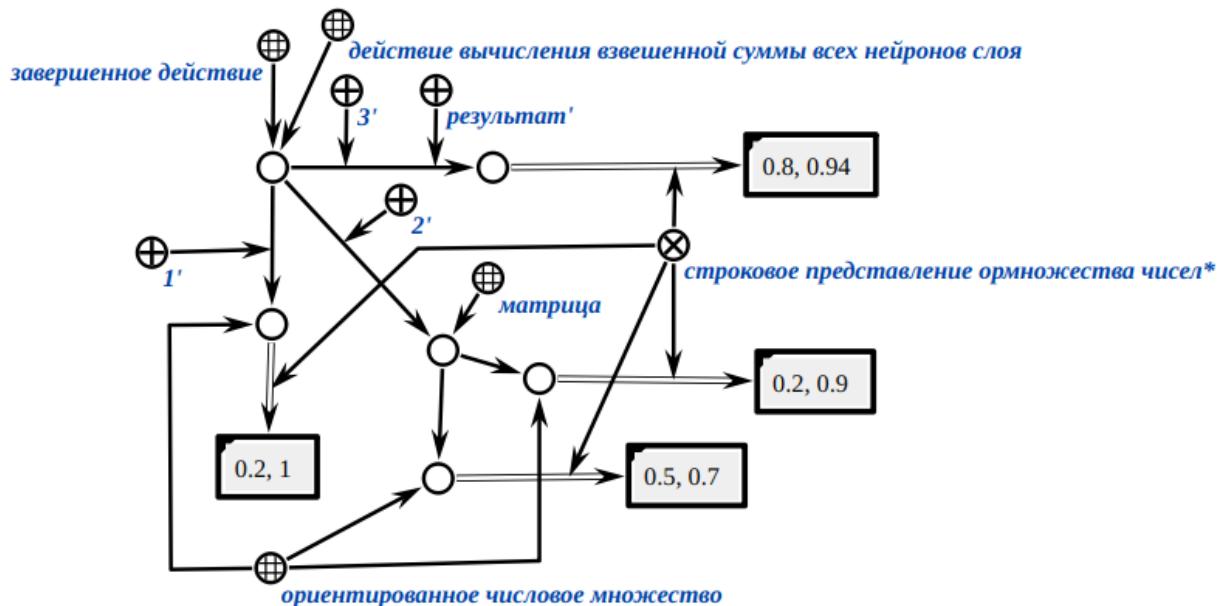
- ⇒ *первый домен*:*
действие интерпретации и.н.с.
- ⇒ *второй домен*:*
ориентированное множество чисел

матрица весовых коэффициентов нейронов слоя'

- ⇒ *первый домен*:*
действие по обработке и.н.с.
- ⇒ *второй домен*:*
матрица

Результатом действия(*результат'*) является ориентированное множество чисел, являющихся взвешенной суммой нейронов соответствующего слоя.

Пример спецификации действия вычисления взвешенной суммы всех нейронов слоя для слоя с двумя нейронами и входным вектором размерностью 2 приведен на рисунке [Пример действия вычисления взвешенной суммы всех нейронов слоя.](#)



= [Пример действия вычисления взвешенной суммы всех нейронов слоя](#)

2. Действие вычисления функции активации всех нейронов слоя

Аргументы этого действия задаются следующими отношениями:

вектор взвешенных сумм нейронов слоя'

- ⇒ *первый домен*:*
действие по обработке и.н.с.
- ⇒ *второй домен*:*
ориентированное множество чисел

вектор порогов нейронов слоя'

- ⇒ *первый домен*:*
действие по обработке и.н.с.
- ⇒ *второй домен*:*
ориентированное множество чисел

функция активации'

- ⇒ *первый домен*:*

действие по обработке и.н.с.

- ⇒ *второй домен*:*
функция

Результатом действия является ориентированное множество чисел, являющихся выходными значениями нейронов слоя.

3. Действие интерпретации сверточного слоя

Аргументы этого действия задаются следующими отношениями:

входная матрица'

- ⇒ *первый домен*:*
действие интерпретации и.н.с.
- ⇒ *второй домен*:*
матрица

ядро свертки'

- ⇒ *первый домен*:*
действие интерпретации сверточного слоя
- ⇒ *второй домен*:*
матрица

шаг свертки'

- ⇒ *первый домен*:*
действие интерпретации сверточного слоя
- ⇒ *второй домен*:*
число

Результатом действия является матрица, полученная в результате свертки входной матрицы с ядром свертки.

4. Действие интерпретации пулинг слоя

Аргументы этого действия задаются следующими отношениями:

входная матрица'

- ⇒ *первый домен*:*
действие интерпретации и.н.с.
- ⇒ *второй домен*:*
матрица

размер окна пулинга'

- ⇒ *первый домен*:*
действие интерпретации пулинг слоя
- ⇒ *второй домен*:*
матрица

размер окна пулинга'

- ⇒ *первый домен*:*
действие интерпретации пулинг слоя
- ⇒ *второй домен*:*
матрица

шаг окна пулинга'

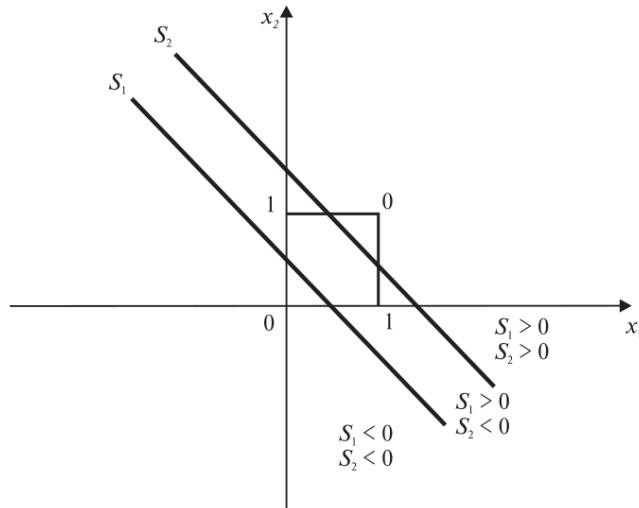
- ⇒ *первый домен*:*
действие интерпретации пулинг слоя
- ⇒ *второй домен*:*
число

Результатом действия является матрица, полученная в результате пулинга входной матрицы.

При необходимости задавать различные аргументы для нейронов одного и того же слоя, можно специфицировать соответствующие действия, однако в этой работе этого не было произведено из-за слабой изученности подобного рода нейросетевых моделей.

Спецификация агентов, соответствующих указанным действиям, задает агентно-ориентированную модель интерпретации искусственных нейронных сетей. Реализация этой модели будет называться интерпретатором искусственных нейронных сетей

Рассмотрим пример описания на языке представления нейросетевого метода SCP, решающего задачу, которая формулируется следующим образом: вычислить результат логической операции "ИСКЛЮЧАЮЩЕЕ ИЛИ" для значений двух логических переменных. На рисунке [Решение задачи "ИСКЛЮЧАЮЩЕЕ ИЛИ"](#) представлено решение этой задачи с помощью сигнальной функции.

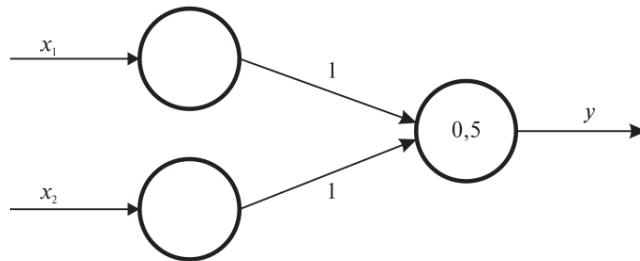


= Решение задачи "ИСКЛЮЧАЮЩЕЕ ИЛИ"

В работе [Головко В.А..НейроТОД-2017кн](#) описан однослойный персептрон, решающий поставленную задачу. Персептрон состоит из двух входных нейронов и одного выходного, с заданным порогом в 0,5 и сигнальной функцией активации:

$$F(S) = \begin{cases} 1, & 0 < S < 0, \\ 0, & \text{else} \end{cases}$$

Весовые коэффициенты синапсов входного слоя равны 1. На рисунке [Схема однослойного персептрана, решающего задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ"](#) представлена схема персептрана.



= Схема однослойного персептрана, решающего задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ"

Данному персептрану соответствует метод, представленный в базе знаний ostis-системы на описанном в данной работе языке представления нейросетевых методов SCP. Данный метод представлен на рисунке [Метод, решающий задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ" представленный с помощью языка представления нейросетевых методов SCP](#).

Описание метода состоит из последовательности двух обобщенных спецификаций действий – действия вычисления взвешенной суммы всех нейронов слоя и действия вычисления функции активации для всех нейронов слоя.

Сигнальная функция активации, использующаяся в персептране, в памяти ostis-системы определяется логической формулой, представленной на рисунке [Представление сигнальной функции активации в памяти ostis-системы](#).

Любой агент, интерпретирующий действия с заданными с помощью отношения *функция активации'* аргументами, должен использовать интерпретатор математических функций, использующихся в качестве функций активации.

```

proc_exclusive_or_ann
<- scp_method;
<- perceptron;
-> rrel_key_sc_element: _process1;

proc_exclusive_or_ann = [*
_process1
_-> scp_process;
_-> rrel_1:: rrel_in:: _input_vector;
_-> rrel_2:: rrel_out:: _output_vector;

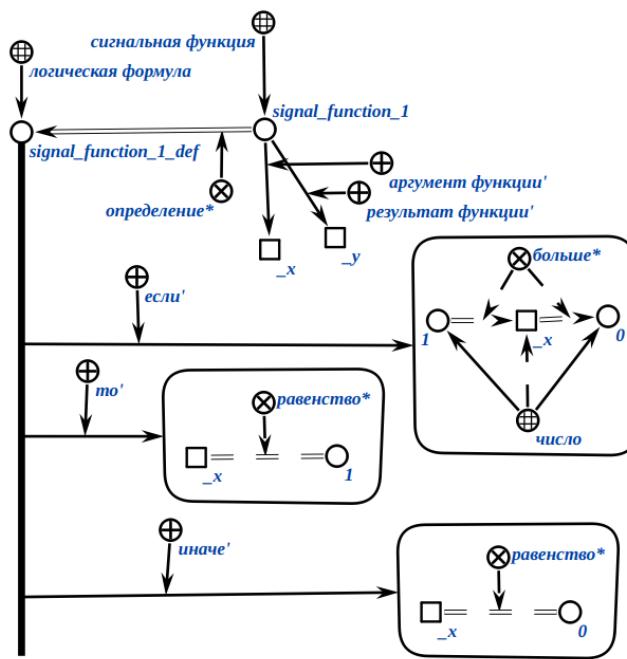
_-<= nrel_decomposition_of_action:: _... (*
    _-> rrel_1:: _..operator1
    (*
        _<- action_calculate_weighted_sum_of_all_neurons_of_layer;;
        _-> rrel_1:: rrel_fixed:: rrel_scp_var:: rrel_input_vector:: _input_vector;;
        _-> rrel_2:: rrel_fixed:: rrel_scp_const:: rrel_synopsis_weight_matrix:: ...
        (*
            <- matrix;;
            -> rrel_1: ...
            (*
                <- number_oriented_set;;
                => nrel_oriented_set_string_representation: [1, 1];
            *);;
        );
        _-> rrel_3:: rrel_assign:: rrel_scp_var:: rrel_result:: _weighted_sum_vector;;
        _=> nrel_goto:: _..operator2;;
    *););
    _-> _..operator2
    (*
        _<- action_calculate_activation_function_of_all_neurons_of_layer;;
        _-> rrel_1:: rrel_fixed:: rrel_scp_var:: _weighted_sum_vector;;
        _-> rrel_2:: rrel_fixed:: rrel_scp_const:: rrel_threshold_set:: ...
        (*
            <- number_oriented_set;;
            => nrel_oriented_set_string_representation: [0.5];
        *);;
        _-> rrel_3:: rrel_fixed:: rrel_scp_const:: rrel_activation_fun:: signal_fun;;
        (*
            <- signal_activation_function;;
            => nrel_definition: signal_function_1_def;;
        *);;
        _-> rrel_4:: rrel_assign:: rrel_scp_var:: _output_vector;;
        _=> nrel_goto:: _..operator3;;
    *););
    _-> _..operator3 (* <- return;; *);
*););
*];

```

= Метод, решающий задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ" представленный с помощью языка представления нейросетевых методов SCP

§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем

Наличия языка представления нейросетевых методов SCP и его интерпретатора позволяет обеспечить интерпретацию нейросетевого метода в памяти ostis-системы. Наличие в единой памяти не только экземпляров методов, но и понятий, их описывающих, создает основу для автоматизации процесса построения нейросетевых методов. В памяти ostis-системы хранятся знания о том, методы какого класса могут решить задачу заданного класса, но экземпляров класса этого метода может не быть представлено в системе. На этот случай система должна иметь возможность сообщить пользователю о возможности решения, для которого, однако, необходимо погрузить в систему определенный метод. Так как система хранит в единой памяти задачу и требования к методу ее решения, появляется возможность спроектировать необходимый метод. Для этого необходимо наличие среди проектиро-



= Представление сигнальной функции активации в памяти ostis-системы

вания методов соответствующих классов. В случае нейросетевого метода, речь идет об интеллектуальной среде построения нейросетевых методов.

В основе интеллектуальной среды построения нейросетевых методов лежат соответствующие другу другу иерархии действий, задач и методов построения и.н.с. Наличие такой иерархии позволит описать язык представления методов построения и.н.с. и разработать интерпретатор этого языка.

Построение иерархии соответствующих действий построения и.н.с. следует начать с изучения этапов проектирования и обучения и.н.с., которые, в общем случае, выполняют все разработчики и.н.с.:

1. Постановка задачи.

Постановка задачи включает в себя описание входных данных (изображения/видео, временные ряды, текст), выходных данных и требований к методу решения (скорость, затраты по памяти и т.д.). Также описывается дополнительная информация, которая может помочь в построении метода решения задачи (к примеру, спецификация обучающей выборки, если таковая имеется). Обычно, на данном этапе разработчик и.н.с. определяет класс задачи, формирует требования к обучающей выборке, если она не предоставлена.

Выполнение данного этапа средой проектирования и.н.с. подразумевает выполнение следующих действий:

- **Действие трансляции условия задачи.** Действие транслирует заданное с помощью интерфейса ostis-системы (к примеру, естественно-языкового интерфейса) описание задачи в память ostis-системы. Действие необходимо в случае, когда условие задачи задается пользователем. Необходимо понимать, что описание задачи поступает в базу знаний не только от пользовательского интерфейса. К примеру, задача может быть сформулирована самой системой в ходе ее жизнедеятельности. Данное действие является общим для всех ostis-систем, поэтому его рассмотрение выходит за рамки рассмотрения процесса построения интеллектуальной среды проектирования и.н.с.
- **Действие классификации задачи.** Действие определяет класс задачи (задача регрессии, детекции, кластеризации и т.д.), исходя из описания задачи в базе знаний.
- **Действие поиска подходящей обучающей выборки.** В базе знаний может храниться набор спецификаций выборок, к которым у ostis-системы есть доступ. Действие производит поиск выборок, которые могут быть использованы в качестве обучающей выборки.
- **Действие формирования требования к обучающей выборке.** Если обучающая выборка не была предоставлена и не была найдена, то необходимо сформировать описание требований к обучающей выборке, которое можно будет транслировать на язык пользовательского интерфейса и запросить необходимую выборку у пользователя.

2. Предобработка выборки: очистка

На этом этапе обнаруживаются признаки, которые имеют в общем случае некорректные значения (например, для каких-то образов значение признака может иметь неопределенное значение, либо значение, не совпадающее по типу, либо аномально большое или очень маленькое значение, которое встречается в редком числе случаев). Для

признаков, имеющих неопределенное значение, может быть применены различные методы устранения, например, такие значения могут быть заменены средним значением этого признака, рассчитанным по всем образам (для непоследовательных данных), либо они могут быть заменены средним значением по соседним образам (в случае временных рядов), либо каким-то фиксированным значением. Радикальная мера решения проблемы – удаление образов, имеющих неопределенные значения признаков из выборки. Однако его лучше применять, если образов с отсутствующими значениями признаков немного. Для выбросов и аномалий применяются схожие стратегии (но только в том случае, если задача не состоит в прогнозировании этих аномалий).

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия очистки выборки**, которое выполняется в случае обработки выборки, которая ранее не была представлена в памяти системы (к примеру, была получена от пользователя). Реализация интерпретатора (агента) данного действия требует описания в памяти классификации стратегий очистки данных и реализации методов применения этих стратегий.

3. Предобработка выборки: выявление содержательных признаков

Осуществляется т.н. инжиниринг признаков, состоящий в отборе признаков, влияющих на результат работы модели, несодержательные признаки, которые никак не коррелируют с выходом модели, удаляются. Цель этого этапа – уменьшение размерности пространства признаков для снижения влияния эффекта переобучения на модель.

Для снижения размерности признакового пространства может применяться методы отбора признаков и выделения признаков.

При отборе признаков, осуществляется формирование подмножества из исходных признаков (алгоритм последовательного обратного отбора, рекурсивный алгоритм обратного устранения признаков, алгоритмы с использованием случайных лесов).

При выделении признаков из набора признаков извлекается информация для построения нового подпространства признаков (алгоритмы с использованием автоэнкодера).

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия выявления содержательных признаков**. Реализация интерпретатора (агента) данного действия требует описания в памяти классификации стратегий уменьшения размерности признакового пространства и реализации методов применения этих стратегий.

4. Предобработка выборки: трансформация

На этом этапе осуществляется подготовка данных к обучению. Здесь следует уделить особое внимание наличию категориальных признаков, чаще всего заданных строковыми типами. Эти признаки могут быть номинальными и порядковыми. Для кодирования порядковых признаков чаще всего применяют последовательный числовой код (1, 2, 3,...). Для кодирования номинальных такое решение неверно, т.к. эти признаки равноправны и не могут сравниваться по числовому коду (например, пол – 0/1). Для номинальных признаков применяется способ прямого кодирования, заключающийся в создании и использовании фиктивных признаков по количеству значений исходного. Например, признак пол (мужской, женский) преобразуется в два новых признака мужской и женский с соответствующими значениями для имеющихся образов.

Масштабирование признаков предполагает приведение значений признаков к одному общему интервалу – это особенно актуально для признаков, имеющих несопоставимые выборочные средние значения по всем образам – например, один признак в среднем имеет значение 10.000, а другой 12. Это может проявиться в выполнении минимизации только по признаку с наибольшими значениями и плохой сходимости метода обучения. Чаще всего масштабирование соответствует выполнению нормализации на отрезок (min-max нормализация):

$$x_{norm}^i = \frac{x^i - x_{min}}{x_{max} - x_{min}}$$

где x^i – значение признака для отдельно взятого образа i , x_{min} – наименьшее значение для признака, x_{max} – наибольшее значение для признака.

Другой вариант масштабирования – применение стандартизации признаков:

$$x_{std}^i = \frac{x^i - \mu(x)}{\sigma(x)},$$

$\mu(x)$ – выборочное среднее отдельного признака, $\sigma(x)$ – стандартное отклонение.

Стандартизация сохраняет полезную информацию о выбросах в исходных данных и делает алгоритм обучения менее чувствительным к ним.

Дискретизация применяется для перехода от вещественного признака к порядковому за счет кодирования интервалов одним значением (например, если признак отражает возраст человека, то может быть произведена дискре-

тизация значений с выделением определенных возрастных групп, где каждая группа будет кодироваться одним целым числом).

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия трансформации выборки**. Реализация интерпретатора (агента) данного действия требует описания в памяти классификации методов масштабирования признаков и реализации методов применения этих стратегий.

5. Разбиение выборки на обучающую, валидационную и тестовую (контрольную)

Производится разбиение всей выборки данных, на обучающую, тестовую и, в некоторых случаях, валидационную.

Валидационная выборка используется для оценки влияния изменения гиперпараметров на результат обучения и может применяться как дополнительный инструмент для этого наравне с сеточным поиском.

Разбиение проводится в соотношении 3:1:1, в процентах (60/20/20), если валидационная выборка не используется, то 80/20.

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия разбиения выборки**.

Все предыдущие этапы применялись к выборке, последующие этапы относятся к используемым моделям и.н.с.

6. Выбор класса нейросетевых методов в соответствии со сформулированной задачей

На этом этапе осуществляется выбор основной архитектуры и.н.с., которая будет использоваться при обучении. Однако, нужно отметить, что этот выбор относительно условный, т.е. исследователь не ограничен использованием только одного типа и.н.с. для решения задачи (как, например, сверточной сети для изображений, поскольку изображения можно обрабатывать и обычным многослойным персептроном). Речь скорее идет именно о рекомендованной архитектуре, но это не исключает использование любых других вариантов архитектур и их сочетаний в рамках одной модели).

Примерами таких рекомендаций являются:

- Изображения/видео – сверточные нейронные сети
- Временные ряды – многослойные персептроны или рекуррентные сети
- Текстовая информация – многослойные персептроны или рекуррентные сети
- Наборы характеристик некоторых объектов (например, спецификации автомобилей) – многослойный персепtron

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия выбора класса нейросетевых методов**.

7. Формирование спецификации на входные и выходные данные

Выполняются дополнительные преобразования данных, связанные с изменением структур хранения (например, преобразование многомерного массива в одномерный, конвертация типов)

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия формирования спецификации входов и выходов и.н.с..**

8. Выбор метода оптимизации

В рамках ПрО и.н.с. описаны следующие методы оптимизации:

- стохастический градиентный спуск (SGD);
- метод Нестерова;
- адаптивный градиент(AdaGrad);
- адаптивная оценка момента (Adam);
- среднеквадратическое распространение (RMSProp).

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия выбора метода оптимизации**.

9. Выбор минимизируемой функции ошибки

На этом этапе задается функция ошибок, которая будет минимизироваться. К примеру, MSE лучше подходит для задач регрессии и для кластеризации, СЕ – для классификационных задач.

Данные функции определяются следующим образом:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \tilde{Y}_i)^2$$

где n – размер обучающей выборки, Y_i – эталонное значение функции, \tilde{Y}_i – результат, полученный НС

$$CE = -\frac{1}{n} \sum_{i=1}^n (Y_i \log(\tilde{Y}_i) + (1 - Y_i) \log(1 - \tilde{Y}_i))$$

(случай 2-классовой классификации)

$$CE = -\frac{1}{n} \sum_{i=1}^n \sum_{c=1}^M Y_i^c \log \tilde{Y}_i^c$$

(случай многоклассовой классификации)

В интеллектуальной среде проектирования данный этап соответствует выполнению *действия выбора минимизируемой функции ошибки*.

10. Начальная инициализация параметров нейронной сети

Наиболее часто используемые варианты инициализации весовых коэффициентов и порогов нейронной сети включают:

- инициализация значениями из равномерного распределения на каком-то небольшом интервале, например, [-0.1, 0.1].
- инициализация значениями из стандартного нормального распределения.
- инициализация по методу Ксавье **glorot2010**.

Применяется для предотвращения резкого уменьшения или увеличения значений выхода нейронных элементов после применения функции активации при прямом прохождении образа через глубокую нейронную сеть. Фактически инициализация этим методом осуществляется посредством выбора значений из равномерного распределения на отрезке $[-\sqrt{6}/\sqrt{n_i + n_{i+1}}, \sqrt{6}/\sqrt{n_i + n_{i+1}}]$, где n_i – это число входящих связей в данный слой, а n_i – число исходящих связей из данного слоя. Таким образом, инициализация этим методом проводится для разных слоев нейронной сети из разных отрезков.

- инициализация, полученная из предобученной модели.

Вариант инициализации, который предполагает использование в качестве "стартовой" модели предобученной модели, взятой из некоторого репозитория предобученных моделей, обученную самим исследователем или в процессе работы интеллектуальной системы.

- инициализация по методу Кайминга **he2015**.

Данный метод инициализации применяется для решения проблемы "затухающего" градиента и "взрывающегося" градиента. Производится посредством выбора значений из равномерного распределения на отрезке $[-\sqrt{2}/\sqrt{(1+a^2)fan}, \sqrt{2}/\sqrt{(1+a^2)fan}]$, где a – угол наклона к оси абсцисс для отрицательной части области определения функции активации типа ReLU (для обычной ReLU функции этот параметр равен 0), fan – параметр режима работы, который для фазы прямого распространения равен количеству входящих связей (для устранения эффекта "взрывающегося" градиента), а для фазы обратного распространения – количеству выходящих (для устранения эффекта "затухающего" градиента).

В интеллектуальной среде проектирования данный этап соответствует выполнению *действия начальной инициализации и.н.с.*

11. Выбора гиперпараметров и.н.с.

На практике некоторые гиперпараметры (такие как количество слоев, их типы, количество нейронов в слое) часто определяются экспериментально, в процессе итеративного поиска лучшего варианта решения задачи. Хотя способы частично автоматизировать этот процесс существуют, они все же рассчитаны на наличие некоторых предусловий проведения эксперимента, в частности интервалов изменения параметра (например, скорости обучения).

К гиперпараметрам, подбираемым на этом этапе, относятся:

- Параметры обучения и.н.с. (скорость обучения, моментный параметр, размер мини-батча).
- Архитектура модели и.н.с., опирающаяся на ранее сформулированные спецификации входных и выходных данных (например, количество нейронов в определенном слое (слоях) или конфигурации целых слоев)

Нахождение оптимальных гиперпараметров может быть получено, например, использованием метода сеточного поиска, который позволяет проверить значения гиперпараметров, взятые с определенным шагом или из определенного интервала (кортежа). С помощью этого метода выбирается оптимальный набор гиперпараметров, который дает лучшие результаты, он используется для последующего дообучения. Или же, если полученные результаты являются приемлемыми, то процесс дальнейшего обучения вообще не проводится. Следует отметить затратность данного метода, т.к. фактически осуществляется перебор различных значений параметров обучения. Для снижения объема работы применяется метод случайного поиска.

Для оптимизации архитектуры определяются типы слоев нейронной сети, количество нейронных элементов в каждом слое, их характеристики – функция активации, для сверточных элементов – размер ядра, а также параметры padding и шаг свертки (stride). Здесь же может осуществляться оценка не только пользовательского варианта сети,

но и предобученной архитектуры. Основное правило при выборе – количество параметров модели не должно превышать размер обучающей выборки. Для предобученных архитектур это ограничение снимается.

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия выбора гиперпараметров и.н.с.**. Действие использует классификацию и спецификации гиперпараметров и.н.с..

12. Обучение модели на обучающей выборке

Производится обучение модели до достижения выбранной точности (оценивается на тестовой выборке) или по другим заданным критериям (достижение заданного количества эпох обучения, неизменность точности на протяжении заданного количества эпох, падение точности на валидационной выборке и т.д.)

Приведем классификацию алгоритмов обучения:

метод обучения и.н.с.

- метод
- метод обучения с учителем
 - ⇒ explanation*:
 - [**метод обучения с учителем** – метод обучения с использованием заданных целевых переменных.]
 - метод обратного распространения ошибки
 - := [м.о.р.о.]
 - ⇒ explanation*:
 - [м.о.р.о. использует заданный метод оптимизации и заданную функцию потерь для реализации фазы обратного распространения ошибки и изменения настраиваемых параметров и.н.с. Одним из самых распространенных методов оптимизации является метод стохастического градиентного спуска.]
 - ⇒ explanation*:
 - [Следует также отметить, что несмотря на то, что метод отнесен к методам обучения с учителем, в случае использования м.о.р.о. для обучения автокодировщиков в классических публикациях он рассматривается как метод обучения без учителя, поскольку в данном случае размеченные данные отсутствуют.]
 - метод обучения без учителя
 - ⇒ explanation*:
 - [**метод обучения без учителя** – метод обучения без использования заданных целевых переменных(в режиме самоорганизации)]
 - ⇒ explanation*:
 - [В ходе выполнения алгоритма метода обучения без учителя выявляются полезные структурные свойства набора. Неформально его понимают как метод для извлечения информации из распределения, выборка для которого не была вручную аннотирована человеком *Гудфеллоу.Я..ГлубОбуч-2017кн.*]

В интеллектуальной среде проектирования данный этап соответствует выполнению **действия обучения и.н.с..**. Действие обучения и.н.с. – действие, в ходе которого реализуется определенный метод обучения и.н.с. с заданными параметрами обучения и.н.с., методом оптимизации и функцией потерь.

При обучении возможно возникновение следующих проблем:

- Переобучение – проблема, возникающая при обучении и.н.с., заключающаяся в том, что сеть хорошо адаптируется к паттернам входной активности из обучающей выборки, при этом теряя способность к обобщению. Переобучение возникает из-за применения неоправданно сложной модели при обучении и.н.с. Это происходит, когда количество настраиваемых параметров и.н.с. намного больше размера обучающей выборки. Возможные варианты решения проблемы заключаются в упрощении модели, увеличении выборки, использовании регуляризации (параметр регуляризации, техника dropout и т.д.). Обнаружение переобученности сложнее, чем недообученности. Как правило, для этого применяется кроссвалидация на валидационной выборке, позволяющая оценить момент завершения процесса обучения. Идеальным вариантом является достижение баланса между переобученностью и недообученностью.
- Недообучение – проблема, возникающая при обучении и.н.с., заключающаяся в том, что сеть дает одинаково плохие результаты на обучающей и контрольной выборках. Чаще всего такого рода проблема возникает при недостаточном времени, затраченном на обучение модели. Однако это может быть вызвано и слишком простой архитектурой модели либо малым размером обучающей выборки. Соответственно решение, которое может быть принято ML-инженером, заключается в устранении этих недостатков: увеличение времени обучения, использование модели с большим числом настраиваемых параметров, увеличение размера обучающей выборки, а также уменьшение регуляризации и более тщательный отбор признаков для обучающих примеров.

Методом обучения и.н.с. называется процесс итеративного поиска оптимальных значений настраиваемых параметров и.н.с., минимизирующих некоторую заданную функцию потерь.

Стоит отметить, что хотя целью применения метода обучения является минимизация функции потерь, "полезность" полученной после обучения модели можно оценить только по достигнутому уровню ее обобщающей способности.

Методы обучения могут быть поделены на две большие группы – **методы обучения с учителем и методы обучения без учителя** (контролируемый и неконтролируемый методы обучения).

Метод обучения с учителем – метод обучения с использованием заданных целевых переменных.

Одним из методов обучения с учителем является метод обратного распространения ошибки.

Приведем его описание в виде алгоритма:

Data: X – данные, E_t – желаемый отклик (метки), E_m – желаемая ошибка (в соответствии с выбранной функцией потерь)

Result: обученная нейронная сеть Net

инициализация весов W и порогов T ;

repeat

foreach $x \in X, e \in E_t$ **do**

фаза прямого распространения сигнала: вычисляются активации для всех слоев и.н.с.;

фаза обратного распространения ошибки: вычисляются ошибки для последнего слоя и всех предшествующих слоев;

изменение настраиваемых параметров и.н.с. в соответствии с вычисленными ошибками;

end

вычисление общей ошибки E на данной эпохе;

until $E < E_m$;

Метод обратного распространения ошибки использует заданный метод оптимизации и заданную функцию потерь для реализации фазы обратного распространения ошибки и изменения настраиваемых параметров и.н.с. Одним из самых распространенных методов оптимизации является метод стохастического градиентного спуска. Приведенный метод используется для реализации последовательного варианта обучения.

Следует также отметить, что несмотря на то, что метод отнесен к методам обучения с учителем, в случае его использования для обучения автокодировщиков в классических публикациях он рассматривается как метод обучения без учителя, поскольку в данном случае размеченные данные отсутствуют.

Метод обучения без учителя – метод обучения без использования заданных целевых переменных (в режиме самоорганизации)

В ходе выполнения алгоритма метода обучения без учителя выявляются полезные структурные свойства набора. Неформально его понимают как метод для извлечения информации из распределения, выборка для которого не была вручную аннотирована человеком [Гудфеллоу.Я..ГлубОбуч-2017kn]. Метод обучения без учителя может рассматриваться как вспомогательный метод для начальной инициализации настраиваемых параметров и.н.с. В этом случае он является методом предобучения.

Среди методов, применяемых для оптимизации целевой функции можно выделить следующие:

- SGD (стохастический градиентный спуск): в данном методе корректировка настраиваемых параметров и.н.с. выполняется в направлении максимального уменьшения функции стоимости, т.е. в направлении, противоположном вектору градиента функции потерь [Хайкин.С.НейрСети-2006kn]
- метод Нестерова: Обучение методом стохастического градиентного спуска иногда происходит очень медленно. Импульсный метод позволяет ускорить обучение, особенно в условиях высокой кривизны, небольших, но устойчивых градиентов или зашумленных градиентов. В импульсном методе вычисляется экспоненциально затухающее скользящее среднее прошлых градиентов и продолжается движение в этом направлении. Метод Нестерова является вариантом импульсного алгоритма, в котором градиент вычисляется после применения текущей скорости [Гудфеллоу.Я..ГлубОбуч-2017kn]
- AdaGrad: Данный метод по отдельности адаптирует скорости обучения всех настраиваемых параметров и.н.с., умножая их на коэффициент, обратно пропорциональный квадратному корню из суммы всех прошлых значений квадрата градиента [Duchi J..AdaptSMfOLaSO-2011art]
- RMSProp: Данный метод является модификацией AdaGrad, которая позволяет улучшить его поведение в невыпуклом случае путем изменения способа агрегирования градиента на экспоненциально взвешенное скользящее среднее. Использование экспоненциально взвешенного скользящего среднего гарантирует повышение скорости сходимости после обнаружения выпуклой впадины, как если бы внутри этой впадины алгоритм AdaGrad был инициализирован заново [Гудфеллоу.Я..ГлубОбуч-2017kn]
- Adam: Данный метод можно рассматривать как комбинацию RMSProp и AdaGrad [Kingma.D.Adam-2014cm]. Помимо усредненного первого момента, данный метод использует усредненное значение вторых моментов градиентов

Отметим, что успешность применения методов оптимизации зависит главным образом от знакомства пользователя с соответствующим алгоритмом [*Гудфеллоу Я..ГлубОбуч-2017кн.*].

Еще одним важным компонентом, влияющим на процесс обучения, является используемая функция потерь.

Функция потерь – функция, используемая для вычисления ошибки, рассчитываемой как разница между фактическим эталонным значением и прогнозируемым значением, получаемым *и.н.с.*

Среди функций потерь, используемые в качестве целевых функций для применяемого метода оптимизации, можно выделить:

- MSE – средняя квадратичная ошибка

$$MSE = \frac{1}{L} \sum_{l=1}^L \sum_{i=1}^m (y_i^l - e_i^l)^2$$

где y_i^l – прогноз модели, e_i^l – ожидаемый (эталонный) результат, m – размерность выходного вектора, L – объем обучающей выборки.

- BCE – бинарная кросс-энтропия

$$BCE = - \sum_{l=1}^L (e^l \log(y^l) + (1 - e^l) \log(1 - y^l))$$

где y^l – прогноз модели, e^l – ожидаемый (эталонный) результат: 0 или 1, L – объем обучающей выборки.

- MCE – мультиклассовая кросс-энтропия

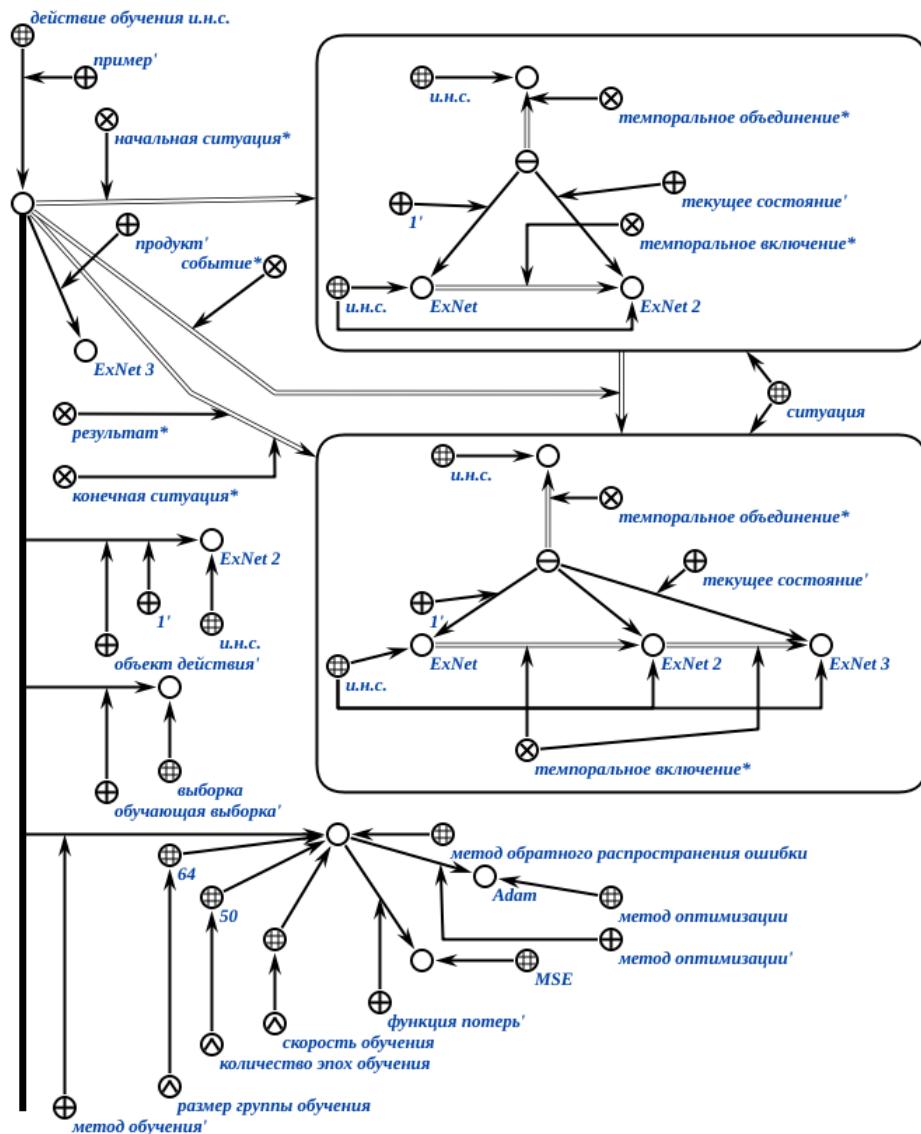
$$MCE = - \sum_{l=1}^L \sum_{i=1}^m e_i^l \log(y_i^l)$$

где y_i^l – прогноз модели, e_i^l – ожидаемый (эталонный результат), m – размерность выходного вектора

Отметим, что для бинарной кросс-энтропии в выходном слое *и.н.с.* будет находиться один нейрон, а для для мультиклассовой кросс-энтропии количество нейронов в выходном слое *и.н.с.* совпадает с количеством классов.

Для решения задачи классификации рекомендуется использовать бинарную или мультиклассовую кросс-энтропийную функцию потерь, для решения задачи регрессии рекомендуется использовать среднюю квадратичную ошибку.

Действие обучения *и.н.с.* можно проиллюстрировать следующим изображением [Действие обучения и.н.с.](#).



= *Действие обучения и.н.с.*

13. Оценка эффективности и.н.с

После выполнения обучения осуществляется оценка полученной модели с помощью метрик оценки качества.

Далее результат оценки может быть визуализирован с помощью матрицы ошибок (confusion matrix) и ROC-кривой.

Матрица ошибок представляется собой матрицу (рис. [Матрица ошибок](#)), в которую помещены сведения о числе истинно-положительных, истинно-отрицательных, ложно-положительных и ложно-отрицательных предсказаниях классификатора.

		Predicted class	
		P	N
Expected class	P	True positive (TP)	False negative (FN)
	N	False positive (FP)	True negative (TN)

= *Матрица ошибок*

ROC-кривая – это график, в котором, основываясь на заданном пороге решения классификатора, рассчитываются доли ложноположительных и истинно положительных исходов. Основываясь на ROC-кривой, высчитывается AUC-показатель (площадь под кривой), которая используется в качестве характеристики качества модели.

В интеллектуальной среде проектирования данный этап соответствует выполнению *действия оценки эффективности и.н.с..*

Рассмотрим пример выполнения описанных этапов разработчиком для конкретной задачи – *классификации цифр из выборки рукописных цифр MNIST*:

1. Исходными данными задачи является: выборка из 70.000 изображений, предварительно разделенная на обучающую (60.000 изображений) и контрольную (10.000 изображений) выборки. Каждое изображение представлено двумерным массивом 28Х28 чисел из интервала [0, 255], числа представляют определенный оттенок серого цвета. Помимо этого каждому изображению соответствует метка класса, соответствующая конкретной цифре от 0 до 9.

Ставится задача: *обучить модель, которая будет принимать на вход двумерный массив данных и возвращать метку класса, соответствующей распознанной цифре.*

Таким образом, тип решаемой задачи – **классификационная**, природа данных задачи – **изображения**.

2. В рассматриваемой выборке отсутствуют аномалии, ошибочные данные, признаки с отсутствующими значениями.

3. В рассматриваемой задаче отсутствуют несодержательные признаки.

4. В качестве метода предобработки данных используем масштабирование признаков, а именно нормализацию на отрезок [0, 1].

5. Выполним разбиение обучающей части данных на обучающую и валидационную выборки в соотношении 4:1 (48.000 в обучающей и 12.000 в валидационной).

6. Так как выборка включает в себя изображения, будем использовать сверточную нейронную сеть.

7. Не требуется.

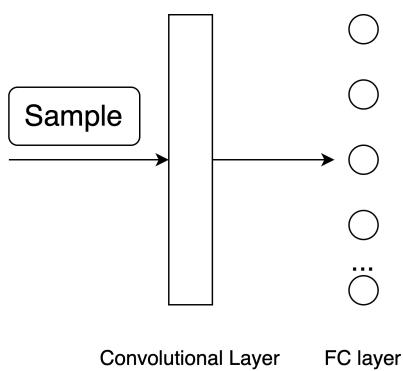
8. В качестве оптимизационного алгоритма будем использовать метод стохастического градиентного спуска (SGD).

9. Так как решается задача классификации, выберем в качестве минимизируемой функции кросс-энтропийную функцию потерь.

10. В качестве начальной инициализации будем использовать инициализацию по методу Кайминга.

11. На этапе 6 и 8 было определено, что для решения задачи будет использоваться сверточная нейронная сеть. При использовании one-hot кодирования в последнем полносвязном слое будет 10 нейронов по числу классов в задаче.

Для упрощения будем использовать архитектуру, изображенную на рис. [Архитектура и.н.с., решающая задачу классификации цифр](#), не содержащую промежуточные слои.



= *Архитектура и.н.с., решающая задачу классификации цифр*

Для нахождения оптимального набора гиперпараметров будем применять метод случайного поиска.

Перечислим кортежи, из которых будут сэмплироваться гиперпараметры:

- Скорость обучения – (0.9, 0.1, 0.01, 0.001)
- Количество нейронов в сверточном слое – (5, 10, 15, 20)
- Размер ядра свертки – (3, 5, 7, 9)
- Моментный параметр – (0, 0.5, 0.9)
- Размер мини-батча – (16, 32, 64, 128)

После определения данных параметров и оценки эффективности работы алгоритма, получим следующую таблицу:

= Результаты решения задачи

(используемые сокращения: *mbs* – mini-batch size, *ks* – kernel size, *lr* – learning rate, *cnc* – convolutional neurons count, *acc* – accuracy, *it* – iterations count)

#	mbs	ks	lr	momentum	cnc	acc	it
1	128	3	0.001	0.5	10	0.9033	10
2	64	9	0.9	0	15	0.1039	1
3	32	3	0.01	0.5	20	0.9741	10
4	32	7	0.01	0.5	15	0.9794	10
5	16	9	0.001	0.5	20	0.9189	2
6	64	3	0.1	0.5	10	0.9736	10
7	64	7	0.001	0.9	15	0.9007	1
8	32	9	0.1	0.5	5	0.9806	10
9	128	5	0.1	0.5	20	0.98	10
10	32	9	0.01	0.9	5	0.9806	10
11	128	3	0.001	0.9	10	0.893	1
12	32	5	0.9	0.9	20	0.1008	1
13	16	9	0.9	0.5	20	0.0976	1
14	32	7	0.9	0.9	15	0.0932	1
15	128	5	0.01	0.5	20	0.9197	2
16	16	3	0.001	0.5	10	0.904	1
17	16	9	0.001	0	20	0.8866	1
18	128	9	0.1	0.5	5	0.9793	10
19	128	3	0.001	0	10	0.6697	1
20	16	3	0.1	0	15	0.9729	4
21	32	7	0.9	0.5	15	0.1048	1
22	128	7	0.9	0	15	0.1113	1
23	64	9	0.01	0.5	10	0.9482	2
24	16	7	0.9	0	20	0.0985	1
25	16	3	0.1	0.5	5	0.9558	2
26	64	7	0.01	0.9	15	0.9839	10
27	16	7	0.1	0	10	0.9836	10
28	16	5	0.01	0	20	0.9608	2
29	16	5	0.01	0.9	20	0.9847	10
30	32	5	0.01	0.5	15	0.9532	2

Можно заметить, что лучший результат (*acc* = 0.9839) по обобщающей способности на валидационной выборке был получен при следующих параметрах: *mbs* = 64, *ks* = 7, *lr* = 0.01, *momentum* = 0.9, *cnc* = 15.

12. В качестве критерия останова нами был выбран самый простой критерий по достижению заданного количества эпох обучения. Дообучение не проводилось, для оценки обобщающей способности использовалась модель, полученная после выполнения процедуры подбора гиперпараметров. Обобщающая способность на тестовой выборке составила **0.9853**, т.е. **98.53%**.

13. Построив матрицу ошибок на основании обученной модели и тестовой выборки, получим результат, проиллюстрированный на рис. [Матрица ошибок для задачи MNIST](#)

Мы получили матрицу с явно выраженным диагональным преобладанием, таким образом полученная модель делает относительно небольшое число ошибок.

Исходя из анализа этапов построения и.н.с., которые выполняют разработчики, можно вывести следующую классификацию действий по построению и.н.с.:

действие по построению и.н.с.

⇒ декомпозиция*:

{• действие по обработке выборки

⇒ декомпозиция*:

{• действие поиска подходящей обучающей выборки

• действие формирования требований к обучающей выборке

• действие очистки выборки

• действие выявления содержательных признаков

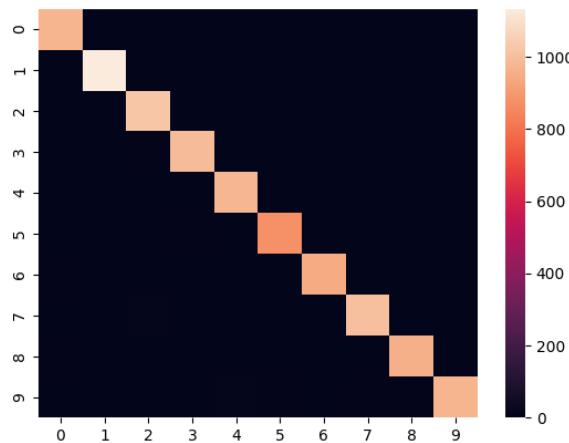
• действие трансформации выборки

• действие разбиения выборки

}

• действие по проектированию и.н.с.

⇒ декомпозиция*:



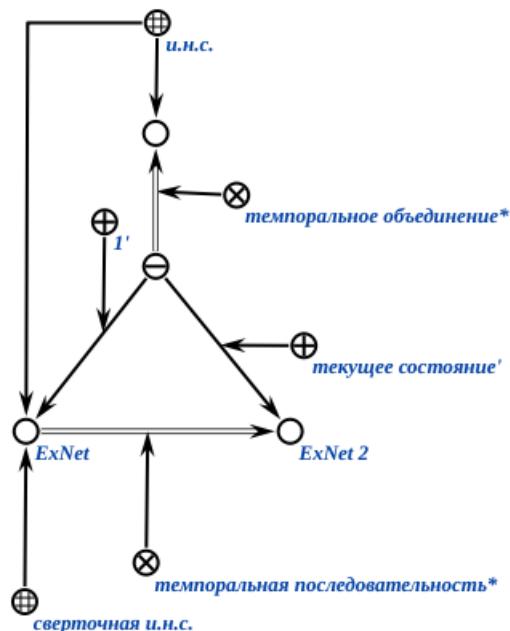
= *Матрица ошибок для задачи MNIST*

- {• действие выбора класса нейросетевых методов
 - действие формирования спецификации входов и выходов и.н.с.
}
- }
- действие обучения и.н.с.
- ⇒ декомпозиция*:
 - {• действие выбора метода оптимизации
 - действие выбора минимизируемой функции ошибки
 - действие начальной инициализации и.н.с.
 - действие выбора гиперпараметров и.н.с.
 - действие обучения и.н.с.
 - действие оценки эффективности и.н.с.
}
}
- }

Реализация интерпретатора описанных в данной главе действий по построению и.н.с. и описания в базе знаний экспертических знаний разработчиков и.н.с. (а значит реализация интеллектуальной среды проектирования и.н.с.) позволит автоматически, исходя из описания задачи, генерировать нейросетевые методы в памяти ostis-системы, что является одним из ключевых направлений развития данной работы.

Так как в результате действий по построению и.н.с. объект этих действий, конкретная и.н.с., может существенно меняться (меняется конфигурация сети, ее весовые коэффициенты), то и.н.с. представляется в базе знаний как темпоральное объединение всех ее версий. Каждая версия является и.н.с. и темпоральной сущностью. На множестве этих темпоральных сущностей задается темпоральная последовательность с указанием первой и последней версии. Для каждой версии описываются специфичные знания.

Общие для всех версий знания описываются для и.н.с., являющейся темпоральным объединением всех версий (рисунок [Темпоральность нейронной сети](#))



= Темпоральность нейронной сети

Далее более подробно рассмотрим действие по обучению и.н.с. и сделаем обзор основных методов, применяемых для обучения.

Заключение к Главе 3.6.

В главе описан подход к интеграции и конвергенции искусственных нейронных сетей с базами знаний в интеллектуальных компьютерных системах нового поколения с помощью представления и интерпретации искусственной нейронной сети в базе знаний.

Описаны синтаксис, денотационная и операционная семантика языка представления нейросетевых методов, который позволяет представить и интерпретировать в памяти интеллектуальной системы любую и.н.с. Наличие такого языка порождает семантическую совместимость нейросетевого метода с другими методами, представленными в памяти системы, что позволяет анализировать саму и.н.с. и этапы ее работы любыми другими методами системы.

Так же наличие языка представления нейросетевых методов позволяет описывать в памяти системы экспертные знания разработчиков и.н.с. В работе приведены этапы построения и.н.с., которые выполняют разработчики и.н.с.. На основании этих этапов, С целью проектирования интеллектуальной среды построения нейросетевых методов, в базе знаний были классифицированы и описаны действия по построению и.н.с.

Проектирования и реализация интеллектуальной среды построения и.н.с. в базе знаний системы является одним из двух основных направлений дальнейшего развития данной работы.

Вторым основным направлением является разработка подхода к обработке фрагментов базы знаний с помощью и.н.с., для чего необходимо разработать универсальный алгоритм взаимно-однозначного соответствия фрагментов базы знаний и входных векторов и.н.с. Язык представления знаний способен представить любое знание. Наличие в системе нейросетевого метода, способного принимать на вход фрагменты знаний, позволит решить новые, слабо изученные классы задач.

Часть 4.

Онтологические модели интерфейсов интеллектуальных компьютерных систем нового поколения

Описание к главе

Глава 4.1.

Общие принципы организации интерфейсов *ostis*-систем

Садовский М.Е.

⇒ *аннотация**:

[Аннотация к главе.]

§ 4.1.1. Структура интерфейсов *ostis*-систем

Интерфейс – совокупность технических, программных и методических (протоколов, правил, соглашений) средств, обеспечивающих обмен информацией между пользователем и устройствами и программами, а также между устройствами и другими устройствами и программами. В широком смысле слова, это способ (стандарт) взаимодействия между объектами. Интерфейс в техническом смысле слова задаёт параметры, процедуры и характеристики взаимодействия объектов.

интерфейс

⇒ *разбиение**:

- {• *пользовательский интерфейс*
 - *программный интерфейс*
 - *физический интерфейс*
- }

Пользовательский интерфейс – один из наиболее важных компонентов компьютерной системы. Представляет собой совокупность аппаратных и программных средств, обеспечивающих обмен информацией между пользователем и компьютерной системой.

Основными типами пользовательского интерфейса являются командный пользовательский интерфейс, WIMP-интерфейс и SILK-интерфейс:

пользовательский интерфейс

- ▷ *командный пользовательский интерфейс*
- ▷ *WIMP-интерфейс*
 - := [Window, Image, Menu, Pointer - интерфейс]
 - := [Окно, Образ, Меню, Указатель - интерфейс]
 - ▷ *пользовательский интерфейс ostis-системы*
- ▷ *SILK-интерфейс*
 - := [Speech, Image, Language, Knowledge - интерфейс]
 - := [Речь, Образ, Язык, Знание - интерфейс]
 - ▷ *естественно-языковой интерфейс*
 - ▷ *речевой интерфейс*

Командный пользовательский интерфейс – пользовательский интерфейс, при котором обмен информацией между компьютерной системой и пользователем осуществляется путем написания текстовых инструкций или команд. *WIMP-интерфейс* – пользовательский интерфейс, при котором обмен информацией между компьютерной системой и пользователем осуществляется в форме диалога при помощи окон, меню и других элементов управления. *SILK-интерфейс* – пользовательский интерфейс, наиболее приближенный к естественной для человека форме общения. Компьютерная система находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд преобразуется в понятную человеку форму, например, в естественно-

языковую форму или изображение. Естественно-языковой интерфейс – SILK-интерфейс, обмен информацией между компьютерной системой и пользователем в котором происходит за счёт диалога. Диалог ведётся на одном из естественных языков. *Речевой интерфейс* – SILK-интерфейс, обмен информацией в котором происходит за счёт диалога, в процессе которого компьютерная система и пользователь общаются с помощью речи. Данный вид интерфейса наиболее приближен к естественному общению между людьми.

Адаптивный интерфейс – пользовательский интерфейс, который изменяется на основе потребностей пользователя или контекста.

Интеллектуальный интерфейс – пользовательский интерфейс, который может предположить дальнейшие действия пользователей и представить информацию на основе этого предположения.

Мультимодальный интерфейс – пользовательский интерфейс, предназначенный для обработки двух или более комбинированных режимов пользовательского ввода, таких как речь, перо, касание, ручные жесты и взгляд, скоординированным образом с выводом мультимедийной системы.

Пользовательский интерфейс ostis-системы представляет собой специализированную *ostis-систему*, ориентированную на решение интерфейсных задач, и имеющую в своем составе базу знаний и решатель задач пользовательского интерфейса *ostis*-системы. Для решения задачи построения пользовательского интерфейса в базе знаний *пользовательского интерфейса ostis-системы* необходимо наличие sc-модели *компонентов пользовательского интерфейса, интерфейсных действий пользователей*, а также классификации *пользовательских интерфейсов* в целом. При проектировании интерфейса используется компонентный подход, который предполагает представление всего интерфейса приложения в виде отдельных специфицированных компонентов, которые могут разрабатываться и совершенствоваться независимо.

Компонент пользовательского интерфейса – знак фрагмента базы знаний, имеющий определённую форму внешнего представления на экране.

Компонент пользовательского интерфейса

⇒ разбиение*:

- {• *атомарный компонент пользовательского интерфейса*
- *неатомарный компонент пользовательского интерфейса*

}

Атомарный компонент пользовательского интерфейса – компонент пользовательского интерфейса, не содержащий в своём составе других компонентов пользовательского интерфейса.

Неатомарный компонент пользовательского интерфейса – компонент пользовательского интерфейса, состоящий из других компонентов пользовательского интерфейса.

Визуальная часть пользовательского интерфейса ostis-системы – часть базы знаний пользовательского интерфейса *ostis*-системы, содержащая необходимые для отображения пользовательского интерфейса компоненты.

визуальная часть пользовательского интерфейса ostis-системы

С *неатомарный компонент пользовательского интерфейса*

Компоненты пользовательского интерфейса могут быть отнесены к одной из трех категорий: *компонент пользовательского интерфейса для отображения, декоративный компонент пользовательского интерфейса, интерактивный компонент пользовательского интерфейса*.

Полная классификация компонентов пользовательского интерфейса приведена далее:

интерактивный компонент пользовательского интерфейса

компонент ввода данных

компонент ввода данных с прямой ответной реакцией

область рисования

ползунок

компонент ввода текста с прямой ответной реакцией (однострочное текстовое поле, многострочное текстовое поле)

компонент выбора (компонент выбора одного значения, компонент выбора нескольких значений)

компонент выбора данных (выбираемый элемент, радиокнопка, переключатель, флаговая кнопка)

компонент ввода данных без прямой ответной реакции

- кнопка-счётчик
- компонент ввода движений
- компонент речевого ввода
- компонент для представления и взаимодействия с пользователем
 - активирующий компонент
 - компонент непрерывной манипуляции
 - компонент редактирования размера
 - полоса прокрутки
 - компонент запроса действий
 - компонент выбора команд
 - пункт меню
 - кнопка
 - компонент ввода команд
- компонент пользовательского интерфейса для отображения
 - компонент вывода
 - компонент вывода видео
 - компонент вывода звука
 - компонент вывода изображения
 - компонент вывода графической информации
 - индикатор выполнения
 - диаграмма
 - карта
 - компонент вывода текста
 - сообщение
 - заголовок
 - параграф
 - декоративный компонент пользовательского интерфейса
 - пустое пространство
 - разделитель
 - контейнер
 - списковый контейнер
 - древовидный контейнер
 - узловый контейнер
 - таблично-строковый контейнер
 - таблично-клеточный контейнер
 - панель вкладок
 - панель вращения
 - меню
 - строка меню
 - панель инструментов
 - строка состояния
 - панель прокрутки
 - окно
 - модальное окно

□□□ немодальное окно

§ 4.1.2. Интерфейсные действия пользователей ostis-систем

Действие, выполняемое пользователем над некоторым компонентом пользовательского интерфейса, называется интерфейсным действием. Для связи данного действия с компонентом пользовательского интерфейса и необходимым к выполнению внутренним действием системы используется отношение инициируемое пользовательским интерфейсом действие*.

Классификация интерфейсных действий:

интерфейсное действие пользователя

- ▷ *действие мышью*
 - ▷ *прокрутка мышью*
 - ▷ *прокрутка мышью вверх*
 - ▷ *прокрутка мышью вниз*
 - ▷ *наведение мышью*
 - ▷ *отпускание мышью*
 - ▷ *нажатие мыши*
 - ▷ *одиночное нажатие мыши*
 - ▷ *двойное нажатие мыши*
 - ▷ *жест мышью*
 - ▷ *отведение мышью*
 - ▷ *перетаскивание мышью*
- ▷ *действие голосом*
- ▷ *действие клавиатурой*
 - ▷ *нажатие функциональной клавиши*
 - ▷ *нажатие клавиши набора текста*
- ▷ *действие осязанием*
- ▷ *действие сенсором*
 - ▷ *нажатие сенсора*
 - ▷ *одиночное нажатие сенсора*
 - ▷ *двойное нажатие сенсора*
 - ▷ *жест по сенсору*
 - ▷ *жест по сенсору одним пальцем*
 - ▷ *жест по сенсору несколькими пальцами*
 - ▷ *отпускание сенсором*
 - ▷ *перетаскивание сенсором*
- ▷ *действие пером*
 - ▷ *нажатие функциональной клавиши пером*
 - ▷ *рисование пером*
 - ▷ *написание текста пером*

Прокрутка мышью – интерфейсное действие пользователя, соответствующее прокрутке содержимого некоторого компонента пользовательского интерфейса при помощи мыши.

Наведение мышью – интерфейсное действие пользователя, соответствующее появлению курсора мыши в рамках компонента пользовательского интерфейса.

Отпускание мышью – интерфейсное действие пользователя, соответствующее отпусканню некоторого компонента пользовательского интерфейса в рамках другого компонента пользовательского интерфейса при помощи мыши.

Нажатие мыши – интерфейсное действие пользователя, соответствующее выполнению нажатия мыши в рамках некоторого компонента пользовательского интерфейса.

Отведение мышью – интерфейсное действие пользователя, соответствующее выходу курсора мыши за рамки компонента пользовательского интерфейса.

перетаскивание мышью – интерфейсное действие пользователя, соответствующее перетаскиванию компонента пользовательского интерфейса при помощи мыши.

Нажатие сенсора – интерфейсное действие пользователя, соответствующее выполнению нажатия сенсора в рамках некоторого компонента пользовательского интерфейса.

Жест по сенсору – интерфейсное действие пользователя, соответствующее выполнению некоторого жеста, выполняемого при помощи движения пальцев на экране сенсора.

отпускание сенсором – интерфейсное действие пользователя, соответствующее отпусканью некоторого компонента пользовательского интерфейса в рамках другого компонента пользовательского интерфейса при помощи сенсора.

перетаскивание сенсором – интерфейсное действие пользователя, соответствующее перетаскиванию компонента пользовательского интерфейса при помощи сенсора.

действие пером – интерфейсное действие пользователя, осуществляющееся при помощи пера на графическом планшете.

Класс интерфейсных действий пользователя – множество, элементами которого являются классы *интерфейсных действий пользователя*.

При взаимодействии пользователя с *компонентом пользовательского интерфейса* могут быть произведены различные интерфейсные действия. В зависимости от выполненного интерфейсного действия и компонента, над которым оно было выполнено, происходит инициирование некоторого *внутреннего действия системы*. Для задания такого инициируемого при взаимодействии с пользовательским интерфейсом действия используется указанное отношение. Первым компонентом связки отношения *инициируемое пользовательским интерфейсом действие** является связка, элементами которой являются элемент множества компонентов пользовательского интерфейса и элемент множества *класс интерфейсных действий пользователя*. Вторым компонентом является элемент множества *класс внутренних действий системы*.

§ 4.1.3. Сообщения, входящие в ostis-систему и выходящие из неё

Сообщение – дискретная информационная конструкция, используемая в процессе передачи от отправителя к получателю.

В качестве отправителя сообщения может выступать как пользователь системы, так и сама система. В случае ostis-системы сообщение может быть эффекторным либо рецепторным.

Эффекторное сообщение ostis-системы – сообщение ostis-системы, формируемое самой ostis-системой при возникновении некоторых ситуаций. К ситуациям, инициирующим возникновение эффекторных сообщений, можно отнести:

- ситуации, возникающие при анализе деятельности самого пользователя. Например, задание аргументов, не соответствующих типу инициируемого действия или появление подсказок при использовании компонентов пользовательского интерфейса;
- ситуации, возникающие при анализе синтаксиса текстов внешних языков. Например, неполнота сформированного предложения на внешнем языке или использование конструкций, нехарактерных или некорректно использованных в контексте отдельно взятого внешнего языка/

Рецепторное сообщение ostis-системы – сообщение ostis-системы, являющееся реакцией на императивное сообщение (сообщение, побуждающее к какому-либо действию). Возможными реакциями ostis-системы на императивное сообщение пользователя являются:

- указание факта завершения выполнения некоторой задачи, что, например, характерно для поведенческих действий;
- получение ответа на поставленную задачу, формируемого либо в результате анализа базы знаний пользовательского интерфейса, либо в результате анализа предметной части базы знаний самой ostis-системы.

Сообщение пользователя ostis-системы может быть сформировано как на внешнем языке (языке, внешнем по отношению к ostis-системе, который не используется для коммуникации внутри системы), так и на внутреннем языке (SC-коде).

Любое сообщение может быть атомарным (не содержащим в своем составе другие сообщения) либо неатомарным (сообщение, в состав которого входят другие сообщения).

Типология сообщений представлена в следующем фрагменте:

сообщение

⇒ *разбиение**:

- {• *сообщение пользователя системы*
 - ⊂ *сообщение пользователя ostis-системы*

- сообщение системы
 - }
- ⇒ разбиение*:
 - {• атомарное сообщение
 - атомарное сообщение
 - }
- ⇒ разбиение*:
 - {• сообщение на естественном языке
 - сообщение на искусственном языке
 - }
- С графическое сообщение
 - := [сообщение, содержащее графическую информацию]
- С видео-сообщение
 - := [сообщение, содержащее видео-информацию]
- С аудио-сообщение
 - := [сообщение, представленное в звуковом формате]
- С обонятельное сообщение
 - := [сообщение, содержащее информацию о запахах]
- С текстовое сообщение
 - := [сообщение, содержащее текстовую информацию]
- С сообщение, требующее трансляции
 - := [сообщение, которое необходимо сформировать системой для дальнейшей передачи его пользователю]
- С протранслированное сообщение
 - := [сообщение, которое было сформировано системой для дальнейшей передачи его пользователю]

§ 4.1.4. Действия и внутренние агенты пользовательского интерфейса ostis-системы

Интерфейсное действие пользователя, как правило, инициирует некоторое внутреннее действие системы.

внутреннее действие системы

- ▷ внутреннее действие ostis-системы

внутреннее действие ostis-системы

- := [действие в sc-памяти]
- := [действие, выполняемое в sc-памяти]

Каждое *внутреннее действие ostis-системы* обозначает некоторое преобразование, выполняемое некоторым *sc-агентом* (или коллективом *sc-агентов*) и ориентированное на преобразование *sc-памяти*.

действие в sc-памяти

- ▷ действие в sc-памяти, инициируемое вопросом
- ▷ действие редактирования базы знаний ostis-системы
- ▷ действие установки режима ostis-системы
- ▷ действие редактирования файла, хранимого в sc-памяти
- ▷ действие интерпретации программы, хранимой в sc-памяти

Среди агентов интерпретации модели пользовательского интерфейса ostis-систем можно выделить агент генерации интерфейса на основе его модели и агент обработки пользовательских действий.

В качестве входного параметра агент генерации интерфейса на основе его модели принимает экземпляр компонента пользовательского интерфейса для отображения. Результатом работы является графическое представление указанного компонента с учетом используемой реализации платформы интерпретации семантических моделей ostis-систем.

Агент обработки пользовательских действий реагирует на появление в базе знаний системы экземпляра интерфейсного действия пользователя, находит связанный с ним класс внутреннего действия и генерирует экземпляр данного внутреннего действия для последующей обработки.

Глава 4.2.

Естественно-языковые интерфейсы *ostis*-систем

Никифоров С.А.

Гойло А.А.

Крощенко А.А.

Захарьев В.А.

Цянь Л.

⇒ *аннотация**:

[В данной главе рассматривается подход к реализации естественно-языковых интерфейсов интеллектуальных компьютерных систем нового поколения, построенных по технологии *OSTIS*, а также предлагается модель контекста диалога. В данном подходе все этапы анализа, включая лексический, синтаксический и семантический анализ могут производиться непосредственно в базе знаний такой системы. Такой подход позволит эффективно решать такие задачи как управление глобальным и локальным контекстами диалога, а также разрешение языковых явлений таких как анафоры, омонимия и эллиптические фразы.

- расписать сопоставление токенов с лексемами;
- перерисовать картинки, часть из них предназначалась для размещения в колонке;
- расписать синтаксический анализ, как находятся составляющие,
- пройтись по подсказкам от плагина, поправить форматирование,
- формализовать пр. о..

Список вопросов:

- что придумать с аналогичными картинками, как те, что были использованы в главе про языки - дублировать в одной книге плохо, но ссылаться через 100 страниц тоже.

]

В настоящее время существует большое количество различных интерфейсов компьютерных систем, что усложняет интероперабельность между такими системами и людьми в силу необходимости ознакомления с интерфейсом каждой новой системы, который зачастую может быть не интуитивно понятен.

Одной из основных особенностей интеллектуальных компьютерных систем нового поколения должен являться пользовательский интерфейс, способный обеспечить эффективное взаимодействие пользователя с системой в условиях его общей профессиональной неподготовленности.

Одной из наиболее естественных и удобных форм передачи информации между людьми является речь, что обуславливает все большее распространение естественно-языковых интерфейсов **GlobalMarket**. В настоящий момент времени уже ни у кого не вызывает сомнения, что данная форма взаимодействия человека и машины играет и будет играть значительную роль во взаимодействии с различными компьютерными системами.

Однако необходимо отметить, что большое многообразие языков (как естественных, так и искусственных) ведет к необходимости упрощения процесса создания таких интерфейсов для каждого отдельно взятого языка.

В основе большинства подходов к обработке и пониманию естественного языка лежит машинное обучение **NLP_as_a_service**, **NLP_in_pharmacology**. Несомненно, для большинства широко распространенных языков модели для обработки естественного языка работают очень хорошо и совершенствуются с каждым днем, но несмотря на успехи в данной области, данный подход имеет ряд недостатков:

- проблемы при работе с различными областями, например, значения слов или предложений могут быть различными в зависимости от предметной области. Таким образом, модели для NLP могут хорошо работать для отдельной предметной области, но не подходит для широкого применения **NLPOverview**,
- создание новой модели требует наличия большого объема данных, а качество таких данных напрямую влияет на качество получаемой модели, что ведет к большим затратам на ее обучение **strubell2019energy large_language_models**;
- данные модели представляют собой "черный ящик", т. к. данные модели не обладают средствами для обоснования своего вывода;
- каждая такая модель решает только свой узкий класс задач, отсутствует общий подход к обработке естественного языка. **NLPOverview**

Данные недостатки используемых методов являются причиной части недостатков современных систем, реализующих естественно-языковой интерфейс, так, несмотря на то, что сейчас существует большое количество речевых

ассистентов, создаваемых разными компаниями `site_url_alexa`, `site_url_siri`, `site_url_gassist`, `site_url_cortana`, они обладают схожими недостатками, например, исключительно распределенной реализацией, в силу недостаточной для запуска ресурсоемких моделей производительности устройств конечных пользователей. Это в свою очередь ведет к проблемам с приватностью РВА.

Подмодуль понимания речи данных систем формирует конструкцию, отражающую смысл сообщения используя фреймовую модель. Упрощенный пример такой конструкции приведен на рисунке [Пример формализованного смысла сообщения](#).

```
{
  "text": "how many people between Tuesday and Friday",
  "intents": [
    { "name": "inquiry" }
  ],
  "entities": {
    "metric": [
      { "role": "metric", "value": "metric_visitor" }
    ],
    "datetime": [
      { "role": "datetime", "type": "interval",
        "from": { "grain": "day", "value": "2020-05-05T00:00:00.000-07:00" },
        "to": { "grain": "day", "value": "2020-05-09T00:00:00.000-07:00" }
      }
    ]
  }
}
```

= [Пример формализованного смысла сообщения](#)

При этом для представления результатов промежуточных этапов обработки используются иные форматы, модули которые их реализуют не имеют какой-либо единой основы и взаимодействуют посредством специализированных программных интерфейсов между ними, что приводит к несовместимости способов представления результатов на различных этапах обработки и конечного результата обработки текстов. Данная несовместимость в свою очередь ведет к существенным накладным расходам при разработке такой системы и в особенности при ее модификации.

В качестве решения проблемы совместимости предлагается использование подхода к обработке естественного языка на основе его формальной модели в виде набора онтологий, сформированных с использованием универсальных средств представления знаний, что будет способствовать интероперабельности как компонента по обработке естественного языка в целом с другими компонентами системы, так и между составляющими самого данного компонента.

Целью главы является формирование модели интерфейса, в основе которой лежит подход к обработке естественного языка на основе онтологий, содержащих формальное описание естественного языка.

§ 4.2.1. Предметная область и онтология естественно-языковых интерфейсов ostis-систем

Естественно-языковой интерфейс – SILK-интерфейс (Speech – речь, Image – образ, Language – язык, Knowledge – знание), обмен информацией между компьютерной системой и пользователем в котором происходит за счёт диалога. Диалог ведётся на одном из естественных языков.

естественно-языковой интерфейс

↪ *речевой интерфейс*

Речевой интерфейс – SILK-интерфейс, обмен информацией в котором происходит за счёт диалога, в процессе которого компьютерная система и пользователь общаются с помощью речи. Данный вид интерфейса наиболее приближен к естественному общению между людьми.

В предлагаемом подходе можно выделить следующие этапы обработки естественного языка:

- лексический анализ;
- синтаксический анализ;
- понимание сообщения.

В свою очередь, лексический анализ включает в себя декомпозицию текста на токены и их сопоставление с лексемами.

Понимание сообщения сводится к генерации вариантов значения сообщения и выбору из них корректного на основании контекста, а также погружение его в данный контекст.

Ниже приведена структура решателя задач естественно-языкового интерфейса.

Решатель задач естественно-языкового интерфейса

⇒ декомпозиция абстрактного sc-агента*:

- {• Абстрактный sc-агент лексического анализа
 - ⇒ декомпозиция абстрактного sc-агента*:
 - {• Абстрактный sc-агент декомпозиции текста на токены
 - Абстрактный sc-агент сопоставления токенов с лексемами
 - Абстрактный sc-агент синтаксического анализа
 - Абстрактный sc-агент понимания сообщения
- }

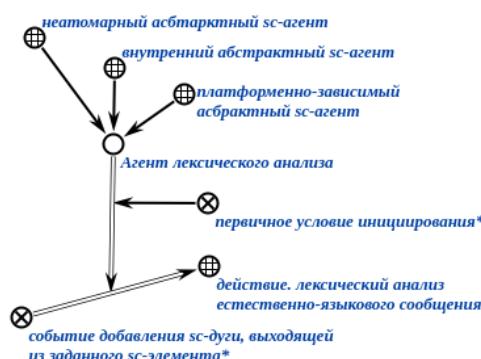
В свою очередь, Абстрактный sc-агент понимания сообщения декомпозируется на:

Агент понимания сообщения

⇒ декомпозиция абстрактного sc-агента*:

- {• Абстрактный sc-агент генерации вариантов значения сообщения
 - Абстрактный sc-агент выбора и обновления контекста
 - ⇒ декомпозиция абстрактного sc-агента*:
 - {• Абстрактный sc-агент разрешения контекста
 - Абстрактный sc-агент выбора смысла сообщения на основе контекста
 - Абстрактный sc-агент погружения сообщения в контекст
- }

Для каждого агента в базе знаний должна находиться спецификация, пример фрагмента такой спецификации приведен на рисунке [Пример спецификации агента..](#).



= Пример спецификации агента.

§ 4.2.2. Предметная область и онтология лексического анализа естественно-языковых сообщений, входящих в ostis-систему

действие. лексический анализ естественно-языкового сообщения

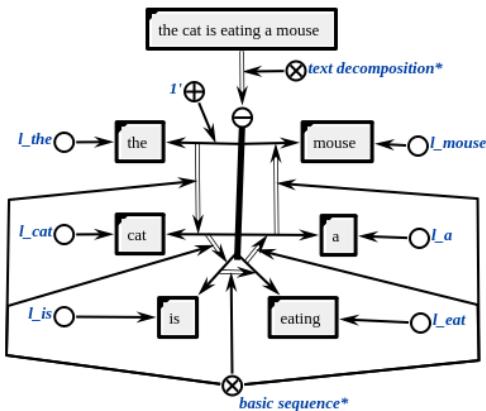
⇒ обобщенная декомпозиция*:

- {• действие. декомпозиция текста на токены
 - действие. сопоставление токенов с лексемами
- }

С точки зрения ostis-системы, любой естественно-языковой текст является *файлом* (т.е. SC-узлом с содержимым).

Этап лексического анализа представляет собой декомпозицию текста на последовательность токенов и сопоставление лексем с получившимися при данной декомпозиции токенами. Следует отметить, что данные токены при необходимости могут сопоставляться не с лексемами, а с их подмножествами, входящими в ее морфологическую парадигму, соответствующими определенным грамматическим категориям: падежу, числу, роду и т.д.

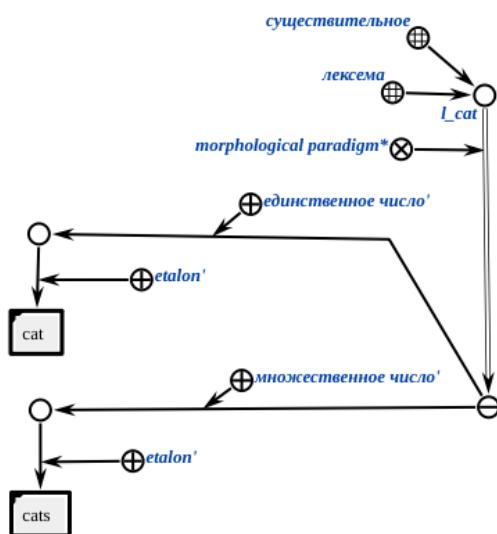
Результат лексического анализа представлен на рисунке [Пример результата лексического анализа..](#)



= [Пример результата лексического анализа.](#)

Для осуществления лексического анализа, в базе знаний системы также должен присутствовать словарь, содержащий лексемы и их различные формы.

Под лексемой понимается единица словарного состава языка, которая представляет собой множество всех форм некоторого слова. Пример спецификации лексемы в базе знаний приведен на рисунке [Пример спецификации лексемы в базе знаний..](#)

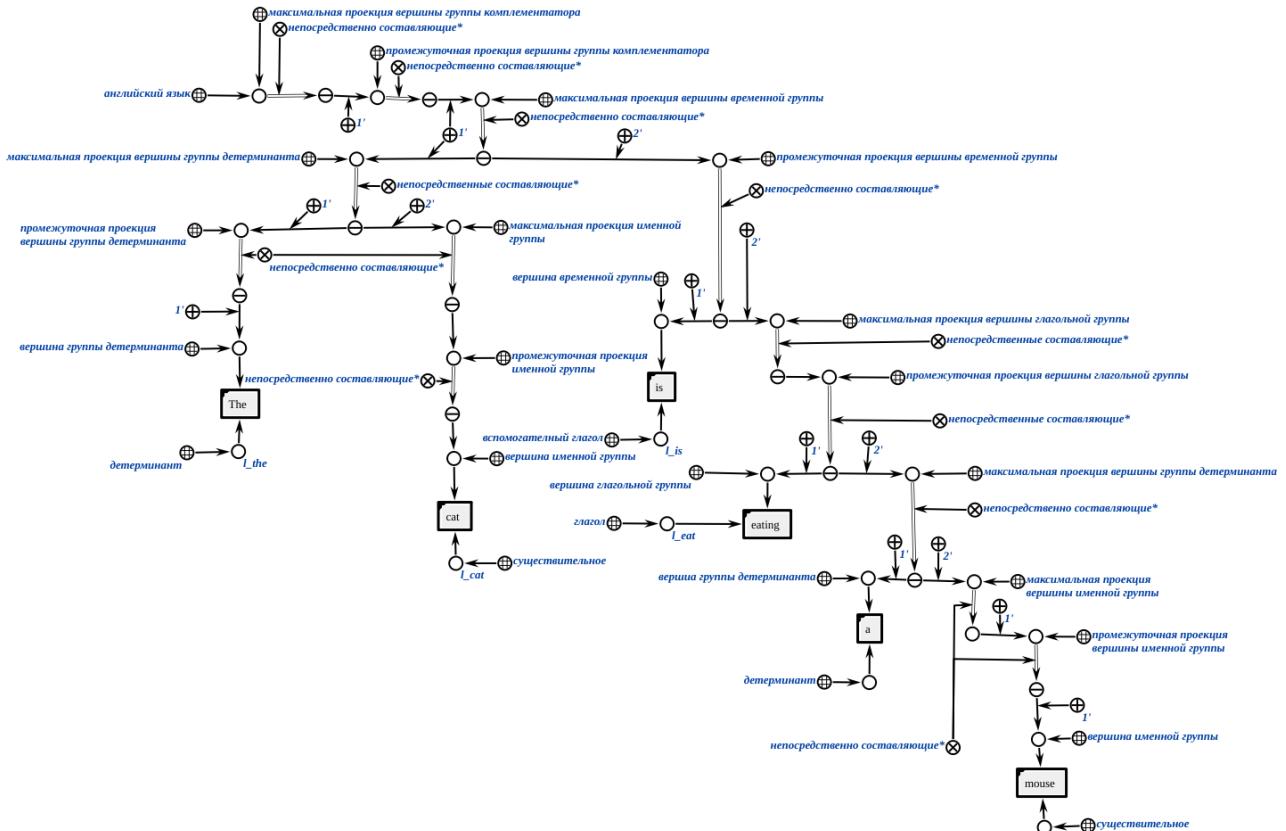


= [Пример спецификации лексемы в базе знаний.](#)

§ 4.2.3. Предметная область и онтология синтаксического анализа естественно-языковых сообщений, входящих в ostis-систему

Агент синтаксического анализа выполняет переход от размеченного на лексемы текста к его синтаксической структуре. При этом из-за невозможности разрешения структурной неоднозначности на этапе синтаксического анализа, его результатом в общем случае будет являться множество потенциальных синтаксических структур.

Пример одной синтаксической структуры представлен на рисунке [Пример синтаксической структуры..](#)



= [Пример синтаксической структуры.](#)

§ 4.2.4. Предметная область и онтология понимания естественно-языковых сообщений, входящих в ostis-систему

действие. понимание естественно-языкового сообщения

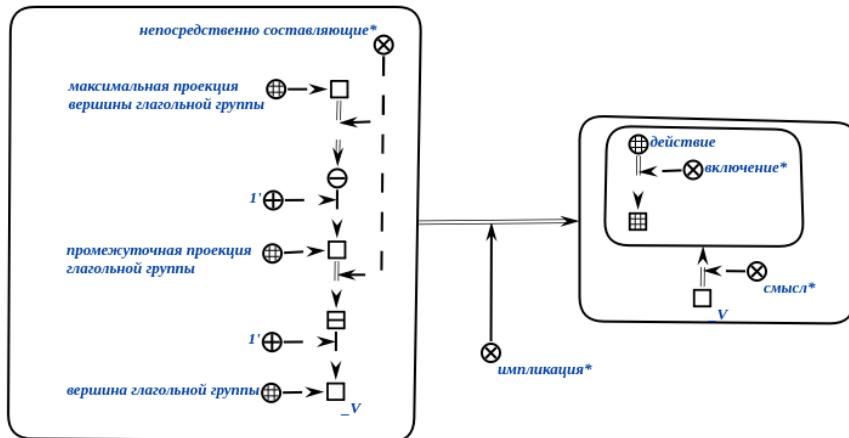
⇒ обобщенная декомпозиция*:

- {• действие. генерация вариантов значения сообщения
 - действие. выбор и обновление контекста
- ⇒ обобщенная декомпозиция*:
- {• действие. разрешение контекста
 - действие. выбор смысла сообщения на основе контекста
 - действие. погружение сообщения в контекст
- }

Действие. генерация вариантов значения сообщения – действие, в ходе которого осуществляется формирование строгой дизъюнкции потенциально эквивалентных структур.

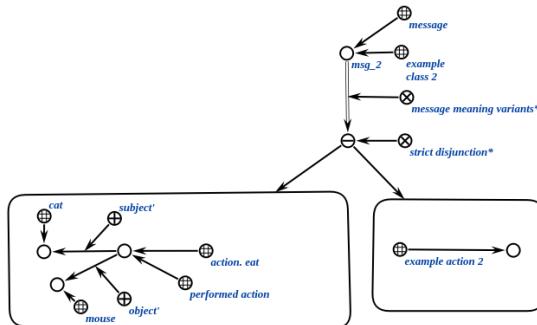
*Потенциально эквивалентная структура** – бинарное ориентированное отношение, связывающее структуру и множество структур, которые потенциально могут быть эквивалентны ей, однако для достоверного определения факта требуются дополнительные действия.

При этом, переход от результата синтаксического анализа к потенциально эквивалентным сообщению структурам осуществляется по правилам, содержащихся в предметной области денотационной семантики. Пример одного из правил представлен на рисунке [Пример правила перехода от синтаксической структуры к семантике..](#)



= [Пример правила перехода от синтаксической структуры к семантике.](#)

В результате данного действия в базе знаний формируется структура, описывающая возможные варианты смысла сообщения, пример такой структуры в терминах грамматики составляющих **X_bar_Syntax** приведен на [Пример конструкции, описывающей потенциальные смыслы сообщения..](#) Наличие нескольких таких структур объясняется тем, что в общем случае на этапе синтаксического анализа выполняется генерация нескольких вариантов синтаксической структуры. Выбор корректного значения сообщения будет осуществлен в ходе выполнения последующих действий.



= [Пример конструкции, описывающей потенциальные смыслы сообщения.](#)

Следует отметить, что при необходимости смысл сообщения может быть сгенерирован не только на основании его синтаксической структуры в терминах грамматики составляющих, но и других знаний о данном сообщении, например выделенных из текста данного сообщения троек вида субъект-отношение-объект, результата его классификации и т. п.

Дальнейшие этапы процесса понимания сообщения выполняются на основе контекста.

Контекст - sc-структура, содержащая знания, которыми оперирует система в ходе одного или нескольких диалогов. В общем случае, данные знания включают в себя как предварительно занесенные в БЗ, так и полученные в ходе работы с сенсорами и/или диалога.

контекст диалога

⊆ контекст
⇒ subdividing*:

Типология контекстов диалога по глобальности[^]

- = {• тематический контекст
 - пользовательский контекст
 - глобальный контекст
}

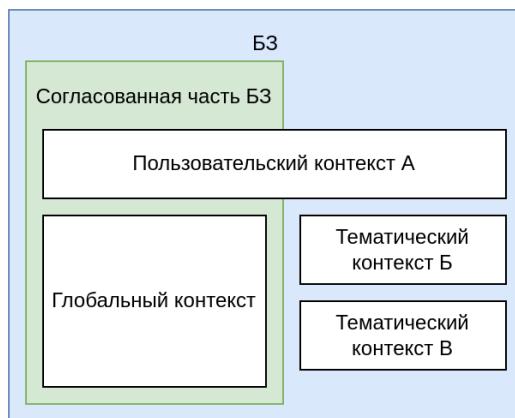
Тематический контекст – контекст диалога, содержащий специфические для темы сведения (сведения, полученные во время ведения диалога, на определенную тематику, например, при диалоге об определенном наборе сущностей).

*Множество тематических контекстов диалога** – бинарное ориентированное отношение, диалог с ориентированным множеством его тематических контекстов.

Пользовательский контекст – контекст диалога, содержащие специфические для пользователя сведения, которые могут быть использованы в диалоге с ним на любую тематику. В общем случае пользовательский контекст имеет пересечение с согласованной частью БЗ (предварительно занесенная в БЗ достоверная информация о пользователе, прошедшая необходимую модерацию), но не включается в нее целиком (часть, полученная в ходе диалога в которой мы не уверены). Пример соотнесения различных типов контекстов с согласованной частью базы знаний приведен на рисунке *Соотношение контекстов с согласованной частью баз знаний..*

Глобальный контекст – контекст диалога, содержащий сведения, которые могут быть необходимы при ведении диалога с любым пользователем. Глобальный контекст – подмножество согласованной части БЗ, содержащее те сведения, что допустимо использовать в диалоге. Например, в диалоге с определенным пользователем не нужно использовать:

- находящуюся в базе знаний служебную информацию, необходимую для работы системы, но не предназначенную для использования в диалоге;
- части пользовательских контекстов иных пользователей.



= *Соотношение контекстов с согласованной частью баз знаний.*

контекст диалога

⇒ subdividing*:

Типология контекста по сроку достоверности знаний[^]

- = {• неизменяемый в ходе работы системы контекст диалога
 - изменяемый в ходе работы системы контекст диалога
}

Неизменяемый в ходе работы системы контекст диалога содержит в себе знания, необходимые для обеспечения выполнения системой своих функций, которые были заложены в нее априорно ее разработчиками и/или администраторами и не изменяются в ходе ее функционирования на постоянной основе.

Изменяемый в ходе работы системы контекст диалога содержит в себе знания, необходимые для обеспечения выполнения системой своих функций, которые были ей получены в ходе ее работы и/или достоверность которых скоротечна.

изменяемый в ходе работы системы контекст диалога

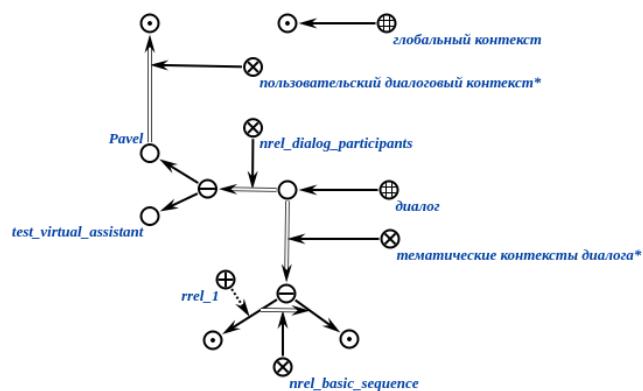
⇒ subdividing*:

Типология изменяемых в ходе работы системы контекстов по источнику знаний[^]

- = {• контекст диалога, содержащий знания из внешних источников
 - контекст диалога, содержащий знания, полученные в ходе диалога
 - }
- ⇒ subdividing*:
- Типология изменяемых контекстов по степени их достоверности[^]**
- = {• достоверный контекст диалога
 - недостоверный контекст диалога
 - }

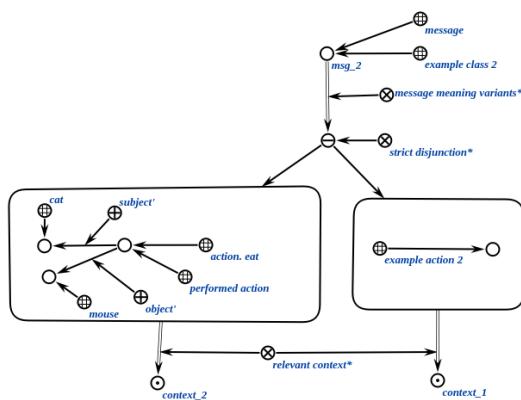
Подмножество контекста может включаться в согласованную часть БЗ, например, если речь идет о каких-то предварительно занесенных в БЗ биографических сведениях – дате рождения и т. п.

В каждый момент времени с пользователем связан 1 пользовательский диалоговый контекст (содержащий, по крайней мере известные заранее факты о нем: имя, возраст и т. п.) и несколько тематических. Пример спецификации контекстов представлен на рисунке [Пример спецификации контекстов..](#)



= [Пример спецификации контекстов.](#)

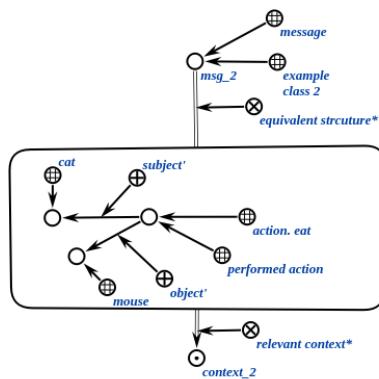
Так, действие. разрешение контекста сводится к сопоставлению каждому варианту его значения соответствующего контекста. Выбор производится на основании значения функции $F_{CTD}(T, C)$, где T - вариант трансляции, С - тематический контекст. Подходящим контекстом для варианта трансляции считается тот, для которого значение этой функции максимально. В случае, если подходящий контекст не найден, генерируется новый. Пример результата данного действия представлен на рисунке [Пример сообщения, всем вариантам значения которого сопоставлен контекст..](#)



= [Пример сообщения, всем вариантам значения которого сопоставлен контекст.](#)

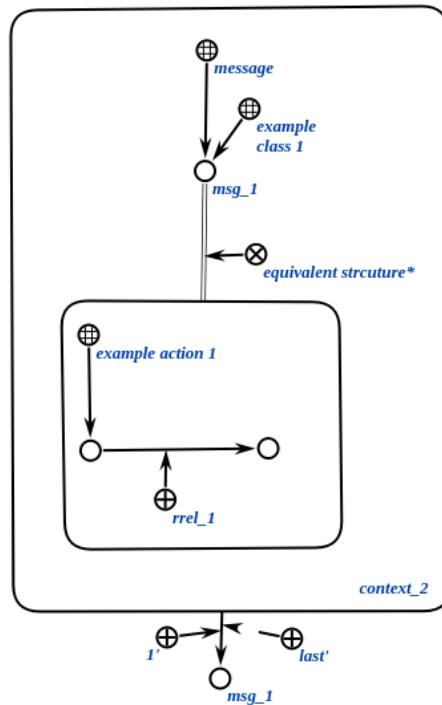
Действие. выбор смысла сообщения представляет собой выбор из множества вариантов трансляции и соответствующих им контекстов одной пары и обозначение ее как эквивалентной сообщению конструкции. В простейшем случае, на данном этапе допустимо выполнить выбор в соответствии с рассчитанными на предыдущем этапе для пар потенциально эквивалентных структур и соответствующих им контекстов значениями функции $F_{CTD}(T, C)$ и выбрать пару, для которой оно максимально, однако при необходимости также возможно введение и отдель-

ной функции. Пример результата данного действия представлен на рисунке *Пример конструкции, описывающей эквивалентную сообщению структуру..*



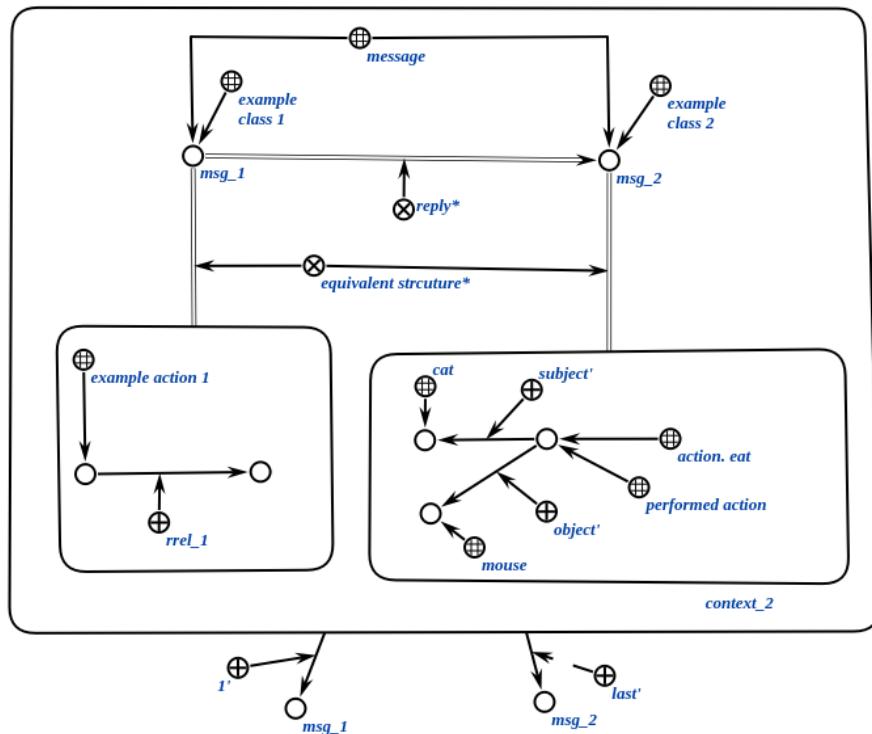
= *Пример конструкции, описывающей эквивалентную сообщению структуру.*

Действие. погружение сообщения в контекст представляет собой погружение полученного смысла сообщения в контекст. Кроме выбранного смысла сообщения, в контекст может добавляться и иная необходимая для обработки сообщения информация. Кроме того, на данном этапе на основе хранящихся в контексте сведений также должно выполняться разрешение местоимений. Примеры контекста до погружения в него сообщения и после погружения представлены на рисунках *Пример контекста до погружения сообщения.* и *Пример контекста после погружения сообщения..*



= *Пример контекста до погружения сообщения.*

Таким образом, актуальная информация собирается в тематический контекст, объединив который с контекстом пользователя и глобальным контекстом можно получить общий контекст, на основании которого должны осуществляться требуемые действия системы, включая генерацию ответа системы.



= Пример контекста после погружения сообщения.

§ 4.2.5. Предметная область и онтология синтеза естественно-языковых сообщений ostis-системы

§ 4.2.6. Модели, методы и средства адаптации пользовательских интерфейсов к носителям китайского языка

Глава 4.3.

Аудиоинтерфейс ostis-систем

⇒ *авторы**:

- Азаров И.С.
- Ващевич М.И.
- Захарьев В.А.
- Лихачев Д.С.
- Петровский Н.А.

⇒ *аннотация**:

[Данная глава посвящена рассмотрению вопросов создания аудио и речевых интерфейсов для интеллектуальных компьютерных систем нового поколения. Предлагается использование подхода на основе онтологического проектирования и формализации системы понятий из предметной области аудиоинтерфейсов, посредством технологии OSTIS. Изложены основные идеи лежащие в основе данного подхода, а также их отличительные особенности от общепринятых. Показано, что в перспективе, использование данного подхода может обеспечить свойства унификации, семантической совместимости и интероперабельности, при разработке аудио и речевых интерфейсов, что в итоге позволит существенным образом сократить издержки при создании интеллектуальных компьютерных систем нового поколения для решения комплексных задач.]

⇒ *ключевое понятие**:

- аудио интерфейс интеллектуальных компьютерных систем
- речевой интерфейс интеллектуальных компьютерных систем
- цифровая обработка сигналов
- обработка речевых сигналов

⇒ *библиографическая ссылка**:

- Pearl2016
- Chen G., *TempoLjfFDoSsGPD-2021art*
- Lu2002content
- Fernandes2022

§ 4.3.1. Анализ существующих подходов к разработке аудиоинтерфейсов интеллектуальных компьютерных систем

Разговорная речь является одной из наиболее естественных и эффективных форм передачи информации между людьми. Этот факт объясняет значительный интерес исследователей к вопросам развития и применения речевых интерфейсов для обеспечения человека-машинного взаимодействия в составе современных коммуникационных, мультимедийных и интеллектуальных систем.

Более всеобъемлющей формой обеспечения взаимодействия с пользователем и окружающей средой по средствам анализа и синтеза акустических сигналов является аудиоинтерфейс. Данную разновидность интерфейса, выступающей родительской по отношению к речевым, можно кратко определить как аппаратно-программный комплекс осуществляющий анализ и синтез сигналов во всем доступном спектре параметров носителей акустической информации. Например, для решения задач анализа обстановки и событий происходящих в акустическом окружении системы, синтеза неречевых сигналов (звуков техногенного и природного характера, сигналов оповещения, музыки, и т.д.) .

Об актуальности направления разработки аудио и речевых интерфейсов свидетельствуют следующие основные тенденции развития данного направления:

- экономические показатели и прогнозы развития рынка речевых технологий, текущие среднегодовые темпы роста которого, по оценкам экспертов, составляют порядка 22%, а совокупный объём будет равен 59,6 млрд. долл. США к 2030;
- появление широкого спектра продуктов на основе речевого интерфейса, получивших массовое распространение. В первую очередь это персональные голосовые ассистенты, таких как «Alexa» (Amazon), «Siri» (Apple), «Cortana» (MicroSoft), «Алиса» (Yandex);

- интерес со стороны научного сообщества, выражющийся в росте публикаций в этом направлении исследований на 15% за последние 5 лет .

Необходимо отметить, что основная масса научных публикаций в данном направлении посвящена развитию базовых технологий, являющихся составляющими речевого интерфейса, таким как синтез речи по тексту, а также распознавание речи в текст . Последние достижения в этих направлениях, связанны с бурным развитием нейросетевых моделей и вычислительных средств. Они позволили довести качественные характеристики использования речевых технологий до комерческого уровня .

Большинство существующих систем, как правило, рассчитаны на решение определенного круга задач и сложно совместимы друг другом. Данный факт в особенности остро проявляется при проектировании сложных систем, наподобие интеллектуальных персональных диалоговых ассистентов, требующих использования многообразия различных видов обрабатываемой информации и различных моделей решения задач. Такие системы кроме стандартных модулей распознавания (ASR, automatic speech recognition) и синтеза (TTS, text to speech), на уровне аудиоинтерфейса также должны содержать модели, определяющие наличие/отсутствие речи в аудиосигнале в сложной акустической обстановке, классификации звуков окружающей среды, распознавание диктора и пр. Помимо этого элементы речевого интерфейса должны быть совместимы с более высокочувствительными модулями обработки естественно-языковой информации, такими как модули понимания (SLU, spoken language understanding) и генерации речи (SLG, spoken language generation), управления диалогом (DM, dialog manager) .

Всё это требует разработки подходов основанных не только на методах машинного обучения и обработке сигналов, но и обработки естественного языка, символьических методах искусственного интеллекта, онтологически проектировании и формализации предметной области аудиоинтерфейса. Это позволит создать системы, которые обладают полным спектром знаний в формализованном виде о типах задач которые они должны решать и методах доступных для их решения.

Необходимым условием для создания таких систем нового поколения, обладающих улучшенными характеристиками по критериям интероперабельности и гибкости является также тот факт, что данные системы должны быть построены на основе базовой технологии, позволяющей обеспечить такое единство формы представления информации на всех её уровнях.

Совокупность данных факторов приводит к необходимости создания интеллектуальных компьютерных систем нового поколения, которые будут включать в себя модули аудио и речевого интерфейса, построенные на основе принципов интероперабельности и семантической совместимости для решения комплексных задач.

§ 4.3.2. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов

Для достижения поставленной цели предлагается прибегнуть к подходу на основе принципов лежащих в основе "Стандарта открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем"или кратко "Стандарта технологии OSTIS".

Суть подхода заключается в рассмотрении процесса проектирования аудиоинтерфейса как интерфейсной подсистемы в рамках общего процесса разработки интеллектуальной компьютерной системы (ИКС) и построении её формальной логико-семантической модели.

Для создания подобной модели интеллектуальных компьютерных систем нового поколения необходимо:

- произвести декомпозицию информационной компьютерной системы на компоненты. Качество декомпозии при этом определяется простотой последующего синтеза общей формальной модели из формальных моделей выделенных компонентов.
- провести конвергенцию выделенных компонентов в целях построения совместимых (легко интегрируемых) формальных моделей этих компонентов;
- провести интеграцию построенных формальных моделей выделенных компонентов и получить общую формальную модель.

Общими методологическими принципами, являющимися основой перехода к ИКС нового поколения при этом являются:

- конвергенция и унификация ИКС и их компонентов;
- структурно-системное упрощение ИКС (принцип "Бритвы Оккама");
- ориентация на универсальные ИКС;
- синтез ИКС из совместимых компонентов;
- ориентация на создание синергетических ИКС.

Из общих методологических принципов вытекают следующие важные особенности предлагаемого подхода, которым необходимо следовать для достижения поставленной цели.

- смысловое представление знаний;
- агентно-ориентированная базовая модель обработки баз знаний, имеющих смысловое представление (инсерционное программирование в смысловом пространстве);
- семантическая структуризация баз знаний в виде иерархической системы предметных областей и соответствующих им онтологий, специфицирующих эти предметные области;
- интерфейс ИКС нового поколения, при этом, трактуется как специализированная интеллектуальная информационная система, ориентированная на решение интерфейсных задач соответствующей индивидуальной ИКС и глубоко интегрированная (встроенная) в эту ИКС.

В качестве технологической основы для реализации предлагаемого подхода будет использоваться Технология OSTIS . Системы, построенные на основе Технологии OSTIS, названы OSTIS-системами, соответственно подсистема аудиоинтерфейса, будет строиться как многократно используемый компонент, который в будущем будет при необходимости встраиваться в различные OSTIS-системы.

В качестве формальной основы для кодирования различной информации в базе знаний используется SC-код , тексты которого (sc-тексты) записываются в виде семантических сетей с базовой теоретико-множественной интерпретацией. Элементы таких сетей получили название sc-элементов (sc-узлов, sc-дуг).

Ориентация данной работы на Технологию OSTIS обусловлена ее следующими основными преимуществами:

- в рамках указанной технологии предложены унифицированные средства представления различных видов знаний, в том числе - метазнаний, что позволяет описать всю необходимую для анализа информацию в одной базе знаний в едином ключе ;
- используемый в рамках технологии формализм позволяет специфицировать в базе знаний не только понятия, но и любые внешние с точки зрения базы знаний файлы (например, фрагменты речевого сигнала), в том числе - синтаксическую структуру таких файлов;
- предложенный в рамках технологии подход к представлению различных видов знаний и моделей их обработки обеспечивает модифицируемость OSTIS-систем, т.е. позволяет легко расширять функциональные возможности системы, вводя новые виды знаний (новые системы понятий) и новые модели обработки знаний;

Поскольку аудиоинтерфейс ИКС нового поколения должен иметь архитектуру, соответствующую общим правилам построения OSTIS-систем, можно выделить и формализовать следующие осевые части:

Аудиоинтерфейс интеллектуальных компьютерных систем нового поколения

- ⇒ **сокращение*:**
[аудиоинтерфейс ИКС]
- ⇒ **обобщенная декомпозиция*:**
- {• база знаний подсистемы аудиоинтерфейса ИКС нового поколения
 - решатель задач подсистемы аудиоинтерфейса ИКС нового поколения
 - интерфейс для взаимодействия с остальными интерфейсными подсистемами OSTIS-системы
- }

Таким образом необходимо отметить что процесс разработки аудиоинтерфейса для ИКС нового поколения, подразумевает прежде всего создание семантически структуризованных баз знаний в виде иерархической системы предметных областей и соответствующих им онтологий, специфицирующих эти предметные области. Следовательно, первым шагом для достижения поставленной цели должен являться этап выделения и формализации сущностей аудио и речевого интерфейса для погружения данной информации в базу знаний интеллектуальной компьютерной системы.

С нашей точки зрения можно провести декомпозицию предметных областей и онтологий входящих в базу знаний аудиоинтерфейса по следующим основным направлениям:

Предметная область и онтология аудиоинтерфейса интеллектуальных компьютерных систем нового поколения

- ⇐ **декомпозиция*:**
- {• Предметная область и онтология задач аудиоинтерфейса
 - Предметная область и онтология моделей параметрического представления сигнала
 - Предметная область и онтология моделей классификации параметров сигнала
- }

Как видно во главу онтологии положен функциональный подход к декомпозиции предметных областей, что является вполне естественным поскольку соответствует природе задач, реализуемых аудиоинтерфейсом.

Представленные выше принципы в совокупности позволяют осуществлять конвергенцию и интеграцию компонентов как на уровне подсистемы аудиоинтерфейса так и на уровне всей ИКС в целом, что в свою очередь позволяет

“перевести” интеллектуальную информационную систему в класс гибридных, интероперабельных и синергетических систем.

Далее перейдём непосредственно к рассмотрению конкретных предметных областей и построению онтологии аудиоинтерфейса.

§ 4.3.3. Предметная область и онтология задач аудиоинтерфейса ostis-систем

Первым шагом на пути к построению базы знаний подсистемы аудиоинтерфейса ИКС нового поколения является формализация онтологии верхнего уровня. В основе данной онтологии предлагается положить формализованное представление основных сущностей предметной области и их свойств, а также функциональных задач, которые аудио и речевой интерфейс призваны решать.

К основным сущностям требующим формализации и погружения в БЗ относятся множества понятий представленные далее.

Одним из ключевых понятий требующих формализации является базовое определение самого сигнала а также основных разновидностей сигналов, в зависимости от их природы, представляющих наибольший интерес в области аудиоинтерфейсов. Чтобы сдеалть процесс описания средствами технологии OSTIS более ясным, перед непосредственным переходом к нему приведём примеры наиболее основных сущностей и понятий, которые требуют формализации и погружения в БЗ:

- сигнал;
- акустический сигнал;
- аудиосигнал;
- речевой сигнал.

В зависимости от способа математического описания обрабатываемого сигнала в OSTIS-системе, можно определить следующие их классы:

- аналоговый сигнал;
- дискретный сигнал;
- цифровой сигнал;
- периодический сигнал;
- апериодический сигнал;
- гармонический сигнал;
- тональный сигнал;
- шумовой сигнал;
- импульсный сигнал;

Формализуем также основные понятия связанные с характеристиками самого сигнала, по следующим основным его атрибутам:

- амплитуда сигнала;
- частота сигнала;
- фаза сигнала;
- интенсивность сигнала;
- длительность сигнала;
- мощность/энергия сигнала;
- осцилограмма сигнала;
- спектр сигнала;
- частота дискретизации сигнала;
- степень квантования сигнала;

К ключевым понятиям предметной области, лежащим в семантической окрестности подпространства функционального назначения аудиоинтерфейсов и обработки аудиосигналов, относятся:

- анализ аудиосигнала;
- синтез аудиосигнала;
- кодирование аудиосигнала;
- шумоочистка аудиосигнала;
- классификация аудиосигнала;
- классификация событий (Environmental Sound Classification, Acoustic Scenes and Events);
- детектирование аномалий (Anomalous Sound Detection);
- идентификация положения источника в пространстве (Sound Source Localization);

Основные понятия аудио и речевого интерфейса также тесно связаны с основными его характеристиками, которые можно разделить на следующие основные группы понятий:

- характеристики речевого сигнала;
- лингвистические характеристики сигнала;
- паралингвистические характеристики сигнала;
- экстралингвистические характеристики сигнала;
- сегментные характеристики речевого сигнала (Segmental Features);
- надсегментные характеристики речевого сигнала (Suprasegmental Features);
- громкость речевого сигнала;
- тембр речевого сигнала;
- темп речевого сигнала;
- частота основного тона сигнала;
- фонемный состав речевого сигнала.

Основными понятиями предметной области лежащими в семантической окрестности функционального назначения речевого сигнала и обработки аудиосигналов находятся следующие понятия:

- анализ речевого сигнала;
- детектирование наличия ключевых слов (Key Words Spotting);
- активация по ключевому слову (Wake Up Word Detection);
- детектирование наличия речевого сигнала (Voice Activity Detection);
- синтез речевого сигнала;
- синтез текста в речь (Text-to-Speech Synthesis);
- эмоциональный синтез текста в речь (Emotional Text-to-Speech Synthesis);
- синтез пения (Sing Synthesis);
- распознавание эмоций в речевом сигнале (Emotional Speech Recognition);
- распознавание диктора;
- сепарация речи (speech separation, speech diarization);
- классификация диктора (Speaker Recognition);
- верификация диктора (Speaker Verification);

Необходимо отметить, что вышеобозначенные понятия зачастую сложным и нетривиальным образом связаны между собой в процессе перехода от источников информации к непосредственным физическим параметрам. Такую сложную структуру сигнала можно представить в виде схемы его информационной структуры (рисунок 3). Этот факт требует от ИКС нового поколения формализации понятий для того чтобы система могла автоматически интерпретировать взаимосвязи между данными характеристиками при работе с аудио и речевыми сигналами. И как следствие выдать ответ пользователю, объясняющий на основе каких характеристик система пришла к тому или иному выводу.

Поскольку для построения ИКС нового поколения в фокусе лежат именно задачи связанные с обработкой речевых сигналов, решение которых необходимо в первую очередь для построения речевого интерфейса, акцент при формализации постараемся сделать на особенности данной предметной области.

Основу SC-модели БЗ составляет иерархическая система предметных областей и соответствующих им онтологий. Ниже показан верхний уровень иерархии части базы знаний, относящейся непосредственно к аудио и речевым интерфейсам.

Преведём формализованное представление некоторых из обозначенных выше понятий:

сигнал

- ⇒ **определение*:**

[физический процесс, несущий сообщение (информацию) о каком-либо событии, состоянии объекта наблюдения либо передающий команды управления, оповещения и т.д.]
- ▷ **акустический сигнал**
 - ⇒ **определение*:**

[сигнал, представляющей собой распространение упругих волн в газообразной, жидкой или твердой среде]
 - ▷ **аудиосигнал**
 - := [звуковой сигнал]
 - ⇒ **определение*:**

[акустический сигнал параметры которого находятся в пределах диапазона значений доступного для восприятия органами чувств человека]
 - ▷ **акустический сигнал**
 - ⇒ **примечание*:**

[диапазон частот звукового сигнала лежит в интервале от 20 до 20 000 Гц.]
 - ▷ **речевой сигнал**
 - ⇒ **определение*:**

[аудиосигнал, который образуется в результате прохождения воздушных потоков через речевой тракт человека. В результате всевозможных акустических преобразований происходит формирование различных звуков речи]

≡ *устная речь*

≡ *речеобразование*

⇒ *примечание**:

[механизм речеобразования человека представляет собой акустическую трубу с динамически изменяющимися параметрами поперечного сечения, возбуждаемую либо квазипериодической последовательностью импульсов, генерируемых голосовыми связками, либо турбулентным потоком воздуха, проталкиваемого сквозь сужения, в разных местах речевого тракта]

Формализация с точки зрения на уровне онтологии верхнего уровня будет представлено только для. Про разновидности моделей математического представления будет сказано в соответствующей главе далее.

математическая модель сигнала

⇐ *объединение**:

{• *аналоговый сигнал*

⇒ *определение**:

[сигнал параметры которого можно измерить в любой момент времени]

⇒ *определение**:

[сигнал, у которого каждый из представленных параметров описывается функцией времени и непрерывным множеством возможных значений]

• *дискретный сигнал*

⇒ *определение**:

[сигнал, у которого хотя бы один из представленных параметров описывается конечным множеством возможных значений]

⇐ *объединение**:

{• *дискретный по времени*
• *дискретный по амплитуде*

}

• *цифровой сигнал*

⇒ *определение**:

[сигнал, у которого каждый из представляющих параметров описывается функцией дискретного времени и конечным множеством возможных]

⇒ *включает**:

{• *дискретный по времени сигнал*
• *квантованный (дискретный) по амплитуде сигнал*

}

• *периодический сигнал*

• *aperiodический сигнал*

• *тональный сигнал*

• *гармонический сигнал*

• *импульсный сигнал*

• *шумовой сигнал*

}

Необходимо отметить, что по причине ограничений на размер материала для характеристик аудиосигнала приведём только иерархию общей их взаимосвязи, поскольку семантика данных понятий вполне характерна и для других областей технических наук и не требует подробных примеров и пояснений.

характеристики аудиосигнала

⇐ *объединение**:

{• *амплитуда сигнала*

• *частота сигнала*

• *фаза сигнала*

• *интенсивность сигнала*

• *длительность сигнала*

• *мощность сигнала*

• *спектр сигнала*

• *осцилограмма сигнала*

⇒ *определение**:

- [функция фиксирующая зависимость изменения характеристик сигнала (в первую очередь амплитуды) от времени]
- *спектrogramma сигнала*
 - ⇒ *определение*:*
[функция фиксирующая зависимость спектральной плотности мощности аудиосигнала от времени]
 - *частота дискретизации сигнала*
 - ⇒ *определение*:*
[значение частоты с которой производилась дискретизация сигнала по времени в процессе аналогово-цифрового преобразования]
 - ⇐ *типовье значения*:*
 - {• 8000 Гц
 - 16000 Гц
 - 22050 Гц
 - 44100 Гц
 - 48000 Гц
 - *уровень квантования сигнала*
 - ⇒ *определение*:*
[допустимое количество дискретных уровней сигнала выраженное как степень двойки и применяемое в процессе квантования сигнала по уровню в процессе аналогово-цифрового преобразования]
 - ⇐ *типовье значения*:*
 - {• 8 бит
 - 10 бит
 - 12 бит
 - 16 бит
 - 24 бита
- }

Описание предметной области моделей сигнала будут более подробно рассмотрены в следующем пункте работы, поэтому не считаем необходимым приводить её здесь.

- анализ аудиосигнала;
- синтез аудиосигнала;
- кодирование аудиосигнала;
- шумоочистка аудиосигнала;
- классификация аудиосигнала;
- классификация событий (Environmental Sound Classification, Acoustic Scenes and Events);
- детектирование аномалий (Anomalous Sound Detection);
- идентификация положения источника в пространстве (Sound Source Localization);

Описание предметной области моделей сигнала будут более подробно рассмотрены в следующем пункте работы, поэтому не считаем необходимым приводить её здесь. Онтологию основных характеристик речевого сигнала формулируем в следующем виде :

характеристики речевого сигнала

- ⇐ *объединение*:*
- {• *коммуникативные характеристики*
 - ⇒ *примечание*:*
[кодируют смысл передаваемого сообщения, зависят от намерений отправителя]
 - *информационные характеристики*
 - ⇒ *примечание*:*
[кодируют дополнительную информацию передаваемого сообщения, не зависят от намерений отправителя]
 - *информационные характеристики*
 - ⇒ *примечание*:*
[кодируют дополнительную информацию передаваемого сообщения, не зависят от намерений отправителя]
 - *сегментные характеристики речевого сигнала (Segmental Features)*
 - ⇒ *примечание*:*
[несут информацию о текущем состоянии источника на протяжении длительности одной или нескольких фонетических единиц]
 - *надсегментные характеристики речевого сигнала (Suprasegmental Features)*

- ⇒ *примечание*:*
[несут информацию о состоянии источника и переходах между ними на протяжении времени всего высказывания]
 - }
 - ▷ *лингвистические характеристики сигнала*
 - Э *вербальные средства коммуникации*
 - Э *коммуникативные характеристики*
 - ⇒ *определение*:*
[характеристики несущие информацию по средствам использованием системы кодирования человеческого языка]
 - ⇒ *примечание*:*
[лингвистические характеристики включают как фонологический код (сегментарный и надсегментный), так и грамматический код (морфологию и синтаксис). Лингвистическая коммуникация информирует получателя о намерениях отправителя с помощью явных словесных форм]
 - ▷ *паралингвистические характеристики сигнала*
 - Э *невербальные средства коммуникации*
 - Э *коммуникативные характеристики*
 - ⇒ *определение*:*
[характеристики несущие информацию по средствам дополнительных средств коммуникации не связанных непосредственно с языком]
 - ⇒ *примечание*:*
[передают информацию об отношении к предмету разговора, чувствах или эмоциональном состоянии говорящего]
 - ⇐ *объединение*:*
 - {• *интонация речи*
 - ⇐ *объединение*:*
 - {• *частота основного тона F_0*
 - *изменение частоты основного тона ΔF_0*
 - *громкость речи*
 - ⇐ *объединение*:*
 - {• *амплитуда сигнала*
 - *интенсивность сигнала*
 - *темп речи*
 - *длительность пауз*
- ▷ *эксталингвистические характеристики сигнала*
 - Э *информационные характеристики*
 - ⇒ *определение*:*
[характеристики, которые не кодируют непосредственно смысл сообщения, но содержат дополнительную информацию об отправителе и условиях коммуникации]
 - ⇒ *примечание*:*
[передают информацию об отношении к предмету разговора, чувствах или эмоциональном состоянии говорящего]
 - ⇐ *объединение*:*
 - {• *характеристики голоса диктора*
 - ⇐ *объединение*:*
 - {• *высота*
 - *тембр*
 - *громкость*
 - *акустическое окружение*

Таким образом, представлен результат формализации средствами языка SCg предметной области и онтологии типовых задач аудио и речевых интерфейсов для ИКС нового поколения.

Необходимо отметить, что в данной работе были представлены примеры формализации только некоторые из обозначенных выше понятий. Вторым важным замечанием является то, что представленный набор понятий ни в коем случае не является исчерпывающим, поскольку для построения. Более полные результаты работы по формализации

предметной области аудиоинтерфейса будут изложены в рамках более объемных работах, таких как монография и следующие версии стандарта OSTIS.

§ 4.3.4. Предметная область и онтология моделей параметрического представления сигнала

Все вышеперечисленные задачи взаимосвязаны, поскольку относятся к одному и тому же объекту исследования – речевому сигналу. Решение каждой из них непосредственно либо косвенно зависит от эффективности моделирования речи как сложного феномена в различных аспектах: параметрическое представление речевого сигнала и выделение его свойств, моделирование процесса фонации, восприятия и интерпретации содержания речевого сообщения (в том числе фонетического, смыслового, эмоционального). Это делает создание универсальных способов обработки речевых сигналов перспективным научным направлением. В контексте перечисленных задач моделирование речи можно условно разделить на три уровня:

- моделирование сигнала в общем виде, используя отсчеты во временной или частотной области;
- моделирование характеристик сигнала, являющихся специфическими для речи и связанных с процессом фонации (таких как частота основного тона, последовательность возбуждения и огибающая амплитудного спектра);
- моделирование высокоуровневых речевых характеристик (голос, акцент, экспрессия, фонетическое и семантическое содержание речевого сообщения). Каждый следующий уровень основывается на предыдущем и подразумевает использование специальных методов параметрического описания.

К первым двум уровням относятся широко известные в цифровой обработке речевых сигналов модели на основе линейного предсказания (ЛП), кепстральных коэффициентов и синусоидальных параметров.

Среди подходов, использующих синусоидальное описание сигнала, в настоящее время наиболее перспективными являются смешанные (гибридные) модели, учитывающие возможность разных режимов фонации с участием голосовых связок (вокализованная речь) и без участия голосовых связок (невокализованная речь), причем каждый из этих двух режимов описывается соответствующей моделью.

Вокализованная речь рассматривается как квазипериодический (детерминистский) сигнал, в то время как невокализованная – как непериодический (стохастический) сигнал. Наиболее известной среди существующих моделей является модель гармоники+шум, которая используется для решения таких сложных задач, как создание речевых интерфейсов, распознавание речи, синтез речи по тексту, конверсия голоса, шумоподавление, повышение разборчивости и субъективного качества речевых сигналов, коррекция акцента и так далее. Ее преимуществом является теоретическая возможность моделирования вокализованных звуков в виде непрерывных функций с изменяющимися параметрами, что позволяет получить эффективное описание процесса фонации и избежать наложения смежных фрагментов, разрыва фаз при синтезе речи. Недостатком модели является высокая сложность алгоритмов анализа и синтеза, обусловленная нестационарностью речевого сигнала.

Поскольку вокализованная речь состоит из квазипериодических компонент с изменяющимися параметрами, для анализа необходимо использовать цифровые фильтры с изменяющимися характеристиками: их полоса пропускания должна меняться в соответствии с контуром частоты основного тона. Это требует использования специальных частотно-временных преобразований, позволяющих производить оценку периодических составляющих с сильной частотной модуляцией, таких как Фан-Чирп и гармоническое преобразование. Точность оценки параметров непосредственно связана с точностью оценки контура основного тона, поэтому использование надежного и точного способа оценки является необходимым условием для успешного использования данной модели.

Также сложной задачей является автоматическое разделение сигнала на детерминистскую и стохастическую составляющие, для чего используются специальные детекторы периодичности.

Моделирование речевого сигнала на основе ЛП является классическим подходом, который применяется в цифровой обработке речи достаточно продолжительное время. Основным преимуществом модели является раздельное описание сигнала в виде огибающей спектра и сигнала возбуждения. Огибающая спектра определяет фонетику произносимого звука и характеризует состояние речевого тракта, в то время как сигнал возбуждения характеризует состояние голосовых связок и высоту (интонацию) вокализованных звуков. Преимуществом ЛП является также низкая вычислительная сложность.

Однако, несмотря на это, в последнее время предпочтение отдается моделям, использующим синусоидальное представление сигнала и в первую очередь это касается приложений, подразумевающих синтез речевого сигнала с измененными параметрами, таких как изменение интонации, конверсия голоса, синтез речи по тексту и других. Данный факт можно объяснить тем, что ЛП не обеспечивает эффективных способов для параметрической обработки сигнала возбуждения и непрерывного синтеза выходного сигнала. Каждый речевой фрагмент (кадр) сигнала представляет собой отдельную независимую единицу и при синтезе возникает проблема согласования соседних

кадров. Несогласованное изменение огибающей амплитудного и фазового спектра при переходе от кадра к кадру вызывает появление слышимых артефактов. Кроме того, оценка огибающей спектра при помощи классических методов ЛП представляет собой усреднение по всему кадру, вследствие чего ее точность ограничена. Порядок предсказателя определяет сложность модели: для предсказателей низких порядков оценка огибающей спектра получается чрезмерно сглаженной, а для предсказателей высоких порядков точность становится избирательной. Для точек спектра, соответствующих гармоникам основного тона, точность повышается, а для всех остальных точек – понижается. Оптимальный порядок предсказателя зависит от высоты голоса, но даже в наиболее благоприятном случае точность оценки огибающей спектра имеет погрешность, приводящую к возникновению слышимых искажений.

Использование кепстральных коэффициентов для моделирования речевых сигналов также является классическим подходом. Наиболее хорошо разработанной системой моделирования речевых сигналов, использующей кепстральные коэффициенты, является TANDEM-STRAIGHT. Так же как и для классических способов анализа на основе ЛП, при оценки кепстральных коэффициентов предполагается стационарность сигнала на протяжении интервала наблюдения. Оценка огибающей амплитудного спектра требует сглаживания и также недостаточно точна по сравнению с моделями на основе синусоидальных параметров.

Благодаря своим широким возможностям гибридная модель на основе синусоидальных параметров является наиболее предпочтительной для использования в большинстве практических случаев. Тем не менее для преодоления существующих ее ограничений, связанных со сложностью оценки параметров, их интерпретации в виде специфических речевых характеристик (параметры речевого тракта, последовательность возбуждения) требуется разработка специальных методов моделирования.

В зависимости от приложения процесс обработки речевого сигнала с использованием той или иной модели обычно включает анализ (определение параметров модели), модификацию (изменение параметров модели в зависимости от цели приложения) и синтез (формирование нового сигнала из измененных параметров модели). Таким образом, для обеспечения наиболее высокой практической значимости разрабатываемые методы моделирования должны включать средства анализа, обработки параметров и синтеза.

Для решения многих современных прикладных задач требуется не только наличие возможности описания речевого сигнала или процесса фонации, но и использование высокоуровневых речевых характеристик, определяющих персональный голос диктора, экспрессию, фонетику и т.д. К таким задачам относятся конверсия голоса, синтез речи по тексту, верификация диктора и многие другие. Высокоуровневое моделирование речи является очень сложной предметной областью, поскольку требует использования интеллектуальных моделей и методов машинного обучения. На данный момент не существует единого универсального способа, применяемого для разных приложений.

Подавляющее большинство высокоуровневых речевых моделей, используемых на практике, являются проблемно-ориентированными и могут применяться только для решения одной, узкоспециализированной задачи. Основным используемым математическим аппаратом являются статистические и вероятностные модели.

Чтобы подвести итог изложенным в разделе идеям приведем формальное представление некоторых из обозначенных выше концепций:

параметрическая модель сигнала

⇒ **определение***:

[математическое выражение, используемое для представления отсчетов сигнала во временной или частотной области]

▷ **параметрическая модель речевого сигнала**

⇒ **определение***:

[математическое описание характеристик сигнала, являющихся специфическими для речи и связанных с процессом фонации (таких как частота основного тона, последовательность возбуждения и огибающая амплитудного спектра)]

⇒ **примечание***:

[к основным моделям речевого сигнала относят: модели на основе линейного предсказания; на основе кепстрального представления; синусоидальные и гибридные модели. Среди гибридных моделей наиболее известна модель гармоники+шум]

Заключение к Главе 4.3. Аудиоинтерфейс ostis-систем

В главе изложены идеи лежащие в основе оригинального подхода к проектированию аудиоинтерфейсов ИКС на основе онтологического проектирования и формализации системы понятий из соответствующей предметной

области, с использованием технологии ОСТИС. Изложены основные принципы лежащие в основе данного подхода, а также их отличительные особенности от общепринятых.

К ограничениям предлагаемого подхода можно отнести следующие основные факторы. Очевидно, что для достижения поставленной цели и реализации задач по формализации любой предметной области, в том числе и аудиоинтерфейсов, требуется в первую очередь большое количество источников знаний для их пополнения. Для преодоления данной проблемы требуется привлечение большого количества экспертов, обладающих соответствующими компетенциями и знаниями в предметной, либо же разработка механизмов надежного автоматического извлечения этих знаний из имеющихся источников.

Прямой доступ к знаниям экспертов весьма ограничен, поскольку это требует значимых усилий по подбору репрезентативной выборки таких экспертов, выстраиванию эффективных и интероперабельных взаимоотношений между сторонами процесса, что зачастую зависит от большого количества субъективных факторов, – соответственно, требует большого количества временных и материальных ресурсов.

Известно, что существенное количество информации накопленного человечеством храниться в виде текстов на естественных языках. Процесс извлечения данной информации и представления её в формализованном виде – в виде знаний, также выглядит нетривиальным.

Исходя из природы данных проблем, по мнению авторов, видятся следующие основные направления их преодоления и, как следствие, две основные стратегии развития предложенного подхода.

1. Создание, для экспертов работающих в домене аудио и речевых интерфейсов, специализированных инструментальных средств по формализации и представлению знаний из данной предметной области, фиксации их в виде стандартов единой формы. Подобные инструменты должны обладать качественно новыми функциональными возможностями обеспечивающими высокий уровень совместимости и интероперабельности в процессе накопления и стандартизации знаний, чтобы сами эксперты были заинтересованы в применении и широком распространении данной технологии для представления знаний. Данный пункт является одной из ключевых задач технологии и стандарта OSTIS.
2. Создание автоматизированных и автоматических средств извлечения знаний из существующих источников информации, в первую очередь текстов на естественном языке. К видам документов в которых, содержаться уже структурированная и отчасти формализованная информация относятся в первую очередь: стандарты, протоколы, рекомендации (RFC), инструкции и т.д. Следовательно процесс автоматизации извлечения знаний должен быть направлен в первую очередь на формализацию уже существующих отраслевых стандартов разработки аудиоинтерфейсов, систем обработки и кодирования аудиоинформации, систем обработки речевых сигналов, таких как стандарты серии ISO, IEEE и AES (Audio Engineering Society).

Реализация подхода, предложенного в работе, позволит обеспечить свойства унификации, семантической совместимости и интероперабельности, при разработке аудио и речевых интерфейсов, (своеобразный аналог модели OSI/ISO в области проектирования интерфейсов ИКС), что в итоге позволит существенным образом сократить издержки при создании интеллектуальных компьютерных систем нового поколения для решения сложных комплексных задач.

Глава 4.4.

3D-модель внешней среды в ostis-системах

⇒ авторы*:

- Головатая Е.А.
- Головатый А.И.

⇒ аннотация*:

[Данная глава посвящена рассмотрению вопросов построения и использования трехмерного представления в различных задачах прикладных интеллектуальных систем, а также соответствующих систем позиционирования и ориентации в пространстве. Описание самого представления, а также принципов его построения осуществляется на основе базы знаний ostis-системы, что позволяет проводить глубокую интеграцию различных задач и методов, а также в последствии приводит к повышению степени конвергенции различных направлений.]

⇒ подраздел*:

- § 4.4.1. Прикладные задачи трехмерной реконструкции
- § 4.4.2. Анализ существующих подходов к трехмерной реконструкции
- § 4.4.3. Предлагаемый подход к трехмерной реконструкции с использованием базы знаний ostis-системы
- § 4.4.4. Семантическое представление объектов и сцены
- § 4.4.5. Трехмерное представление объектов в сцене
- § 4.4.6. Трехмерная реконструкция объектов окружающего мира
- § 4.4.7. Системы локального позиционирования, использующиеся в задачах трехмерной реконструкции
- § 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции
- § 4.4.9. Онтология действий, выполняемых в предметной области трехмерной реконструкции

⇒ ключевое понятие*:

- трехмерное представление
- трехмерная реконструкция
- семантическое описание сцены

⇒ библиографическая ссылка*:

- Luhmann T.. CloseRPa3DI-2018bk

§ 4.4.1. Прикладные задачи трехмерной реконструкции

Для взаимодействия интеллектуальной компьютерной системы с объектами внешней среды в прикладных задачах необходимо создание внутреннего представления этих объектов. Одним из видов такого внутреннего представления может являться описание объектов в виде трехмерной модели. При этом формирование такого внутреннего представления относится к классу задач анализа сенсорной информации и требует определения точного расположения объекта, на основании которого прикладные системы могут реализовывать различные сценарии взаимодействия. В соответствии с этим особую важность приобретают системы ориентации в пространстве и трехмерной реконструкции, способные формировать и обрабатывать такие представления.

К прикладным областям, в которых требуется решение подобных задач, можно отнести *Luhmann T.. CloseRPa3DI-2018bk*:

- Интеллектуальные робототехнические системы. Примеры задач: анализ окружения, построение траекторий движения, восстановление трехмерных координат объектов.
- Интеллектуальные системы управления производством. Примеры задач: анализ деформаций объектов и структурных изменений, бесконтактное измерение размеров объектов произвольного масштаба и конфигурации, контроль производственного процесса.
- Интеллектуальные системы комплексного медицинского мониторинга и обслуживания. Примеры задач: анализ результатов обследований, отслеживание динамики развития заболеваний, планирование лечения.
- Научно-исследовательская деятельность.
- Другие прикладные области (архитектура, картография и т.д.).

Предметная область трехмерного представления объектов затрагивает, как само описание объекта, так и методов его получения. На основании данных представлений могут быть решены следующие классы задач:

- построение трехмерного представления объекта, группы объектов или окружения,
- определение размеров объекта, включающее и определение отклонений от заданного шаблона или параметров, например, в системах медицинской диагностики,
- проведение дополнительных построений, доопределяющих созданное сформированное трехмерное представление,
- построение траектории движения,
- и т. д.

Несмотря на то, что часть указанных выше задач требует дополнительных действий, все они основываются на получении трехмерного представления объекта.

Таким образом, декларативной формулировкой задачи трехмерной реконструкции является получение внутреннего представления объекта, относящегося к классу трехмерных представлений.

§ 4.4.2. Анализ существующих подходов к трехмерной реконструкции

На данный момент существует большое количество методов реконструкции, оперирующих разными представлениями входных данных и информации: фотограмметрическое восстановление по фото/видеосъемке, радиочастотные методы в разных диапазонах, магнитные и инерциальные методы, нейросетевой анализ и т.д. Например, искусственные нейронные сети, решающие задачу трехмерной реконструкции, могут принимать в качестве входного значения отдельные изображения, наборы изображений, панорамные и стереоскопические изображения, комбинацию изображений и данных с различных видов датчиков, наборы ключевых точек, воксельное облако. Для каждого метода может быть определен набор характеристик внешней среды (помещение, освещение, наличие движения) и входного представления (размер, тип поверхности), в пределах которых данный метод является корректным и демонстрирует наилучшие результаты работы по одному из целевых критериев. Полученное внутреннее представление также может отличаться: часть методов позволяет восстановить внутреннюю структуру, другие - только поверхность (внешнюю форму) объекта.

Кроме этого в задачах анализа сенсорной информации, как правило, есть возможность установить несколько разных типов датчиков, но они должны быть, во-первых, пригодны для исследования данного типа объекта, а, во-вторых, полученная информация должна дополнять друг друга (увеличивать детализацию модели, разрешающую способность, точность и т.д.), то есть система должна иметь возможность адаптироваться под конкретную задачу и внешние условия.

Все эти факторы накладывают серьезные ограничения на возможность применения различных методов трехмерной реконструкции при решении конкретной прикладной задачи, которые должны быть учтены при проектировании системы.

К проблемам существующих решений можно отнести:

- Отсутствие согласованности систем понятий и описаний методов в различных источниках. Встречаются различные описания одного и того же метода, существует много модификаций, постоянно возникают сложности и, как минимум, недопонимания в связи с этим.
- Отсутствие привязки и недостаточное внимание вопросам конвергенции предметной области трехмерной реконструкции с предметной областью формирования трехмерных сцен и окружений интерактивного пользовательского взаимодействия с трехмерной средой, например, в системах трехмерного моделирования, виртуальной и дополненной реальности.
- Высокая сложность разработки прикладных систем, использующих методы трехмерной реконструкции, и необходимость привлечения опытных высококвалифицированных разработчиков к решению соответствующих задач.
- Отсутствие комплексной технологии проектирования. Несмотря на обилие алгоритмов и методов, анализ их применимости к различным видам прикладных задач крайне поверхностный. Вследствие этого в большинстве случаев лучший метод выбирается перебором или эмпирически.
- Отсутствие средств интеграции отдельных компонентов, этапов различных методов, различного типа данных при описании объекта и полученных внутренних представлений. Кроме вариативности отдельных действий, в результате каждый метод работает со своим классом внутреннего представления (поверхность, заданная полигонально, воксельное облако с регулярной сеткой, набор отдельных координат в трехмерном пространстве и т.д.).

Таким образом, систематизация знаний данной предметной области, а также создание технологии разработки и автоматизации процесса проектирования интеллектуальных систем в данной области являются актуальными и нерешенным на данный момент задачами.

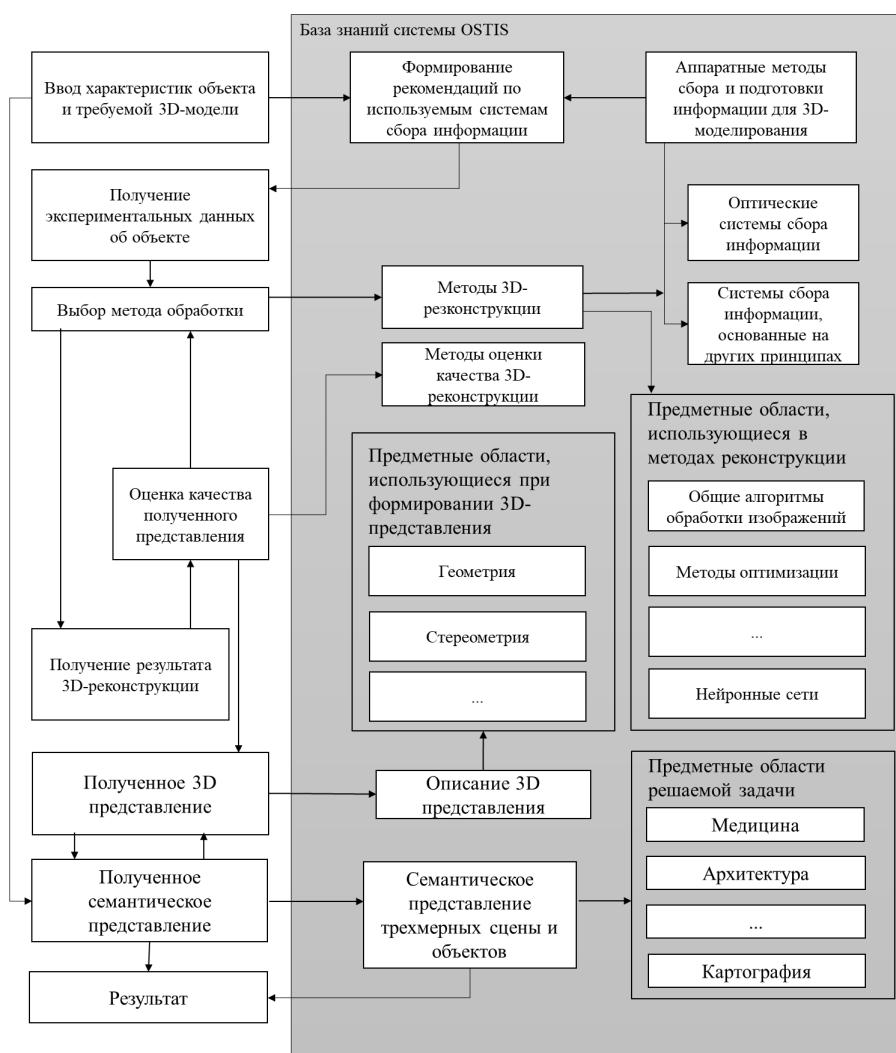
§ 4.4.3. Предлагаемый подход к трехмерной реконструкции с использованием базы знаний ostis-системы

Описание самого представления, а также принципов его построения осуществляется в виде базы знаний ostis-системы. В рамках формирования базы знаний и платформы для разработки интеллектуальных систем в данной предметной области выделены следующие этапы:

- выделение семантического представления трехмерных сцен;
- систематизация предметной области, существующих подходов и установление связей со смежными областями;
- разработка набора агентов, определяющих подходящие методы и средства для конкретных прикладных задач;
- разработка набора агентов, проводящих агрегацию разных методов, с целью уточнения или проверки параметров трехмерного представления (положения) объекта.

Последовательность основных этапов процесса трехмерной реконструкции и их связь с базой знаний OSTIS представлена на рисунке *Описание основных этапов процесса трехмерной реконструкции и их связи с базой знаний*. В рамках базы знаний выделяются следующие блоки:

- описание характеристик наблюдаемых объектов;
- описание типов и принципов задания трехмерных представлений;
- описание физических принципов работы и спецификаций оборудования, с помощью которого может быть собрана информация об исследуемом объекте;
- набор различных методов реконструкции и локализации с ограничениями и решателями конкретных задач;
- методы оценки результатов полученных 3D-представлений;
- описание семантического представления трехмерных сцен и объектов.



= Описание основных этапов процесса трехмерной реконструкции и их связи с базой знаний

Далее более подробнее рассмотрим основные указанные предметные области и онтологии.

§ 4.4.4. Семантическое представление объектов и сцены

Важной составляющей интеллектуальных систем, использующих внутреннее трехмерное представление, является описание семантики этого представления. Информация об индивидуальных точках, поверхностях, полигонах или других примитивах не позволяет сформировать комплексное представление о смысловом наполнении этого представления – точно так же, как отдельные буквы не позволяют оценить семантическую составляющую текстовых сообщений.

Семантическое описание объекта подразумевает ассоциацию множества точек, соответствующего некоторому объекту в трехмерном пространстве, с некоторым объектом имеющейся базы знаний. Отношения в таких ассоциациях могут быть представлены в виде «объект А представляет собой трехмерный образ некоторой сущности В», где объект А задаётся внутренним трехмерным представлением, а сущность В – некоторым описанием в базе знаний.

Некоторые свойства сущности В, ассоциируемой с объектом А, могут быть естественным образом выведены из трехмерного представления этого объекта – в частности, информация о форме, свойствах поверхности, окраске и текстурировании могут уже присутствовать в трехмерном описании. Таким образом, с помощью дополнительной обработки трехмерное представление объекта А может являться источником фактологической информации о связанной сущности В.

Аналогично семантическому описанию отдельного объекта может быть введено семантическое описание составных объектов. Следует иметь в виду, что семантическое наполнение индивидуальных составляющих не позволяет в полной мере определить семантику всего составного объекта целиком, поэтому аналогичную ассоциацию также необходимо задать и для всей совокупности. Например, объект «велосипед» может быть декомпозирован на составные объекты «цепь», «рама», «колесо» и т.д., однако набор семантических описаний трехмерных представлений составляющих по отдельности не позволяет формировать или учитывать семантику всего составного объекта целиком.

Семантическое описание части объекта должно содержать как информацию о базовом объекте, так и о дополнительном контексте относительно смыслового наполнения рассматриваемой части. Например, при видеоэндоскопическом анализе участка пищевода важной становится также информация о том, к какой именно части пищевода в организме относится этот участок.

Сцена представляет собой сложную композицию нескольких простых или составных объектов в некотором общем пространстве, дополненную данными об их взаимном расположении. Как и в случае с составными объектами, семантическое описание сцен должно включать не только описания индивидуальных составляющих, но и также смысловое наполнение эмержентных свойств всех этих составляющих в совокупности.

Таким образом, выделены следующие виды основных сущностей в рамках семантического представления трехмерных сцен и объектов:

объект

:= [множество точек в пространстве, соединенных между собой и имеющих одно смысловое представление]

составной объект

:= [объект, подразумевающий декомпозицию на отдельные индивидуальные объекты]

часть объекта

:= [множество точек в пространстве, принадлежащих некоторому объекту, которое может быть выделено по геометрическому или смысловому представлению]

сцена

:= [совокупность нескольких объектов и данных об их взаимном расположении в пространстве]

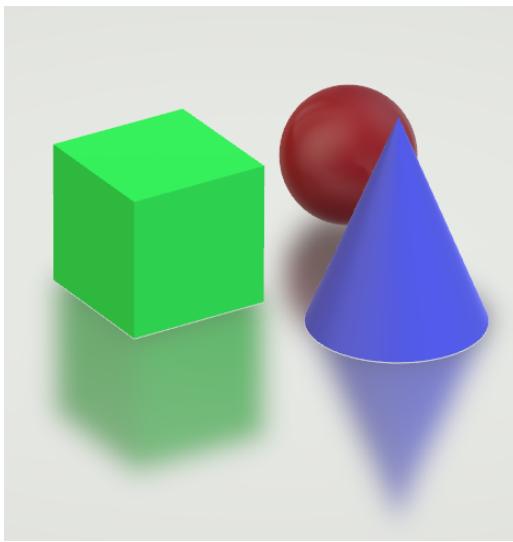
Для сцены важными являются семантические характеристики визуального восприятия сцены с позиции некоторого помещенного в неё наблюдателя или системы машинного зрения. В этом контексте сцена может быть представлена в виде двумерной проекции (или пары двумерных проекций в случае стереоскопического зрения), при этом семантика описаний исходных трехмерных объектов и соответствующих им участков полученных проекций также может отличаться – например, некоторые объекты могут находиться вне поля видимости, или восприниматься наблюдателем по-другому из-за наличия некоторых оптических, перспективных или психофизиологических эф-

фектов (например, разница в освещении, оптическая дисторсия, различные иллюзии восприятия цвета или глубины, например, комната Эймса, и т.п.).

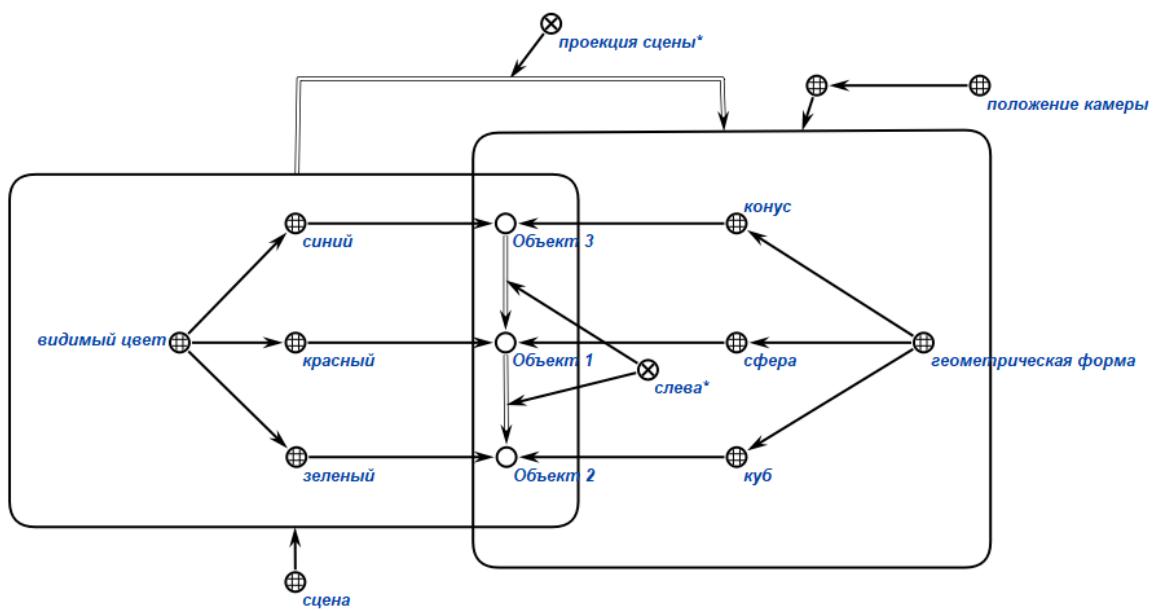
Важным является тот факт, что семантическое описание сцены должно включать как определение принадлежности индивидуальных объектов сцены некоторым сущностям имеющейся базы знаний, так и описание возможных взаимосвязей между этими объектами, возникающих в силу их присутствия в сцене, или в силу их попарного взаимного расположения с позиции некоторого наблюдателя. Эта информация может естественным образом использоваться в качестве референтных свойств объектов сцены. Например, при наличии в сцене двух одинаковых объектов типа А и одного объекта типа В, некоторое логическое высказывание или запрос на естественном языке может использовать факт их взаимного расположения, например, для идентификации – «тот объект типа А, который находится слева от объекта типа Б».

- Информация о присутствии на сцене или проекции сцены
 - Объект отсутствует на сцене
 - Объект присутствует, но не виден на проекции
 - Объект присутствует и на проекции виден частично
 - Объект присутствует и на проекции виден полностью
- Информация о взаимном расположении на сцене
 - По высоте
 - Выше другого объекта
 - На одном уровне с другим объектом
 - Ниже другого объекта
 - Информация о взаимном расположении на проекции сцены
 - По глубине
 - За другим объектом
 - Перед другим объектом
 - По высоте
 - Выше другого объекта
 - На одном уровне с другим объектом
 - Ниже другого объекта
 - По расположению вдоль линии горизонта
 - Слева от другого объекта
 - Справа от другого объекта
 - Информация об относительном размере объектов на проекции
 - Больше другого объекта
 - Одного размера с другим объектом
 - Меньше другого объекта
 - Информация о визуальном сходстве объектов
 - Похож или не похож на другой объект по форме
 - Похож или не похож на другой объект по цвету
 - Похож или не похож на другой объект по размеру

Пример простой трехмерной сцены, а также её семантического описания с точки зрения взаимного расположения объектов, представлен на рисунках [Пример простой трехмерной сцены, визуализированной в виде двумерной проекции с определенного положения камеры](#) и [Семантическое описание некоторых связей взаимного расположения объектов в сцене с положения наблюдателя](#).



= Пример простой трехмерной сцены, визуализированной в виде двумерной проекции с определенного положения камеры



= Семантическое описание некоторых связей взаимного расположения объектов в сцене с положения наблюдателя

§ 4.4.5. Трехмерное представление объектов в сцене

Системы позиционирования, распознавания и отображения объектов реального мира опираются не только на качественную составляющую описания, но и взаимное расположение объектов в пространстве или их отдельных частей. В соответствии с восприятием окружающего мира человеком трехмерное представление является наиболее информативным, хотя и не обязательным. Полученные двумерные изображения, используемые во многих задачах, являются проекциями трехмерного пространства. Поэтому для рассматриваемых в данной главе задач максимальным классом объектов исследования выбрано трехмерное представление объектов, включающее класс описаний, содержащих информацию о взаимном расположении объектов или их частей в трех координатах.

Трехмерное представление

- ⇒ разбиение*:
 {• поверхностное представление
 ⇒ включает*:
 {• полигональные сетки
 • NURBS-поверхности
 • поверхности разделения
 • поверхности на основе T-сплайнов
 }
 • воксельное представление
 ⇒ включает*:
 {• облако точек
 • карта глубины
 • структурная сетка
 }
 • специальные представления
 ⇒ включает*:
 {• видео 360
 }
 }

точка

:= [местоположение в пространстве, однозначно определяемое в системе координат]

координатная плоскость

:= [плоскость, на которой каждой точке Р сопоставлена некоторая уникальная пара чисел (x, y), называемая координатами]

эпиполярная геометрия

:= [один из подразделов оптики, который занимается геометрической интерпретацией формирования двумерных изображений по трехмерной сцене. Чаще всего в этом контексте рассматривается задача геометрических преобразований лучей света, которые отражаются от наблюдаемой трехмерной сцены, улавливаются некоторой оптической системой или объективом и проецируются на фоточувствительную двумерную поверхность (например, плёнку или цифровую матрицу) для формирования изображения]

§ 4.4.6. Трехмерная реконструкция объектов окружающего мира

Получить трехмерное представление на основе экспериментальных данных позволяет трехмерная реконструкция. Предметная область трехмерного представления объектов затрагивает, как само описание объекта, так и методы его получения.

трехмерная реконструкция

:= [задача определения истинной формы объектов в трехмерном пространстве на основании информации об этих объектах, получаемой в результате измерений, наблюдений или опытов]

Несмотря на то, что часть указанных выше задач требует дополнительных действий, все они основываются на получении трехмерного представления объекта. В соответствии с этим каждая из задач также дополнительно может быть определена как внутренним состоянием, так и описанием условий, в рамках которых она осуществляется:

- описание цели – тип трехмерного представления, его разрешающая способность и точность,
- условия по возможности осуществления действий – например, имеющееся оборудование или допустимое время выполнения,
- вид входных данных – описание типа входного объекта.

Трехмерное представление объекта или окружения, независимо от конкретного подкласса, может быть получено разными методами, в то же время конкретный метод может определять только одно представление.

методы трехмерной реконструкции

- ⇒ включает*:
 {• методы фотограмметрической реконструкции

- ⇒ подразделяется по взаимному положению объекта и камеры на*:
 - {• мобильные системы
 - макрофотограмметрия
 - спутниковая фотограмметрия
 - аэрофотограмметрия
 - наземная фотограмметрия
 - близкая фотограмметрия
- ⇒ подразделяется по виду входной информации на*:
 - {• одно изображение
 - стерео изображения
 - многокадровые
- методы томографической реконструкции
- ⇒ включает*:
 - {• реконструкция на основе Фурье-проекций
 - алгоритм обратной проекции
 - алгоритм итерационной реконструкции
- ⇒ включает*:
 - {• ART
 - SART
 - SAMV
- реконструкция по коническому лучу
- реконструкция на основе методов глубокого обучения
- }
- методы структурированного подсвета
- ⇒ включает*:
 - {• методы на основе световых сечений
 - методы на основе проекций полос
 - методы на основе фазового сдвига
- методы по оценке отраженного сигнала
- ⇒ включает*:
 - {• измерение расстояния методами оптической модуляции
 - импульсная модуляция
- методы оценки по фокусировке
- ⇒ включает*:
 - {• методы оценки из фокусировки
 - методы оценки из расфокусировки
- методы на основе теоремы о Фурье-проекциях
- нейросетевые модели

}

Выше представлен один из вариантов онтологического описания методов создания трехмерных представлений. Каждый из методов трехмерной реконструкции в рамках этого описания может быть определен также кроме физического принципа его разрешающей способностью, типом входных данных, размером реконструируемых объектов и т.д. Полное описание представляет собой неиерархическую онтологическую модель. Для дальнейшего взаимодействия агентов с данной структурой все эти описания ставятся в соответствие некоторым характеристикам. Характеристики могут относится как к отдельному методу, так и к группе методов, например, разрешающая способность может быть общей для всего подкласса электромагнитных волновых методов. Данные характеристики должны получаться агентами из самого представления знаний динамически, что позволяет дополнять и модифицировать общую структуру. Для удобства представления данные характеристики можно выделить в спецификацию метода, на основании которой описывается область и возможность его применения. Она дает возможность использования методов для решения конкретных прикладных задач. В рамках спецификации (и, соответственно, структуры представления методов в базе знаний) задаются:

- тип возможных входных параметров;
- тип выходного представления – в данном случае соответствующее трехмерное представление;
- время работы;
- разрешающая способность метода;

- расстояние от объекта до камеры;
- тип реконструируемого объекта;
- композиция сцены (отдельный объект, группа объектов, окружающее пространство);
- тип поверхности (глянцевость, прозрачность, цветность);
- наличие внутренней структуры;
- размер объекта.

Описанная спецификация может быть интерпретируема промежуточным отношением для каждого метода. При наличии такого описания для промежуточных этапов данный подход позволяет проводить также комбинацию методов. Например, методы радиочастотного диапазона не дают возможность построить карту глубины, но позволяют получить положение камеры в глобальной системе координат в конкретный момент времени.

§ 4.4.7. Системы локального позиционирования, использующиеся в задачах трехмерной реконструкции

В основе многих систем трехмерной реконструкции лежит решение задачи локального позиционирования. Следует отметить, что в целом задача локального позиционирования является более общей, однако обычно рассматривается относительно наблюдателя, а не объекта.

система локального позиционирования

:= [автоматизированная система, обеспечивающая идентификацию, определение координат, отображение на плане местонахождения контролируемых объектов или их частей в пределах территории, охваченной необходимой инфраструктурой]

Характеристики систем позиционирования:

- точность позиционирования,
- достоверность позиционирования,
- периодичность опроса,
- надежность,
- габаритность.

методы позиционирования

⇒ включает*:

{• ***триангуляционные методы***

:= [методы, основанные на определении направления на источник сигнала]

- ***угол прихода сигнала***
:= [angle of arrival]
:= [AoA]
:= [направление, из которого принимается сигнал (например, радио, оптический или акустический)]
- ***угол вылета***
:= [angle of departure]
:= [AoD]
:= [направление, в котором отправляется сигнал (например, радио, оптический или акустический)]

• ***трилатерационные методы***

:= [методы, основанные на трилатерации, то есть на основе построения на местности смежных треугольников]

- ***время прибытия***
:= [time of arrival]
:= [ToA]
:= [абсолютный момент времени, когда радиосигнал, исходящий от передатчика, достигает удаленного приемника]
- ***разница во времени прибытия***
:= [time difference of arrival]
:= [TDoA]
:= [разница между временем прибытия сигнала до двух базовых станций]

- ***время полета***

:= [time of flight]
:= [ToF]
:=

- [время, затрачиваемое объектом, частицей или волной (акустической, электромагнитной и т. д.) на преодоление расстояния через среду]
- *двустороннее определение дальности*
 - := [two-way ranging]
 - := [TWR]
 - := [метод определения дальности, который использует две задержки, которые обычно возникают при передаче сигнала, для определения дальности между двумя станциями: задержка распространения сигнала между двумя беспроводными устройствами, задержка обработки подтверждений в беспроводном устройстве]
 - *симметричное двустороннее определение дальности*
 - := [symmetrical double-sided two-way ranging]
 - := [SDS TWR]
 - := [метод определения дальности, который использует двустороннее определение дальности дважды: относительно базовой станции и относительно мобильного устройства]
 - *методы на основе измерения силы сигнала*
 - := [received signal strength indicator]
 - := [RSSI]
 - := [индикатор уровня мощности принимаемого сигнала. Данный метод позволяет определить местоположение устройства, основываясь на уровне силы сигнала, полученного БС или наоборот]
 - *одометрия*
 - := [использование данных о движении приводов для оценки перемещения]
 - *визуальная одометрия*
- }

физические принципы позиционирования

⇒ включает*:

- {● *акустические*
 - := [основаны на использовании ультразвуковых (высокочастотных) звуковых волн]
- *радиочастотные*
- *магнитные*
 - := [магнитный трекинг основан на измерении интенсивности магнитного поля в различных направлениях. Как правило, в таких системах есть базовая станция, которая генерирует переменное или постоянное магнитное поле]
- *оптические*
 - := [совокупность методов позиционирования на основе данных с камер видимого или инфракрасного диапазона]
 - *с внешней камерой*
 - *с внутренней камерой*
 - *визуальная одометрия*
 - := [метод оценки положения и ориентации робота или иного устройства с помощью анализа последовательности изображений, снятых установленной на нем камерой (или камерами)]
 - *лазерное позиционирование*
 - := [группа методов, основанных на оценке времени прохождения лазерных импульсов определенной периодичности]
 - *инерциальные*
 - := [инерциальное позиционирование основано на свойствах инерции тел. основная особенность этих методов состоит в том, что они не требуют внешних ориентиров или поступающих извне сигналов]
 - *гироскоп*
 - *акселерометр*
 - *магнитометр*
 - *барометр*
 - *гибридные*

}

§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции

Пункт 4.4.8.1. Оптические системы компьютерного зрения

Трехмерную модель рассматриваемого объекта по нескольким снимкам или видеоряду можно получить на основе построения проекций, связанных с каждым из снимков, и выстраиванием этих проекций в одном пространстве. При наличии пересекающихся лучей, направленных в одну и ту же точку реального объекта, по проекциям можно восстановить параметры объекта и сцены в связанной с этим пространством системе координат.

проекция

- := [отображение точек, фигур, векторов пространства любой размерности на его подпространство любой размерности]
- ⇒ включает*:
 - {• ортогональная
 - центральная

Для описания преобразования между трехмерной моделью и проекцией используется модель проекции. Для полно- го задания модели вводятся параметры, называемые параметрами ориентирования, однозначно характеризующие положение снимка относительно некоторой глобальной системы координат. Выделяют параметры внутреннего и внешнего ориентирования. Параметры внутреннего ориентирования задают относительное положение точки фотографирования и самого снимка. В модели центральной проекции к параметрам внутреннего ориентирования относятся 3 величины: двумерные координаты главной точки снимка (x_0, y_0) и фокусное расстояние f . К параметрам внешнего ориентирования относят 6 величин, задающих связь проецирующих лучей в момент съемки с глобальной системой координат: координаты точки положения камеры (x_C, y_C, z_C) , два угла, определяющие положение главного луча камеры (ω, φ) и угол поворота снимка β . Для перехода в новое пространство необходимо осуществить пространственное преобразование между системами координат. Для этого используется преобразование Гельмерта, зависящее от 7 параметров: координаты начала системы координат снимка x_{uz} в глобальной координатной системе OXYZ (X_0, Y_0, Z_0), три угла поворота (ω, φ, β) относительно осей X, Y, Z и коэффициент масштаба t . На основании данных углов можно построить матрицу поворота, задающую преобразование координат из одной системы в другую при последовательном повороте вокруг каждой из осей.

связь координат объекта на изображении и в трехмерном пространстве

- ⇒ основана на*:
 - {• параметры внутреннего ориентирования
- ⇒ включает*:
 - {• фокусное расстояние
 - координаты главной точки снимка (x_0, y_0)
- параметры внешнего ориентирования
- ⇒ включает*:
 - {• координаты точки положения камеры (x_C, y_C, z_C)
 - углы, определяющие положение главного луча камеры (ω, φ)
 - угол поворота снимка β

бинокулярное зрение

- := [способность одновременно чётко видеть изображение предмета обоими глазами; в этом случае человек видит одно изображение предмета, на который смотрит]

стереоскопическое зрение

- := [вид зрения, при котором возможно восприятие формы, размеров и расстояния до предмета, благодаря восприятию двух изображений одновременно]

диспаранность

- := [различие взаимного положения двух точек, отображаемых на двух изображениях и соответствующих одной точке пространства]

калибровка камеры

:= [задача получения внутренних и внешних параметров камеры по имеющимся фотографиям или видео, отснятыми ею]

Калибровка камеры часто используется на начальном этапе решения многих задач компьютерного зрения и в особенности дополненной реальности. Кроме того, калибровка камеры помогает исправлять дисторсию на фотографиях и видео.

дисторсия

:= [аберрация оптических систем, при которой коэффициент линейного увеличения изменяется по мере удаления отображаемых предметов от оптической оси. При этом нарушается геометрическое подобие между объектом и его изображением.]

⇒ включает*:

- {• радиальная
- тангенциальная

}

Пункт 4.4.8.2. Локальные признаки изображений

Одним из первых этапов трехмерной реконструкции по изображениям, полученным с различных ракурсов, является нахождение всевозможных комбинаций соответствующих точек на изображениях, которые являются одной и той же точкой исходной сцены. Традиционно для решения этой задачи используются алгоритмы для работы с локальными признаками изображения.

локальный признак

:= [некоторое подмножество точек изображения, пространственные окрестности которых определённым образом характеризуют изображение целиком или некоторые из присутствующих на нём объектов]

⇒ включает*:

- {• ключевая точка

}

Основной сложностью при реализации и работе с алгоритмами нахождения локальных признаков является тот факт, что универсального или однозначно точного определения того, что именно является локальным признаком, не существует. Конкретное определение локального признака в произвольной точке произвольного изображения зависит от вида используемого алгоритма и решаемой задачи. Алгоритмы детектирования локальных признаков на изображении, как правило, являются входной точкой для последующей обработки другими методами.

методы построения карты диспарантности

⇒ включает*:

- {• *Sum of Absolute Differences (SAD)*
- *Sum of Squared Differences (SSD)*
- *Normalized Cross Correlation (NCC)*
- *нейросетевые алгоритмы для вычисления карты диспарантности*
- *детекторы и дескрипторы ключевых точек*
 - *детекторы*
 - :=** [алгоритмы, которые по входному изображению генерируют набор ключевых точек]
 - *эмпирические детекторы*
 - *Fast*
 - *детектор Харриса*
 - *Susan*
 - *статистические детекторы*
 - *SIFT*
 - *AGAST*
 - *детекторы на основе машинного обучения*
 - *FAST-ER*
 - *TILDE*
 - *дескрипторы*
 - :=** [алгоритмы, которые по входному изображению и набору координат точек на них, генерируют вектор признаков, описывающий эти точки в некотором метрическом пространстве]

- гистограммные дескрипторы
 - SIFT
 - SURF
 - бинарные дескрипторы
 - BRIEF
 - ORB
 - BRISK
 - методы оптического потока
 - := [методы отображения видимого движения объектов, поверхностей или краев сцены, получаемое в результате перемещения наблюдателя (глаз или камеры) относительно сцены]
 - фазовая корреляция
 - := [инверсия нормализованного перекрестного спектра]
 - блочные методы
 - := [минимизация суммы квадратов или суммы модулей разностей]
 - дифференциальные методы оценки оптического потока, основанные на частных производных сигнала
 - алгоритм Лукаса — Канаде
 - := [рассматриваются части изображения и аффинная модель движения]
 - Horn-Schunck
 - := [минимизация функционала, описывающего отклонение от предположения о постоянстве яркости и гладкость получаемого векторного поля]
 - Buxton-Buxton
 - := [основан на модели движения границ объектов в последовательности изображений]
 - общие вариационные методы
 - := [модификации метода Horn-Schunck, использующие другие ограничения на данные и другие ограничения на гладкость]
 - дискретные методы оптимизации
 - := [поисковое пространство квантуется, затем каждому пикселю изображения ставится в соответствие метка таким образом, чтобы расстояние между последовательными кадрами было минимальным]
- }

§ 4.4.9. Онтология действий, выполняемых в предметной области трехмерной реконструкции

На верхнем уровне любой метод трехмерной реконструкции может быть представлен в виде процедурного знания, например, в виде последовательности этапов, в соответствии с которыми входные данные для трехмерной реконструкции преобразовываются в итоговое трехмерное представление.

Общая черта многих методов трехмерной реконструкции – использование промежуточного (или конечного) представления об объектах окружающего мира в трех координатах. Другими словами, метод трехмерной реконструкции можно представить в виде последовательности действий по формированию набора элементов, характеризующихся координатами в общем трехмерном пространстве, которые в дальнейшем, при необходимости, могут быть достроены до поверхностей, скомбинированы с двумерными представлениями и т.д. для формирования нужного выходного представления:

$$r : Im(R^3) \rightarrow O \quad (4.4.1)$$

где I – входные данные, R^3 – общее трехмерное декартово пространство, $m(R^3)$ – описание элемента в системе координат общего трехмерного пространства, O – выходное трехмерное представление. Чаще всего в качестве элементов промежуточного представления выступают отдельные точки трехмерного пространства – в этом случае такое представление называют облаком точек.

В качестве элементов могут также выступать отрезки кривых, аналитически задаваемые поверхности, полигоны и другие виды объектов.

В качестве источников данных о трехмерных координатах для промежуточного представления могут выступать:

- Непосредственно абсолютные значения трехмерных координат точек, т.е. в этом случае входные данные I представляют собой набор точек R^3 . Это характерно для всех методов, которые позволяют строить карту

глубины сцены, т.к. представление в виде карты глубины с известными координатами положения наблюдателя и фокусного расстояния карты позволяет определить трехмерные координаты любой точки на ней.

- Данные, по которым с помощью некоторой дополнительной обработки могут быть получены значения трехмерных координат отдельных точек. Такие представления, как правило, намного более распространены и просты в получении.

Следует отметить, что различные источники данных при формировании промежуточного представления могут использоваться совместно, при наличии у каждого из источников некоторой привязки к общей системе координат.

Пункт 4.4.9.1. Действия по формированию промежуточного трехмерного представления

Многие методы дистанционного зондирования полагаются на наличие априорной информации о трехмерных координатах исследуемого объекта, по которой можно построить облако точек для промежуточного трехмерного представления. К таким относятся методы, которые позволяют оценивать расстояние от измерительного прибора до объекта. Тем не менее, использование этих методов требует более сложного оборудования, применение которого может быть затруднено в некоторых сценариях.

В этой связи, в качестве отдельной категории методов исследования можно выделить группу методов формирования промежуточного представления в виде облака точек по более традиционным видам информации. К таким относятся методы генерации по одному изображению, стереопаре, набору изображений или по видеоряду, в которых также может использоваться информация об оптической системе камеры, которая была использована для получения изображения.

Таким образом можно выделить следующие виды входных данных:

- Статическое изображение или набор статических изображений;
- Видеоряд (набор статических изображений с временной меткой);
- Стереопара (2 статических изображения с параметрами оптической системы) или набор стереопар;
- Стерео-видеоряд (набор стереопар с временной меткой).
- Информация об оптической системе камеры.

В качестве выходного представления в данном случае выступает разреженное облако точек трехмерного пространства, с привязкой каждой из точек этого пространства к одной или нескольким точкам исходных входных изображений.

Формирование разреженного облака точек включает в себя следующие этапы, для каждого из которых могут быть определены указанные параметры:

- Предобработка
- Детектирование ключевых точек
 - алгоритм-детектор
- Сопоставление ключевых точек
 - алгоритм-дескриптор или алгоритм оптического потока
 - прореживание
- Оценка положений камеры
 - модель проекции
 - алгоритм оценки связок
- Постобработка

Детектирование и сопоставление ключевых точек позволяет определить точки, принадлежащие одному и тому же объекту исследуемого трехмерного пространства, при наличии достаточного количества изображений. На этапе оценки положения камеры используется математическая модель проекции, задающая взаимосвязь между двумерными координатами точки на изображении и соответствующими ей трехмерными координатами в пространстве моделирования; на этом же этапе может осуществляться эмпирическая оценка глубины, например, при помощи нейросетевых методов. Каждое соответствие, описанное математически с помощью модели проекции, в дальнейшем используется в алгоритме оценки связок для того, чтобы восстановить в трехмерном пространстве положения камеры для каждого из исследуемых изображений, а также определить расстояния от камер до точек, исходя из некоторого критерия минимизации ошибки обратной проекции.

Например, классический метод трехмерной реконструкции Structure from Motion по нескольким входным изображениям в рамках представленного конвейера можно описать следующими характеристиками:

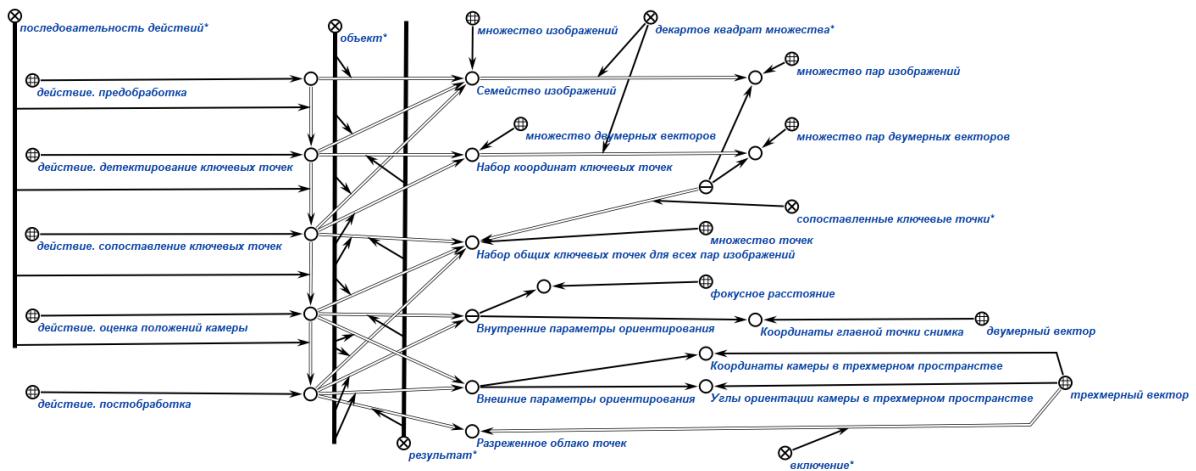
- Предобработка – преобразование к оттенкам серого;
- Алгоритм-детектор – SIFT, SURF, FAST;
- Алгоритм-дескриптор – SIFT, SURF, ORB;

- Прореживание – RANSAC;
- Модель проекции – центральная проекция;
- Алгоритм оценки связок – глобальный метод связок с оптимизацией методом Левенберга-Марквардта.

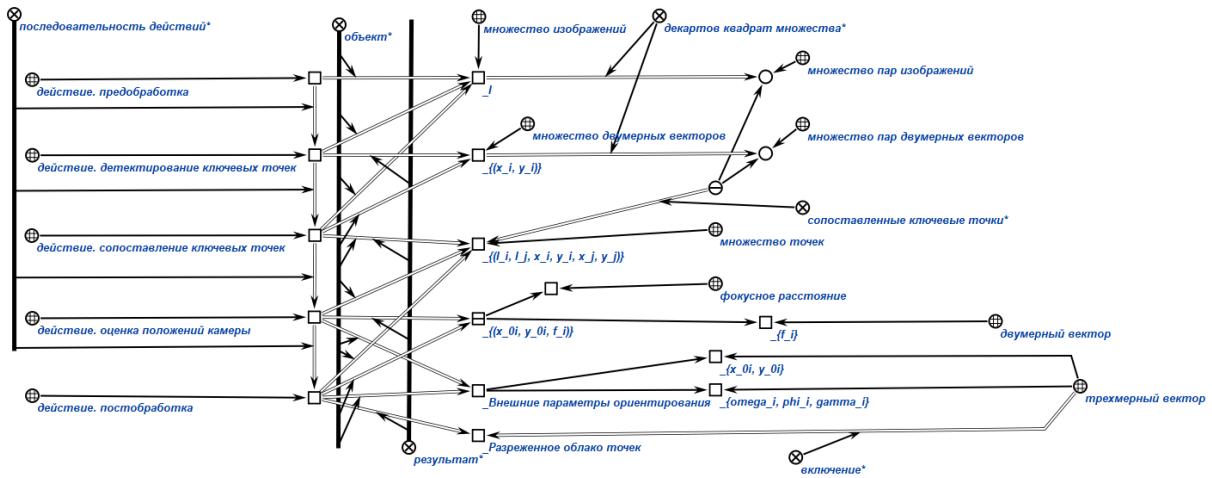
В качестве ещё одного примера рассмотрим метод одновременной локализации и построения карты ORB-SLAM, использующий в качестве входного представления видеоряд:

- Предобработка – прореживание кадров;
- Алгоритм-детектор – FAST;
- Алгоритм-дескриптор – ORB + метод оптического потока Лукаса-Канаде;
- Прореживание – RANSAC, прореживание по инвариантам движения;
- Модель проекции – центральная проекция;
- Алгоритм оценки связок – инкрементальный метод связок с фиксированным положением камеры и фиксированным положением ключевых точек между кадрами, метод построения карты по движению;
- Постобработка – уточнение траектории и детектирование петель.

Онтологическое описание представленной последовательности действий, а также пример конкретного алгоритма, реализованного по этой последовательности действий, представлены на рисунках *Онтологическое описание последовательности действий по построению промежуточного трехмерного представления в виде разреженного облака точек* и *Пример алгоритма построения разреженного облака точек на основе представленного описания*.



= *Онтологическое описание последовательности действий по построению промежуточного трехмерного представления в виде разреженного облака точек*



= Пример алгоритма построения разреженного облака точек на основе представленного описания

Поскольку каждый из предложенных этапов описан в виде функционального отображения, предполагается добавление, удаление или модификацию этапов при обработке, если соблюдается соответствие типов входных и выходных представлений в контексте решаемой задачи.

Пункт 4.4.9.2. Действия по формированию итогового трехмерного представления

Как уже упоминалось, в некоторых задачах разреженное облако точек в трехмерном пространстве может выступать достаточным представлением, и может считаться итогом работы алгоритма трехмерной реконструкции. Также достаточно популярным является представление в виде уплотненного цветного облака точек.

Тем не менее, во многих задачах такого представления недостаточно, поэтому можно выделить класс действий при формировании более сложного трехмерного представления, в зависимости от типа требуемого выходного представления. В качестве входного представления для этого этапа выступает разреженное облако точек, а также дополнительная информация о связи конкретных точек облака с исходными представлениями.

Описание методов формирования итогового трехмерного представления можно представить в виде совокупностей следующих этапов, с соответствующими параметрами:

- Уплотнение облака точек
 - алгоритм уплотнения
 - связь с исходными данными
- Формирование поверхностей
 - тип поверхности
 - алгоритм формирования поверхностей
- Уточнение поверхностей
 - алгоритм уточнения поверхностей
 - связь с исходными данными
- Текстурирование поверхностей
 - метод текстурирования
 - связь с исходными данными
 - разрешение конфликтов

На этапе уплотнения облака точек информация о связи трехмерных координат точек разреженного облака с исходными данными используется для переноса дополнительных точек напрямую из исходного представления в трехмерную модель. Далее путём анализа полученного плотного облака точек и исходных данных формируется грубая оценка итоговой трехмерной поверхности в виде некоторого трехмерного примитива, как правило, путём формирования полигональной сетки объединением ближайших точек в треугольники. На этапе уточнения поверхностей может происходить сглаживание, прореживание и объединение примитивов, полученных на предыдущем этапе, на основании некоторой информации из исходных данных. Наконец, на этапе текстурирования исходное

представление переносится в построенную трехмерную модель, чтобы обеспечить её реалистичность; также на этом этапе происходит разрешение конфликтов для выбора корректной стратегии текстурирования в условиях наличия нескольких равноправных источников информации о текстуре.

Например, алгоритм реконструкции поверхностей, предложенный в рамках наиболее популярной на сегодняшний день реализации метода связок можно описать в виде следующего набора параметров:

- алгоритм уплотнения – перенос окрестностей ключевых точек
- тип поверхностей – полигональные с прямоугольными полигонами
- алгоритм формирования поверхностей – оценка переносов камеры с помощью триангуляции Делоне, оценка планарных переносов камеры, интерполяция между положениями камеры, ручная подстройка
- алгоритм уточнения поверхностей – отсутствует
- метод текстурирования – прямой перенос с исходных изображений
- метод разрешения конфликтов текстурирования – альфа-смешение пропорционально расстоянию до точки.

Пункт 4.4.9.3. Действия по подбору и генерации алгоритма

Подробное описание структуры методов не является достаточным для непосредственного выполнения трехмерной реконструкции по некоторому алгоритму. Поскольку для каждого из этапов итогового конвейера существует большое множество реализаций, возникает также задача оптимального выбора из набора реализаций каждого из этапов для наиболее эффективного решения поставленной задачи.

Как для выбора этапов алгоритма трехмерной реконструкции, так и для его непосредственной реализации может использоваться агентно-ориентированный подход к обработке информации. Коммуникация агентов осуществляется на основе обращения к общему представлению в базе знаний, по которой генерируется ряд вопросов, специфицирующих дальнейшие действия.

В качестве основных вопросов, на основании которых может осуществляться генерация алгоритма, можно выделить следующие:

- Какие входные данные будут использоваться для реконструкции?
 - Можно ли по входным данным определить расстояние до точки в трехмерном пространстве, или непосредственно её трехмерные координаты?
 - В качестве данных будут использованы изображения?
 - Известны ли оптические параметры камеры?
 - Известны ли положения или ориентации камеры в пространстве для каждого изображения?
 - Является ли набор изображений непрерывным видеорядом с известной частотой кадров?
 - При наличии нескольких источников входных данных, имеется ли информация о привязке описываемых ими данных к общей системе координат?
- Какой тип трехмерной модели должен быть сформирован по этим данным?
 - Требуется ли реконструкция трехмерных поверхностей?
 - Могут ли исходные данные использоваться для формирования текстур поверхностей?

На основе рассмотренного подхода может осуществляться как подбор метода, удовлетворяющего требованиям задачи, так и подбор отдельных этапов алгоритма, например, выбор наиболее оптимального алгоритма детектирования и описания ключевых точек. Кроме этого, может осуществляться комбинация 3D-представлений, полученных разными методами.

Заключение к Главе 4.4.

В главе рассмотрено онтологическое описание предметной области трехмерной реконструкции объектов и действий по их обработке в базах знаний на основе технологии OSTIS. Описание представлено в виде доменной области самих представлений, методов их получения и соответствующих действий по их обработке.

Преимуществами использования технологии OSTIS для рассмотренных задач являются:

- Введение общей системы понятий и описаний методов в унифицированном и согласованном виде.
- Возможность конвергенции предметной области трехмерной реконструкции с предметной областью формирования трехмерных сцен и окружений и смежными областями.
- Упрощение разработки прикладных систем, использующих методы трехмерной реконструкции.
- Возможность построения комплексной технологии проектирования с использованием интеллектуальных агентов, потребляющих предложенное описание.

- Возможность создания средств интеграции отдельных компонентов, этапов различных методов и полученных внутренних представлений.

С помощью предложенного подхода становится возможным создавать интеллектуальные системы, которые могут получать и оперировать трехмерным представлением для дальнейшей обработки в прикладных задачах.

Часть 5.

Методы и средства проектирования интеллектуальных компьютерных систем нового поколения

Описание к главе

Глава 5.1.

Комплексная библиотека многократно используемых семантически совместимых компонентов *ostis*-систем

Орлов М.К.
Шункевич Д.В.
Ковалёв М.В.
Садовский М.Е.
Загорский А.Г.
Банцевич К.А.

⇒ *автор**:
• Орлов М.К

⇒ *тезис**:
[Всё, что можно сделать одинаково, нужно делать одинаково.]

⇒ *аннотация**:
[Важнейшим этапом эволюции любой технологии является переход к компонентному проектированию на основе постоянно пополняемой библиотеки многократно используемых компонентов. Идея библиотеки компонентов не нова, но семантическая мощность *Библиотеки OSTIS* значительно выше аналогов за счет того, что подавляющее большинство компонентов библиотеки – компоненты базы знаний, представленные на унифицированном языке смыслового представления знаний (*SC-коде*). Таким образом, в Библиотеке OSTIS обеспечивается высокий уровень семантической совместимости компонентов, что приводит к высокому уровню семантической совместимости *ostis*-систем, использующих комплексную библиотеку многократно используемых семантически совместимых компонентов *ostis*-систем.]

⇒ *библиографическая ссылка**:
• Шункевич.Д.В..*Средства ПКПСУЗ-2015*ст
• Iyengar A..*CompoDfRD-2021*art
• Ford B..*CompoD-2019*art
• Wilkes M.V..*PrepaoPfaEDCWSRttEatUoaLoS-1951*bk
• Уилкс М..*Состав ПдЭСМ-1953*кн
• Волченкова Н.И..*ТехноМРиЖБПВМнЯФ-1984*ст
• Blahser J..*ThineAfaFTDPMBoHP-2021*art
• Fritzson P..*ModelLO-2014*art
• Prakash S.P..*MicroPA-2022*art
• Memduhoglu M..*PossiCoSSMaTtMPSDP-2018*art
• Москаленко Ф.М..*ТехноСРЗИСdОБИнОРРОИ-2016*ст

⇒ *подраздел**:
• 5.1. Введение в Главу 5.1.
• § 5.1.1. Анализ современных библиотек многократно используемых компонентов
• § 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов *ostis*-систем
• § 5.1.3. Менеджер многократно используемых компонентов *ostis*-систем
• § 5.2.7. Многократно используемые компоненты баз знаний *ostis*-систем
• § 5.1.4. Многократно используемые встраиваемые *ostis*-системы

Введение в Главу 5.1.

⇒ *ключевое понятие**:
• компонентное проектирование интеллектуальных систем

⇒ *ключевое знание*:*

- *Проблемы в реализации компонентного проектирования интеллектуальных систем*

Повторное использование готовых компонентов широко применяется во многих отраслях, связанных с проектированием различного рода систем, поскольку позволяет уменьшить трудоемкость разработки и ее стоимость (путем минимизации количества труда за счет отсутствия необходимости разрабатывать какой-либо компонент), повысить качество создаваемого контента и снизить профессиональные требования к разработчикам компьютерных систем. Таким образом, осуществляется переход от программирования компонентов или целых систем к их дизайну на основе готовых компонентов. Использование готовых компонентов предполагает, что распространяемый компонент верифицирован и документирован, а возможные ошибки и ограничения устраниены либо специфицированы и известны.

Рассмотрим проблемы в реализации компонентного подхода к проектированию интеллектуальных систем: *Шункевич.Д.В..Средст
2015cm*

компонентное проектирование интеллектуальных систем

⇒ *проблемы текущего состояния*:*

Проблемы в реализации компонентного проектирования интеллектуальных систем

- = {• [Отсутствие средств унификации любых видов знаний и моделей представления знаний.]
 - [Отсутствие формальных средств структуризации, позволяющих представить иерархию компонентов на различных уровнях детализации.]
 - [Несовместимость компонентов, разработанных в рамках разных проектов, вследствие отсутствия унификации в принципах представления различных видов знаний в рамках одной базы знаний, и, как следствие, отсутствие унификации в принципах выделения и спецификации многократно используемых компонентов.]
 - [Невозможность автоматической интеграции компонентов в систему без ручного вмешательства пользователя.]
 - [Не проводится тестирование, верификация и анализ качества компонентов, не выделяются преимущества, недостатки, ограничения компонентов.]
 - [Не ведётся разработка стандартов, обеспечивающих совместимость этих компонентов.]
 - [Многие компоненты используют для идентификации языка разработчика (как правило, английский), и предполагается, что все пользователи будут использовать этот же язык. Однако для многих приложений это недопустимо – понятные только разработчику идентификаторы должны быть скрыты от конечных пользователей, которые должны быть в состоянии выбрать язык для идентификаторов, которые они видят.]
 - [Отсутствие средств поиска компонентов, удовлетворяющих заданных критериям.]
- }

Для того, чтобы решить возникшие проблемы, при проектировании интеллектуальных систем и библиотек их многократно используемых компонентов необходимо выполнить следующие требования:

компонентное проектирование интеллектуальных систем

⇒ *предъявляемые требования*:*

- {• [Использование универсального языка представления знаний, используемого в интеллектуальной системе.]
 - [Наличие универсальной процедуры интеграции знаний в рамках указанного языка.]
 - [Наличие стандарта, обеспечивающего семантическую совместимость интегрируемых знаний (таким стандартом является согласованная система используемых понятий и унифицированная спецификация компонентов).]
- }

Большинство существующих систем создано как автономные программные продукты, которые не могут быть использованы в качестве компонентов других систем. Необходимо использовать либо целую систему, либо ничего. Небольшое число систем поддерживает компонентно-ориентированную архитектуру способную интегрироваться с другими системами *Iyengar A.CompoDfRD-2021art Ford B..CompoD-2019art*. Однако, их интеграция возможна при условии использования одинаковых технологий и только при проектировании одной командой разработчиков.

Многократная повторная разработка уже имеющихся технических решений обусловлена либо тем, что известные технические решения плохо интегрируются в разрабатываемую систему, либо тем, что эти технические решения трудно найти. Данная проблема актуальна как в целом в сфере разработки компьютерных систем, так и в сфере разработки систем, основанных на знаниях, поскольку в системах такого рода степень согласованности различных видов знаний влияет на возможность системы решать нетривиальные задачи.

На данный момент не существует комплексной библиотеки многократно используемых семантически совместимых компонентов компьютерных систем в целом, не говоря об интеллектуальных. Существуют некоторые попытки создания библиотек типовых методов и программ для традиционных компьютерных систем, однако такие библиотеки не решают вышеперечисленные проблемы.

Термин “библиотека подпрограмм”, одними из первых упомянули Уилкс М., Уиллер Д. и Гилл С. в качестве одной из форм организации вычислений на компьютере (*Wilkes M.V.PrepaوPfaEDCWSRtiEatUoaLoS-1951bk Уилкс М..СоставПdЭСМ-1953kn*). Исходя из изложенного в их книге, под библиотекой понимался набор «коротких, заранее заготовленных программ для отдельных, часто встречающихся (стандартных) вычислительных операций» (*Волченкова Н.И.ТехноМРиЖБПВМиЯФ-1984ст*). Стоит отметить, что компонентами библиотек являются не только программы, но и компоненты интерфейса и базы знаний.

К традиционным решениям относятся **пакетные менеджеры** языков программирования и операционных систем, а также отдельные системы и платформы с встроенными компонентами и средствами для сохранения создаваемых компонентов.

Компоненты библиотеки могут быть реализованы на разных языках программирования (что приводит к тому, что для каждого языка программирования разрабатываются свои библиотеки со своими решениями различных часто встречающихся ситуаций), а также могут располагаться в разных местах, что приводит к тому, что в библиотеке необходимо средство для поиска компонентов и их установки.

§ 5.1.1. Анализ современных библиотек многократно используемых компонентов

⇒ *ключевое понятие**:

- пакетный менеджер

Современные пакетные менеджеры, такие как npm, pip, poetry, maven, apt, pacman и другие имеют преимущество в том, что они способны разрешать конфликты при установке зависимых компонентов, однако они не учитывают семантику компонентов, а только лишь устанавливают компоненты по идентификатору (*Blahser J.ThineAfaFTDPMBoHP-2021art*). Библиотеки таких компонентов являются только лишь хранилищем компонентов, никак не учитывающим назначение компонентов, их преимущества и недостатки, сферы применения, иерархию компонентов и другую информацию, необходимую для интеллектуализации компонентного проектирования компьютерных систем. Поиск компонентов в библиотеках компонентов, соответствующих данных пакетным менеджерам сводится к поиску по идентификатору компонента. Также существенным недостатком современного подхода является платформенная зависимость компонентов. Современные библиотеки компонентов ориентированы только на какой-то определённый язык программирования, операционную систему или платформу.

Пакетный менеджер pip является системой управления пакетами, которая используется для установки пакетов из Python Package Index, который является некоторой библиотекой таких пакетов. Зачастую pip устанавливается вместе с Python. Пакетный менеджер pip используется только для языка программирования Python. Он имеет множество функций для работы с пакетами:

- установка пакета;
- установка пакета специализированной версии;
- удаление пакета;
- переустановка пакета;
- отображение установленных пакетов;
- поиск пакетов;
- верификация зависимостей пакетов;
- создание файла конфигурации со списком установленных пакетов и их версий;
- установка множества пакетов из файла конфигурации;

```

1 py==1.8.1
2 pip==19.0.3
3 Mako==1.1.1
4 MarkupSafe==1.1.1
5 six==1.14.0
6 attrs==19.3.0
7 pytest==5.3.5
8 pluggy==0.13.1
9 setuptools==40.8.0
10 parse==1.14.0
11 glob2==0.7

```

= Файл конфигурации pip

Пакетный менеджер pip хорошо работает с зависимостями, отображает безуспешно установленные пакеты, а также отображает информацию о требуемой версии пакета при конфликте с другим пакетом.

Альтернативой пакетного менеджера pip является пакетный менеджер poetry, который также ориентирован на язык программирования Python. Преимущество poetry перед pip в том, что он автоматически работает с виртуальными окружениями, способен самостоятельно их находить и создавать. Файл конфигурации для пакетов poetry является более богатым, чем у pip, он хранит такие сведения, как имя проекта, версия проекта, его описание, лицензия, список авторов, URL проекта, его документации и сайта, список ключевых слов проекта и список PyPI классификаторов. Poetry позволяет более гибко настраивать пакеты для проектов Python, файл конфигурации poetry представляет собой более богатую спецификацию проекта, однако эта спецификация не позволяет достичь совместимости между компонентами даже в рамках Python проектов и предназначена преимущественно только для чтения разработчиком. Автоматизировать проектирование компьютерных систем с помощью пакетного менеджера poetry или pip невозможно, так как требуется вмешательство разработчика, который должен вручную совместить интерфейсы устанавливаемых пакетов. Другие пакетные менеджеры языков программирования и операционных систем устроены по такому же принципу: существует хранилище компонентов (библиотека), которая представляет собой множество пакетов этого языка программирования или операционной системы и с которым взаимодействует менеджер компонентов.

```

[tool.poetry]
name = "first"
version = "0.1.0"
description = ""
authors = [...]
readme = "README.md"

[tool.poetry.dependencies]
python = "^3.10"

[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"

```

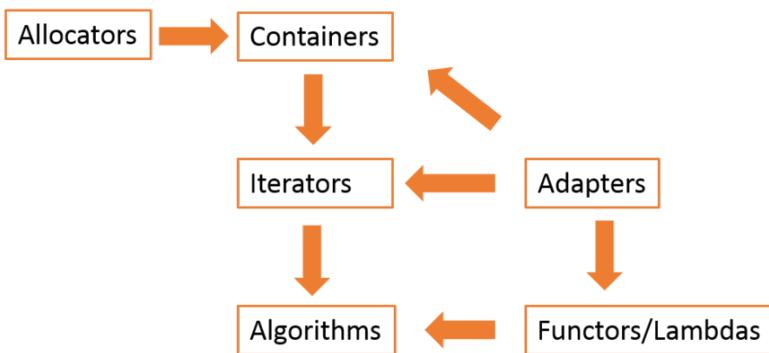
= Файл конфигурации poetry

В качестве компонентного подхода к проектированию программ можно рассмотреть библиотеки подпрограмм современных языков программирования, например, библиотеку стандартных шаблонов C++.

Библиотека стандартных шаблонов – набор согласованных обобщённых алгоритмов, контейнеров, средств доступа к их содержимому и различных вспомогательных функций в C++.

В библиотеке выделяют пять основных компонентов:

- контейнер – хранение набора объектов в памяти;
- итератор – обеспечение средств доступа к содержимому контейнера;
- алгоритм – определение вычислительной процедуры;
- адаптер – адаптация компонентов для обеспечения различного интерфейса;
- функциональный объект — сокрытие функции в объекте для использования другими компонентами.



= Структура библиотеки STL

Разделение позволяет уменьшить количество компонентов. Например, вместо написания отдельной функции поиска элемента для каждого типа контейнера обеспечивается единственная версия, которая работает с каждым из них, пока соблюдаются основные требования.

Совместимость компонентов (контейнеров) в библиотеке стандартных шаблонов обеспечивается общим интерфейсом использования этих компонентов.

Компонентный подход к проектированию компьютерных систем может реализовываться в рамках различных языков, платформ и приложений. Рассмотрим некоторые из них.

На основе языка **Modelica** разработано большое число свободно-доступных библиотек компонентов, одной из которых является библиотека **Modelica_StateGraph2**, включающая компоненты для моделирования дискретных событий, реактивных и гибридных систем с помощью иерархических диаграмм состояния *Fritzson P. ModelLO-2014art*. Основным недостатком систем на базе языка Modelica является отсутствие совместимости компонентов и достаточной документации, а также узкая направленность разрабатываемых компонентов.

Microsoft Power Apps — это набор приложений, служб и соединителей, а также платформа данных, которая предоставляет среду разработки для эффективного создания пользовательских приложений для бизнеса. Платформа Power Apps имеет средства для создания библиотеки многократно используемых компонентов графического интерфейса, а также предварительно созданные модели распознавания текста (чтение визитных карточек или чеков) и средство обнаружение объектов, которые можно подключить к разрабатываемому приложению *Prakash S.P.MicroPA-2022art*. Библиотека компонентов Power Apps представляет собой множество создаваемых пользователем компонентов, которые можно использовать в любых приложениях. Преимущество библиотеки в том, что компоненты могут настраивать свойства по умолчанию, которые можно гибко редактировать в любых приложениях, использующих компоненты. Недостаток в том, что отсутствует семантическая совместимость компонентов, спецификация компонентов, не решена проблема существования семантически эквивалентных компонентов, нету иерархии компонентов и средств поиска этих компонентов. Компоненты платформы Microsoft Power Apps являются многократно используемыми только для однотипных приложений, которые создаются одним и тем же разработчиком.

WebProtege представляет собой многопользовательский веб-интерфейс, позволяющий редактировать и хранить онтологии в формате OWL в совместной среде *Memduhoglu M..PossiCoSSMaTtMPSDP-2018art*. Данный проект позволяет не только создавать новые онтологии, но также загружать уже существующие онтологии, которые хранятся на сервере университета Стэнфорда. К преимуществу данного проекта можно отнести автоматическую проверку ошибок в процессе создания объектов онтологий. Данный проект является примером попытки решения проблемы накопления, систематизации и повторного использования уже существующих решений, однако, недостатком данного решения является обособленность разрабатываемых онтологий. Каждый разработанный компонент имеет свою иерархию понятий, подход к выделению классов и сущностей, которые зависят от разработчиков данных онтологий, так как в рамках данного подхода не существует универсальной модели представления знаний, а также

формальной спецификации компонентов, представленных в виде онтологий. Следовательно, возникает проблема их семантической несовместимости, что, в свою очередь, приводит к невозможности повторного использования разработанных онтологий при проектировании баз знаний. Данный факт подтверждается наличием на сервере университета Стэнфорда многообразия различных онтологий на одни и те же темы.

Платформа IACPaaS (Intelligent Applications, Control and Platform as a Service) *Москаленко Ф.М.. ТехноПРЗИСдОБИОРРОИ-2016см* – облачная платформа для разработки, управления и удаленного использования интеллектуальных облачных сервисов. Она предназначена для обеспечения поддержки разработки, управления и удаленного использования прикладных и инструментальных мультиагентных облачных сервисов (прежде всего интеллектуальных) и их компонентов для различных предметных областей. Платформа предоставляет доступ:

- прикладным пользователям (специалистам в различных предметных областях) (в качестве реализации модели SaaS) - к прикладным сервисам;
- разработчикам прикладных и инструментальных сервисов и их компонентов (в качестве реализации модели PaaS) - к инструментальным сервисам;
- управляющим интеллектуальными сервисами (в качестве реализации модели Caas);
- к сервисам управления.

Платформа IACPaaS поддерживает:

- базовую технологию разработки прикладных и специализированных инструментальных (интеллектуальных) сервисов с использованием базовых инструментальных сервисов платформы, поддерживающих эту технологию;
- множество специализированных технологий разработки прикладных и специализированных инструментальных (интеллектуальных) сервисов, с использованием специализированных инструментальных сервисов платформы, поддерживающих эти технологии.

Платформа IACPaaS также не имеет средств для унифицированного представления компонентов интеллектуальных компьютерных систем и средств для их спецификации и автоматической интеграции.

Исходя из проведённого анализа можно сказать, что на текущем состоянии развития информационных технологий не существует комплексной библиотеки многократно используемых семантически совместимых компонентов компьютерных систем. Таким образом, предлагается комплексная библиотека многократно используемых семантически совместимых компонентов *ostis*-систем.

§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов *ostis*-систем

⇒ *ключевое понятие**:

- *многократно используемый компонент ostis-систем*
- *материнская ostis-система*
- *библиотека многократно используемых компонентов ostis-систем*

⇒ *ключевое знание**:

- *архитектура Экосистемы OSTIS*
- *классификация многократно используемых компонентов ostis-систем*
- *спецификация многократно используемых компонентов ostis-систем*

Основным требованием, предъявляемым к *Технологии OSTIS*, является обеспечение возможности совместного использования в рамках *ostis*-систем различных видов знаний и различных моделей решения задач с возможностью неограниченного расширения перечня используемых в *ostis*-системе видов знаний и моделей решения задач без существенных трудозатрат, а также различных видов интерфейсов. Следствием данного требования является необходимость реализации компонентного подхода на всех уровнях, от простых компонентов баз знаний, решателей задач и интерфейсов до целых *ostis*-систем. Разработчики любой ostis-системы могут включить в ее состав библиотеку, которая позволит им накапливать и распространять результаты своей деятельности среди других участников Экосистемы *OSTIS* в виде *многократно используемых компонентов*. Решение о включении компонента в библиотеку принимается экспертным сообществом разработчиков, ответственным за качество этой библиотеки. Верификацию компонентов можно автоматизировать. В рамках Экосистемы *OSTIS* существует множество библиотек многократно используемых компонентов *ostis*-систем, являющихся подсистемами соответствующих материнских *ostis*-систем. Материнская *ostis*-система отвечает за какой-то класс компонентов и является САПРом для этого класса, содержит методики разработки таких компонентов, их классификацию, подробные пояснения ко всем подклассам компонентов. Таким образом, формируется иерархия *материнских ostis-систем*. Материн-

ская ostis-система в свою очередь может являться дочерней ostis-системой для какой-либо другой ostis-системы, заимствуя компоненты из библиотеки, входящей в состав этой другой ostis-системы.

ostis-система

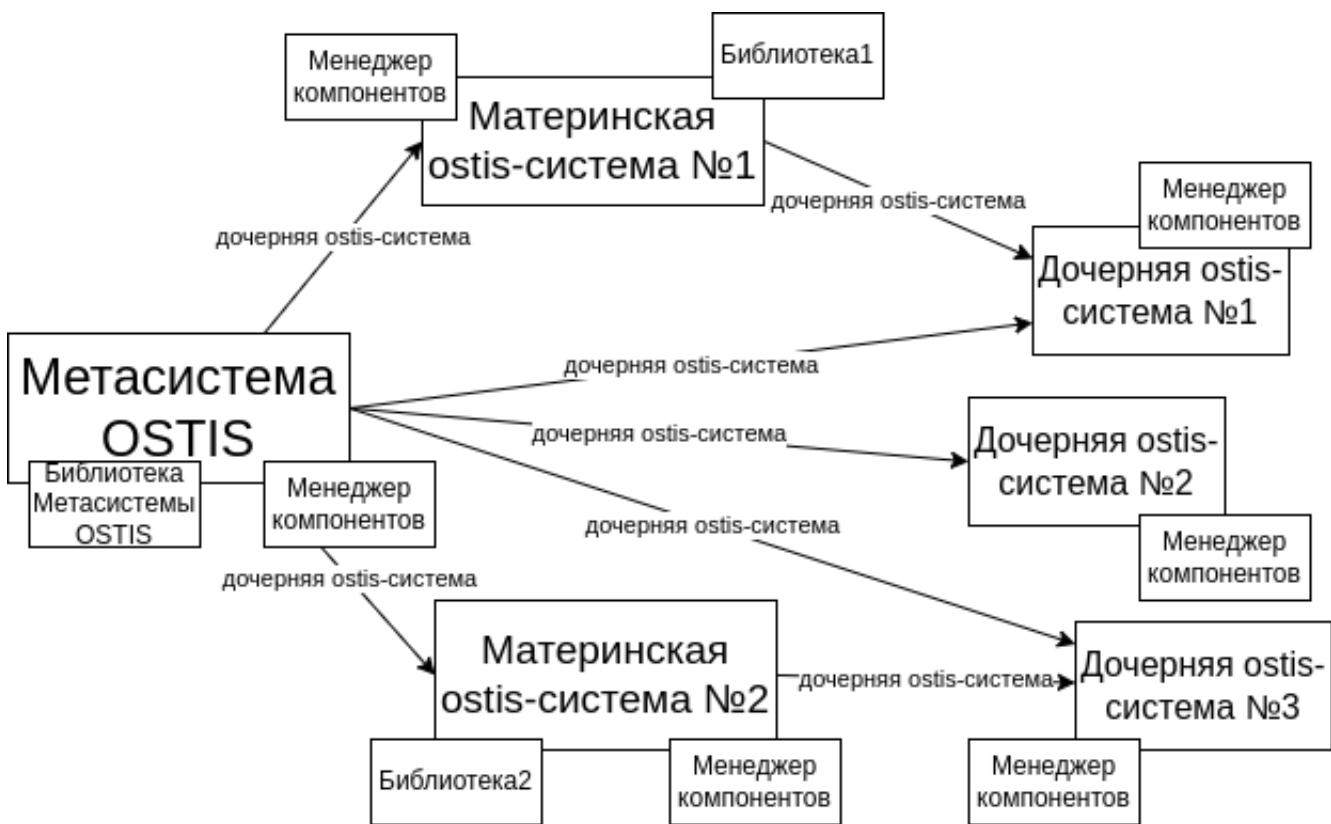
- ▷ *материнская ostis-система*
 - := [ostis-система, имеющая в своем составе библиотеку многократно используемых компонентов.]
 - ≡ *Метасистема OSTIS*
- ▷ *дочерняя ostis-система*
 - := [ostis-система, в составе которой имеется компонент, заимствованный из какой-либо библиотеки многократно используемых компонентов.]

Библиотека многократно используемых компонентов ostis-систем позволяет использовать проектный опыт по разработке и модернизации ostis-систем различного назначения.

библиотека многократно используемых компонентов ostis-систем

- ⇒ *сокращение**:
 - [библиотека ostis-систем]
- := [библиотека многократно используемых компонентов OSTIS]
- := [комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем]
- ≡ *типичный пример*':
 - Библиотека OSTIS**
 - := [Библиотека многократно используемых компонентов ostis-систем в составе Метасистемы OSTIS]
 - := [Библиотека Метасистемы OSTIS]
- ⇐ *объединение**:
 - {• *библиотека типовых подсистем ostis-систем*
 - *библиотека шаблонов типовых компонентов ostis-систем*
 - *библиотека ostis-платформ*
 - *библиотека многократно используемых компонентов баз знаний*
 - *библиотека многократно используемых компонентов решателей задач*
 - *библиотека многократно используемых компонентов пользовательских интерфейсов*
- }

Главной библиотекой многократно используемых компонентов ostis-систем является *Библиотека Метасистемы OSTIS*. *Метасистема OSTIS* выступает *материнской системой* для всех разрабатываемых ostis-систем, поскольку содержит все базовые компоненты (рисунок [Архитектура Экосистемы OSTIS](#)).



= Архитектура Экосистемы OSTIS

Основу для реализации компонентного подхода в рамках *Технологии OSTIS* составляет **Библиотека OSTIS**. *Метасистема OSTIS* ориентирована на разработку и практическое внедрение методов и средств компонентного проектирования семантически совместимых интеллектуальных систем, которая предоставляет возможность быстрого создания интеллектуальных систем различного назначения. В состав Метасистемы OSTIS входит **Библиотека OSTIS**. Сфера практического применения технологии компонентного проектирования семантически совместимых интеллектуальных систем ничем не ограничены.

Основное назначение Библиотеки OSTIS – создание условий для эффективного, осмысленного и массового проектирования ostis-систем и их компонентов путём создания среды для накопления и совместного использования компонентов ostis-систем. Такие условия осуществляются путём неограниченного расширения постоянно эволюционируемых семантически совместимых ostis-систем и их компонентов, входящих в Экосистему OSTIS.

Постоянно расширяемая Библиотека OSTIS существенно сокращает сроки разработки новых интеллектуальных компьютерных систем. Библиотека многократно используемых компонентов ostis-систем позволяет избавиться от дублирования семантически эквивалентных информационных компонентов. А также от многообразия форм технической реализации используемых моделей решения задач.

Функциональные возможности библиотеки многократно используемых компонентов ostis-систем:

библиотека многократно используемых компонентов ostis-систем

⇒ **функциональные возможности***:

- [Хранение многократно используемых компонентов ostis-систем и их спецификаций. При этом часть компонентов, специфицированных в рамках библиотеки, могут физически храниться в другом месте ввиду особенностей их технической реализации (например, исходные тексты ostis-платформы могут физически храниться в каком-либо отдельном репозитории, но специфицированы как компонент будут в соответствующей библиотеке). В этом случае спецификация компонента в рамках библиотеки должна также включать описание (1) того где располагается компонент и (2) сценария его автоматической установки в дочернюю ostis-систему.]
- [Просмотр имеющихся компонентов и их спецификаций, а также поиск компонентов по фрагментам их спецификации.]
- [Хранение сведений о том, в каких ostis-системах-потребителях какие из компонентов библиотеки и какой версии используются (были скачаны). Это необходимо как минимум для учета востребованности того или иного компонента, оценки его важности и популярности.]
- [Систематизация многократно используемых компонентов ostis-систем.]

- [Обеспечение версионирования многократно используемых компонентов ostis-систем.]
 - [Поиск зависимостей между многократно используемыми компонентами в рамках библиотеки компонентов.]
 - [Обеспечение автоматического обновления компонентов, заимствованных в дочерние ostis-системы. Данная функция может включаться и отключаться по желанию разработчиков дочерней ostis-системы. Одновременное обновление одних и тех же компонентов во всех системах, его использующих, не должно ни в каком контексте приводить к несогласованности между этими системами. Это требование может оказаться довольно сложным, но без него работа Экосистемы OSTIS невозможна.]
- }

библиотека многократно используемых компонентов ostis-систем является подсистемой ostis-систем, которая имеет свою базу знаний, свой решатель задач и свой интерфейс. Однако не каждая ostis-система обязана иметь библиотеку.

библиотека многократно используемых компонентов ostis-систем

⇒ обобщенная декомпозиция*:

- {• база знаний библиотеки многократно используемых компонентов ostis-систем
 - решатель задач библиотеки многократно используемых компонентов ostis-систем
 - интерфейс библиотеки многократно используемых компонентов ostis-систем
- ⇒ декомпозиция*:
- {• минимальный интерфейс библиотеки многократно используемых компонентов ostis-систем
 - расширенный интерфейс библиотеки многократно используемых компонентов ostis-систем
- := [графический интерфейс библиотеки многократно используемых компонентов ostis-систем]
- }
- }

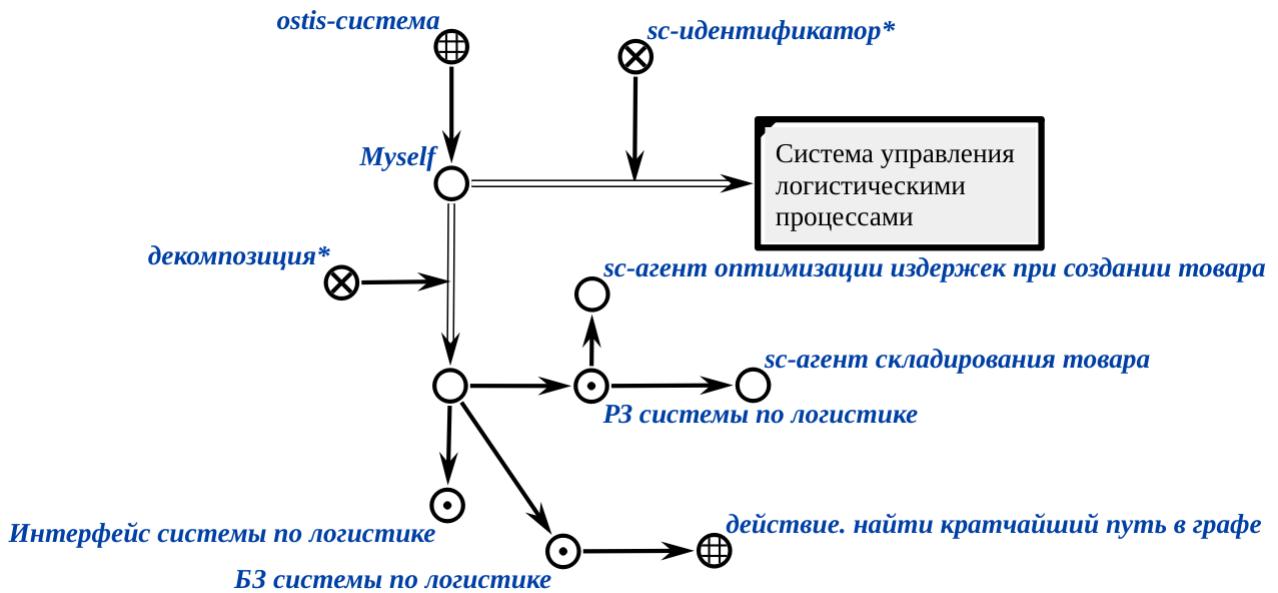
решатель задач библиотеки многократно используемых компонентов ostis-систем реализует функциональные возможности библиотеки ostis-систем.

база знаний библиотеки многократно используемых компонентов ostis-систем представляет собой иерархию многократно используемых компонентов ostis-систем и их спецификаций.

интерфейс библиотеки ostis-систем обеспечивает доступ к многократно используемым компонентам и возможностям библиотеки ostis-систем для пользователя и других систем. Существует минимальный и расширенный интерфейс библиотеки многократно используемых компонентов ostis-систем. Минимальный интерфейс позволяет менеджеру многократно используемых компонентов ostis-систем, входящему в состав какой-либо дочерней ostis-системы, подключиться к библиотеке многократно используемых компонентов ostis-систем и использовать ее функциональные возможности, то есть, например, получить доступ к спецификации компонентов и установить выбранные компоненты в дочернюю ostis-систему, получить сведения до доступных версиях компонента, его зависимостях и т.д. Расширенный интерфейс, в отличие от минимального интерфейса, позволяет не только получить доступ к компонентам для дальнейшей работы с ними, но и просматривать существующую структуру библиотеки, а также компоненты и их элементы в удобном и интуитивно понятном для пользователя виде.

Интеграция многократно используемых компонентов ostis-систем сводится к отождествлению ключевых узлов и устраниению возможных дублирований и противоречий исходя из спецификации компонента и его содержания. Отождествление sc-элементов происходит в ходе выполнения *действие. отождествить два указанных sc-элемента*, которое рассматривается в главе 3.2.

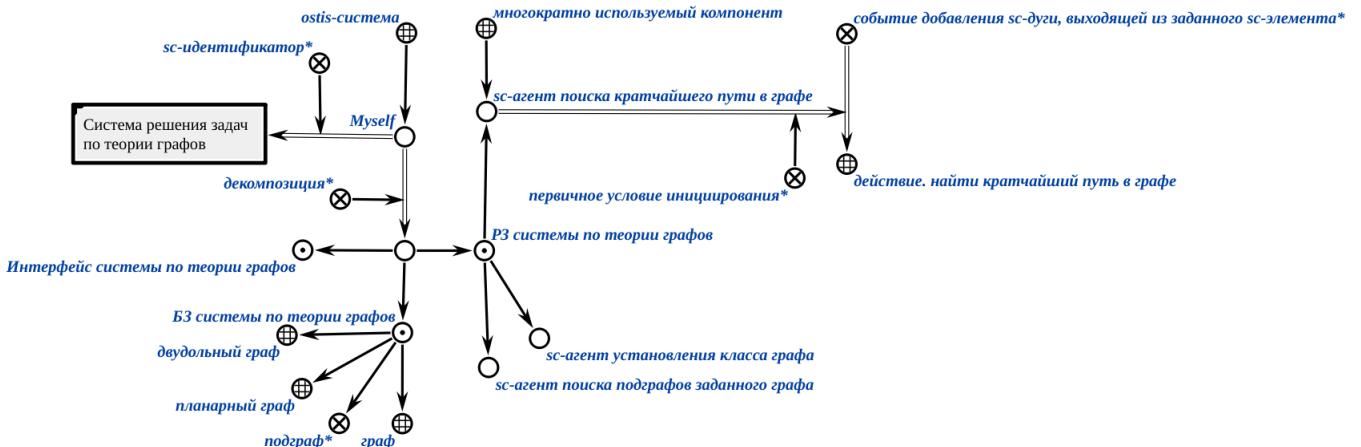
Рассмотрим пример интеграции многократно используемого компонента решателя задач, который представляет собой sc-агент поиска кратчайшего пути в графе, в систему управления логистическими процессами. Допустим, система управления логистическими процессами умеет решать задачи, связанные с оптимизацией издержек в процессе создания товара и со складированием товаров.



= Структура системы управления логистическими процессами

При этом в базе знаний системы описано, какие задачи должна решать система и соответствующие им действия. Например, действие. найти кратчайший путь в графе, которое система пока что не умеет выполнять.

Система решения задач по теории графов имеет богатую базу знаний и решатель задач, в том числе имеет sc-агент поиска кратчайшего пути в графе, который специфицирован как многократно используемый компонент и может быть использован в системе по логистике.



= Структура системы решения задач по теории графов

В результате установки многократно используемого компонента в виде sc-агента поиска кратчайшего пути в графе вся его sc-модель погружается в систему решения задач по логистике. При интеграции многократно используемого компонента, который представляет собой sc-агент поиска кратчайшего пути в графе, в систему по логистике ключевой узел *действие. найти кратчайший путь в графе* системы по логистике отождествляется с таким же узлом из установленного компонента из системы по теории графов. Таким образом, при выполнении логистических задач, система сможет интерпретировать действие по поиску кратчайших путей с помощью интегрированного компонента.

Интеграция любых компонентов ostis-систем происходит автоматически, без вмешательства разработчика. Это достигается за счёт использования SC-кода и его преимуществ, унификации спецификации многократно используемых компонентов и тщательного отбора компонентов в библиотеках эксперты сообществом, ответственным за эту библиотеку.

Рассмотрим понятие многократно используемого компонента ostis-систем. Под многократно используемым компонентом ostis-систем понимается компонент некоторой ostis-системы, который может быть использован в рамках другой ostis-системы. Это компонент ostis-системы, который может быть использован в других ostis-системах (*дочерних ostis-системах*) и содержит все те и только те sc-элементы, которые необходимы для функционирования компонента в дочерней ostis-системе. Другими словами это компонент некоторой *материнской ostis-системы*, который может быть использован в некоторой дочерней ostis-системе. Многократно используемые компоненты должны иметь унифицированную спецификацию и иерархию для поддержки совместимости с другими компонентами. Совместимость многократно используемых компонентов приводит систему к новому качеству, к дополнительному расширению множества решаемых задач при интеграции компонентов.

многократно используемый компонент ostis-систем

\coloneqq [многократно используемый компонент OSTIS]

\coloneqq часто используемый sc-идентификатор*:

[многократно используемый компонент]

\subset компонент ostis-системы

компонент ostis-системы – это целостная часть ostis-системы, которая содержит все те (и только те) sc-элементы, которые необходимы для её функционирования в ostis-системе. многократно используемый компонент ostis-систем – это общий компонент (общая часть) для некоторого множества ostis-систем, который многократно используется, дублируется и входит в состав некоторого множества ostis-систем. Отличие многократно используемого компонента ostis-систем от компонента ostis-системы в том, что многократно используемый компонент имеет спецификацию, достаточную для установки этого компонента в дочернюю ostis-систему. Спецификация является частью базы знаний библиотеки многократно используемых компонентов соответствующей материнской ostis-системы. Есть техническая возможность встроить многократно используемый компонент в дочернюю ostis-систему.

Требования, предъявляемые к многократно используемым компонентам ostis-систем, наследуют общие требования к проектированию программных компонентов, а также включают следующие:

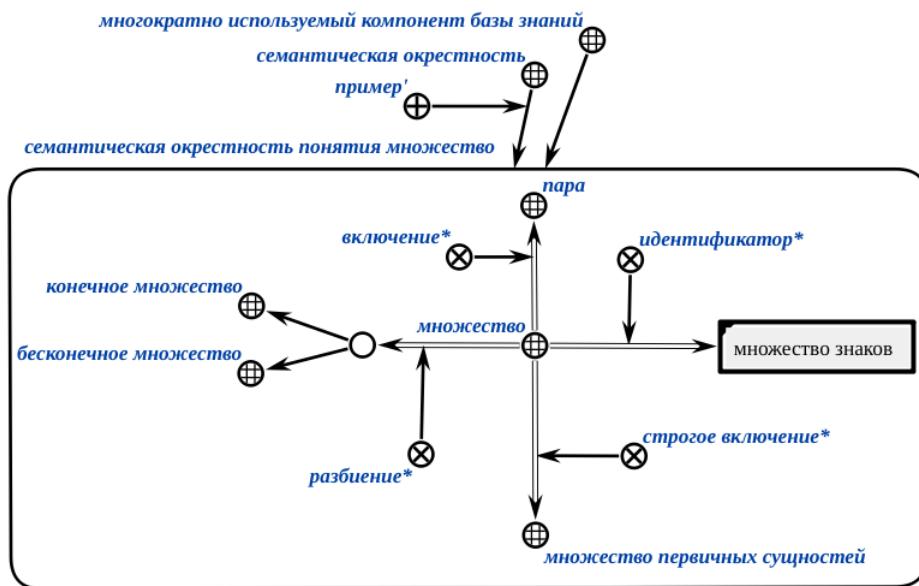
- Существует техническая возможность встроить многократно используемый компонент в дочернюю ostis-систему.
- Многократно используемый компонент должен выполнять свои функции наиболее общим образом, чтобы круг возможных систем, в которые он может быть встроен, был наиболее широким.
- Совместимость многократно используемого компонента: компонент должен стремиться повышать уровень договороспособности ostis-систем, в которые он встроен и иметь возможность автоматической интеграции в другие системы;
- Самодостаточность компонентов (или групп компонентов) технологии, т.е. способности их функционировать отдельно от других компонентов без утраты целесообразности их использования.

Рассмотрим классификацию многократно используемых компонентов ostis-систем. Класс многократно используемого компонента ostis-систем является важной частью спецификации компонента, позволяющей лучше понять назначение и область применения данного компонента, а также класс многократно используемого компонента является важнейшим критерием поиска компонентов в библиотеке ostis-систем. Основной признак классификации многократно используемых компонентов является признак предметной области, к которой относится компонент. Здесь структура может быть довольно богатой в соответствии с иерархией областей человеческой деятельности.

многократно используемый компонент ostis-систем

\Rightarrow разбиение*:

- {• **многократно используемый компонент базы знаний**
 - \subset семантическая окрестность
 - Э семантическая окрестность города Минска
 - Э семантическая окрестность понятия множество
 - \subset предметная область и онтология
 - Э предметная область и онтология треугольников
 - \subset шаблон многократно используемого компонента ostis-систем
 - Э Шаблон описания предметной области
 - Э Шаблон описания отношения
- **многократно используемый компонент решателя задач**
 - \subset автоматный агент обработки знаний
 - Э агент подсчёта мощности множества
 - \subset программа обработки знаний
- **многократно используемый компонент интерфейса**
 - \subset многократно используемый компонент пользовательского интерфейса для отображения
 - \subset интерактивный многократно используемый компонент пользовательского интерфейса



= Пример многократно используемого компонента базы знаний

Для компонентов баз знаний важнейшим признаком классификации многократно используемых компонентов является **вид используемых знаний**. Для компонентов решателя задач - **модель решения задач**. Для компонентов интерфейса - **вид интерфейса** в соответствии с классификацией компонентов пользовательских интерфейсов.

многократно используемый компонент ostis-систем

⇒ разбиение*:

Типология компонентов ostis-систем по атомарности[^]

- = {• атомарный многократно используемый компонент ostis-систем
 - Э агент подсчёта мощности множества
- неатомарный многократно используемый компонент ostis-систем
 - Э агент решения геометрических задач

}

атомарный многократно используемый компонент ostis-систем – это компонент, который в текущем состоянии библиотеки ostis-систем рассматривается как неделимый, то есть не содержит в своем составе других компонентов. Принадлежность многократно используемого компонента ostis-систем классу атомарных компонентов зависит от того, как специфицирован этот компонент и от существующих на данный момент компонентов в библиотеке. Неатомарный многократно используемый компонент в текущем состоянии библиотеки ostis-систем содержит в своем составе другие атомарные или неатомарные компоненты, он не зависит от своих частей. Без какой-либо части неатомарного компонента его назначение сужается. В общем случае атомарный компонент может перейти в разряд неатомарных в случае, если будет принято решение выделить какую-то из его частей в качестве отдельного компонента. Все вышеизложенное, однако, не означает, что даже в случае его платформенной независимости, атомарный компонент всегда хранится в sc-памяти как сформированная sc-структура. Например, платформенно-независимая реализация sc-агента всегда будет представлена набором scp-программ, но будет атомарным многократно используемым компонентом OSTIS в случае, если ни одна из этих программ не будет представлять интереса как самостоятельный компонент. В общем случае неатомарный компонент может перейти в разряд атомарных в случае, если будет принято решение по каким-либо причинам исключить все его части из рассмотрения в качестве отдельных компонентов. Следует отметить, что неатомарный компонент необязательно должен складываться полностью из других компонентов, в его состав могут также входить и части, не являющиеся самостоятельными компонентами. Например, в состав реализованного на языке SCP sc-агента, являющего неатомарным многократно используемым компонентом могут входить как scp-программы, которые могут являться многократно используемыми компонентами (а могут и не являться), а также агентная scp-программа, которая не имеет смысла как многократно используемый компонент.

многократно используемый компонент ostis-систем

⇒ разбиение*:

Типология компонентов ostis-систем по зависимости[^]

- = {• зависимый многократно используемый компонент ostis-систем
 - Э визуализатор для системы по химии

- независимый многократно используемый компонент ostis-систем
 - Э предметная область чисел
- }

зависимый многократно используемый компонент ostis-систем зависит хотя бы от одного другого компонента библиотеки ostis-систем, т.е. не может быть встроен в дочернюю ostis-систему без компонентов, от которых он зависит. независимый многократно используемый компонент ostis-систем не зависит ни от одного другого компонента библиотеки ostis-систем.

многократно используемый компонент ostis-систем

⇒ разбиение*:

- Типология компонентов ostis-систем по способу их хранения[^]**
- = {• многократно используемый компонент ostis-систем, хранящийся в виде внешних файлов
 - ⇒ разбиение*:
 - {• многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов
 - многократно используемый компонент ostis-систем, хранящийся в виде скомпилированных файлов
 - многократно используемый компонент, хранящийся в виде sc-структуры
- }

На данном этапе развития Технологии OSTIS более удобным является хранение компонентов в виде исходных текстов.

многократно используемый компонент ostis-систем

⇒ разбиение*:

- Типология компонентов ostis-систем по зависимости от ostis-платформы[^]**
- = {• платформенно-зависимый многократно используемый компонент ostis-систем
 - ▷ ostis-платформа
 - ▷ абстрактный sc-агент, не реализуемый на Языке SCP
 - платформенно-независимый многократно используемый компонент ostis-систем
 - ▷ многократно используемый компонент базы знаний
 - ▷ платформенно-независимый абстрактный sc-агент
 - ▷ scp-программа
- }

Под **платформенно- зависимым многократно используемым компонентом ostis-систем** понимается компонент, частично или полностью реализованный при помощи каких-либо сторонних с точки зрения Технологии OSTIS средств. Недостаток таких компонентов в том, что интеграция таких компонентов в интеллектуальные системы может сопровождаться дополнительными трудностями, зависящими от конкретных средств реализации компонента. В качестве возможного преимущества платформенно- зависимых многократно используемых компонентов ostis-систем можно выделить их, как правило, более высокую производительность за счет реализации их на более приближенном к платформе уровне. В общем случае платформенно- зависимый многократно используемый компонент ostis-систем может поставляться как в виде набора исходных кодов, так и скомпилиированном виде. Процесс интеграции платформенно- зависимого многократно используемого компонента ostis-систем в дочернюю систему, разработанную по Технологии OSTIS, сильно зависит от технологий реализации данного компонента и в каждом конкретном случае может состоять из различных этапов. Каждый платформенно- зависимый многократно используемый компонент ostis-систем должен иметь соответствующую подробную, корректную и понятную инструкцию по его установке и внедрению в дочернюю систему. Под **платформенно-независимым многократно используемым компонентом ostis-систем** понимается компонент, который целиком и полностью представлен на SC-коде. В случае неатомарного многократно используемого компонента это означает, что все более простые компоненты, входящие в его состав также обязаны быть платформенно-независимыми многократно используемыми компонентами ostis-систем. Процесс интеграции платформенно- зависимого многократно используемого компонента ostis-систем в дочернюю систему, разработанную по Технологии OSTIS, существенно упрощается за счет использования общей унифицированной формальной основы представления и обработки знаний.

Наиболее ценными являются платформенно-независимые многократно используемые компоненты ostis-систем.

многократно используемый компонент ostis-систем

⇒ разбиение*:

- Типология компонентов ostis-систем по динамичности их установки[^]**

- = {• *динамически устанавливаемый многократно используемый компонент ostis-систем*
 - := [многократно используемый компонент, при установке которого система не требует перезапуска]
 - ⇒ *декомпозиция**:
 - {• *многократно используемый компонент, хранящийся в виде скомпилированных файлов*
 - *многократно используемый компонент базы знаний*
- *многократно используемый компонент, при установке которого система требует перезапуска*

Процесс интеграции компонентов разных видов на разных этапах жизненного цикла osits-систем бывает разным. Наиболее ценными являются компоненты, которые могут быть интегрированы в работающую систему без прекращения её функционирования. Некоторые системы, особенно системы управления, нельзя останавливать, а устанавливать и обновлять компоненты нужно.

многократно используемый компонент ostis-систем

- ▷ *типовая подсистема ostis-систем*
 - Э Среда коллективной разработки баз знаний ostis-систем
 - Э Визуальный web-ориентированный редактор sc.g-текстов

Для того, чтобы хранить многократно используемые компоненты ostis-систем, необходимо некоторое хранилище. Таким хранилищем может выступать как какая-либо ostis-система, так и стороннее хранилище, например, облачный сервис. Помимо внешних файлов компонента в хранилище должна находиться его спецификация.

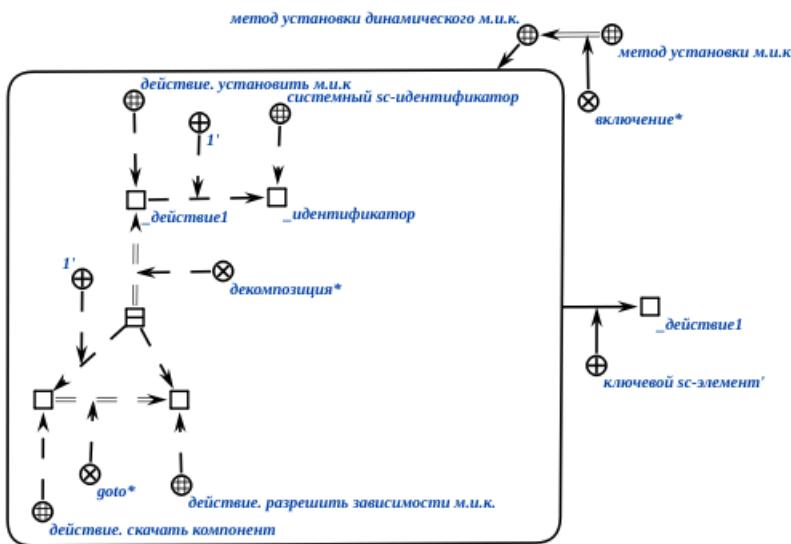
Каждый *многократно используемый компонент ostis-систем* должен быть специфицирован в рамках библиотеки. Данные спецификации включают в себя основные знания о компоненте, которые позволяют обеспечить построение полной иерархии компонентов и их зависимостей, а также обеспечивают беспрепятственную интеграцию компонентов в дочерние ostis-системы. Для спецификации компонента используются, как отношения, так и классы компонента.

Чтобы многократно используемый компонент мог быть принят в библиотеку, нужно специфицировать его используя необходимое для установки отношение, специфицирующее многократно используемый компонент ostis-систем. В то же время необязательное для установки отношение, специфицирующее многократно используемый компонент ostis-систем помогает лучше понять суть компонента, упрощает поиск, но не является обязательным для того, чтобы компонент мог быть установлен в ostis-систему.

отношение, специфицирующее многократно используемый компонент ostis-систем[^]

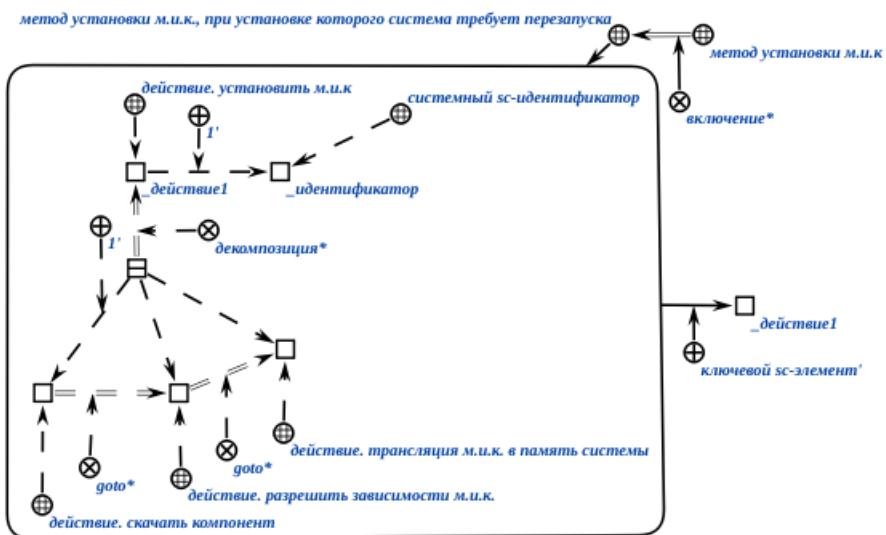
- := [отношение, которое используется при спецификации многократно используемого компонента ostis-систем]
- ⇒ *разбиение**:
 - {• *необходимое для установки отношение, специфицирующее многократно используемый компонент ostis-систем*
 - Э метод установки*
 - Э адрес хранилища*
 - Э зависимости компонента*
 - *необязательное для установки отношение, специфицирующее многократно используемый компонент ostis-систем*
 - Э сопутствующий компонент*
 - Э история изменений*
 - Э авторы*
 - Э примечание*
 - Э пояснение*
 - Э идентификатор*
 - Э ключевой sc-элемент*
 - Э назначение*

Метод установки позволяет пользователю установить компонент вручную, а **менеджеру компонентов** - автоматически. Основные два метода установки многократно используемых компонентов - метод установки динамически устанавливаемого многократно используемого компонента ostis-систем и метод установки многократно используемого компонента, при установке которого система требует перезапуска. При динамической установке необходимо только скачать компонент, и он сразу же функционирует в системе.



= Метод установки динамически устанавливаемого многократно используемого компонента *ostis*-систем

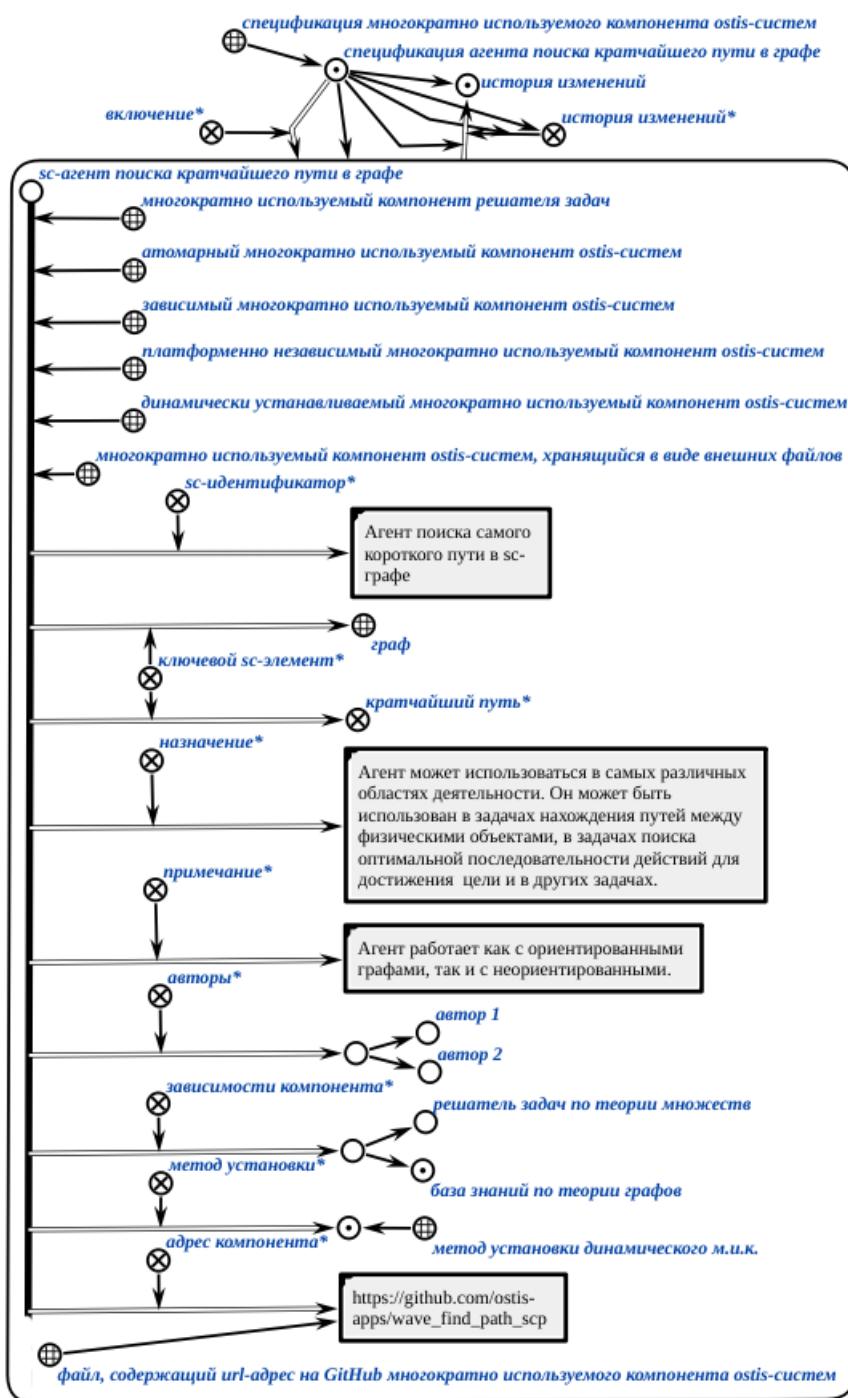
При установке компонента, при установке которого система требует перезапуска необходимо помимо скачивания компонента транслировать его в память системы.



= Метод установки динамически устанавливаемого многократно используемого компонента, при установке которого система требует перезапуска

Связки отношения *адрес хранилища** связывают многократно используемый компонент, хранящийся в виде внешних файлов и файл, содержащий url-адрес многократно используемого компонента ostis-систем. Таким файлом может быть файл, содержащий url-адрес на GitHub многократно используемого компонента ostis-систем, файл, содержащий url-адрес на Google Drive многократно используемого компонента ostis-систем, файл, содержащий url-адрес на Docker Hub многократно используемого компонента ostis-систем и другие.

Связки отношения *зависимости компонента** связывают многократно используемый компонент, и множество компонентов, без которых устанавливаемый компонент не может быть встроен в дочернюю *ostis*-систему.



= Пример спецификации многократно используемого компонента ostis-систем

В некоторых случаях может оказаться, что для использования одного многократно используемого компонента OSTIS целесообразно или даже необходимо дополнительно использовать несколько других многократно используемых компонентов OSTIS. Например, может оказаться целесообразным вместе с каким либо sc-агентом информационного поиска использовать соответствующую команду интерфейса, которая представлена отдельным компонентом и позволит пользователю задавать вопрос для указанного sc-агента через интерфейс системы. В таких случаях для связи компонентов используется отношение *相伴隨 компонент**. Наличие таких связей позволяет устранить возможные проблемы неполноты знаний и навыков в дочерней системе, из-за которых какие-либо из компонентов могут не выполнять свои функции. Связки отношения相伴隨 компонент* связывают многократно используемые компоненты ostis-систем, которые целесообразно использовать в дочерней системе вместе. Каждая такая связка может дополнительно быть снабжена sc-комментарием или sc-пояснением, отражающим суть указываемой зависимости.

§ 5.1.3. Менеджер многократно используемых компонентов ostis-систем

- ⇒ *ключевое понятие**:
 - менеджер многократно используемых компонентов ostis-систем

- ⇒ *ключевое знание**:
 - Функциональные возможности менеджера многократно используемых компонентов ostis-систем

менеджер многократно используемых компонентов ostis-систем – подсистема ostis-системы, с помощью которой проходит взаимодействие с библиотекой компонентов ostis-систем.

менеджер многократно используемых компонентов ostis-систем

- := часто используемый sc-идентификатор*:
 - [менеджер многократно используемых компонентов]
- := часто используемый sc-идентификатор*:
 - [менеджер компонентов]
- ⇒ обобщенная декомпозиция*:
 - {• база знаний менеджера многократно используемых компонентов ostis-систем
 - решатель задач менеджера многократно используемых компонентов ostis-систем
 - интерфейс менеджера многократно используемых компонентов ostis-систем

База знаний менеджера компонентов содержит все те знания, которые необходимы для установки многократно используемого компонента в дочернюю ostis-систему. К таким знаниям относятся знания о спецификации многократно используемых компонентов, методы установки компонентов, знание о библиотеках ostis-систем, с которыми происходит взаимодействие. Решатель задач менеджера компонентов взаимодействует с библиотекой ostis-систем и позволяет установить и интегрировать многократно используемые компоненты в дочернюю ostis-систему, также выполнять поиск, обновление, публикацию, удаление компонентов. Интерфейс менеджера многократно используемых компонентов обеспечивает удобное для пользователя и других систем использование менеджера компонентов.

Функциональные возможности менеджера компонентов ostis-систем следующие:

- **Поиск многократно используемых компонентов ostis-систем.** Множество возможных критериев поиска соответствует спецификации многократно используемых компонентов. Такими критериями могут быть классы компонента, его авторы, идентификатор, фрагмент примечания, назначение, принадлежность какой-либо предметной области, вид знаний компонента и другие.
- **Установка многократно используемого компонента ostis-систем.** Установка многократно используемого компонента происходит вне зависимости от типологии, способа установки и местоположения компонента. Необходимое условие для возможности установки многократно используемого компонента – наличие *спецификации многократно используемого компонента ostis-систем*. Перед установкой многократно используемого компонента в дочернюю систему необходимо разрешить все зависимости путём установки зависимых компонентов. После успешной установки компонента, в базе знаний дочерней системы генерируется информационная конструкция, обозначающая факт установки компонента в систему с помощью отношения *установленные компоненты**. После установки компонента в ostis-систему могут возникнуть противоречия в базе знаний, которые устраняются с помощью средств обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы.
- **Спецификация многократно используемого компонента ostis-систем.** Менеджер компонентов позволяет специфицировать компоненты, входящие в состав библиотеки ostis-систем, а также специфицировать новые создаваемые компоненты, которые будут публиковаться в библиотеку ostis-систем. При этом спецификация может происходить как автоматически, так вручную. Например, менеджер компонентов может обновить спецификацию используемого компонента в соответствии с тем, в какие новые ostis-системы его установили, обновление спецификации авторства компонента при его редактировании в библиотеке ostis-систем, спецификация ошибок, выявленных в процессе эксплуатации компонента и так далее.
- **Формирование многократно используемого компонента ostis-систем по шаблону с заданными параметрами.**
- **Публикация многократно используемого компонента ostis-систем в библиотеку ostis-систем.** При публикации компонента в библиотеку ostis-систем происходит верификация на основе спецификации компонента. Также существует возможность обновления версии опубликованного компонента сообществом его разработчиков.
- **Обновление установленного многократно используемого компонента ostis-систем**
- **Удаление установленного многократно используемого компонента.** Как и в случае установки после удаления многократно используемого компонента из ostis-системы в базе знаний системы устанавливается факт удаления компонента. Эта информация является важной частью истории эксплуатации ostis-системы.

- **Редактирование многократно используемого компонента в библиотеке *ostis*-систем.**
- **Добавление и удаление отслеживаемых *ostis*-системой библиотек.** Менеджер компонентов содержит информацию о множестве источников для установки компонентов, перечень которых можно дополнять вручную. По умолчанию менеджер компонентов отслеживает Библиотеку Метасистемы OSTIS, однако можно создавать и добавлять дополнительные библиотеки *ostis*-систем.

§ 5.1.4. Многократно используемые встраиваемые *ostis*-системы

Глава 5.2.

Методика и средства проектирования и анализа качества баз знаний ostis-систем

**Бутрин С.В.
Шункевич Д.В.
Банцевич К.А.**

⇒ *аннотация**:
[Аннотация к главе.]

Правки:

- Написать аннотацию
- Описать индивидуальный аспект редактора базы знаний
- Перенести материал по коллективной разработке от Давыденко
- Изменить содержание главы:
 - Действия и методики проектирования баз знаний ostis-систем;
 - Индивидуальный аспект проектирования и разработки баз знаний ostis-систем (название временное) (иdea - отразить локальный процесс работы с бз, т.е. редакторы, исходники, развертка и запуск всего);
 - Коллективный аспект проектирования и разработки баз знаний ostis-систем (название временное) (иdea - отразить процесс согласования локальных фрагментов бз в согласованную бз, т.е. создание предложений в бз, их модерирование и внедрение);
 - Логико-семантическая модель ostis-системы автоматизации управления взаимодействием разработчиков различных категорий в процессе проектирования базы знаний ostis-системы (больше рассматривается с точки зрения коллективного процесса, но возможно стоит учесть влияние на индивидуальный процесс создания фрагментов бз) (вообщем возможно не стоит выделять в отдельный раздел или же объединить со вторым разделом);
 - Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы (возможно объединить с логическими дырами);
- Описать библиографию

§ 5.2.1. Действия и методики проектирования баз знаний ostis-систем

Проектирование базы знаний включает в себя два аспекта: индивидуальный и коллективный.

Индивидуальный представляет собой наполнение изолированной части базы знаний конкретным пользователем.

Коллективный представляет собой согласование все базы знаний и включает в себя процесс внедрения изменений и внесения предложений.

Таким образом индивидуальный аспект включает в себя средства и методы позволяющие пользователю непосредственно наполнять базу знаний посредством редакторов, трансляторов и т.д. При этом процесс редактирования базы знаний должен быть максимально удобным и понятным для пользователя.

В идеале редактор должен являться частью системы и должен быть описан в самой базе знаний, так как система имеющая всю информацию о редакторе может позволить пользователю эффективно с ним взаимодействовать засчет системы поддержки. Такая система поддержки должна позволить пользователю непрерывно и (незаметно? естественно?) овладевать редактором учитывая при это особенности как самого редактора (его интерфейса, возможностей и т.д.) , так и самого пользователя (его предпочтения, ограничения и т.д.)

Коллективный же процесс включает в себя средства и методы для автоматического или полуавтоматического согласования общей базы знаний с индивидуальными базами знаний пользователей. Т.е. он включает в себя процессы создания предложений по изменению, их рассмотрение и внесение в общую базу знаний.

Процесс разработки базы знаний включает в себя следующие стадии:

- Формирование начальной структуры гибридной базы знаний, которая предполагает:

- формирование структуры разделов базы знаний, соответствующей варианту структуризации базы знаний с точки зрения разработчиков, описанному в подразделе 2.6.3 данной диссертационной работы;
- выявление описываемых предметных областей;
- построение иерархической системы описываемых предметных областей;
- построение иерархии разделов базы знаний в рамках предметной части базы знаний, учитывающей построенную на предыдущем этапе иерархию предметных областей.
- Выявление компонентов базы знаний, которые могут быть заимствованы из библиотеки многократно используемых компонентов баз знаний, и включение их в состав разрабатываемой базы знаний;
- Формирование проектных заданий на разработку недостающих фрагментов базы знаний и распределение заданий между разработчиками;
- Разработка и согласование фрагментов базы знаний, которые, в свою очередь, могут в дальнейшем быть включены в состав библиотеки многократно используемых компонентов баз знаний;
- Верификация и отладка базы знаний.

Следует отметить, что в процессе совершенствования базы знаний этапы 3 - 5 выполняются циклически.

Для обеспечения свойства рефлексивности интеллектуальной системы, в частности, возможности автоматизации анализа истории эволюции базы знаний и генерации планов по ее развитию, вся деятельность, связанная с разработкой базы знаний, должна специфицироваться в самой этой базе знаний теми же средствами, что и предметная часть.

§ 5.2.2. Индивидуальный аспект проектирования и разработки баз знаний ostis-систем

§ 5.2.3. Коллективный аспект проектирования и разработки баз знаний ostis-систем

Для решения указанной задачи разработана модель деятельности, направленной на создание гибридных баз знаний коллективом разработчиков.

Данная модель базируется на модели деятельности различных субъектов и реализована в виде онтологии предметной области деятельности разработчиков, направленной на разработку и модификацию гибридных баз знаний.

Процесс создания и редактирования базы знаний ostis-системы сводится к формированию разработчиками предложений по редактированию того или иного раздела базы знаний и последующему рассмотрению этих предложений администраторами базы знаний.

Кроме того, предполагается, что в случае необходимости для верификации поступающих предложений по редактированию базы знаний могут привлекаться эксперты, а управление процессом разработки осуществляется менеджерами соответствующих проектов по разработке базы знаний.

При этом формирование проектных заданий и их спецификация осуществляются также при помощи механизма предложений по редактированию соответствующего раздела базы знаний.

Таким образом, вся информация, связанная с текущими процессами разработки базы знаний, историей и планами ее развития, хранится в той же базе знаний, что и ее предметная часть, т. е. часть базы знаний, доступная конечному пользователю системы. Такой подход обеспечивает широкие возможности автоматизации процесса создания баз знаний, а также последующего анализа и совершенствования базы знаний.

Каждое предложение по редактированию базы знаний представляет собой структуру, содержащую sc-текст, который предлагается включить в состав согласованной части базы знаний. В состав таких предложений могут входить знаки действий по редактированию базы знаний, которые автоматически инициируются и выполняются соответствующими агентами после утверждения предложения.

§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы

Важным этапом в разработке любой системы является контроль качества, так как именно на этом этапе определяется степень жизнеспособности и эффективности системы.

Верификация является видом анализа качества и часть процесса разработки. Она заключается в проверке информации на правильность и точность. Ее целью является выявление ошибок, различных дефектов и недоработок для своевременного их устранения.

Существующие на данный момент методы верификации хорошо развиты и разработано большое количество различных моделей верификации, использующих расширенные таблицы решения, сети ПетриPetri, различные логики, например, логики с векторной семантикойVTF1VTF2 и другие модели. Более того формируются специализированные онтологии для описания многообразия средств и моделей верификации баз знанийRybinaAlgo. Однако механизма взаимодействия средств, использующих данные методы, нет.

Поэтому средства верификации баз знаний, существующие на данный момент, обладают рядом таких проблем как **ZhangProblems**:

- зависимость от формата представления информации, из-за чего приходится тратить дополнительное время на конвертирование информацию
- проблема невозможности быть переиспользованными, так как средства обычно делаются с учетом особенностей конкретной системы
- проблема отсутствия механизма взаимодействия средств верификации и анализа знаний
- высокая роль человека в процессе верификации, так как самым распространенным способом верификации баз знаний является ручная проверка базы экспертом, человек выступает как администратор, принимающий единогласное решение, навязывая свое мнение системе
- современные средства не учитывают и не рассматривают процесс верификации в рамках взаимодействия систем друг с другом.

Эти проблемы могли бы быть решены, если:

- использовать унифицированный и удобный формат представления знаний
- системы создавались бы по общей методологии и были бы совместимы друг с другом
- продумать и реализовать механизм позволяющий системе стремиться самой принимать решение относительно своего состояния и наличия в нем проблемных моментов и ошибок, система может допускать ошибки и не всегда принимать верные решения, но это должны быть ее ошибки, а не экспертов и разработчиков

Преимуществами Технологии OSTIS в рамках задачи верификации являются:

- наличие общей методологии проектирования интеллектуальных систем, позволяющая решить проблему совместимости систем при их коллективном взаимодействии;
- все знания представлены в унифицированном виде, что позволяет эффективно их обрабатывать, сводя затраты на конвертирование к минимуму;
- средства, с помощью которых производится выявление, анализ и устранение противоречий описаны в самой базе знаний, а так же их спецификация представлены в самой базе знаний системы, тем самым обеспечивая легкость их расширения и позволяя системе знать, каким инструментарием она обладает;
- отсутствие семантических эквивалентных фрагментов, что обеспечивает локальность вносимых исправлений и исключает необходимость вносить исправления многократно в разных местах;
- многоагентный подход, который позволяет рассматривать средства анализа и верификации баз знаний как коллектив агентов, способных взаимодействовать друг с другом и дальнейшем принимать общее решение касательно состояния базы знаний.

Предлагаемый подход сводится к разработке:

- специализированной *предметной области и онтологии*, которая бы содержала бы в себе необходимые знания о возможных видах проблемных фрагментов базы знаний и способах их исправления
- алгоритма, позволяющего системе выявить в себе проблемные фрагменты и устранить их, при этом обеспечив согласованность работы средств самой системы
- специализированного решателя задач, содержащего необходимые агенты для выявления и устранения проблемных фрагментов

Качество базы знаний во многом определяется уровнем наличия/отсутствия не-факторов **Наринъяни А.С.НЕФакКВ-2004ст** в базе знаний.

не-фактор

- := [группа семантических свойств, определяющих качество информации, хранимой в памяти кибернетической системы]
- = {• корректность/некорректность информации, хранимой в памяти кибернетической системы
 - однозначность/неоднозначность информации, хранимой в памяти кибернетической системы
 - целостность/нечелостность информации, хранимой в памяти кибернетической системы
 - чистота/загрязненность информации, хранимой в памяти кибернетической системы
 - достоверность/недостоверность информации, хранимой в памяти кибернетической системы
 - точность/неточность информации, хранимой в памяти кибернетической системы
 - четкость/нечеткость информации, хранимой в памяти кибернетической системы
 - определенность/недоопределенность информации, хранимой в памяти кибернетической системы
}

проблемная структура

- := [структура, описывающая проблемный фрагмент базы знаний]
 - := [структура, описывающая некачественный фрагмент базы знаний]
 - ⇐ *объединение**:
 - {• *некорректная структура*
 - := [структура, содержащая фрагменты, противоречащие каким либо правилам или закономерностям описанным в базе знаний]
 - *структура, описывающая неполноту в базе знаний*
 - := [структура, в которой имеется неполнота (то есть имеется некоторое количество информационных дыр)]
 - ⇒ *примечание**:
[Под структурой, описывающей неполноту в базе знаний, понимается структура, содержащая фрагмент базы знаний, в котором отсутствует какая-либо информация, которая необходима (или, по крайней мере, желательна) для однозначного и полного понимания смысла данного фрагмента.]
 - *информационный мусор*
 - := [структура, удаление которой существенно не усложнит деятельность системы]
 - := [структура, содержащая фрагмент базы знаний, который по каким-либо причинам стал ненужным и требует удаления]
- }

противоречие*

- := [пара противоречащих друг другу фрагментов информации, хранимой в памяти кибернетической системы*]
 - ⇒ *примечание**:
[Чаще всего противоречащими друг другу информационными фрагментами являются:
 - явно представленная в памяти некоторая закономерность (некоторое правило)
 - информационный фрагмент, не соответствующий (противоречащий) указанной закономерности
 В этом случае некорректность может присутствовать:
 - либо в информационном фрагменте, который противоречит указанной закономерности;
 - либо в самой этой закономерности;
 - либо и там и там.
-]

некорректная структура

- ⇐ *включение**:
 - {• *дублирование системных идентификаторов*
 - *несоответствие элементов связи доменам отношения*
 - *цикл по отношению порядка*
 - *структура, противоречащая свойству единственности*
- }

структура, описывающая неполноту в базе знаний

- ⇐ *включение**:
 - {• *не указан максимальный класс объектов исследования предметной области*
 - *для сущности указан системный, но не указаны основные идентификаторы для всех внешних языков*
 - *не указаны домены отношения*
 - *понятие не соотнесено ни с одной предметной областью*
- }

требующее внимание разработчика

- := [проблемная структура, для исправления которой требуется участие разработчика]

множество элементов, которые должны быть удалены для исправления структуры*

- := [множество элементов, удаление которых из структуры позволяет устраниить в ней противоречие]

множество элементов, которые должны быть добавлены для исправления структуры*

- := [множество элементов, добавление которых в структуру позволяет устраниить в ней противоречия]

структура, которую система не способна исправить сама

:= [структура, в которой система не способна автоматически устраниТЬ противоречия]

следует отличать*

- ∅ {• структура, которая система не может решить сама
 - ⇒ примечание*: [Здесь структура, которую система не может решить сама, не может быть исправлена при взаимодействии с разработчиком и требует полного исправления от самого разработчика]
 - требующее внимания разработчика
 - ⇒ примечание*: [Здесь структура, требующая внимания разработчика, может быть решена в процессе верификации, но потребуется участие разработчика]
- }

Решатель задач средств выявления и устранения противоречий

⇐ декомпозиция абстрактного sc-агента*:

- {• Неатомарный агент выявления противоречий
 - := [Множество агентов, обеспечивающих поиск и фиксирование противоречий в структуре]
 - Неатомарный агент устранения противоречий
 - := [Множество агентов, создающих предложения по исправлению противоречий]
 - ⇒ примечание*: [Результатом работы таких агентов будут множества предлагаемых к удалению из структуры или добавлению в структуру элементов]
 - Агент слияния структур
 - := [Агент, создающий структуру содержащую все элементы сливаемых структур]
 - Агент применения предложений по устранению противоречий
 - Агент внесения исправлений в базу знаний
 - ⇒ примечание*: [Внесение изменений подразумевает не только исправление в базе знаний изначальной проблемной структуры, но и фиксацию самого факта изменения состояния базы знаний]
 - Неатомарный агент верификации структуры
 - ⇒ примечание*: [Агент обеспечивающий полный цикл верификации структуры координирую другие агенты]
- }

§ 5.2.5. Логико-семантическая модель ostis-системы обнаружения и анализа информационных дыр в базе знаний ostis-системы

Нужно согласовать и возможно объединить в раздел выше

§ 5.2.6. Логико-семантическая модель ostis-системы автоматизации управления взаимодействием разработчиков различных категорий в процессе проектирования базы знаний ostis-системы

пользователь базы знаний ostis-системы*

- := [бинарное отношение, связывающее sc-модель базы знаний ostis-системы и sc-элемент, обозначающий персону, участвующую в разработке или эксплуатации этой базы знаний]
- ∈ бинарное отношение
- ∈ ориентированное отношение
- ⇒ примечание*: [Данное отношение отражает связь пользователя и базы знаний в целом, при этом тот же сам пользователь может быть связан другими более частными отношениями с какими-либо фрагментами этой же базы знаний.]
- ⇒ разбиение*: *Типология отношений между базами знаний ostis-систем и их пользователями по наличию факта прохождения регистрации в этих ostis-системах[^]*

= {• зарегистрированный пользователь*
 • незарегистрированный пользователь*
 }

пользователь, обладающий правом просмотра sc-структурой*

:= [бинарное отношение, связывающее sc-элемент, обозначающий sc-структуру (например, фрагмент sc-модели базы знаний), и sc-элемент, обозначающий пользователя этой ostis-системы, который обладает правом просмотра этой sc-структуры.]
 ∈ бинарное отношение
 ∈ ориентированное отношение
 ⇒ примечание**:
 [Пользователь, обладающий правом просмотра sc-структурой базы знаний ostis-системы может быть зарегистрирован или не зарегистрирован в sc-модели базы знаний.]

пользователь, обладающий правом редактирования sc-структурой*

:= [бинарное отношение, связывающее sc-элемент, обозначающий sc-структуру (например, фрагмент sc-модели базы знаний), и sc-элемент, обозначающий зарегистрированного пользователя ostis-системы, который обладает правом редактирования этой sc-структуры.]
 ∈ бинарное отношение
 ∈ ориентированное отношение
 ▷ пользователь, обладающий правом просмотра sc-структурой*
 ⇒ пояснение*:
 [Связки отношения пользователя, обладающий правом редактирования sc-структурой ostis-системы* связывают sc-структуру (не обязательно всю sc-модель базы знаний) и пользователя, зарегистрированного в этой sc-модели базы знаний.]
 ⇒ покрытие*:
 {• пользователь, обладающий правом редактирования sc-структурой посредством формирования предложений по внесению изменений в согласованную часть базы знаний этой ostis-системы*
 • пользователь, обладающий правом редактирования sc-структурой с автоматическим формированием и принятием предложений по внесению изменений в согласованную часть базы знаний этой ostis-системы*
 }
 }

разработчик*

⊂ пользователь, обладающий правом редактирования sc-структурой*
 := [бинарное отношение, связывающее sc-элемент, обозначающий некоторый раздел базы знаний (в пределе – всю базы знаний), и sc-элемент, обозначающий пользователя ostis-системы, который может быть разработчиком данного раздела базы знаний, т. е. выполнять проектные задачи в рамках данного раздела]
 ⇒ разбиение*:
 Типология разработчиков баз знаний ostis-систем[^]
 = {• администратор*
 • менеджер*
 • эксперт*
 }

§ 5.2.7. Многократно используемые компоненты баз знаний ostis-систем

На сегодняшний день разработано большое число баз знаний по самым различным предметным областям. Однако в большинстве случаев каждая база знаний разрабатывается отдельно и независимо от других, в отсутствие единой унифицированной формальной основы для представления знаний, а также единых принципов формирования систем понятий для описываемой предметной области. В связи с этим разработанные базы оказываются, как правило, несовместимы между собой и не пригодны для повторного использования. Компонентный подход к разработке интеллектуальных компьютерных систем, реализуемый в виде **библиотеки многократно используемых компонентов ostis-систем**, позволяет решить описанные проблемы.

Глава 5.3.

Методика и средства компонентного проектирования решателей задач ostis-систем

Шункевич Д.В.

Марковец В.С.

⇒ *аннотация**:

[Аннотация к главе.]

В области разработки *решателей задач* существует большое количество конкретных реализаций, однако вопросы совместимости различных решателей при решении одной задачи практически не рассматриваются. Гипотетически возможно существование универсального решателя задач, объединяющего в себе все известные способы и методы решения задач. Однако использование такого решателя в прикладных целях не является целесообразным. Таким образом, наиболее приемлемым вариантом становится создание библиотеки совместимых между собой компонентов, из которых впоследствии может быть скомпилирован решатель, удовлетворяющий необходимым требованиям.

§ 5.3.1. Действия и методики проектирования решателей задач ostis-систем

Методика построения и модификации решателей задач включает несколько этапов.

- формирование требований и спецификация решателя задач;
- формирование коллектива sc-агентов, входящих в состав разрабатываемого решателя;
- разработка алгоритмов атомарных sc-агентов;
- реализация scr-программ;
- верификация разработанных компонентов;
- отладка разработанных компонентов, исправление ошибок.

Методика может быть применена как при разработке объединенных решателей, так и при разработке решателей частного вида, поскольку с формальной точки зрения все они трактуются как неатомарный абстрактный sc-агент.

Этап 1. Формирование требований и спецификация решателя задач

На данном этапе необходимо:

- четко выделить задачи, решение которых должен обеспечивать решатель задач;
- продумать предполагаемые способы их решения и на основе данного анализа определить место будущего решателя в общей иерархии решателей.

Важность данного этапа заключается в том, что при правильной классификации существует вероятность того, что в составе библиотеки многократно используемых компонентов ostis-систем уже есть реализованный вариант требуемого решателя. В противном случае, у разработчика появляется возможность включить разработанный решатель в библиотеку многократно используемых компонентов ostis-систем для последующего использования. Данные факты обусловлены тем, что структура библиотеки многократно используемых компонентов решателей задач основана на семантической классификации таких решателей и, соответственно, их компонентов.

При недостаточно четкой спецификации и классификации разрабатываемого решателя повышается вероятность того, что подходящий решатель не будет найден в библиотеке компонентов даже в случае, если он там есть, а вновь разработанный решатель не сможет быть включен в библиотеку. Таким образом, идея многократного использования уже разработанных компонентов будет нарушена, что существенно повысит затраты на разработку такого решателя.

Этап 2. Формирование коллектива sc-агентов, входящих в состав разрабатываемого решателя

В случае, когда найти в библиотеке готовый решатель, удовлетворяющий всем предъявляемым требованиям, не представляется возможным, необходимо выделить и специфицировать все компоненты такого решателя.

Результатом данного этапа является перечень полностью специфицированных sc-агентов, которые войдут в состав разрабатываемого решателя, с их иерархией вплоть до *атомарных sc-агентов*. В рамках данного этапа очень важно

проектировать коллектив агентов таким образом, чтобы максимально задействовать уже имеющиеся в библиотеке многократно используемые компоненты ostis-систем, а при отсутствии нужного компонента — иметь возможность включить его в библиотеку после реализации.

При разработке перечня агентов (в том числе их спецификаций) необходимо соблюдать ряд принципов:

- каждый разрабатываемый sc-агент должен быть по возможности предметно независим, т. е. во множестве ключевых узлов данного sc-агента не должны входить понятия, имеющие отношение непосредственно к рассматриваемой предметной области. Исключение составляют понятия из общих предметных областей, которые носят междисциплинарный характер (например, отношение *включение** или понятие *действие*). Данное правило также может быть нарушено в случае, если sc-агент является вспомогательным и ориентирован на обработку какого-либо конкретного класса объектов (например, sc-агенты, выполняющие арифметические вычисления, могут напрямую работать с конкретными отношениями *сложение** и *умножение** и т. п.). Всю необходимую для решения задачи информацию sc-агент должен извлекать из семантической окрестности соответствующего инициированного действия. Очевидно, что sc-агент, разработанный с учетом указанных требований, может быть использован при проектировании большего числа ostis-систем, чем в случае, если бы он был реализован с ориентацией на конкретную частную предметную область. После завершения разработки и отладки такой sc-агент должен быть включен в *библиотеку многократно используемых абстрактных sc-агентов*, которая входит в состав *библиотеки многократно используемых компонентов решателей задач*;
- не стоит путать понятия sc-агент и агентная программа (в том числе агентная scr-программа). Взаимодействие sc-агентов осуществляется исключительно посредством спецификации информационных процессов в общей памяти, каждый sc-агент реагирует на некоторый класс событий в sc-памяти. Таким образом, каждому sc-агенту соответствует некоторое условие инициирования и одна агентная программа, которая запускается автоматически при возникновении в sc-памяти соответствующего условия инициирования. При этом в рамках данной программы могут сколько угодно раз вызываться различные подпрограммы. Однако не стоит путать инициирование sc-агента, которое осуществляется при появлении в sc-памяти соответствующей конструкции, и вызов подпрограммы другой программой, который предполагает явное указание вызываемой подпрограммы и перечня ее параметров;
- каждый sc-агент должен самостоятельно проверять полноту соответствия своего условия инициирования текущему состоянию sc-памяти. В процессе решения какой-либо задачи может возникнуть ситуация, когда на появление одной и той же структуры среагировали несколько sc-агентов. В таком случае выполнение продолжают только те из них, условие инициирования которых полностью соответствует сложившейся ситуации. Остальные sc-агенты в данном случае прекращают выполнение и возвращаются в режим ожидания. Выполнение данного принципа достигается за счет тщательного уточнения спецификаций разрабатываемых sc-агентов. В общем случае условия инициирования у нескольких sc-агентов могут совпадать, например, в случае, когда одна и та же задача может быть решена разными способами и заранее неизвестно, какой из них приведет к желаемому результату;
- необходимо помнить, что неатомарный sc-агент с точки зрения других sc-агентов, не входящих в его состав, должен функционировать как целостный sc-агент (выполнять логически атомарные действия), что накладывает определенные требования на спецификации атомарных sc-агентов, входящих в его состав: как минимум, необходимо, чтобы в составе неатомарного sc-агента присутствовал хотя бы один атомарный sc-агент, условие инициирования которого полностью совпадает с условием инициирования данного неатомарного sc-агента;
- при необходимости реализации нового sc-агента следует руководствоваться следующими принципами выделения атомарных абстрактных sc-агентов:
 - проектируемый sc-агент должен быть максимально независим от предметной области, что позволит в дальнейшем использовать его при разработке решателей максимально возможного числа ostis-систем. При этом универсальность предполагает не только минимизацию числа ключевых узлов sc-агента, но и выделение класса действий, выполняемых данным sc-агентом таким образом, чтобы имело смысл включить данный sc-агент в *библиотеку многократно используемых абстрактных sc-агентов* и использовать его при разработке решателей других ostis-систем. Не следует искусственно увязывать ряд действий в один sc-агент и, наоборот, расчленять одно самодостаточное действие на поддействия: это вызовет сложности восприятия принципов работы sc-агента разработчиками и не позволит использовать sc-агент в ряде систем (например, в обучающих системах, которые должны объяснять ход решения пользователю);
 - выполняемое данным sc-агентом действие должно быть логически целостным и завершенным. Следует помнить, что все sc-агенты взаимодействуют исключительно через общую sc-память и избегать ситуаций, в которых инициирование одного sc-агента осуществляется путем явной генерации известного условия инициирования другим sc-агентом (т. е., по сути, явным непосредственным вызовом одного sc-агента другим);
 - имеет смысл выделять в отдельные sc-агенты относительно крупные фрагменты реализации некоторого общего алгоритма, которые могут выполняться независимо друг от друга.
- при объединении sc-агентов в коллективы рекомендуется проектировать их таким образом, чтобы они могли быть использованы не только как часть рассматриваемого неатомарного абстрактного sc-агента. В случае,

- если это не представляется возможным и некоторые sc-агенты, будучи отделенными от коллектива, теряют смысл, необходимо указать данный факт при документировании рассматриваемых sc-агентов;
- фактическим инициатором запуска sc-агента посредством общей памяти (автором соответствующей конструкции) может быть как непосредственно пользователь системы, так и другой sc-агент, что никак не должно отражаться в работе самого sc-агента.

Этап 3. Разработка алгоритмов атомарных sc-агентов

В рамках данного этапа необходимо продумать алгоритм работы каждого разрабатываемого *атомарного sc-агента*. Разработка алгоритма подразумевает выделение в нем логически целостных фрагментов, которые могут быть реализованы как отдельные *scp-программы*, в том числе выполняемые параллельно. Таким образом, появляется необходимость говорить не только о *библиотеке многократно используемых абстрактных sc-агентов*, но и о *библиотеке многократно используемых программ обработки sc-текстов* на различных языках программирования, в том числе *библиотеке многократно используемых scp-программ*. Благодаря этому часть scp-программ, реализующих алгоритм работы некоторого sc-агента, может быть заимствована из соответствующей библиотеки.

Важно помнить, что если в процессе работы *sc-агент* генерирует в памяти какие-либо временные структуры, то при завершении работы он обязан удалять всю информацию, использование которой в системе более нецелесообразно (убрать за собой информационный мусор). Исключение составляют ситуации, когда подобная информация необходима нескольким *sc-агентам* для решения одной задачи, однако после решения задачи информация становится бесполезной или избыточной и требует удаления. В данном случае может возникнуть ситуация, когда ни один из *sc-агентов* не в состоянии удалить информационный мусор. В таком случае возникает необходимость говорить о включении в состав решателя специализированных *sc-агентов*, задачей которых является выявление и уничтожение информационного мусора.

Этап 4. Реализация scp-программ

Конечным этапом непосредственно разработки является реализация специфицированных ранее *scp-программ* или при необходимости программ, реализуемых на уровне платформы.

Этап 5. Верификация разработанных компонентов

Верификация разработанных компонентов может осуществляться как вручную, так и с использованием специфицированных средств, входящих в состав системы автоматизации проектирования решателей задач *ostis*-систем.

Этап 6. Отладка разработанных компонентов. Исправление ошибок Этап отладки разработанных компонентов, в свою очередь, можно также условно разделить на более частные этапы:

- отладка отдельных scp-программ или программ, реализуемых на уровне платформы;
- отладка отдельных атомарных sc-агентов;
- отладка неатомарных sc-агентов, входящих в состав решателя задач;
- отладка всего решателя задач.

Этап 5 и Этап 6 могут выполняться параллельно и повторяются до тех пор, пока разработанные компоненты не будут удовлетворять необходимым требованиям.

Методика построения и модификации решателей задач основана на онтологии деятельности разработчиков таких решателей. Решатель задач представляет собой *абстрактный sc-агент*, в связи с чем разработка решателя сводится к разработке такого агента.

Фрагмент онтологии деятельности, направленной на построение и модификацию решателей задач:

действие. разработать решатель задач ostis-системы

= **действие. разработать абстрактный sc-агент**

⇐ **разбиение*:**

- {• **действие. разработать атомарный абстрактный sc-агент**
 - ⇒ **включение*:**
 - действие. разработать платформенно-независимый атомарный абстрактный sc-агент**
 - **действие. разработать неатомарный абстрактный sc-агент**
}

⇒ **абстрактное поддействие*:**

- **действие. специфицировать абстрактный sc-агент**
- **действие. найти в библиотеке абстрактный sc-агент, удовлетворяющий заданной спецификации**
- **действие. верифицировать sc-агент**
- **действие. отладить sc-агент**

действие. разработать платформенно-независимый атомарный абстрактный sc-агент

⇒ **абстрактное поддействие*:**

- действие. декомпозировать платформенно-независимый атомарный абстрактный sc-агент на scp-программы
- действие. разработать scp-программу

действие. разработать неатомарный абстрактный sc-агент

⇒ абстрактное поддействие*:

- действие. декомпозировать неатомарный абстрактный sc-агент на более частные
- действие. разработать абстрактный sc-агент

действие. разработать scp-программу

⇒ абстрактное поддействие*:

- действие. специфицировать scp-программу
- действие. найти в библиотеке scp-программу, удовлетворяющую заданной спецификации
- действие. реализовать специфициированную scp-программу
- действие. верифицировать scp-программу
- действие. отладить scp-программу

действие. верифицировать sc-агент

⇐ разбиение*:

- {• действие. верифицировать атомарный sc-агент
- действие. верифицировать неатомарный sc-агент

}

действие. отладить sc-агент

⇐ разбиение*:

- {• действие. отладить атомарный sc-агент
- действие. отладить неатомарный sc-агент

}

Наличие такой онтологии позволяет:

- частично автоматизировать процесс построения и модификации решателей;
- повысить эффективность информационной поддержки разработчиков, поскольку данная онтология может быть включена в базу знаний Метасистемы OSTIS.

§ 5.3.2. Логико-семантическая модель комплекса ostis-систем автоматизации проектирования решателей задач ostis-систем

К числу задач системы автоматизации проектирования решателей задач ostis-систем относится техническая поддержка разработчиков решателей, в том числе – обеспечение корректного и эффективного выполнения этапов, предусмотренных методикой проектирования решателей задач ostis-систем.

При разработке любых компонентов ostis-систем используются схожие принципы. Одним из основных принципов является принцип использования готовых компонентов различного рода, уже имеющихся в библиотеке многократно используемых компонентов ostis-систем, входящей в состав Метасистемы OSTIS.

Система автоматизации проектирования решателей задач сама по себе также является ostis-системой и имеет соответствующую структуру. Таким образом, модель данной системы включает sc-модель базы знаний, sc-модель объединенного решателя задач и sc-модель пользовательского интерфейса.

В рамках системы условно выделяются две подсистемы – подсистема автоматизации проектирования агентов обработки знаний и подсистема автоматизации проектирования scp-программ.

Система может использоваться тремя способами:

- Как подсистема в рамках Метасистемы OSTIS. Данный вариант использования предполагает отладку необходимых компонентов в рамках метасистемы с последующим переносом их в дочернюю ostis-систему.
- Как самостоятельная ostis-система, предназначенная исключительно для разработки и отладки компонентов решателей задач. В этом случае проектируемые компоненты отлаживаются в рамках такой системы, а затем должны быть перенесены в дочернюю ostis-систему.
- Как подсистема в рамках дочерней ostis-системы. В таком варианте отладка компонентов осуществляется непосредственно в той же системе, в которой предполагается их использование, и дополнительного переноса не требуется.

Независимо от выбранного способа использования системы, разработанные компоненты впоследствии могут быть включены в состав библиотеки многократно используемых компонентов ostis-систем.

Выделяются два принципиально разных уровня отладки решателя задач:

- отладка на уровне sc-агентов;
- отладка на уровне scp-программ.

В случае отладки на уровне sc-агентов акт выполнения каждого агента считается неделимым и не может быть прерван. При этом может выполняться отладка как атомарных sc-агентов, так и неатомарных. Инициирование того или иного агента, в том числе входящего в состав неатомарного, осуществляется путем создания соответствующих конструкций в sc-памяти, таким образом, отладка может осуществляться на разных уровнях детализации агентов, вплоть до атомарных.

Система поддержки проектирования агентов может служить основой для моделирования систем агентов, использующих другие принципы коммуникации, например, непосредственный обмен сообщениями между агентами.

Отладка на уровне scp-программ осуществляется аналогично существующим современным подходам к отладке процедурных программ и предполагает возможность установки точек останова, пошагового выполнения программы и т. д.

Система автоматизации проектирования решателей задач и, соответственно, ее sc-модель, разделяется на две более частные:

Система автоматизации проектирования решателей задач ostis-систем

⇐ базовая декомпозиция*:

- Система автоматизации проектирования агентов обработки знаний
 - ⇐ базовая декомпозиция*:
 - База знаний системы автоматизации проектирования агентов обработки знаний
 - Решатель задач системы автоматизации проектирования агентов обработки знаний
 - Пользовательский интерфейс системы автоматизации проектирования агентов обработки знаний
- Система автоматизации проектирования scp-программ
 - ⇐ базовая декомпозиция*:
 - База знаний системы автоматизации проектирования scp-программ
 - Решатель задач системы автоматизации проектирования scp-программ
 - Пользовательский интерфейс системы автоматизации проектирования scp-программ

Пункт 5.3.2.1. Семантическая модель базы знаний системы автоматизации проектирования решателей задач ostis-систем

База знаний системы автоматизации проектирования решателей задач ostis-систем включает в себя кроме Ядра и предметной области Базового языка программирования ostis-систем также описание ключевых понятий, связанных с верификацией и отладкой scp-программ.

Основные понятия, специфичные для базы знаний системы автоматизации проектирования scp-программ.

точка останова*

∈ квазибинарное отношение

Связки отношения *точки останова** связывают *scp-программу* с некоторым множеством sc-переменных, соответствующих *scp-операторам* в рамках этой программы. При генерации каждого *scp-процесса*, соответствующего этой *scp-программе*, все *scp-операторы*, соответствующие таким переменным, будут добавлены во множество *точка останова*, т. е. выполнение данного *scp-процесса* будет прерываться при достижении каждого из этих *scp-операторов*. Использование данного отношения приводит к указанию точек останова для всех *scp-процессов*, формируемых на основе заданной *scp-программы*. Для указания точки останова в рамках отдельно взятого *scp-процесса* нужный *scp-оператор* явно включается во множество *точка останова*.

точка останова

⇐ включение*:
scp-оператор

Во множество *точка останова* входят все *scp-операторы*, являющиеся точками останова в рамках какого-либо *scp-процесса*. Это означает, что в момент, когда в соответствии с переходами между *scp-операторами* по связкам отношения *последовательность действий** указанный *scp-оператор* должен стать *активным действием*, он становится *отложенным действием*, и, соответственно, выполнение всего *scp-процесса* по данной ветке приостанавливается. Чтобы продолжить выполнение, необходимо удалить указанный *scp-оператор* из множества *отложенных действий* и добавить его во множество *активных действий*.

Под *некорректностью* в *scp-программе* понимается *некорректная структура*, описывающая некорректность (не обязательно делающую невозможным выполнение соответствующих данной *scp-программе scp-процессов*), выявленную в рамках какой-либо конкретной *scp-программы*.

некорректность в scp-программе

- ⇐ *включение**:
 - некорректная структура
- ⇒ *включение**:
 - ошибка в scp-программе
- ⇒ *включение**:
 - недостижимый scp-оператор
 - потенциально бесконечный цикл

Под *ошибкой* в *scp-программе* понимается такая *некорректность в scp-программе*, которая делает невозможным успешное выполнение любого *scp-процесса*, соответствующего данной *scp-программе*, или даже создание такого *scp-процесса*.

ошибка в scp-программе

- ⇐ *разбиение**:
 - синтаксическая ошибка в scp-программе
 - ⇒ *пояснение**:

[Под *синтаксической ошибкой* в *scp-программе* понимается *ошибка в scp-программе*, в состав которой входит некоторая конструкция, не соответствующая синтаксису *scp-программы* или какой-либо ее части, например, конкретного *scp-оператора*.]
 - семантическая ошибка в scp-программе
 - ⇒ *пояснение**:

[Под *семантической ошибкой* в *scp-программе* понимается *ошибка в scp-программе*, в состав которой входит некоторая конструкция, корректная с точки зрения синтаксиса, но некорректная с семантической точки зрения, например, нарушающая логическую целостность *scp-программы*.]
- ⇒ *разбиение**:
 - ошибка в scp-программе на уровне программы
 - ошибка в scp-программе на уровне множества параметров
 - ошибка в scp-программе на уровне множества операторов
 - ошибка в scp-программе на уровне оператора
 - ошибка в scp-программе на уровне операнда

Каждая *ошибка в scp-программе на уровне программы* описывает некорректный фрагмент, выявление которого требует анализа всей *scp-программы* как единого целого, и не может быть выполнено путем анализа ее отдельных частей, например, конкретных *scp-операторов*.

ошибка в scp-программе на уровне программы

- ⇒ *включение**:
 - отсутствует scp-процесс, соответствующий данной scp-программе
 - ∈ синтаксическая ошибка в scp-программе
 - не указана декомпозиция scp-процесса, соответствующего данной scp-программе
 - ∈ синтаксическая ошибка в scp-программе

Каждая *ошибка в scp-программе на уровне множества параметров* описывает некорректный фрагмент, для выявления которого достаточно анализа параметров некоторой *scp-программы*, т. е. явным образом выделенных аргументов *действия* (*scp-процесса*), соответствующего данной *scp-программе*. К такого рода ошибкам относятся, например, неверное указание ролей этих аргументов в рамках данного *действия*.

ошибка в scp-программе на уровне множества параметров

⇒ включение*:

- не указан тип параметра *scp-программы*
∈ синтаксическая ошибка в *scp-программе*
- не указан порядковый номер параметра *scp-программы*
∈ синтаксическая ошибка в *scp-программе*

Каждая ошибка в *scp-программе на уровне множества операторов* описывает некорректный фрагмент, для выявления которого достаточно анализа множества операторов некоторой *scp-программы*, т. е. элементов декомпозиции действия (*scp-процесса*), соответствующего данной *scp-программе*. К таким ошибкам относится, например, факт отсутствия начального оператора' *scp-программы* или факт отсутствия в программе *scp-оператора завершения выполнения программы*.

ошибка в scp-программе на уровне множества операторов

⇒ включение*:

- декомпозиция *scp-процесса* не содержит ни одного элемента
∈ синтаксическая ошибка в *scp-программе*
- отсутствует *scp-оператор завершения выполнения программы*
∈ синтаксическая ошибка в *scp-программе*
- *scp-оператор*, к которому осуществляется переход, не является частью текущего *scp-процесса*
∈ синтаксическая ошибка в *scp-программе*
- не указана последовательность действий после выполнения текущего *scp-оператора*
∈ синтаксическая ошибка в *scp-программе*
- отсутствует начальный оператор *scp-программы*
∈ синтаксическая ошибка в *scp-программе*

Каждая ошибка в *scp-программе на уровне оператора* описывает некорректный фрагмент, для выявления которого достаточно анализа одного конкретного *scp-оператора*, при этом не важно, в состав какой *scp-программы* он входит. К такого рода ошибкам относится, например, факт указания количества operandов *scp-оператора*, не соответствующего спецификации соответствующего класса *scp-операторов*.

ошибка в scp-программе на уровне оператора

⇒ включение*:

- *scp-оператор* не принадлежит ни одному из атомарных классов *scp-операторов*
∈ синтаксическая ошибка в *scp-программе*
- ни один operand *scp-оператора удаления* не помечен как удаляемый sc-элемент
∈ синтаксическая ошибка в *scp-программе*
- в *scp-операторе поиска пятиэлементной конструкции* совпадает второй и четвертый operand
∈ синтаксическая ошибка в *scp-программе*
- *scp-оператор поиска* не содержит ни одного operand'a с заданным значением
∈ синтаксическая ошибка в *scp-программе*
- *scp-оператор поиска с формированием множества* не содержит ни одного operand'a с атрибутом формируемое множество
∈ синтаксическая ошибка в *scp-программе*
- атрибутом формируемое множество отмечен operand, которому соответствует operand с заданным значением
∈ синтаксическая ошибка в *scp-программе*
- количество operandов *scp-оператора* не совпадает со спецификацией
∈ синтаксическая ошибка в *scp-программе*

Каждая ошибка в *scp-программе на уровне operand'a* описывает некорректный фрагмент, для выявления которого достаточно анализа одного конкретного operand'a в рамках *scp-программы*, точнее sc-дуги принадлежности, связывающей указанный operand и соответствующий *scp-оператор*, при этом не важно, какой именно *scp-оператор*. К такого рода ошибкам относится, например, факт отсутствия ролевого отношения, указывающего на номер operand'a в рамках *scp-оператора*.

ошибка в scp-программе на уровне operand'a

⇒ включение*:

- не указан номер operand'a в рамках *scp-оператора*
∈ синтаксическая ошибка в *scp-программе*

некорректность в scp-программе*

∈ бинарное отношение

- ⇒ *первый домен**:
некорректность в *scp-программе*
- ⇒ *второй домен***:
scp-программа

Отношение *scp-программа* поиска некорректности в *scp-программе** связывает класс некорректностей в *scp-программе* и *scp-программу*, которая может использоваться для выявления соответствующей некорректности в какой-либо другой *scp-программе*.

Указанная *scp-программа* должна иметь единственный параметр, который является *in-параметром*' и, в зависимости от соответствующего класса некорректностей в *scp-программе*, обозначает:

- саму *scp-программу* в случае выявления некорректности в *scp-программе* вообще или ошибки в *scp-программе на уровне программы*;
- *scp-процесс*, являющийся ключевым sc-элементом данной *scp-программы* в случае выявления ошибки в *scp-программе на уровне множества параметров*;
- множество операторов данной *scp-программы* в случае выявления ошибки в *scp-программе на уровне множества операторов*;
- знак конкретного *scp-оператора* в случае выявления ошибки в *scp-программе на уровне оператора*;
- *sc-дугу принадлежности* в случае выявления ошибки в *scp-программе на уровне операнда*.

Если в результате верификации *scp-программы* выявлена некорректность, то формируется соответствующая структура и генерируется связка отношения *некорректность в scp-программе**

Пункт 5.3.2.2. Семантическая модель решателя задач системы автоматизации проектирования решателей задач ostis-систем

Семантическая модель решателя задач системы автоматизации проектирования агентов обработки знаний

Решатель задач системы автоматизации проектирования агентов обработки знаний

- ⇒ разбиение*:
 - {• Множество методов решателя задач системы автоматизации проектирования агентов обработки знаний
 - Машина обработки знаний системы автоматизации проектирования агентов обработки знаний
 - ⇐ декомпозиция sc-агента*:
 - {• абстрактный sc-агент верификации sc-агентов
 - ⇐ декомпозиция sc-агента*:
 - {• абстрактный sc-агент верификации спецификации sc-агента
 - абстрактный sc-агент проверки неатомарного sc-агента на непротиворечивость его спецификации спецификациям более частных sc-агентов в его составе
 - }
 - абстрактный sc-агент отладки коллективов sc-агентов
 - ⇐ декомпозиция sc-агента*:
 - {• абстрактный sc-агент поиска всех выполняющихся процессов, соответствующих заданному sc-агенту
 - абстрактный sc-агент инициализации заданного sc-агента на заданных аргументах
 - абстрактный sc-агент активации заданного sc-агента
 - абстрактный sc-агент деактивации заданного sc-агента
 - абстрактный sc-агент установки блокировки заданного типа для заданного процесса на заданный sc-элемент
 - абстрактный sc-агент снятия всех блокировок заданного процесса
 - абстрактный sc-агент снятия всех блокировок с заданного sc-элемента
 - }

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту поиска всех выполняющихся процессов, соответствующих заданному sc-агенту, абстрактному sc-агенту активации заданного sc-агента, абстрактному sc-агенту деактивации заданного sc-агента*, является знак этого sc-агента.

Класс действий, соответствующий *абстрактному sc-агенту инициирования заданного sc-агента на заданных аргументах*, имеет два аргумента. Первый аргумент является знаком инициируемого sc-агента, второй – знаком связки, в которую под соответствующими атрибутами входят sc-элементы, которые станут аргументами соответствующего действия в sc-памяти.

Класс действий, соответствующий *абстрактному sc-агенту установки блокировки заданного типа на заданный sc-элемент*, имеет три аргумента. Первый аргумент является знаком класса блокировок, второй – знаком процесса в sc-памяти, третий – sc-элементом, на который должна быть установлена блокировка.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту снятия всех блокировок заданного процесса*, является знак этого процесса в sc-памяти.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту снятия всех блокировок с заданного sc-элемента*, является знак этого sc-элемента.

Семантическая модель решателя задач системы автоматизации проектирования scp-программ

Решатель задач системы автоматизации проектирования scp-программ

⇒ разбиение*:

- {• Множество методов решателя задач системы автоматизации проектирования агентов обработки знаний
- Машина обработки знаний системы автоматизации проектирования агентов обработки знаний
 - ⇐ декомпозиция sc-агента*:
 - {• абстрактный sc-агент верификации scp-программ
 - абстрактный sc-агент отладки scp-программ
 - ⇐ декомпозиция sc-агента*:
 - {• абстрактный sc-агент запуска заданной scp-программы для заданного множества входных данных
 - абстрактный sc-агент запуска заданной scp-программы для заданного множества входных данных в режиме пошагового выполнения
 - абстрактный sc-агент поиска всех scp-операторов в рамках scp-программы
 - абстрактный sc-агент поиска всех точек останова в рамках scp-процесса
 - абстрактный sc-агент добавления точки останова в scp-программу
 - абстрактный sc-агент удаления точки останова из scp-программы
 - абстрактный sc-агент добавления точки останова в scp-процесс
 - абстрактный sc-агент удаления точки останова из scp-процесса
 - абстрактный sc-агент продолжения выполнения scp-процесса на один шаг
 - абстрактный sc-агент продолжения выполнения scp-процесса до точки останова или завершения
 - абстрактный sc-агент просмотра информации об scp-процессе
 - абстрактный sc-агент просмотра информации об scp-операторе

}

Алгоритм работы *абстрактного sc-агента верификации scp-программ* сводится к поиску некорректностей в рамках scp-программы на основе определений, соответствующих различным классам таких некорректностей, а также посредством запуска соответствующих данным классам некорректностей scp-программ поиска некорректности в scp-программе*.

Результатом работы *абстрактного sc-агента верификации scp-программ* является формирование в sc-памяти структур, описывающих некорректности в исследуемой scp-программе, если таковые имеются.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту верификации scp-программ*, является знак верифицируемой scp-программы.

Классы действий, соответствующие *абстрактному sc-агенту запуска заданной scp-программы для заданного множества входных данных* и *абстрактному sc-агенту запуска заданной scp-программы для заданного множества входных данных в режиме пошагового выполнения*, имеют два аргумента. Первый аргумент является знаком запускаемой scp-программы, второй – знаком связки, в которую под соответствующими атрибутами входят sc-элементы, которые станут аргументами соответствующего scp-процесса.

В режиме пошагового выполнения предполагается, что на каждом шаге инициируется выполнение всех scp-операторов в рамках заданного scp-процесса, для которых предыдущий scp-оператор стал прошлой сущностью (выполнился). В свою очередь, шаг заканчивается, когда все инициированные таким образом операторы закончат

выполнение. Таким образом, в случае, если в рамках scp-программы есть параллельные ветви, то на одном шаге могут одновременно инициироваться два и более scp-оператора.

Классы действий, соответствующие *абстрактному sc-агенту добавления точки останова в scp-программу*, *абстрактному sc-агенту удаления точки останова из scp-программы*, *абстрактному sc-агенту добавления точки останова в scp-процесс* и *абстрактному sc-агенту удаления точки останова из scp-процесса*, имеют два аргумента. Первый аргумент является знаком scp-программы или scp-процесса соответственно, второй – знаком scp-оператора, входящего в состав этой scp-программы или scp-процесса.

Единственным аргументом классов действий, соответствующих *абстрактному sc-агенту поиска всех точек останова в рамках scp-процесса*, *абстрактному sc-агенту продолжения выполнения scp-процесса на один шаг*, *абстрактному sc-агенту продолжения выполнения scp-процесса до точки останова или завершения* и *абстрактному sc-агенту просмотра информации об scp-процессе*, является знак scp-процесса, с которым будет выполнено соответствующее действие.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту поиска всех scp-операторов в рамках scp-программы*, является знак этой scp-программы.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту просмотра информации об scp-операторе*, является знак scp-оператора, входящего в состав некоторого scp-процесса. Результатом работы данного агента является структура, описывающая значения операндов данного scp-оператора, его атомарный тип и другую служебную информацию, полезную для разработчика.

Пункт 5.3.2.3. Семантическая модель пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем

Пользовательский интерфейс системы автоматизации проектирования решателей задач ostis-систем представлен набором интерфейсных команд, позволяющих пользователю инициировать деятельность нужного агента, входящего в состав этой системы.

команда пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем

⇐ разбиение*:

- {• команда пользовательского интерфейса системы автоматизации проектирования агентов обработки знаний
- команда пользовательского интерфейса системы автоматизации проектирования программ языка SCP

}

команда пользовательского интерфейса системы автоматизации проектирования агентов обработки знаний

⇐ разбиение*:

- {• команда верификации sc-агентов

⇐ разбиение*:

 - {• команда верификации спецификации sc-агента
 - команда верификации неатомарного sc-агента на непротиворечивость его спецификации спецификациям более частных sc-агентов в его составе

}

- команда отладки коллективов sc-агентов

⇐ разбиение*:

- {• команда поиска всех выполняющихся процессов, соответствующих заданному sc-агенту
- команда инициализации заданного sc-агента на заданных аргументах
- команда активации заданного sc-агента
- команда деактивации заданного sc-агента
- команда установки блокировки заданного типа для заданного процесса на заданный sc-элемент
- команда снятия всех блокировок заданного процесса
- команда снятия всех блокировок с заданного sc-элемента

}

}

команда пользовательского интерфейса системы автоматизации проектирования scp-программ

⇐ разбиение*:

- {• команда верификации *scp*-программ
- команда отладки *scp*-программ

⇐ разбиение*:

 - {• команда запуска заданной *scp*-программы для заданного множества входных данных
 - команда запуска заданной *scp*-программы для заданного множества входных данных в режиме пошагового выполнения
 - команда поиска всех *scp*-операторов в рамках *scp*-программы
 - команда поиска всех точек останова в рамках *scp*-процесса
 - команда добавления точки останова в *scp*-программу
 - команда удаления точки останова из *scp*-программы
 - команда добавления точки останова в *scp*-процесс
 - команда удаления точки останова из *scp*-процесса
 - команда продолжения выполнения *scp*-процесса на один шаг
 - команда продолжения выполнения *scp*-процесса до точки останова или завершения
 - команда просмотра информации об *scp*-процессе
 - команда просмотра информации об *scp*-операторе

}

}

§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем

Библиотека многократно используемых компонентов решателей задач является важнейшим фрагментом Метасистемы OSTIS, обеспечивающим надежность проектируемых решателей задач и повышение скорости их разработки.

Библиотека включает:

- множество компонентов решателей задач;
- средства спецификации компонентов решателей задач;
- средства поиска компонентов решателей задач на основе их спецификации.

Если многократно используемый компонент решателей задач является платформенно-зависимым многократно используемым компонентом ostis-системы, то его интеграция производится в соответствии с инструкцией, предоставляемой разработчиком в зависимости от платформы, как и для любого компонента такого рода. В противном случае процесс интеграции можно конкретизировать в зависимости от подклассов данного типа компонентов.

Рассмотрим классификацию многократно используемых компонентов решателей задач ostis-систем.

многократно используемый компонент решателей задач

- ▷ программа
- ▷ пакет программ
- ▷ абстрактный sc-агент
- ▷ решатель задач ostis-системы

⇒ примечание*:

[Целевые решатели задач могут быть многократно используемыми компонентами в случае разработки интеллектуальных систем, назначение которых совпадает.]

метод

- := [программа]
- ▷ программа на основе нейросетевых моделей
- ▷ программа на основе генетических алгоритмов
- ▷ императивная программа
 - ▷ процедурная программа
 - ▷ объектно-ориентированная программа
- ▷ декларативная программа
 - ▷ логическая программа
 - ▷ функциональная программа
- ▷ программа sc-агента

абстрактный sc-агент

- ⇒ разбиение*:
- {• неатомарный абстрактный sc-агент

- атомарный абстрактный sc-агент
 - }
- ⇒ разбиение*:
- {• внутренний абстрактный sc-агент
 - эффекторный абстрактный sc-агент
 - рецепторный абстрактный sc-агент
- ⇒ разбиение*:
- {• абстрактный sc-агент, не реализуемый на Языке SCP
 - абстрактный sc-агент, реализуемый на Языке SCP
- ⇒ разбиение*:
- {• абстрактный sc-агент интерпретации scp-программ
 - абстрактный программный sc-агент
 - абстрактный sc-метаагент
- ⇒ разбиение*:
- {• платформенно-зависимый абстрактный sc-агент
 - ▷ абстрактный sc-агент, не реализуемый на Языке SCP
 - платформенно-независимый абстрактный sc-агент
- ⇒ разбиение*:
- {• абстрактный sc-агент информационного поиска
 - абстрактный sc-агент погружения интегрируемого знания в базу знаний
 - абстрактный sc-агент выравнивания онтологии интегрируемого знания с основной онтологией текущего состояния базы знаний
 - абстрактный sc-агент планирования решения явно сформулированных задач
 - абстрактный sc-агент логического вывода
 - абстрактный sc-агент верификации базы знаний
 - абстрактный sc-агент редактирования базы знаний
 - абстрактный sc-агент автоматизации деятельности разработчиков базы знаний

Классификацию и описание решателей задач ostis-систем можно найти в [§ 3.2.6. Решатели задач ostis-систем](#);

Под *многократно используемым абстрактным sc-агентом* подразумевается компонент, соответствующий некоторому *абстрактному sc-агенту*, который может быть использован в других системах, возможно, в составе более сложных *неатомарных абстрактных sc-агентов*. Указанный абстрактный sc-агент входит в соответствующую компоненту *структурную под атрибутом ключевой sc-элемент*^{*}. Каждый *многократно используемый абстрактный sc-агент* должен содержать всю информацию, необходимую для функционирования соответствующего sc-агента в дочерней ostis-системе.

Таким образом, соответствующая *многократно используемому абстрактному sc-агенту структура* формируется следующим образом:

- В нее включается *sc-узел*, обозначающий соответствующий *абстрактный sc-агент*, и вся его спецификация, т. е., как минимум, указание *ключевых sc-элементов sc-агента**, *условия инициирования и результат**, *первичного условия инициирования**, *sc-описание поведения sc-агента* и класса решаемых им задач;
- В случае, если входящий в *многократно используемый sc-агент* *абстрактный sc-агент* рассматривается как *неатомарный абстрактный sc-агент*, то *многократно используемый sc-агент* будет содержать *sc-узлы*, обозначающие все более частные *абстрактные sc-агенты*, а также все их спецификации;
- Для каждого *атомарного абстрактного sc-агента*, знак которого вошел в *многократно используемый абстрактный sc-агент*, необходимо выбрать вариант его реализации (т. е. элемент класса *платформенно-независимый абстрактный sc-агент* или *платформенно-зависимый абстрактный sc-агент*, связанный с исходным *атомарным абстрактным sc-агентом* связкой отношения *включение**) и включить в *многократно используемый абстрактный sc-агент* *sc-узел*, обозначающий указанную реализацию, а также знаки всех программ, входящие во множество, связанное с указанной реализацией отношением *программа sc-агента**. Выбранная реализация включается в *многократно используемый абстрактный sc-агент* под атрибутом *ключевой sc-элемент*^{*};
- В *многократно используемый абстрактный sc-агент* включаются также все связки отношений, связывающие уже включенные в его состав sc-элементы, а также сами знаки этих отношений (например, *включение**, *программа sc-агента** и т. д.).

После того как *многократно используемый абстрактный sc-агент* был скопирован в дочернюю ostis-систему, необходимо сгенерировать *sc-узел*, обозначающий конкретный *sc-агент*, работающий в данной системе и принад-

лежащий выбранной реализации *абстрактного sc-агента*, и добавить его во множество *активных sc-агентов* при необходимости.

Под *многократно используемой программой* подразумевается компонент, соответствующий программе, записанной на произвольном языке программирования, которая ориентирована на обработку *структур*, хранящихся в памяти *ostis*-системы. Приоритетным в данном случае является использование *scp-программ* по причине их платформенной независимости, за исключением случаев проектирования некоторых компонентов интерфейса, когда полная платформенная независимость невозможна (например, при проектировании *эффекторных sc-агентов* и *рецепторных sc-агентов*).

Также каждую *scp-программу*, попавшую в *дочернюю ostis-систему* при копировании *многократно используемого компонента решателя задач*, необходимо добавить во множество *корректных scp-программ* (корректность верифицируется при попадании в библиотеку компонентов).

Для удобства работы с библиотекой многократно используемых компонентов необходимы такие же средства автоматизации поиска компонентов на основе заданной спецификации, представляющие собой неатомарный sc-агент, который декомпозируется на более частные.

Ниже представлена структура такого агента:

Средства автоматизации библиотеки многократно используемых компонентов решателей задач

⇐ *декомпозиция sc-агента**:

- {• *абстрактный sc-агент формирования неатомарного компонента из атомарных*
 - *абстрактный sc-агент поиска всех неатомарных компонентов, частью которых является заданный атомарный компонент*
 - *абстрактный sc-агент поиска всех сопутствующих компонентов*
 - *абстрактный sc-агент поиска sc-агента по условию иницирования*
 - *абстрактный sc-агент поиска sc-агента по результату работы*
 - *абстрактный sc-агент поиска scp-программы по входным/выходным параметрам*
 - *абстрактный sc-агент поиска sc-агентов, для которых элементы заданного множества являются ключевыми sc-элементами*
- }

Под *неатомарным компонентом решателей задач* понимается такой компонент, в составе которого можно выделить другие компоненты, которые могут использоваться самостоятельно, отдельно от исходного компонента. Чаще всего в роли таких неатомарных компонентов выступают неатомарные sc-агенты, в составе которых могут быть выделены самодостаточные sc-агенты, которые могут быть использованы отдельно от исходного неатомарного, или scp-программы, которые являются общими для нескольких агентов и могут быть использованы не только в составе неатомарного sc-агента. Таким образом, задачей *абстрактного sc-агента формирования неатомарного компонента из атомарных* является формирование структуры, содержащей в себе полный sc-текст неатомарного компонента, включая спецификации всех sc-агентов в его составе, а также тексты всех необходимых scp-программ. Формирование такой структуры необходимо для того, чтобы упростить процесс копирования указанного компонента в другие *ostis*-системы.

Под *сопутствующим компонентом* понимается компонент, который часто используется в *ostis*-системе одновременно с некоторым другим компонентом. Такая связь между компонентами задается явно при помощи отношения *сопутствующий компонент**. Примерами таких компонентов являются некоторый sc-агент и команда пользовательского интерфейса, позволяющая пользователю инициировать выполнение указанного агента с заданными аргументами. При этом sc-агент будет функционировать и без наличия в системе такой команды, однако для его инициирования придется сформировать соответствующую конструкцию в sc-памяти вручную.

абстрактный sc-агент поиска sc-агентов, для которых элементы заданного множества являются ключевыми sc-элементами играет важную роль при внесении изменений в базу знаний, в частности, при переопределении каких-либо понятий. Указанный sc-агент позволяет выявить те sc-агенты, для которых могут потребоваться изменения в алгоритме работы в связи с изменением семантической трактовки каких-либо понятий.

Рассмотрим отношения необходимые для спецификации многократно используемого компонента решателя задач, его поиска и установки в дочернюю *ostis*-систему.

отношение, специфицирующее многократно используемый компонент решателей задач ostis-систем

С *отношение, специфицирующее многократно используемый компонент ostis-систем*

Э *первичное условие иницирования**

⇒ *пояснение*:

[Связки отношения *первичное условие иницирования** связывают между собой sc-узел, обозначающий *абстрактный sc-агент* и бинарную ориентированную пару, описывающую первичное условие инициро-

вания данного абстрактного sc-агента, т.е. такой ситуации в sc-памяти, которая побуждает sc-агента перейти в активное состояние и начать проверку наличия своего полного условия инициирования.

Первым компонентом данной ориентированной пары является знак некоторого подмножества понятия события, например событие добавления выходящей sc-дуги, т.е. по сути конкретный тип события в sc-памяти.

Вторым компонентом данной ориентированной пары является произвольный в общем случае sc-элемент, с которым непосредственно связан указанный тип события в sc-памяти, т.е., например, sc-элемент, из которого выходит либо в который входит генерируемая либо удаляемая sc-дуга, либо sc-ссылка, содержимое которой было изменено.

После того, как в sc-памяти происходит некоторое событие, активизируются все активные sc-агенты, первичное условие инициирования* которых соответствует произошедшему событию.]

- ⇒ *первый домен*:*
абстрактный sc-агент
- ⇒ *второй домен*:*
бинарная ориентированная пара

∃ *условие инициирования и результат**

- ⇒ *пояснение*:*

[Связки отношения условие инициирования и результат* связывают между собой sc-узел, обозначающий абстрактный sc-агент и бинарную ориентированную пару, связывающую условие инициирования данного абстрактного sc-агента и результаты выполнения данного экземпляров данного sc-агента в какой-либо конкретной системе.

Указанную ориентированную пару можно рассматривать как логическую связку импликации, при этом на sc-переменные, присутствующие в обеих частях связки, неявно накладывается квантор всеобщности, на sc-переменные, присутствующие либо только в посылке, либо только в заключении неявно накладывается квантор существования.

Первым компонентом указанной ориентированной пары является логическая формула, описывающая условие инициирования описываемого абстрактного sc-агента, то есть конструкции, наличие которой в sc-памяти побуждает sc-агент начать работу по изменению состояния sc-памяти. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

Вторым компонентом указанной ориентированной пары является логическая формула, описывающая возможные результаты выполнения описываемого абстрактного sc-агента, то есть описание произведенных им изменений состояния sc-памяти. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.]

- ⇒ *первый домен*:*
абстрактный sc-агент
- ⇒ *второй домен*:*
бинарная ориентированная пара

∃ *эквивалентный компонент**

∃ *неориентированное отношение*

- ⇒ *пояснение*:*

[Бинарное отношение связывающее функционально эквивалентные многократно используемые компоненты решателей задач.]

- ⇒ *первый домен*:*
многократно используемый компонент решателей задач
- ⇒ *второй домен*:*
многократно используемый компонент решателей задач

§ 5.3.4. Многократно используемые внутренние агенты ostis-систем

внутренний абстрактный sc-агент

- ⇒ *пояснение*:*

[Каждый *внутренний абстрактный sc-агент* обозначает класс sc-агентов, которые реагируют на события в sc-памяти и осуществляют преобразования исключительно в рамках этой же sc-памяти.]

Глава 5.4.

Методика и средства компонентного проектирования интерфейсов ostis-систем

⇒ автор*:

- Садовский М.Е.
- Жмырко А.В.

⇒ аннотация*:

[Проектирование интерфейса – это один из наиболее важных этапов разработки любой системы. Пользователь при обращении с интерфейсом должен представить себе, какая информация о выполняемой задаче у него существует, и в каком состоянии находятся средства, с помощью которых он будет решать данную задачу. Эффективность работы пользователя и его интерес обеспечивает правильно сформулированная методика разработки и проектирования пользовательского интерфейса.

В настоящее время организация взаимодействия пользователя с компьютерной системой лежит парадигма **грамотного пользователя**, который знает, как управлять системой и несёт полную ответственность за качество взаимодействия с ней. Многообразие форм и видов интерфейсов приводит к необходимости пользователя адаптироваться к каждой конкретной системе, обучаться принципам взаимодействия с ней для решения необходимых ему задач.

Проектирование пользовательских интерфейсов включает в себя ряд последовательных этапов. В рамках главы будут рассмотрены этапы проектирования традиционных пользовательских интерфейсов и этапы проектирования адаптивных интеллектуальных мультимодальных пользовательских интерфейсов.]

§ 5.4.1. Действия и методики проектирования интерфейсов ostis-систем

Среди существующих методик проектирования адаптивных интеллектуальных мультимодальных пользовательских интерфейсов можно выделить методики, предложенные в *Ehlert P..IntellUIIntroS-2003bk* и *Kong J..Desig oHCAMI-2011art*.

В рамках работы *Ehlert P..IntellUIIntroS-2003bk* выделяется 4 основных этапа проектирования:

- анализ;
- разработка интерфейса;
- оценка интерфейса;
- доработка и усовершенствование.

Этап анализа является, вероятно, самой важной фазой в любом процессе проектирования, но тем более в проектировании интерфейсов ostis-систем. В процессе проектирования обычного неинтеллектуального интерфейса необходимо проанализировать, кто является обычным пользователем, какие задачи интерфейс должен поддерживать.

В пользовательском интерфейсе часто нет среднего пользователя. В идеале, пользовательский интерфейс должен быть способен адаптироваться к любому пользователю в любой среде. Поэтому используемая техника адаптации должна быть разработана таким образом, чтобы она могла поддерживать все типы пользователей.

Этап *анализа* включает выполнение четырех взаимосвязанных видов анализа:

- функциональный анализ;
- анализ данных;
- анализ пользователей;
- анализ среды.

В рамках *функционального анализа* необходимо дать ответ на вопрос: "каковы основные функции системы?". В рамках *анализа данных* необходимо определить значение и структуру данных, используемых в приложении. В рамках *анализа пользователей* необходимо выделить типы пользователей и их возможности в интеллектуальном и когнитивном плане. В рамках *анализа среды* необходимо определить требования, предъявляемые к среде, в которой будет работать система.

Результатом данного этапа является спецификация целей и информационных потребностей пользователя, а также спецификация функций и информации, которые требуются системе.

Разработка интерфейса включает следующие шаги:

- *создание модели интерфейса* в соответствии с этапом анализа;
- реализация модели интерфейса.

Результатом данного этапа является пользовательский интерфейс, который, по мнению разработчика, удовлетворяет требованиям пользователей и соответствует требованиям, сформулированным на этапе анализа.

Оценка интерфейса предполагает, что:

- требования, которые были сформулированы на этапе анализа, должны быть удовлетворены;
- эффективность модели интерфейса должна быть исследована.

На этапе *оценки интерфейса* необходимо вернуться к требованиям *этапа анализа*. Требования, которые были сформулированы на *этапе анализа*, должны быть выполнены, а также должна быть исследована эффективность модели интерфейса. Чтобы определить эту эффективность, необходимо определить критерии эффективности.

Очень важным, но субъективным критерием является удовлетворенность пользователя. Поскольку пользователь должен работать с интерфейсом, он имеет право голоса в вопросе о том, удобно ли работать с интерфейсом и т.п.

Критериями эффективности могут выступать различные показатели, такие как:

- количество ошибок;
- время выполнения задачи;
- отношение пользователя к интерфейсу;
- т.д.

Доработка и усовершенствование осуществляется на основе проблем, выявленых на этапе оценки. В рамках данного этапа вносится ряд улучшений в модель интерфейса. Затем начинается новый цикл проектирования. Этот итеративный процесс будет продолжаться до тех пор, пока результат оценки не будет удовлетворять обозначенным требованиям.

Методика, предложенная в *Kong J..Desig oHCAMI-2011art* включает 6 этапов:

- моделирование пользовательского интерфейса (описание абстрактного пользовательского интерфейса);
- проектирование пользовательского интерфейса по умолчанию (стандартная версия, конкретный пользовательский интерфейс);
- разработка пользовательского интерфейса (расширение или замена пользовательского интерфейса по умолчанию) - этот шаг опускается, когда система генерирует пользовательский интерфейс по умолчанию автоматически;
- создание контекста использования (идентификация и создание контекста использования - модели пользователя, модель устройства и модель среды/платформы);
- адаптация пользовательского интерфейса - автоматически - (адаптация пользовательского интерфейса во время выполнения для соответствия конкретного контекста использования);
- кастомизация пользовательского интерфейса - настройка пользовательского интерфейса самим пользователем (адаптируемость).

На основе рассмотренных методик проектирования интерфейсов можно выделить следующие общие этапы:

- анализ контекста использования и задач пользователей;
- проектирование и разработка интерфейса;
- оценка качества спроектированного интерфейса.

Среди недостатков предложенных подходов можно выделить:

- знания по каждому этапу проектирования находятся у разных специалистов в неформализованном неуничтоженном виде;
- отсутствие этапа формализованного документирования этапов проектирования приводит в дальнейшем в необходимости создания отдельных help-систем для пользователей, разработчиков и т.д.

Предлагается использовать онтологический подход на основе семантической модели в процессе проектирования и реализации адаптивного интеллектуального мультимодального пользовательского интерфейса. Такой интерфейс предлагается рассматривать как специализированную подсистему для решения интерфейсных задач пользователя, состоящую из базы знаний и решателя интерфейсных задач.

Описание модели базы знаний и решателя предлагается осуществлять на основе универсального унифицированного языка представления знаний, что обеспечит совместимость между этими компонентами.

Архитектура интерфейса такой системы была рассмотрена на рисунке "Архитектура интеллектуального интерфейса".

Архитектура интеллектуального интерфейса

Таким образом, предлагаемая методика проектирования интерфейсов ostis-систем будет включать:

- анализ пользователя, его задач и целей, а также контекста использования;
- анализ требований к пользовательскому интерфейсу;
- проектирование пользовательского интерфейса по умолчанию;



- разработка пользовательского интерфейса;
- анализ пользовательского интерфейса и его адаптации.

Поскольку знания о конкретном этапе обычно находятся у разных экспертов, особенностью предлагаемого подхода является обязательное формализованное документирование знаний в унифицированном виде и применение на каждом из этапов компонентного подхода.

Для применения компонентного подхода предлагается использовать *библиотеку многократно используемых компонентов* базы знаний, решателя задач и интерфейса.

Анализ пользователя, его задач и целей, а также контекста использования

Результаты первого этапа, такие как: модель конкретного пользователя, его потребности и контекст использования системы (устройство, окружающая среда) должны быть формализованы в рамках соответствующих онтологий базы знаний интерфейса. При этом в процессе формализации по необходимости должны быть переиспользованы компоненты базы знаний из библиотеки многократно используемых компонентов, а новые компоненты могут пополнить эту же библиотеку.

Анализ требований к пользовательскому интерфейсу

Результатом второго этапа являются конечные требования к интерфейсу, которые должны быть сформулированы относительно модели пользователя и его цели, а также относительно контекста использования.

Результаты должны быть также формализованы, а в процессе выполнения могут быть использованы существующие компоненты базы знаний из библиотеки многократно используемых компонентов.

Проектирование пользовательского интерфейса по умолчанию

В соответствии с требованиями к пользовательскому интерфейсу, строится модель адаптивного интеллектуального мультимодального пользовательского интерфейса, которая является результатом третьего этапа.

Такая модель будет включать в себя формализованную модель базы знаний и решателя задач.

При проектировании могут быть использованы компоненты интерфейса, базы знаний и решателя задач. Компоненты будут записаны в унифицированном виде, что позволит обеспечить их автоматическую совместимость.

Разработка пользовательского интерфейса

Результатом четвертого этапа является реализация спроектированного пользовательского интерфейса. В данном случае следует использовать готовые компоненты интерфейса из библиотеки многократно используемых компонентов интерфейса.

Анализ пользовательского интерфейса и его адаптации

На данном этапе используются готовые компоненты решателя задач.

Таким образом будет сформирована база знаний проектируемого интерфейса, которая автоматически может быть использована в качестве help-системы для пользователей, разработчиков и т.д.

§ 5.4.2. Логико-семантическая модель ostis-системы автоматизации проектирования интерфейсов ostis-систем

§ 5.4.3. Многократно используемые компоненты интерфейсов ostis-систем

Часть 6.

Платформы реализации интеллектуальных компьютерных систем нового поколения

Описание к главе

Глава 6.1.

Универсальная модель интерпретации логико-семантических моделей ostis-систем

⇒ *автор**:

- Шункевич Д.В.

⇒ *аннотация**:

[В главе рассматривается подход к решению проблемы платформенной независимости компьютерных систем, предполагающий унификацию принципов реализации таких систем и обеспечения их семантической совместимости на основе Технологии OSTIS. Приводится формализованная система понятий, определяющая принципы реализации данного подхода]

⇒ *подраздел**:

- § 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению
- § 6.1.2. Методы и средства реализации ostis-систем
- § 6.1.3. Уточнение понятия ostis-платформы

⇒ *ключевое понятие**:

- sc-машина
- ostis-платформа
- базовая ostis-платформа
- расширенная ostis-платформа
- специализированная ostis-платформа
- минимальная конфигурация ostis-системы
- программный вариант ostis-платформы
- ассоциативный семантический компьютер

⇒ *библиографическая ссылка**:

- Колесников А.В. Гибрид ИСТиР-2001кн
- Комарцова Л.Г. Нейро-2004кн
- Hopcroft J.E.. Intro iATLaC-2000арт
- Godfrey M.D.. iCompu avNPi-1993арт
- USBAccel-2022ел

§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению

В общем случае разработка любой искусственной системы, в частности, интеллектуальной компьютерной системы, предполагает выполнение двух этапов:

- этапа проектирования, то есть построения формальной модели системы, достаточной для понимания принципов ее устройства и выполнения последующего этапа ее реализации;
- этапа реализации, то есть непосредственно воплощения разработанной модели с использованием конкретных средств (инструментов, материалов, комплектующих и т.д.). В случае компьютерных систем выполнение данного этапа обычно предполагает выбор конкретных языков программирования, библиотек, сторонних средств, таких как с.у.б.д. и различные сервисы и т.д., а также собственно программирование и отладку системы с использованием выбранных средств.

Для каждого из указанных этапов могут существовать свои методики, а также средства автоматизации соответствующих процессов.

Если этап проектирования компьютерной системы как правило требует участия высококвалифицированных специалистов и экспертов в предметных областях, в которых осуществляется автоматизация, то этап реализации, с одной стороны, как правило является более простым (при условии качественного выполнения этапа проектирования), а с другой стороны требует значительных ресурсов. Одной из причин этого является необходимость работы компьютерной системы на различных plataформах (устройствах), каждое из которых в общем случае может

иметь свои особенности и ограничения, которые необходимо учитывать на этапе реализации. Решением данной проблемы является обеспечение платформенной независимости (или кроссплатформенности) разрабатываемых компьютерных систем.

Сама по себе идея обеспечения платформенной независимости давно и широко используется в современных компьютерных системах. Данная проблема, как правило, рассматривается на двух уровнях:

- проблема обеспечения возможности работы программной системы в разных операционных системах;
- проблема обеспечения совместимости операционной системы с различными аппаратными архитектурами. Для решения этой проблемы могут существовать разные сборки ядра операционной системы для разных аппаратных архитектур, как это делается для операционных систем семейства Linux. При этом следует отметить, что речь в подавляющем большинстве случаев идет не о принципиально разных архитектурах, а о вариантах реализации базовой архитектуры фон Неймана.

В случае, когда разрабатываемая компьютерная система проектируется на более низком уровне, чем операционная система как таковая (например, при программировании контроллеров управления различными устройствами), проблема обеспечения платформенной независимости значительно усугубляется и чаще всего может быть решена только для набора аппаратных средств определенного класса, для которого стандартизируется интерфейс доступа, то есть, по сути, набор низкоуровневых команд обработки информации.

Таким образом, можно сказать, что большее внимание при проектировании современных компьютерных систем на данный момент уделяется первому из перечисленных уровней платформенной независимости, то есть обеспечению работы программной системы на разных операционных системах. Это может достигаться разными путями:

- Использование кроссплатформенных языков программирования, которые, в свою очередь, можно разделить на "полностью" интерпретируемые языки (Python, JavaScript и языки на его основе, PHP и другие) и языки, использующие компиляцию в сохраняющий независимость от платформы низкоуровневый байт-код, с его возможной последующей компиляцией в машинный код непосредственно в процессе исполнения (Just-in-time компиляция или JIT-компиляция). К языкам второго класса относятся, например, Java и C#. Реализация такого подхода требует установки на целевой компьютер с операционной системой интерпретатора соответствующего языка программирования или байт-кода.

Несмотря на популярность такой вариант имеет ряд ограничений:

- в среднем производительность интерпретируемых программ ниже, чем компилируемых. Одним из подходов к решению данной проблемы является JIT-компиляция;
- строго говоря, кроссплатформенность при таком варианте обеспечивается не для всех операционных систем, а для класса операционных систем и соответствующего класса устройств, например, операционных систем, предназначенных для персональных компьютеров. Так, например, приложение, написанное на языке Java для персонального компьютера не может быть напрямую перенесено на мобильное устройство, поскольку при разработке мобильных приложений учитываются другие принципы работы пользователя с интерфейсом системы, отсутствие многооконности и многое другое.
- Реализация системы в виде web-приложения, работа с которым осуществляется через web-браузер и интерфейс которого, таким образом, реализуется на базе общепринятых стандартов Всемирной паутины (HTML, CSS, JavaScript и языки и библиотеки на его основе). Такой вариант обеспечивает возможность работы с приложением с любого устройства, имеющего web-браузер, в том числе, мобильного. К недостаткам такого варианта относятся:
 - как правило, высокая требовательность к производительности конечного устройства. Современный web-браузер является одним из самых ресурсоемких приложений почти на любом устройстве;
 - остается за кадром проблема обеспечения платформенной-независимости серверной части web-приложения, которая должна решаться каким-то другим способом;
 - несмотря на стандартизацию, разработчикам часто приходится учитывать особенности конкретных web-браузеров и тестировать работоспособность приложений для каждого из них;
 - потенциально одним и тем же web-приложением можно пользоваться на любом устройстве, однако для обеспечения удобства и наглядности как правило приходится разрабатывать отдельные версии web-приложения, адаптированные под разные устройства, имеющие, например разные размеры экрана.
- Виртуализация (контейнеризация, эмуляция). Перечисленные термины не являются полностью синонимичными, но в целом обозначают подход, при котором в рамках операционной системы создается некоторое изолированное локальное окружение (виртуальная машина, контейнер, среда эмуляции), содержащее все необходимые для работы приложения настройки и гарантирующее его работу на любых операционных системах и устройствах, где может интерпретироваться соответствующая виртуальная машина или контейнер. Соответственно, запуск таких окружений требует установки на конечное устройство соответствующего интерпретатора или эмулятора.

Данный подход бурно развивается и набирает популярность в настоящее время, поскольку позволяет решить не только проблему кроссплатформенности, но и избавить потребителя от установки большого числа зависимостей и выполнении настройки приложения на конечном устройстве.

Среди популярных средств реализующих данный подход можно указать средства виртуализации (VirtualBox, DosBox, VMWare Workstation), контейнеризации (Docker), эмуляции приложений Android для настольных операционных систем (Genymotion, Bluestacks, Anbox) и многие другие.

К недостаткам такого подхода можно отнести его ресурсоемкость и снижение производительности, а также ограниченность применения (как правило, соответствующие интерпретаторы разрабатываются только для наиболее популярных и востребованных операционных систем). Кроме того, возникает проблема следующего уровня, связанная уже с зависимостью от выбранного средства виртуализации (контейнеризации).

Важно также отметить, что даже для интерпретируемых языков программирования существует проблема зависимости приложения от используемого набора библиотек и фреймворков. Так, при разработке интерфейса web-приложения могут использоваться популярные фреймворки AngularJS и ReactJS, при этом после выбора одного из них быстрый перевод приложения на другой фреймворк невозможен.

Таким образом, можно сделать вывод о том, что проблеме обеспечения платформенной независимости в современных компьютерных системах уделяется достаточно много внимания, однако в полной мере она не решена. В то же время, существует большое количество успешных частных решений, которые, однако, обладают серьезными ограничениями, связанными, в первую очередь, с отсутствием унификации современных подходов к разработке компьютерных систем.

Еще более актуальной проблема обеспечения платформенной независимости становится в контексте разработки интеллектуальных компьютерных систем. Это обусловлено следующими особенностями таких систем:

- значительно более сложная по сравнению с традиционными компьютерными системами структура представляемой информации и, соответственно, многообразие форм ее представления, хранение и обработка которых на разных платформах могут быть организованы совершенно по-разному;
- высокие требования к производительности для некоторых классов систем, в частности, систем, использующих машинное обучение, что приводит к созданию специализированных аппаратных архитектур, таких как, например, нейрокомпьютеры (*Комарцова Л.Г. Нейро-2004kn, USBAccel-2022el*);
- многообразие моделей решения задач, которые в общем случае реализуются по-разному в разных системах;
- актуальность разработки гибридных интеллектуальных систем *Колесников А.В. ГибриИСТиTP-2001kn*, в рамках которых интегрируются различные виды знаний и различные модели решения задач. Ввиду отсутствия на настоящий момент общепринятой унифицированной основы для их интеграции такие системы создаются в основном с ориентацией на какую-то определенную платформу и трудно переносимы на другие платформы.

Таким образом, можно сказать, что проблема обеспечения платформенной независимости для интеллектуальных систем обусловлена во многом отсутствием семантической совместимости компонентов таких систем между собой, что, в свою очередь, создает препятствия даже для реализации подходов к обеспечению платформенной независимости, реализуемых в процессе разработки традиционных компьютерных систем. То есть, для решения проблемы обеспечения платформенной независимости интеллектуальных систем, требуется вначале обеспечить семантическую совместимость компонентов таких систем между собой, что, в свою очередь, предполагает:

- унификацию представления различного рода информации, хранимой в базах знаний таких систем;
- унификацию базовых моделей обработки информации, хранимой в базах знаний таких систем, то есть выделение универсального низкоуровневого языка программирования, позволяющего осуществлять обработку информации, хранимой в унифицированном виде;
- унификацию принципов реализации различных моделей решения задач и, как следствие, возможность их интеграции в рамках гибридных интеллектуальных систем;
- унификацию принципов разработки интерфейсов компьютерных систем, которая бы позволила реализовать в рамках одной интеллектуальной системы возможность взаимодействия с другими системами и пользователями таких систем на разных внешних языках, включая естественные языки.

Указанные принципы реализуются в рамках *Технологии OSTIS*, которая, таким образом, может стать основой для решения проблемы обеспечения семантической совместимости компонентов интеллектуальных компьютерных систем в целом и обеспечения платформенной независимости таких систем. С одной стороны, принципы, лежащие в основе *Технологии OSTIS* (см. Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения) обеспечивают принципиальную возможность реализации платформенной независимости компьютерных систем, разрабатываемых на ее основе (*ostis-систем*). С другой стороны, благодаря своей универсальности *Технология OSTIS* позволяет преобразовать любую современную компьютерную систему в *ostis-систему*, которая будет функционально эквивалентна исходной компьютерной системе, но при этом будет обладать всеми перечисленными выше свойствами, создающими предпосылки для решения проблемы платформенной независимости.

Для реализации данного подхода в рамках *Технологии OSTIS* требуется разработать семейство онтологий, обеспечивающих уточнение таких понятий, как *ostis-система*, *ostis-платформа*, их структуры, типологии и предъявляемых к ним требований. Рассмотрению указанных онтологий и посвящена данная глава.

Что касается обозначенной выше проблемы зависимости компьютерных систем от конкретных фреймиков, то аналогичная проблема может возникнуть и при дальнейшем развитии *Технологии OSTIS*, в ситуации, когда соответствующие библиотеки будут содержать достаточно большое количество функционально эквивалентных компонентов. Однако, благодаря принципам, лежащим в основе *Технологии OSTIS*, в частности, смысловому представлению информации и семантической совместности компонентов, данная проблема будет значительно менее острой, поскольку:

- число функционально эквивалентных компонентов будет значительно ниже, чем в традиционных информационных технологиях, нет необходимости создавать синтаксически разные компоненты, отличия будут только на семантическом уровне;
- сами по себе компоненты будут являться более универсальными, то есть смогут быть использованы в значительно большем количестве систем;
- есть возможность автоматически выявить близкие компоненты, их сходства, различия, потенциальные конфликты и зависимости компонентов;
- есть возможность построения достаточно простых (по сравнению с традиционными технологиями) процедур перехода от одного фреймворка к другому, поскольку все компоненты и фреймворким имеют общую формальную смысловую основу, более высокоуровневую, чем в традиционных технологиях.

§ 6.1.2. Методы и средства реализации *ostis*-систем

⇒ *ключевое понятие**:

- *sc-модель кибернетической системы*
- *ostis-система*
- *ostis-платформа*
- *sc-память*
- *sc-модель базы знаний*
- *sc-модель решателя задач*
- *sc-модель интерфейса кибернетической системы*
- *sc-машина*
- *scr-машина*

Рассмотрим предлагаемый подход к организации реализации *ostis-систем*. Одним из ключевых принципов *Технологии OSTIS* является обеспечение платформенной независимости *ostis-систем*, то есть строгое разделение логико-семантической модели кибернетической системы (*sc-модели кибернетической системы*) и платформы интерпретации *sc-моделей кибернетических систем* (*ostis-платформы*). Преимущества такого строгого разделения достаточно очевидны:

- Перенос *ostis-системы* с одной платформы на другую (например более новую и эффективную или ориентированную на определенный класс устройств) выполняется с минимальными накладными расходами (в идеальном случае – вообще сводится просто к загрузке *sc-модели кибернетической системы* на платформу);
- Компоненты *ostis-систем* становятся универсальными, то есть могут использоваться в любых *ostis-системах*, где их использование является целесообразным;
- Развитие платформы и развитие *sc-моделей систем* может осуществляться параллельно и независимо друг от друга, в общем случае отдельными независимыми коллективами разработчиков по своим собственным правилам и методикам.

Рассмотрим более детально понятие *логико-семантической модели кибернетической системы*.

логико-семантическая модель кибернетической системы

- := [формальная модель (формальное описание) функционирования кибернетической системы, состоящая из (1) формальной модели информации, хранимой в памяти кибернетической системы и (2) формальной модели коллектива агентов, осуществляющих обработку указанной информации.]
- ▷ *sc-модель кибернетической системы*
- := [логико-семантическая модель кибернетической системы, представленная в SC-коде]
- := [логико-семантическая модель *ostis*-системы, которая, в частности, может быть функционально эквивалентной моделью какой-либо кибернетической системы, не являющейся *ostis*-системой]

кибернетическая система

- ▷ *компьютерная система*
- := [искусственная кибернетическая система]
- ▷ *ostis-система*
- := [компьютерная система, построенная по Технологии *OSTIS* на основе интерпретации спроектированной логико-семантической *sc*-модели этой системы]

***ostis*-система**

Субъект

⇒ обобщенная декомпозиция*:

- {• sc-модель кибернетической системы
 - *ostis*-платформа
- }

sc-модель кибернетической системы

⇒ обобщенная декомпозиция*:

- {• sc-память
 - sc-модель базы знаний
 - sc-модель решателя задач
 - sc-модель интерфейса кибернетической системы
- }

sc-память

- := [абстрактная sc-память]
- := [sc-хранилище]
- := [семантическая память, хранящая конструкции SC-кода]
- := [хранилище конструкций SC-кода]

sc-память представляет собой с одной стороны общую среду для хранения *базы знаний*, а с другой стороны – среду для взаимодействия *sc-агентов*. При этом каждый *sc-агент* опирается при работе на некоторые известные ему *sc-элементы*, хранящиеся в *sc-памяти* (*ключевые sc-элементы* данного *sc-агента*).

В общем случае *sc-память* реализует следующие функции:

- хранение конструкций *SC-кода*;
- хранение внешних по отношению к *SC-коду* информационных конструкций (файлов). В общем случае хранение файлов может быть реализовано отличным от хранения *sc-конструкций* образом;
- доступ (чтение, создание, удаление) к конструкциям *SC-кода*, реализуемый через соответствующий программный или аппаратный интерфейс. Такой интерфейс по сути представляет собой язык микропрограммирования, позволяющий реализовывать на его основе более сложные процедуры обработки хранимых конструкций, в том числе – операторы *Языка SCP*, набор которых по сути определяет перечень команд такого языка микропрограммирования. Сама *sc-память* в этом плане является пассивной и просто выполняет команды, инициируемые извне какими-либо субъектами.

Отметим, что разделение функции хранения и доступа является достаточно условным, поскольку реализовать функцию хранения конструкций без возможности доступа к ним хотя бы на самом низком уровне представляется нецелесообразным, ведь пользоваться таким хранилищем будет невозможно.

Термины “*sc-память*” и “*абстрактная sc-память*” являются синонимами в том смысле, что говоря об *sc-памяти* мы подразумеваем некоторую абстракцию, для которой не уточняется ее максимальный объем (максимальное количество *sc-элементов*, которые могут одновременно храниться в такой памяти), конкретный способ хранения *sc-элементов*, средства обеспечения надежности хранения и т.д. Все указанные особенности уточняются на уровне реализации *sc-памяти* в аппаратном варианте или варианте программной модели на базе какой-либо другой архитектуры.

Явное выделение *sc-модели базы знаний*, *sc-модели решателя задач* и *sc-модели интерфейса кибернетической системы* в рамках *sc-модели кибернетической системы* является в известной мере условным, поскольку для обеспечения платформенной независимости *sc-модели кибернетической системы* и *решатель задач*, и *интерфейс системы* описываются средствами *SC-кода* и, таким образом, тоже являются частью *базы знаний*. Такое явное выделение указанных компонентов обусловлено удобством проектирования и сопровождения системы.

Таким образом, при условии строгого разделения *sc-модели кибернетической системы* и *ostis-платформы*, а также обеспечении универсальности *ostis-платформы*, то есть возможности интерпретировать любую *sc-модель кибернетической системы* на любом варианте *ostis-платформы*, этап реализации *ostis-системы* фактически сводится к загрузке *sc-модели кибернетической системы* на выбранный вариант *ostis-платформы*.

Важно отметить, что универсальность конкретного варианта реализации *ostis-платформы* очевидно ограничивается физической (аппаратной) частью этой реализации. Например, если аппаратная часть выбранного варианта платформы представляет собой обычный персональный компьютер, то без добавления дополнительных аппаратных компонентов система не сможет решать задачи, связанные с физическим перемещением себя и других объектов в пространстве, даже если программная часть системы способна выполнить необходимые расчеты. Говоря другими словами, любая *ostis-платформа* всегда будет ограничена в решении *поведенческих задач* каких-либо классов, какими бы мощными физическими ресурсами она не обладала. Таким образом, корректнее говорить об

универсальности *ostis*-платформы в контексте решения информационных задач, то есть возможности интерпретировать любые *sc*-модели кибернетических систем независимо от того, какого рода *информационные задачи* решают эти системы.

Исходя из этого можно сформулировать ключевое требование, предъявляемое к *sc*-модели кибернетической системы – ни на одном из этапов решения любой *информационной задачи* в данной системе не должны учитываться особенности той платформы, на которой в дальнейшем будет интерпретироваться указанная *sc*-модель. Аналогично ключевым требованием к *ostis*-платформе является обеспечение интерфейса доступа (поиска и преобразования) к хранимой в *sc*-памяти информации некоторым универсальным способом, не зависящим от особенностей реализации конкретной платформы. Таким образом, важнейшей задачей для обеспечения платформенной независимости *ostis*-систем является четкая спецификация требований, предъявляемых к каждой реализации *ostis*-платформы, то есть стандартизация *ostis*-платформ. Важно отметить, что такая стандартизация не должна зависеть от того, в каком виде реализуется *ostis*-платформа, и, соответственно, подходить и для аппаратного варианта реализации.

Для уточнения требований, предъявляемых к *ostis*-платформе, введем понятие *sc*-машины, которое является аналогом таких моделей как Машина Поста и Машина Тьюринга (*Hopcroft J.E. Intro to ATLaC-2000art*), Машина фон Неймана (*Godfrey M.D. Computer and NPi-1993art*).

абстрактная sc-машина

- := [абстрактная модель ассоциативного семантического компьютера (компьютера, построенного на основе SC-кода)]
- := [абстрактная ассоциативная графодинамическая машина, в основе которой лежит SC-код]
- ↔ аналог*:
 - машина Поста
 - машина Тьюринга
 - абстрактная машина фон Неймана

Точно так же, как *абстрактная машина фон Неймана* является основой для большинства современных универсальных компьютеров, *абстрактная sc-машина* является основой для построения универсального ассоциативного семантического компьютера.

sc-машина

- ⇒ *принципы, лежащие в основе**:
 - ⟨ • [Память sc-машины (sc-память) представляет собой множество знаковых (информационных) элементов (элементарных фрагментов) памяти. Знаковый элемент sc-памяти может быть свободным (пустым), а может быть использованным для хранения информации. Каждому используемому знаковому элементу sc-памяти взаимно однозначно соответствует один из элементов хранимой (записанной) в sc-памяти sc-конструкции.]
 - [Запись sc-конструкции в sc-память осуществляется путем формирования ориентированных бинарных связей между знаковыми элементами sc-памяти, взаимно однозначно соответствующих связям инцидентности между sc-элементами записываемой (хранимой) в sc-памяти sc-конструкции. Такие связи инцидентности, в частности, могут формироваться с помощью коммутационных элементов sc-памяти.]
 - [Обработка информации в sc-памяти сводится не столько к изменению состояния знаковых элементов sc-памяти, сколько к изменению конфигурации связей между ними.]
 - [Изменение состояния знаковых элементов sc-памяти осуществляется:
 - либо в случае изменения синтаксического типа sc-элемента, соответствующего этому знаковому элементу sc-памяти;
 - либо, для знаковых элементов, обозначающих хранимые файлы, в случае изменения этих файлов.
 - [Некоторым sc-элементам, хранимым в sc-памяти, взаимно однозначно соответствуют их основные внешние идентификаторы, представленные в sc-памяти соответствующими файлами. Каждый такой знаковый элемент sc-памяти, соответствующий идентифицируемому файлом хранимому sc-элементу соответствующим образом связан со знаковым элементом sc-памяти, который обозначает и хранит указанный файл.]
 - [Доступ к sc-элементам, хранимым в sc-памяти, осуществляется:
 - либо по его основному внешнему идентификатору (если это ключевой sc-элемент);
 - либо по sc-конструкции, которая его однозначно специфицирует.
 - [процессорные элементы]
 - [коммутационные элементы – это всегда инцидентность]
 - [волновая передача сообщений по скоммутированным связям инцидентности]
 - [типология процессорных элементов – это типология sc-элементов]

sc-машина

- := [абстрактная *sc-машина*]
- := [обобщение всевозможных реализаций *ostis*-платформ, для которого задаются общие функциональные требования]
- := [обобщенная модель, описывающая функционирование любой *ostis*-платформы независимо от способа ее реализации]
- := [обобщенная модель, определяющая общие закономерности любой *ostis*-платформы независимо от способа ее реализации]
- := [обобщенный информационный образ *ostis*-платформы]
- ⇐ обобщенная модель*:
ostis-платформа
- ⇒ обобщенная декомпозиция*:
 - {• sc-память
 - ⇐ обобщенная модель*:
реализация sc-памяти
 - абстрактная машина обработки знаний
 - ⊂ абстрактный sc-агент
- }
- ▷ *scp-машина*
 - ⇐ обобщенная модель*:
scp-интерпретатор
 - := [*sc-машина*, обеспечивающая интерпретацию базового языка программирования *ostis*-систем]
 - := [обобщенная модель интерпретатора базового языка программирования *ostis*-систем]
 - := [обобщенная модель, определяющая общие принципы интерпретации базового языка программирования *ostis*-систем]
 - := [обобщенная модель операционной семантики базового языка программирования *ostis*-систем]

Потенциально можно говорить о нескольких возможных функционально эквивалентных вариантах *scp-машины*, которые будут соответствовать разным вариантам базового языка программирования. В рамках текущей версии *Технологии OSTIS* фиксируется как денотационная семантика Языка *SCP*, так и его операционная семантика, реализуемая в виде Абстрактной *scp-машины*. Более подробно об этом говорится в Главе 3.2. *Агентно-ориентированные модели гибридных решателей задач ostis-систем*.

Важно подчеркнуть, что несмотря на преимущества платформенно-независимой реализации *ostis*-систем иногда оказывается целесообразным реализовывать некоторые компоненты *ostis*-систем (например, конкретные *sc-агенты* или компоненты пользовательского интерфейса) на уровне *ostis*-платформы. В случае подобной реализации программ *sc-агентов* можно провести аналогию с реализацией каких-либо подпрограмм на уровне языков микропрограммирования для современных компьютеров. Чаще всего целесообразность такого решения обусловлена повышением производительности таких компонентов и системы в целом, поскольку реализация компонента с учетом особенностей платформы в общем случае является более производительной. В то же время заметим, что последнее утверждение не всегда верно, поскольку при реализации компонента на уровне логико-семантической модели может быть реализованы, например, модели параллельной обработки информации, не всегда легко и понятно реализуемые на уровне платформы.

Таким образом, при проектировании каждой конкретной *ostis*-системы разработчику необходимо принимать решение о реализации тех или иных компонентов на платформенно-независимом уровне или уровне платформы. При этом очевидно, что с точки зрения развития технологии и накопления проектного опыта более приоритетной является реализация компонентов *ostis*-систем на платформенно-независимом уровне.

Исходя из сказанного, можно предположить существование *ostis*-систем, в которых все *sc-агенты* реализованы на уровне платформы, которая в таком случае по сути "заточена" под конкретную *ostis*-систему и может рассматриваться как аналог специализированного компьютера, ориентированного на решение задач только определенного ограниченного класса. Назовем такой вариант реализации *ostis*-систем *минимальной конфигурацией ostis-системы*. Для того, чтобы *минимальная конфигурация ostis-системы* вообще могла считаться *ostis*-системой, то есть системой, построенной в соответствии с принципами *Технологии OSTIS*, она должна удовлетворять следующему минимальному набору требований:

- использование *SC-кода* как базового языка кодирования информации в базе знаний, и, соответственно, наличие памяти, хранящей конструкции *SC-кода*;
- наличие базы знаний, определяющей денотационную семантику понятий, используемых системой;
- наличие хотя бы одного внутреннего *sc-агента*, осуществляющего обработку знаний в памяти *ostis*-системы. Этот *sc-агент* может быть реализован на уровне платформы, соответственно база знаний такой системы может не содержать процедурных знаний (методов);

Такой вариант *минимальной конфигурации ostis-системы* обладает только *внутренним sc-агентом* и, соответственно, не имеет возможности общаться с внешним миром (можно сказать, что такая *ostis-система* не обладает "органами чувств"). Для того, чтобы система имела возможность общаться с внешним миром, необходимо добавить к *минимальной конфигурации ostis-системы* хотя бы один *рецепторный sc-агент* и хотя бы один *эффекторный sc-агент*.

Важно отметить, что, как видно из представленного описания *минимальной конфигурации ostis-системы*, в общем случае *ostis-система* не обязана по умолчанию быть *интеллектуальной системой*. Применение *Технологии OSTIS* для разработки компьютерных систем не делает их автоматически интеллектуальными, оно позволяет обеспечить возможность последующей неограниченной интеллектуализации таких систем с минимальными накладными расходами при условии соблюдения при их разработке всех принципов *Технологии OSTIS*.

§ 6.1.3. Уточнение понятия *ostis*-платформы

⇒ *ключевое понятие**:

- *ostis*-платформа
- *базовая ostis*-платформа
- *расширенная ostis*-платформа
- *специализированная ostis*-платформа
- *реализация sc-памяти*
- *реализация файловой памяти sc-машины*
- *scp-интерпретатор*
- *базовая подсистема взаимодействия ostis-системы с внешней средой*
- *подсистема обеспечения жизнедеятельности ostis-системы*
- *специализированная платформенно-зависимая машина обработки знаний*
- *минимальная конфигурация ostis-системы*
- *однопользовательская ostis*-платформа
- *многопользовательская ostis*-платформа
- *программный вариант ostis*-платформы
- *ассоциативный семантический компьютер*

ostis-платформа

- := [платформа интерпретации sc-моделей компьютерных систем]
- := [интерпретатор sc-моделей кибернетических систем]
- := [интерпретатор унифицированных логико-семантических моделей компьютерных систем]
- := [Семейство платформ интерпретации sc-моделей компьютерных систем]
- := [платформа реализации sc-моделей компьютерных систем]
- := [встроенная пустая ostis-система]
- := [реализация sc-машины]
- ⊂ *встроенная ostis*-система
- ⊂ *платформенно- зависимый многократно используемый компонент ostis*-систем

Реализация *ostis*-платформы (интерпретатора sc-моделей кибернетических систем) может иметь большое число вариантов – как программно, так и аппаратно реализованных. При необходимости, в *ostis*-платформу могут быть заранее на платформенно-зависимом уровне включены какие-либо компоненты решателей задач или баз знаний, например, с целью упрощения создания первой версии *прикладной ostis-системы*. Реализация *ostis*-платформы может осуществляться на основе произвольного набора существующих технологий, включая аппаратную реализацию каких-либо ее частей. С точки зрения компонентного подхода любая *ostis*-платформа является *платформенно- зависимым многократно используемым компонентом ostis-систем*.

ostis-платформа

⇒ *разбиение**:

- {• *базовая ostis*-платформа
 - := [базовый интерпретатор логико-семантических моделей ostis-систем]
 - := [*минимальная универсальная ostis*-платформа, обеспечивающая интерпретацию sc-модели любой ostis-системы и включающая интерпретатор базового языка программирования ostis-систем (Языка SCP)]
 - := [универсальный интерпретатор sc-моделей ostis-систем]
 - := [универсальная базовая ostis-система, обеспечивающая имитацию любой ostis-системы путем интерпретации sc-модели имитируемой ostis-системы]

- *расширенная ostis-платформа*
:= [*ostis*-платформа, содержащая дополнительные компоненты, реализованные на уровне платформы]
:= [базовая *ostis*-платформа и множество компонентов, реализованных на уровне платформы]
 - *специализированная ostis-платформа*
:= [*ostis*-платформа, не содержащая реализацию интерпретатора языка SCP]
:= [неуниверсальная *ostis*-платформа]
- }

Понятие *базовой ostis-платформы* является ключевым с точки зрения обеспечения платформенной независимости *ostis*-систем. Универсальность *базовой ostis-платформы* подразумевает возможность интерпретации на ее основе любой sc-модели кибернетической системы. Это достигается за счет наличия в рамках Технологии *OSTIS* средств, позволяющих описывать на уровне sc-модели базу знаний, решатель задач и интерфейс кибернетической системы, а также наличия Базового универсального языка программирования для *ostis*-систем (Языка SCP). Язык SCP в таком случае выступает в роли базового низкоуровневого стандарта (ассемблера) обработки конструкций SC-кода, гарантирующего полноту с точки зрения обработки, то есть, обеспечивающего возможность осуществить любое преобразование любого фрагмента SC-кода при условии сохранения синтаксической корректности этого фрагмента. Следует отметить, что в общем случае таких функционально эквивалентных ассемблеров может быть несколько (и, как следствие, соответствующих им scp-машин), но для обеспечения совместимости в рамках Технологии *OSTIS* один из таких вариантов выбирается в качестве стандарта и описывается в главе 3.2. Агентно-ориентированные модели гибридных решателей задач *ostis*-систем.

Таким образом, основным и единственным требованием, предъявляемым ко всем *базовым ostis-платформам* для обеспечения их универсальности, является необходимость обеспечения интерпретации Языка SCP, стандартизированного в рамках Технологии *OSTIS*. При этом важно отметить, что все *базовые ostis-платформы* обязаны быть функционально эквивалентными, поскольку интерпретируют один и тот же стандарт Языка SCP.

Каждая *базовая ostis-платформа* включает в себя:

- реализацию средств хранения конструкций SC-кода (sc-памяти), включая реализацию файловой памяти;
- реализацию средств обработки конструкций SC-кода – scp-интерпретатора;
- реализацию базового набора рецепторных sc-агентов и эффекторных sc-агентов, обеспечивающих минимально необходимый обмен информацией между *ostis*-системой и внешней средой. Конкретный перечень таких агентов требует уточнения, однако можно сказать, что в общем случае они могут быть реализованы как в составе scp-интерпретатора (в этом случае им будут соответствовать определенные классы scp-операторов), так и отдельно от него в составе платформы.
- реализацию набора sc-агентов, обеспечивающих базовые функции *ostis*-системы, связанные с обеспечением ее жизнедеятельности, которые принципиально не могут быть реализованы на платформенно-независимом уровне. К таким функциям относятся, например, запуск системы, загрузка базы знаний в память системы, запуск scp-интерпретатора и т.д.

Более формально модель *базовой ostis-платформы* можно записать следующим образом:

базовая ostis-платформа

⇒ обобщенная декомпозиция*:

- { • реализация sc-памяти
 - ⇒ обобщенная часть*:
 - реализация файловой памяти sc-машины
 - scp-интерпретатор
 - базовая подсистема взаимодействия *ostis*-системы с внешней средой
 - подсистема обеспечения жизнедеятельности *ostis*-системы
- }

расширенная ostis-платформа представляет собой *базовую ostis-платформу*, дополненную каким-либо множеством компонентов (хотя бы одним), реализованных на уровне платформы, при условии сохранения при этом всех возможностей *базовой ostis-платформы*. Таким образом, *расширенная ostis-платформа* по сути представляет собой *базовую ostis-платформу*, адаптированную для более эффективного решения задач определенных классов в рамках конкретного класса *ostis*-систем. Компонент, реализуемый на уровне платформы, становится частью этой платформы и, таким образом, преобразует *базовую ostis-платформу* в *расширенную ostis-платформу*.

Введение понятия *расширенной ostis-платформы* позволяет сформулировать ряд дополнительных принципов реализации *ostis*-систем:

- Может существовать произвольное количество *ostis*-систем, каждая из которых будет иметь свою уникальную *расширенную ostis-платформу*, но при этом все они будут основаны на одном и том же варианте *базовой ostis-платформы*.

- Для каждого варианта *базовой ostis-платформы* может существовать своя *библиотека многократно используемых компонентов ostis-платформ*, совместимых с данным вариантом *базовой ostis-платформы*, и позволяющая компоновать различные варианты *расширенной ostis-платформы* на основе *базовой ostis-платформы*.

специализированная ostis-платформа представляет собой ограниченный вариант реализации *ostis-платформы*, не содержащий *sc-интерпретатора*. Таким образом, все sc-агенты, в рамках *ostis-системы*, основанной на *специализированной ostis-платформе* должны быть реализованы на платформенно-зависимом уровне. Такая *специализированная ostis-платформа* является аналогом специализированного компьютера, реализованного для конкретной компьютерной системы. Таким образом, в общем случае каждая *ostis-система*, реализуемая на *специализированной ostis-платформе* будет иметь свою уникальную специализированную ostis-платформу.

специализированная ostis-платформа может быть получена из *базовой ostis-платформы* путем исключения из нее реализации *sc-интерпретатора* и реализации всех необходимых *sc-агентов* на уровне платформы (или заимствования всех или части агентов из соответствующей данному варианту *базовой ostis-платформы библиотеки многократно используемых компонентов ostis-платформ*).

специализированная ostis-платформа

⇒ обобщенная декомпозиция*:

- { • реализация *sc-памяти*
 - ⇒ обобщенная часть*:
 - реализация *файловой памяти sc-машины*
 - базовая подсистема взаимодействия *ostis-системы с внешней средой*
 - подсистема обеспечения жизнедеятельности *ostis-системы*
 - специализированная платформенно- зависимая машина обработки знаний
 - := [sc-агент, как правило неатомарный, обеспечивающий выполнение всех функций некоторой специализированной *ostis-платформы*, связанных с обработкой знаний]
 - ⊂ платформенно- зависимый sc-агент
- }

Введенное ранее понятие *минимальной конфигурации ostis-системы* может быть уточнено с учетом понятия *специализированной ostis-платформы*.

минимальная конфигурация ostis-системы

⇒ обобщенная декомпозиция*:

- { • *sc-модель базы знаний*
 - специализированная *ostis-платформа*
- }

Применение *специализированных ostis-платформ* может быть целесообразным на стартовом этапе развития *Технологии OSTIS*, а также с целью повышения производительности конкретных наиболее высоконагруженных *ostis-систем*, однако активное развитие таких *специализированных ostis-платформ* и их компонентов с точки зрения *Технологии OSTIS* является неподходящим, поскольку:

- если какой-либо компонент разработан с ориентацией на конкретную платформу, то нет гарантий возможности его повторного использования в других вариантах реализации *ostis-платформы* (как минимум, компоненты, разработанные для *программного варианта реализации ostis-платформы* не смогут быть использованы в рамках *ассоциативного семантического компьютера*);
- наличие большого числа платформенно- зависимых компонентов требует развития и сопровождения отдельной инфраструктуры библиотек для хранения и повторного использования таких компонентов. Чем больше будет вариантов *ostis-платформ* и чем больше будет число платформенно- зависимых компонентов, тем более сложной и громоздкой будет такая инфраструктура. Как минимум, необходимо будет отслеживать совместимость компонентов с разными версиями разных вариантов реализации *ostis-платформ*;
- изменения в *специализированной ostis-платформе*, например, связанные с переходом на более новую и эффективную версию *базовой ostis-платформы*, на основе которой построена данная *специализированная ostis-платформа* в общем случае могут привести к необходимости внесения изменений в компоненты, зависящие от данного варианта реализации *ostis-платформы*. Чем больше таких платформенно- зависимых компонентов, тем больше потенциальных изменений может потребоваться и, соответственно, тем сложнее будет осуществляться эволюция платформы при условии сохранения работоспособности *ostis-систем*, в которых она используется.

Перечисленные тезисы справедливы и для *расширенных ostis-платформ*, однако в случае *расширенной ostis-платформы* проблемы, связанные с переходом на более новую версию платформы и изменениями в соответствующих компонентах всегда могут быть решены путем временной замены платформенно- зависимых компонентов на

их платформенно-независимые версии с соответствующим снижением производительности, но зато сохранением функциональной целостности системы.

ostis-платформа

⇒ разбиение*:

- {• однопользовательская *ostis*-платформа
 - := [вариант реализации платформы интерпретации sc-моделей компьютерных систем, рассчитанный на то, что с конкретной *ostis*-системой взаимодействует только один пользователь (владелец)]
 - многопользовательская *ostis*-платформа
 - := [вариант реализации платформы интерпретации sc-моделей компьютерных систем, рассчитанный на то, что с конкретной *ostis*-системой одновременно или в разное время могут взаимодействовать разные пользователи, в общем случае обладающие разными правами, сферами ответственности, уровнем опыта, и имеющие свою конфиденциальную часть хранимой в базе знаний информации]
- }

При однопользовательском варианте реализации платформы оказывается невозможным реализовать некоторые важные принципы *Технологии OSTIS*, например, коллективную согласованную разработку базы знаний системы в процессе ее эксплуатации. При этом могут использоваться различные сторонние средства, например для разработки базы знаний на уровне исходных текстов.

ostis-платформа

⇒ разбиение*:

- {• программный вариант *ostis*-платформы
 - := [платформа интерпретации sc-моделей *ostis*-систем, реализованная в виде программной системы на базе традиционной компьютерной архитектуры]
 - := [программная платформа интерпретации sc-моделей *ostis*-систем]
 - := [программный интерпретатор sc-моделей *ostis*-систем]
 - аппаративный семантический компьютер
 - := [аппаратная платформа интерпретации sc-моделей *ostis*-систем]
 - := [аппаратно реализованный базовый интерпретатор sc-моделей *ostis*-систем]
- }

Важно отметить, что в любом варианте реализации *ostis*-платформы всегда присутствует как программная, так и аппаратная часть. Так, любой *программный вариант ostis*-платформы предполагает его последующую интерпретацию на какой-либо аппаратной основе, например, на персональном компьютере с традиционной архитектурой. В то же время, разработка *ostis*-платформы в виде *ассоциативного семантического компьютера* предполагает разработку набора микропрограмм, реализующих базовые операции поиска и преобразования sc-конструкций, хранящихся в sc-памяти.

Таким образом, разделение множества возможных реализаций *ostis*-платформы на программный и аппаратный варианты скорее отражает вариант аппаратной архитектуры, на которую в конечном итоге ориентирован тот или иной вариант реализации платформы – либо на традиционную фон-неймановскую архитектуру, либо на специализированную архитектуру *ассоциативного семантического компьютера* со структурно-перестраиваемой (графодинамической) памятью. *Программный вариант ostis*-платформы по сути является моделью (виртуальной машиной) *ассоциативного семантического компьютера*, построенной на базе традиционной фон-неймановской архитектуры, а Язык SCP выступает в роли ассемблера для *ассоциативного семантического компьютера* и также может интерпретироваться как в рамках аппаратной реализации такого компьютера, так и в рамках его программной модели.

Целесообразность разработки *программных вариантов ostis*-платформы на настоящий момент обусловлена очевидной распространностью фон-неймановской архитектуры и, соответственно, необходимостью реализации *ostis*-систем на современных компьютерах различного вида. В то же время очевидно, что разработка специализированных *ассоциативных семантических компьютеров* позволит существенно повысить эффективность работы *ostis*-систем, а четкое разделение sc-модели кибернетической системы и платформы ее интерпретации позволит осуществить перевод уже работающих *ostis*-систем с традиционных архитектур на *ассоциативные семантические компьютеры* с минимальными накладными расходами.

Каждой конкретной *ostis*-системе однозначно соответствует конкретная *ostis*-платформа, которая может относиться к разному набору классов *ostis*-платформ. В тоже время очевидно, что на этапе разработки платформы проектируется и реализуется некоторый вариант *ostis*-платформы, который затем тиражируется в разные *ostis*-системы. Впоследствии в каждой *ostis*-системе в этот вариант *ostis*-платформы могут быть внесены изменения, но в общем случае в большом количестве *ostis*-систем могут использоваться полностью эквивалентные *ostis*-платформы. Таким образом, целесообразно говорить о *типовых ostis*-платформах, которые:

- являются объектом разработки для разработчиков *ostis*-платформ;
- являются многократно используемым компонентом *ostis*-систем и специфицируются в рамках соответствующих библиотек;
- являются образцом для тиражирования (копирования) при создании новых *ostis*-систем.

Заключение к Главе 6.1.

В данной главе рассмотрены современные проблемы в области обеспечения платформенной независимости *компьютерных систем* в целом, а также особенности обеспечения платформенной независимости *интеллектуальных компьютерных систем*. Предложен подход к решения указанных проблем путем преобразования современных компьютерных систем в *ostis*-системы, для которых решение данной проблемы значительно упрощается благодаря принципам, лежащим в основе *Технологии OSTIS*.

Детально рассмотрены принципы обеспечения платформенной независимости *ostis*-систем, уточнено понятие *ostis*-платформы, их классификация и архитектура.

Глава 6.2.

Ассоциативные семантические компьютеры для *ostis*-систем

⇒ *автор**:

- Голенков В.В.
- Шункевич Д.В.
- Гулякина Н.А.
- Захарьев В.А.

⇒ *аннотация**:

[В главе рассмотрены принципы реализации аппаратной платформы для реализации систем, построенных на основе Технологии OSTIS, – ассоциативного семантического компьютера.]

⇒ *подраздел**:

- ***

⇒ *ключевое понятие**:

- ***

⇒ *библиографическая ссылка**:

- ***

Введение в Главу 6.2.

Применение для разработки *ostis*-систем современных программно-аппаратных платформ, ориентированных на адресный доступ к хранящимся в памяти данным, не всегда оказывается эффективным, поскольку при разработке интеллектуальных систем фактически приходится моделировать нелинейную память на базе линейной. Повышение эффективности решения задач интеллектуальными системами требует разработки специализированных платформ, в том числе аппаратных, ориентированных на унифицированные семантические модели представления и обработки информации. Таким образом, основной целью создания *ассоциативных семантических компьютеров* является повышение производительности *ostis*-систем.

§ 6.2.1. Современное состояние в области разработки компьютеров для интеллектуальных систем

Рассмотрим основные принципы, лежащие в основе *Абстрактной машины фон-Неймана*.

машина фон-Неймана

С *абстрактная машина обработки информации*

⇒ *принципы, лежащие в основе**:

- [Информация в памяти представляется в виде последовательности строк символов.]
- [Память машины представляет собой последовательность адресуемых ячеек памяти.]
- [В каждую ячейку может быть записана любая строка символов в бинарном алфавите (“0” или “1”).
При этом длина строк для всех адресуемых ячеек одинакова (в текущем стандарте ячеек, называемых байтами, равна 8 бит).]
- [Каждой ячейке памяти взаимно однозначно соответствует битовая строка, обозначающая эту ячейку и являющаяся ее адресом.]
- [каждому типу элементарных действий (операций), выполняемых в памяти машины фон-Неймана, взаимно однозначно ставится ее идентификатор, который в памяти представляется в виде битовой строки]
- [каждая конкретная операция (команда), выполняемая в памяти, представляется (специфицируется) в памяти в виде строки, состоящей
 - из кода соответствующего типа операции;

- из последовательности адресов фрагментов памяти, в которых находятся операнды, над которыми выполняются операции – исходные аргументы и результаты. Любой такой фрагмент задается адресом первого байта и количеством байт. Количество операндов однозначно задается кодом типа операции;
-]
- [Программа, выполняемая в памяти, хранится в памяти в виде последовательности спецификаций конкретных операций (команд).]
- }

Рассмотрим более детально особенности логической организации традиционной (фон-Неймановской) архитектуры компьютерных систем, существенно затрудняющие эффективную реализацию *ostis*-систем на ее основе:

- последовательная обработка, ограничивающая эффективность компьютеров физическими возможностями элементной базы;
- низкий уровень доступа к памяти, т.е. сложность и громоздкость выполнения процедуры ассоциативного поиска нужного фрагмента знаний;
- линейная организация памяти и чрезвычайно простой вид конструктивных объектов, непосредственно хранимых в памяти. Это приводит к тому, что в интеллектуальных системах, построенных на базе современных компьютеров, манипулирование знаниями осуществляется с большим трудом. Во-первых, приходится оперировать не самими структурами, а их громоздкими линейными представлениями (списками, матрицами смежности, матрицами инцидентности); во-вторых, линеаризация сложных структур разрушает локальность их преобразований;
- представление информации в памяти современных компьютеров имеет уровень весьма далекий от смыслового, что делает переработку знаний довольно громоздкой, требующей учета большого количества деталей, касающейся не смысла перерабатываемой информации, а способа ее представления в памяти;
- в современных компьютерах имеет место весьма низкий уровень аппаратно реализуемых операций над нечисловыми данными и полностью отсутствует аппаратная поддержка логических операций над фрагментами знаний, имеющих сложную структуру, что делает манипулирование такими фрагментами весьма сложным.

Перечисленные особенности, по существу, не устраняются также и в развивающихся в настоящее время подходах к построению нетрадиционных высокопроизводительных компьютеров (например, компьютеров, предназначенных для обучения и интерпретации искусственных нейронных сетей **Neurocomputers**; **USBAccel-2022el**), ибо, в основном, все эти подходы (даже если они достаточно далеко отходят от предложенных фон Нейманом базовых принципов организации вычислительных машин) неявно сохраняют точку зрения на компьютер как на большой арифмометр и тем самым сохраняют ее ориентацию на задачи числового характера.

Существует ряд статей **Tran2018; Shi2018; Lu2021; Afanasyev2021; Zhang2017; Hu2021; Minati2019; Song2016** и патентов **Somsubhra_2006; Allen_1989; Moussa_2013**, направленных на разработку аппаратных архитектур, предназначенных для обработки информации, представленной в более сложных формах, чем в традиционных архитектурах, однако они не получили широкого распространения и применения, по причине, во-первых, частности предлагаемых решений, и во-вторых из-за отсутствия общего универсального и унифицированного языка кодирования любой информации, в роли которого в рамках Технологии *OSTIS* выступает SC-код.

SC-код, являющийся формальной основой *Технологии OSTIS* изначально разрабатывался как язык кодирования информации в памяти *ассоциативных семантических компьютеров*, таким образом в нем изначально заложены такие принципы, как универсальность (возможность представить знания любого рода) и унификация (единообразие) представления, а также минимизация *Алфавита SC-кода*, которая, в свою очередь, позволяет облегчить создание аппаратной платформы, позволяющей хранить и обрабатывать тексты *SC*-кода.

Основная методологическая особенность нашего подхода к разработке средств аппаратной реализации поддержки интеллектуальных систем заключается в том, что такие средства должны разрабатываться не до, а после того, как основные положения соответствующей технологии проектирования и эксплуатации интеллектуальных систем будут апробированы на современных технических средствах. Более того, в рамках Технологии *OSTIS* четко продумана методика перехода на новые аппаратные средства, которая затрагивает только самый нижний уровень технологии – уровень реализации базовой машины обработки семантических сетей.

§ 6.2.2. Общие принципы, лежащие в основе ассоциативных семантических компьютеров для *ostis*-систем

Необходимо описать абстрактную *sc*-машину по аналогии с тем, как описывается *абстрактная машина фон Неймана*, взяв за основу принципы Языка *SCP*. Необходимо уточнить:

- как кодируется машинная команда
- что происходит в памяти в результате ее выполнения

Память sc-машины представляет собой динамический sc-текст.

Перечень команд sc-машины:

- команда-продукция
 - область действия команды
 - для всех найденных структур/для одной найденной структуры
 - образец (шаблон) поиска (с указанием констант и переменных)
 - образец (шаблон) генерируемой структуры, в состав которой могут входить
 - команды удаления sc-элементов
 - команды переноса инцидентных связей от одного sc-элемента к другому
- команда удаления sc-элемента
- команда удаления sc-элемента в рамках указываемой структуры
- команда переноса инцидентных связей от одного sc-элемента к другому

Склейивание двух sc-элементов реализуется в несколько этапов:

- генерация нового sc-элемента, который станет результатом склеивания;
- для каждой связи первого склеиваемого sc-элемента генерируется аналогичная связь с новым sc-элементом, после чего эти перенесенные связи удаляются (команда переноса инцидентных связей от одного sc-элемента к другому);
- аналогично осуществляется перенос всех связей второго склеиваемого sc-элемента (команда переноса инцидентных связей от одного sc-элемента к другому);
- после этого оба склеиваемых sc-элемента удаляются;

Рассмотрим принципы, лежащие в основе реализации *ассоциативных семантических компьютеров*:

- нелинейная память – каждый элементарный фрагмент хранимого в памяти текста может быть инцидентен неограниченному числу других элементарных фрагментов этого текста. Таким образом, ячейки памяти, в отличие от обычной памяти, связываются не фиксированными условными связями, задающими фиксированную последовательность (порядок) ячеек в памяти, а реально (физически) проводимыми связями произвольной конфигурации. Эти связи соответствуют дугам, ребрам, гиперребрам записанного в памяти графа (sc-текста);
- структурно-перестраиваемая (реконфигурируемая) память – процесс отработки хранимой в памяти информации сводится не только к изменению состояния элементов, но и к реконфигурации связей между ними. То есть, в ходе переработки информации в структурно-перестраиваемой памяти меняются не только и даже не столько состояния ячеек памяти, как это имеет место в обычной памяти, сколько конфигурация связей между этими ячейками. Т.е. в структурно-перестраиваемой памяти в ходе переработки информации не только перераспределяются метки на вершинах записанного в памяти графа, но и меняется структура самого этого графа;
- в качестве внутреннего способа кодирования знаний, хранимых в памяти *ассоциативного семантического компьютера*, используется универсальный (!) способ нелинейного (графоподобного) смыслового представления знаний – SC-код;
- обработка информации осуществляется коллективом агентов, работающих над общей памятью. Каждый из них реагирует на соответствующую ему ситуацию или событие в памяти (компьютер, управляемый хранимыми знаниями);
- есть программно реализуемые агенты, поведение которых описывается хранимыми в памяти агентно-ориентированными программами, которые интерпретируются соответствующими коллективами агентов;
- есть базовые агенты, которые не могут быть реализованы программно (в частности, это агенты интерпретации агентных программ, базовые рецепторные агенты-датчики, базовые эффекторные агенты);
- все агенты работают над общей памятью одновременно. Более того, если для какого-либо агента в некоторый момент времени в различных частях памяти возникает сразу несколько условий его применения, разные информационные процессы, соответствующие указанному агенту в разных частях памяти могут выполняться одновременно;
- для того, чтобы информационные процессы агентов, параллельно выполняемые в общей памяти не "мешали" друг другу, для каждого информационного процесса в памяти фиксируется и постоянно актуализируется его текущее состояние. То есть каждый информационный процесс сообщает всем остальным о своих намерениях и пожеланиях, которым остальные информационные процессы не должны препятствовать. Реализация такого подхода может выполняться, например, на основе механизма блокировок элементов семантической памяти, рассмотренного в главе [Агентно-ориентированные модели гибридных решателей задач ostis-систем](#);
- процессор и память *ассоциативного семантического компьютера* глубоко интегрированы и составляют единую процессор-память. Процессор *ассоциативного семантического компьютера* равномерно "распределен" по его памяти так, что процессорные элементы одновременно являются и элементами памяти компьютера. То есть каждая ячейка дополняется функциональным (процессорным) элементом, а перестраиваемые связи между ячейками становятся коммутируемыми каналами связи между функциональными элементами. Каждый функциональный элемент при этом имеет свою специальную внутреннюю регистровую память, отражающую

важные для данного функционального элемента аспекты текущего состояния процесса выполнения элементарных операций внутреннего языка.

Обработка информации в *ассоциативном семантическом компьютере* сводится к реконфигурации каналов связи между процессорными элементами, следовательно память такого компьютера есть не что иное, как коммутатор (!) указанных каналов связи. Таким образом, текущее состояние конфигурации этих каналов связи и есть текущее состояние обрабатываемой информации. Этот принцип обеспечивает значительное ускорение переработки информации благодаря исключению этапов передачи информации из памяти в процессор и обратно, но оплачивается ценой большой избыточности функциональных (процессорных) средств, равномерно распределяемых по памяти.

§ 6.2.3. Архитектура ассоциативных семантических компьютеров для ostis-систем

В разработке компьютера выделим три этапа:

- Согласование интерфейса между платформенно-независимой частью технологии и платформенно-зависимой частью. Разработка частной спецификации системы команд (желательно очень ограниченной) для компьютера.
- Обоснование системы ограничений. Разработка архитектуры компьютера (разработка альтернативных вариантов, экспресс-оценка и выбор и детализация одного из вариантов). Моделирование архитектуры. Верификация проекта. Оценка эффективности архитектуры.
 - Алфавит компьютера
 - Система команд
 - Формат команд
 - Способ программного управления
 - Базовая структура компьютера
- Выбор элементной базы, аппаратной платформы, обоснование системы ограничений. Разработка структуры и схемотехники аппаратного прототипа компьютера. Верификация проекта. Испытания и оценка эффективности компьютера.

ассоциативный семантический компьютер

- С компьютер с графодинамической ассоциативной памятью
- := [associative semantic computer]
- := [sc-компьютер]
- := [scp-компьютер]
- := [аппаратно реализованный базовый интерпретатор семантических моделей (sc-моделей) компьютерных систем]
- := [аппаратно реализованная ostis-платформа]
- := [аппаратный вариант ostis-платформы]
- := [семантический ассоциативный компьютер, управляемый знаниями]
- := [компьютер с нелинейной структурно перестраиваемой (графодинамической) ассоциативной памятью, переработка информации в которой сводится не к изменению состояния элементов памяти, а к изменению конфигурации связей между ними]
- := [универсальный компьютер нового поколения, специально предназначенный для реализации семантически совместимых гибридных интеллектуальных компьютерных систем]
- := [универсальный компьютер нового поколения, ориентированный на аппаратную интерпретацию логико-семантических моделей интеллектуальных компьютерных систем]
- := [универсальный компьютер нового поколения, ориентированный на аппаратную интерпретацию ostis-систем]
- := [ostis-компьютер]
- := [компьютер для реализации ostis-систем]
- := [компьютер, управляемый знаниями, представленными в SC-коде]
- := [компьютер, ориентированный на обработку текстов SC-кода]
- := [компьютер, внутренним языком которого является SC-код]
- := [компьютер, осуществляющий реализацию sc-памяти и интерпретацию scp-программ]
- := [предлагаемый нами компьютер нового поколения, ориентированный на реализацию интеллектуальных компьютерных систем и использующий SC-код в качестве внутреннего языка]
- ⇒ разбиение*:
 - {• *scp-компьютер*
 - := [sc-компьютер с минимальным набором аппаратно реализованных sc-агентов]
 - *sc-компьютер с расширенным набором аппаратно реализуемых sc-агентов*

scp-компьютер

- := [минимальная конфигурация аппаратно реализованной ostis-платформы, в рамках которой все небазовые sc-агенты реализованы в виде агентных scp-программ]
- := [минимальная конфигурация аппаратно реализованной ostis-платформы, в рамках которой аппаратно реализуются только базовые sc-агенты]
- ⇒ пояснение*: [В рамках scp-компьютера аппаратно реализуется (1) sc-память, (2) базовые sc-агенты, обеспечивающие интерпретацию scp-программ, (3) элементарные рецепторные sc-агенты, (4) элементарные эффекторные sc-агенты]

Компоненты sc-компьютера:

- коммутатор
 - узлы которого – sc-элементы
 - коммутируемые каналы связи – дуги инцидентности sc-элементов
 - узлы – процессорные элементы, которые обмениваются между собой фрагментами микропрограмм, обеспечивающих интерпретацию хранимых в памяти scp-программ.
 - волновой язык микропрограмм
- процессорный элемент sc-памяти. В его памяти хранится:
 - содержимое (если это узел с содержимым)
 - метка типа sc-элемента
 - метка блокировки sc-элементов с указанием метки соответствующего процесса
 - микропрограммы с микрокомандами типа "переслать по всем выходящим sc-дугам указанного типа и с указанными метками блокировки данную микропрограмму"(волновые микропрограммы)
 - уникальные имена (метки) процессов, к которым относятся передаваемые микропрограммы (с описанием иерархии надпроцессов).

Типология микрокоманд sc-компьютера

- Переслать указанную микропрограмму для исполнения из данного процессорного элемента по всем указанным каналам (инцидентным sc-коннекторам указываемого типа) всем смежным sc-элементам указываемого типа;
- Выполнить указанное преобразование содержимого данного sc-узла;
- Заменить метку типа sc-элемента;
- Заменить блокировку данного sc-элемента для указанного процесса (в том числе, отменить метку);
- Удалить sc-элемент;
- Сгенерировать инцидентный sc-коннектор (новый канал связи), возможно, вместе со смежным sc-элементом;
- Сгенерировать оба или один sc-элемент, соединяемые данным sc-коннектором

Языки для sc-компьютера:

- SC-код;
- Язык SCP;
- Язык описания состояния процессорного элемента (sc-элемента);
- Язык микропрограмм, которым обмениваются процессорные элементы между собой, и которые исполняются этими процессорными элементами.

Кроме *процессорных элементов* есть *коммутационные элементы*. Множество *коммутационных элементов* – это связующая среда, обеспечивающая формирование коммутируемых каналов связи между *процессорными элементами*.

Любой *коммутационный элемент* может входить в не более чем в один канал связи.

Связь между *коммутационными элементами* имеет два состояния:

- свободна – не входит в состав коммутируемого канала связи;
- занята – является звеном в канале связи.

Глава 6.3.

Программная платформа *ostis*-систем

- ⇒ *автор**:
- Зотов Н.В.
 - Шункевич Д.В.
- ⇒ *аннотация**:
- [В данной главе формально описывается один из вариантов реализации программной платформы массовой коллективной разработки и эксплуатации интеллектуальных к.с. нового поколения. В данной части монографии детально иллюстрируется пример использования онтологического подхода к разработке и спецификации подкласса программных систем, являющихся системами автоматизации проектирования и реализации других программных систем.]
- ⇒ *подраздел**:
- § 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных систем
 - § 6.3.2. Принципы, лежащие в основе, и аналоги предлагаемого Программного варианта реализации *ostis*-платформы
 - § 6.3.3. Реализация sc-памяти в Программной платформе *ostis*-систем
 - § 6.3.4. Реализация файловой памяти в Программной платформе *ostis*-систем
 - § 6.3.5. Реализация подсистемы взаимодействия с sc-памятью на основе языка JSON
 - § 6.3.6. Реализация интерпретатора sc-моделей пользовательских интерфейсов
- ⇒ *ключевой знак**:
- Программный вариант реализации *ostis*-платформы
 - Реализация sc-памяти *ostis*-платформы
 - Реализация файловой памяти *ostis*-платформы
 - Реализация подсистемы взаимодействия *ostis*-платформы с внешней средой
 - Реализация интерпретатора sc-моделей пользовательских интерфейсов
- ⇒ *ключевое знание**:
- Спецификация Программного варианта реализации *ostis*-платформы
- ⇒ *библиографическая ссылка**:
- ***

Введение в Главу 6.3. Программная платформа *ostis*-систем

До настоящего времени существует обширное множество различных решений в области автоматизации проектирования и разработки к.с. *iliadis2019tower*, позволяющих решать задачи достаточно серьёзного уровня. Однако ни одна из таких систем не способна обеспечить платформенную независимость, а значит и семантическую совместимость и интероперабельность создаваемых к.с. Актуальность проблемы объясняется необходимостью создания к.с. нового поколения, способных быстро и качественно решать задачи любого рода деятельности.

Современные компьютерные системы, а также средства автоматизации проектирования и разработки таких систем, обладают рядом значительных недостатков:

- Проектируемые к.с. в большей мере остаются зависимыми от реализации конкретных платформ, на которых они проектируются, что, в свою очередь, приводит к существенным затратам на приведение в соответствие методов и средств проектирования систем в случае их перехода на новые платформы.
- Проектирование и разработка реализации конкретной системы ведётся при помощи разных методов и моделей проектирования программных к.с. Тем самым, описание целевого состояния системы и описание текущей реализации могут не соответствовать друг другу, а интеграция таких решений трудно достижима *Соколов А.П. СистемАПКМ-2021см.*
- Место спецификации программных систем отводится на второй план, а иногда и вовсе не предусматривается проектом разработки конкретной компьютерной системы. Следовательно, увеличиваются затраты на поддержание процесса перманентного реинжиниринга таких систем *Dillon T. OBSESE-2008art.*

- При разработке современных систем отсутствует понимание необходимости разработки и описания методов проектирования этих систем, в т. ч. описания процесса реализации, направлений использования и т. д. По этой причине новое поколение разработчиков программных к.с. не используют уже имеющийся накопленный опыт, а изобретают одни и те же либо похожие решения *Dillon T. GSGSotWETaS-2007art*, *Dillon T. OBSESE-2008art*.
- Отсутствуют единые универсальные инструментальные средства разработки **kabilan2007ontology** и реинжиниринга других систем, позволяющие не только автоматизировать их проектирование, но и свести к минимуму саму разработку за счёт унификации моделей представления этих систем и наличия семантически мощной комплексной библиотеки многократно используемых компонентов.
- Даже узкоспециализированные программные компьютерные системы должны обладать хорошим уровнем интеллекта и хорошим уровнем обучаемости для расширения возможностей в решении более сложных задач. Программные компьютерные системы нового поколения в отличие от современных к.с., должны оперировать смыслом того, что они знают и обрабатывают: они должны понимать друг друга, **ouksel1999semantic**, **neiva2016towards** находить точки соприкосновения и образовывать коллективы для решения задач любого класса *Lu K..RethiMCfSCiSC-2022art*, *zhou2022cognitive*.
- Для эффективной реализации даже существующих моделей представления знаний и моделей решения трудно формализуемых задач современные компьютеры оказываются плохо приспособленными, что требует разработки принципиально новых платформ и компьютеров, обеспечивающих унификацию представления этих знаний **hagoort2009semantic**, **siekmann1984universal**.

Обычно системы такого рода проектируются и разрабатываются для решения узких прикладных задач. В результате образуются системы с описанными выше проблемами. Так, большая часть из описанных проблем, к сожалению, не были решены при создании САПР, описанных в работах **lenat1990cyc**, **reed2002mapping**, **Gomolkaa2022**, **Gribova2018**, **Filippov2016**, **Zagorulko2015**.

При разработке спецификации таких систем важной задачей является задача выбора средств и методов проектирования будущей платформы разработки ostis-систем. Она должна обеспечивать:

- Однозначность интерпретации и представления логико-семантических моделей ostis-систем, обеспечиваемые используемым унифицированным языком представления знаний и онтологии проектирования платформ ostis-систем;
- Семантическую совместимость логико-семантических моделей ostis-систем и их компонентов между собой;
- Платформенную независимость реализуемых и интерпретируемых на ней логико-семантических моделей ostis-систем;
- Простоту и гибкость расширения своих функциональный возможностей и круга решаемых задач;
- Функциональную полноту для создания логико-семантических моделей ostis-систем за счёт наличия формальной методологии проектирования её реализации;
- Разделение обязанностей между компонентами (машинами) платформы.

§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных систем

⇒ **ключевой знак***:

- *SMILA*
- *SYC*
- *Neo4j*

Реализация хранилищ данных, используемых в подавляющем большинстве компьютерные системы основываются на *реляционной модели данных*. Примерами таких систем для обработки неструктурированных и слабоструктурированных данных являются платформы SMILA (SeMantic Information Logistic Architecture, Семантическая информационная логистическая архитектура), Teradata Aster Discovery Platform **ballinger2009teradata**, реализующие реляционную, поколоночную и гибридную модели хранения записей в базе данных с массово-параллельной архитектурой MPP, платформа CYC CycPlatform2022, средства Semantic Web **sem_web**.

Однако, в настоящее время информационные системы подвергаются интенсивной интеллектуализации. В первую очередь, это вызвано повышением уровня сложности решаемых задач. Интеллектуализация информационных систем, как и любая технология разработки программных систем, требует учета слабоформализуемой, возможно не полностью определенной, нечеткой, темпоральной, пространственно-распределенной информации и, как следствия, получения структурированных, слабоструктурированных и неструктурных данных **Rudikova2018**. Увеличение числа интеллектуальных задач обработки больших объемов данных во всех сферах деятельности человека приводит к потребности создания универсальных средств хранения, представления и обработки мультистроктурированной информации.

Наличие таких задач стимулирует переход от обычных б.д. к графовым аналогам. Это объясняется не столько эффективностью организации памяти и обработки данных в графовых б.д., сколько важностью представления конфигураций связей (т.е. смысла) между ними *Bai J., fPlatf tKGЕoLA-2022art*. Подробное изъяснение принципов организации графовых данных в базах данных можно найти в работе авторов популярной графовой с.у.б.д. Neo4j **GDB**.

Для общего понимания всей проблемы, связанной с представлением и обработкой данных и знаний рассмотрим несколько современных реализаций графовых моделей данных в виде программных продуктов. Любая из описанных ниже б.д. предназначена для удобного хранения и доступа к данным, представленным в виде графов сверхбольшой размерности. По организации памяти и процессу обработки данных графовые б.д. можно классифицировать на следующие виды:

- б.д. с локальным хранением и обработкой графов (Neo4j, HyperGraphDB, AllegroGraph);
- б.д. с распределенным хранением и обработкой данных (Horton, InfiniteGraph);
- б.д. в формате "ключ-значение" (Trinity, CloudGraph, RedisGraph, VertexDB);
- документо-ориентированные б.д. (OrientDB);
- надстройки над SQL-ориентированными б.д. (Filament, G-store);
- графовые б.д. с моделью MapReduce (Pregel, Apache Giraph, GraphLab).

С подробным описанием каждой из представленных б.д. можно ознакомиться в работах, посвящённых сравнению реляционных и графовых б.д. *Абрамский М.М. СравнИРиГБдвРЦОС-2018ст*, *vicknair2010comparison*, *Климанская Е.В. СоврПИАОИГБД-2014ст*, *Chen C..MPEoRaGD-2022art*.

Мотивация перехода от обычных графовых баз данных объясняется преимуществами организации модели памяти и обработки в них:

- Производительность обработки данных улучшается на единицу или более порядков при представлении данных в виде графовых структур, что объясняется свойствами самого графа. В отличие от реляционных баз данных, где производительность запросов с увеличением интенсивности запросов ухудшается по мере увеличения набора данных, производительность графовой б.д. имеет тенденцию оставаться относительно постоянными, даже когда набор данных растет. Это связано с тем, что обработка данных локализуется в некоторой части графа. В результате время выполнения каждого запроса равно пропорционально только размеру части графа, пройденной для удовлетворения этого запроса, а не размеру всего графа **Khasahmadi2020Memory-Based**.
- Графовые структуры имеют огромную выразительную силу. Графовые базы данных предлагают чрезвычайно гибкую модель данных и способ представления **DiscreteMathSys-эл**, **Reinhard**. Графовые структуры аддитивны, это обеспечивает гибкость добавления новых связей между данными, новые узлы и новые подграфы к уже существующей структуре, не нарушая её целостности и связности.

Как говорилось ранее, компьютерные системы нового поколения в силу своих свойств должны оперировать не просто данными, а знаниями. Чтобы понимать смысл знаний, необходимо представлять эти знания в понятной форме для каждого: и для человека, и для машины. Говоря об унификации представления всех видов знаний, важным считается использование графовых баз данных не просто как средств для хранения структурированных данных, а для хранения семантически целостных и связанных между собой знаний **SenKB2017**. В контексте проектирования к.с. нового поколения будем говорить о базах знаний, проектируемых по принципам графовых б.д.

Стоит также отметить, что акцент в данной работе ставится не на разработку к.с. поддержки проектирования других систем, а на разработку комплексных инструментальных средств поддержки автоматического проектирования интеллектуальных к.с. нового поколения, управляемых знаниями. Такие инструментальные средства можно сравнивать с системами управления базами знаний *Наумов А.Н.. СистеУБдз-1991кн*, *Gavrilova2000*, *Башлыков А.А. СистеУБЗ-2010ст*, *Башлыков А.А. МетодПСУБЗдИС-2013ст*.

В основе *ostis-платформы* должны лежать основополагающие принципы:

- Все тексты, представляемые на SC-коде являются графовыми конструкциями. Поэтому задача разработки *Программного варианта реализации ostis-платформы* сводится к разработке средств хранения и обработки таких графовых конструкций. Другими словами, будущая платформа должна обеспечивать функционально полную и однозначную интерпретацию хранимых графовых конструкций.
- Проектирование платформы интерпретации sc-моделей к.с., в том числе её компонентов, должно чётко специфицироваться и формулироваться в рамках моделей, методов и средств описания сложных систем, предлагаемых Технологией OSTIS. Именно онтологический подход к проектированию, эксплуатации и реинжиниринга такого подкласса к.с. позволит эффективно и универсально разрабатывать другие *ostis*-системы самого различного назначения *Molorodov2019*, *Zapata J..Ontol iSEATGKA*.

Спецификация такого сложного программного объекта, как *ostis-платформа*, должна быть представлена на каком-то формальном языке представления знаний, в данном случае на SC-коде, поскольку она его должна хранить и обрабатывать. С точки зрения связи с самим SC-кодом, язык, который должен описывать *Программную реализацию ostis-платформы*, должен являться подъязыком SC-кода, то есть должен наследовать все свойства синтаксиса и денотационной семантики SC-кода, и метаязыком описания представления конструкций SC-кода в памяти про-

граммного эмулятора семантического ассоциативного компьютера. Такая модель представления спецификации программных к.с. даёт безусловно сильные преимущества по сравнению с другими возможными вариантами представления спецификаций:

- Язык, тексты которого система хранит и обрабатывает, и язык спецификации того, как система представляет тексты первого языка в памяти самой себя, являются подмножествами одного и того же языка. Это упрощает не только становление понимание разработчика, который разрабатывает сложную программную систему, за счёт того, что форма представления обрабатываемого этой системой языка и языка её спецификации, унифицирована, но и позволяет открыть для этой системы новые функциональные возможности в познании самой себя. Таким образом, такой подход позволяет полностью реализовывать свойства интеллектуальной системы, например, рефлексивности.
- Нельзя проектировать и реализовывать интеллектуальные компьютерные системы на компьютерной системе, которая сама таковой не является. Представление спецификации системы в такой форме позволяет повысить уровень её интеллекта.
- Поскольку форма представления языка описания системы унифицирована с языком, которые она обрабатывает, то нет необходимости в создании дополнительных средств для верификации и анализа работы системы.

Вообще этот sc-язык можно представить в виде некоторого семейства более частных sc-языков, которые позволяют описывать:

- то, как информационные конструкции SC-кода (sc-конструкции) представляются внутри памяти ostis-системы;
- то, как информационные конструкции, которые не принадлежат SC-коду (внешние информационные конструкции) представляются внутри памяти ostis-системы;
- то, как различные подсистемы платформы, на которой базируется некоторая ostis-система, взаимодействуют между собой;
- то, какие методы и соответствующие им агенты взаимодействуют с памятью ostis-системы;
- то, как представляются и работают различного рода интерпретаторы sc-моделей (базы знаний, решателя, интерфейса) ostis-системы;
- и так далее.

В таком случае, это позволяет не только уменьшить количество используемых средств при проектировании и реализации ostis-платформы, но и позволяет унифицировать информацию, хранимую в ostis-платформе и описывающую ostis-платформу, с целью использования этой информации в процессе эволюции компонентов ostis-платформы. При этом спецификация ostis-платформы остаётся платформенно-независимой, поэтому при смене одного варианта реализации ostis-платформы на другой подход к описанию ostis-платформы остаётся одним тем же.

Одним из путей, позволяющих осуществлять апробацию, развитие, а в ряде случаев и внедрение новых моделей и технологий вне зависимости от наличия соответствующих аппаратных средств является разработка программных моделей этих аппаратных средств, которые были бы функционально эквивалентны этим аппаратным средствам, но при этом интерпретировались на базе традиционной аппаратной архитектуры (в данной работе традиционной архитектурой будем считать архитектуру фон Неймана, как доминирующую в настоящее время). Очевидно, что производительность таких программных моделей в общем случае будет ниже, чем самих аппаратных решений, однако в большинстве случаев она оказывается достаточной для того, чтобы развивать соответствующую технологию параллельно с разработкой аппаратных средств и осуществления постепенного перевода уже работающих систем с программной модели на аппаратные средства.

Популярность и развитость графовых б.д. приводит к тому, что на первый взгляд целесообразным и эффективным кажется вариант реализации *Программного варианта реализации ostis-платформы* на базе одного из таких средств. Однако, существует ряд причин, в силу которых это сделать невозможно. К ним относятся следующие:

- Для обеспечения эффективности хранения и обработки информационных конструкций определенного вида (в данном случае – конструкций SC-кода, sc-конструкций), должна учитываться специфика этих конструкций. В частности, описанные в работе *Корончик.Д.Н.УСМПИИСТИКП-2013см* эксперименты показали значительный прирост эффективности собственного решения по сравнению с существующими на тот момент.
- В отличие от классических графовых конструкций, где дуга или ребро могут быть инцидентны только вершине графа (это справедливо и для rdf-графов) в SC-коде вполне типичной является ситуация, когда sc-коннектор инцидентен другому sc-коннектору или даже двум sc-коннекторам. В связи с этим существующие средства хранения графовых конструкций не позволяют в явном виде хранить sc-конструкции (sc-графы). Данная проблема также решаема при переходе от неориентированного графа к орграфу *Иващенко.В.П.МоделиАИЗООСС-2015см*.
- В основе обработки информации в рамках Технологии OSTIS лежит многоагентный подход **iotti2018agent**, в рамках которого агенты обработки информации, хранимой в sc-памяти (sc-агенты) реагируют на события, происходящие в sc-памяти и обмениваются информацией посредством спецификаций выполняемых ими действий в sc-памяти *Shunkevich.D.V.AgentОММТСРСДИС-2018см*. В связи с этим одной из важнейших задач является реализация в рамках *Программного варианта реализации ostis-платформы* возможности подписки на события, происходящие в программной модели sc-памяти, которая на данный момент практически не поддерживается в рамках современных средств хранения и обработки графовых конструкций.

- SC-код позволяет описывать также внешние информационные конструкции любого рода (изображения, текстовые файлы, аудио- и видеофайлы и т.д.), которые формально трактуются как содержимое *sc*-элементов, являющихся знаками *внешних файлов ostis-системы*. Таким образом, компонентом *Программного варианта ostis-платформы* должна быть реализация файловой памяти, которая позволяет хранить указанные конструкции в каких-либо общепринятых форматах. Реализация такого компонента в рамках современных средств хранения и обработки графовых конструкций также не всегда представляется возможной.

По совокупности перечисленных причин было принято решение о реализации *Программного варианта реализации ostis-платформы* "с нуля" с учетом особенностей хранения и обработки информации в рамках Технологии OSTIS.

§ 6.3.2. Принципы, лежащие в основе, и аналоги предлагаемого Программного варианта реализации *ostis*-платформы

- ⇒ *ключевой знак**:
- *Программный вариант реализации ostis-платформы*

Текущий *Программный вариант реализации ostis-платформы* является web-ориентированным, поэтому с этой точки зрения каждая *ostis*-система представляет собой web-сайт доступный онлайн посредством обычного браузера. Такой вариант реализации обладает очевидным преимуществом – доступ к системе возможен из любой точки мира, где есть Интернет, при этом для работы с системой не требуется никакого специализированного программного обеспечения. С другой стороны, такой вариант реализации обеспечивает возможность параллельной работы с системой нескольких пользователей. Реализация является кроссплатформенной и может быть собрана из исходных текстов в различных операционных системах. В то же время, взаимодействие клиентской и серверной части организовано таким образом, что web-интерфейс может быть легко заменен на настольный или мобильный интерфейс, как универсальный, так и специализированный.

Программный вариант реализации ostis-платформы

- ∈ *специализированная ostis-платформа*
 - ⇐ *ключевой знак**:
 - Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем
- ∈ *web-ориентированный вариант реализации ostis-платформы*
 - := [вариант реализации платформы интерпретации sc-моделей компьютерных систем, предполагающий взаимодействие пользователей с системой посредством сети Интернет]
- ∈ *многопользовательский вариант реализации ostis-платформы*
 - ⇐ *ключевой знак**:
 - Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем
- ∈ *неатомарный многократно используемый компонент ostis-систем*
 - ⇐ *ключевой знак**:
 - Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем
- ∈ *зависимый многократно используемый компонент ostis-систем*
 - ⇐ *ключевой знак**:
 - Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем
- ⇒ *декомпозиция программной системы**:
 - {• Реализация памяти ostis-платформы
 - Реализация подсистемы взаимодействия с внешней средой с использованием языков сетевого взаимодействия
 - Реализация интерпретатора sc-моделей пользовательских интерфейсов
 - Реализация базового набора платформенно-зависимых sc-агентов и их общих компонентов
- ⇒ *зависимости компонента**:
 - {• Реализация памяти ostis-платформы
 - Реализация подсистемы взаимодействия с внешней средой с использованием языков сетевого взаимодействия
 - Реализация интерпретатора sc-моделей пользовательских интерфейсов

Ядром платформы является *Реализация памяти ostis-платформы*, которая одновременно может взаимодействовать как с *Реализацией интерпретатора sc-моделей пользовательских интерфейсов*, так и с любыми сторонними

приложениями по соответствующим языкам сетевого взаимодействия (сетевым протоколам). С точки зрения общей архитектуры *Реализация интерпретатора sc-моделей пользовательских интерфейсов* выступает как один из множества возможных внешних компонентов, взаимодействующих с *Реализацией памяти ostis-платформы* по сети. Стоит отметить, что текущий вариант реализации *ostis*-платформы является специализированным, то есть не включает реализацию интерпретатора базового языка SCP.

Текущая *Реализация памяти ostis-платформы* позволяет хранить и представлять sc-конструкции, с помощью которых описывается любая логико-семантическая модель *ostis*-систем, внешние информационные конструкции, не принадлежащие SC-коду, а также предоставлять различные уровни доступа для обработки этих конструкций. В контексте данного *Программного варианта реализации ostis-платформы* *Реализация памяти ostis-платформы* состоит из таких компонентов как: *Реализация sc-памяти ostis-платформы*, внутри которой представляются sc-конструкции логико-семантических моделей *ostis*-систем, *Реализация файловой памяти ostis-платформы*, внутри которой представляются внешние информационные конструкции, не принадлежащие SC-коду, то есть содержимое внутренних файлов *ostis*-системы, но дополнительно описывающие, поясняющие и детализирующие sc-конструкции логико-семантических моделей *ostis*-систем.

Реализация памяти ostis-платформы

- := [Реализация sc-машины]
- := [Реализация sc-памяти и файловой памяти ostis-платформы]
- ∈ неатомарный многократно используемый компонент *ostis*-систем
- ∈ зависимый многократно используемый компонент *ostis*-систем
- ⇒ декомпозиция программной системы*:
 - {• Реализация sc-памяти ostis-платформы
 - Реализация файловой памяти ostis-платформы
- ⇒ зависимости компонента*:
 - {• Библиотека методов и структур данных GLib
 - Библиотека методов и структур данных C++ Standart Library
 - Реализация файловой памяти ostis-платформы

Стоит отметить, что при переходе с *Реализации памяти ostis-платформы* на её аппаратную реализацию *файловую память ostis*-системы целесообразно будет реализовывать на основе традиционной линейной памяти (во всяком случае, на первых этапах развития семантического компьютера). Текущий вариант *Реализации памяти ostis-платформы* является открытым и доступен на [SoftISNSP-эл.](#).

§ 6.3.3. Реализация sc-памяти в Программной платформе *ostis*-систем

- ⇒ подраздел*:
 - Пункт 6.3.3.1. Формальная спецификация sc-языка внутреннего представления sc-конструкций в sc-памяти ostis-платформы
 - Пункт 6.3.3.2. Синтаксис SCin-кода
 - Пункт 6.3.3.3. Денотационная семантика SCin-кода
 - Пункт 6.3.3.4. Описание процесса и пример трансляции sc-текста в sc-память ostis-платформы
 - Пункт 6.3.3.5. Достоинства и недостатки текущего варианта Реализации sc-памяти и языка внутреннего представления sc-конструкций в sc-памяти ostis-платформы
- ⇒ ключевой знак*:
 - Реализация sc-памяти ostis-платформы
 - SCin-код
 - := [scin-текст]
 - Алфавит SCin-кода[^]
 - элемент sc-памяти
 - элемент sc-памяти, соответствующий sc-узлу
 - элемент sc-памяти, соответствующий sc-дуге
 - элемент sc-памяти, имеющий нулевой sc-адрес
 - sc-адрес
 - sc-адрес*
 - Синтаксис SCin-кода
 - Денотационная семантика SCin-кода

Под текущей *Реализацией sc-памяти ostis-платформы* понимается компонент программной модели, осуществляющий хранение sc-конструкций и доступ к ним через программный интерфейс.

В рамках данного *Программного варианта реализации ostis-платформы sc-память* моделируется в виде набора *сегментов*, каждый из которых представляет собой фиксированного размера упорядоченную последовательность *элементов sc-памяти*, каждый из которых соответствует конкретному sc-элементу. В настоящее время каждый сегмент состоит из $2^{16} - 1 = 65535$ элементов sc-памяти. Каждый сегмент состоит из набора структур данных, описывающих конкретные sc-элементы (элементов sc-памяти). Независимо от типа описываемого sc-элемента каждый элемент sc-памяти имеет фиксированный размер (в текущий момент – 36 байт), что обеспечивает удобство их хранения.

Выделение *сегментов sc-памяти* позволяет, с одной стороны, упростить адресный доступ к *элементам sc-памяти*, с другой стороны – реализовать возможность выгрузки части sc-памяти из оперативной памяти на файловую систему при необходимости. Во втором случае сегмент sc-памяти становится минимальной (атомарной) выгружаемой частью sc-памяти. Механизм выгрузки сегментов реализуется в соответствии с существующими принципами организации виртуальной памяти в современных операционных системах.

Максимально возможное число сегментов ограничивается настройками программной реализации sc-памяти (в настоящее время по умолчанию установлено количество $2^{16} - 1 = 65535$ sc-сегментов, но в общем случае оно может быть другим). Таким образом, технически максимальное количество хранимых sc-элементов в текущей реализации составляет около 4.3×10^9 sc-элементов. По умолчанию все сегменты физически располагаются в оперативной памяти, если объема памяти не хватает, то предусмотрен механизм выгрузки части sc-сегментов на жесткий диск (механизм виртуальной памяти).

Текущий вариант *Реализации sc-памяти ostis-платформы* предполагает возможность сохранения состояния (слепка) памяти на жесткий диск и последующей загрузки из ранее сохраненного состояния. Такая возможность необходима для перезапуска системы, в случае возможных сбоев, а также при работе с исходными текстами базы знаний, когда сборка из исходных текстов сводится к формированию слепка состояния памяти, который затем помещается в *Реализации sc-памяти ostis-платформы*.

Реализация sc-памяти ostis-платформы

- := [Программный вариант реализации графодинамической ассоциативной памяти в Программной платформе ostis-систем]
- Є реализация sc-памяти
- Є неатомарный многократно используемый компонент ostis-систем
- Є зависимый многократно используемый компонент ostis-систем
- ⇐ программная модель*:
 - sc-память
 - ⇐ ключевой знак*:
 - Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем
 - ⇐ семейство подмножеств*:
 - сегмент sc-памяти
 - := [страница sc-памяти]
 - ⇐ семейство подмножеств*:
 - элемент sc-памяти
 - ⇒ зависимости компонента*:
 - {• Библиотека методов и структур данных GLib
 - Библиотека методов и структур данных C++ Standart Library
 - ⇒ используемый язык программирования*:
 - C
 - C++
 - ⇒ внутренний язык*:
 - SCin-код

В общем случае *sc-память* может быть реализовано по-разному. Так, например, другой вариант sc-памяти ostis-платформы можно реализовать при помощи программной реализации памяти платформы Neo4j. Отличие такого возможного варианта реализации sc-памяти от текущего состоит в том, что хранение графовых конструкций и управление потоков операций над ними должно осуществляться в большей мере средствами, предоставляемыми платформой Neo4j, в том же время представление графовых конструкций должно реализовываться по-своему, поскольку зависит от синтаксиса SC-кода.

Такую модель sc-памяти достаточно просто описывать на sc-языке, то есть на подъязыке SC-кода. Такой язык позволяет описывать то, как внутри памяти ostis-платформы представляются тексты языка, на этом же языке (!). При этом соблюдается не только унификация представления информации, обрабатываемой ostis-платформой, и

информации, описывающую саму ostis-платформу, но и даются возможности для расширения и использования языка в процессе эволюции ostis-платформы и её компонентов, в том числе в процессе эволюции *Реализации sc-памяти ostis-платформы*.

Далее будет рассмотрена формальная спецификация такого sc-языка внутреннего представления sc-конструкций в памяти ostis-системы.

Пункт 6.3.3.1. Формальная спецификация sc-языка внутреннего представления sc-конструкций в sc-памяти ostis-платформы

Раннее упоминалось об sc-языках, которые используются при описании компонентов текущего *Программного варианта реализации ostis-платформы*. Одним из таких языков будем называть sc-языком внутреннего представления конструкций SC-кода, или, кратко, *SCin-кодом (Semantic Code interior)*. Sc-память текстов SC-кода (как некоторую абстрактную модель, по которой можно реализовать sc-память ostis-платформы) можно рассматривать как подмножество scin-текста.

SCin-код

:= [Semantic Code interior]
 := [Язык внутреннего смыслового представления SC-кода внутри памяти ostis-системы]
 := [метаязык описания представления SC-кода внутри памяти ostis-системы]
 ⇒ часто используемый неосновной внешний идентификатор sc-элемента*: [scin-текст]
 ∈ имя нарицательное
 ∈ абстрактный язык
 ∈ метаязык
 ∈ sc-язык
 ⊆ SC-код
 ⊇ sc-память

Пункт 6.3.3.2. Синтаксис SCin-кода

Синтаксис SCin-кода задается: (1) Алфавитом SCin-кода, (2) отношением инцидентности *sc-адрес**. Важно отметить, что Синтаксис SCin-кода, как и синтаксис других sc-языков, является подмножеством Синтаксиса SCin-кода, то есть Алфавит SCin-кода[^], а также отношение инцидентности *sc-адрес** являются подмножествами Алфавит SC-кода[^] и отношений инцидентности SC-кода см. Главу 2.2. *Смысловое представление знаний в памяти ostis-систем* соответственно, при этом правила Синтаксиса SCin-кода включают правила Синтаксиса SC-кода и правила, которые уточняют нюансы Синтаксиса SCin-кода.

Алфавит SCin-кода[^]

:= [синтаксический тип элемента sc-памяти]
 := [Множество типов элементов sc-памяти]
 ⇐ алфавит*: SCin-код
 = {• элемент sc-памяти, соответствующий sc-узлу
 • элемент sc-памяти, соответствующий sc-дуге
 • элемент sc-памяти, имеющий нулевой sc-адрес
 ∈ синглетон
 }

Алфавит SCin-кода[^] состоит из трёх синтаксически выделяемых типов элементов sc-памяти: элемента sc-памяти, соответствующего sc-узлу общего вида, элемента sc-памяти, соответствующего sc-дуге общего вида и элемента sc-памяти, имеющего нулевой sc-адрес. Такой алфавит не только позволяет задавать в памяти минимальный набор объектов с которым можно производить вычислительные операции, но и, при необходимости, удобен для расширения. Так, например, в данный алфавит языка можно расширить, добавив в его элемент *sc-памяти, соответствующий внутреннему файлу ostis-системы* либо элемент *sc-памяти, соответствующий sc-ребру*.

элемент sc-памяти, соответствующий sc-элементу
 ∈ sc-элемент

- \coloneqq [элемент sc-памяти]
- \coloneqq [ячейка sc-памяти]
- \coloneqq [образ sc-элемента в рамках sc-памяти]
- \coloneqq [структура данных, каждый экземпляр которой в рамках sc-памяти соответствует одному sc-элементу]
- \Rightarrow разбиение*:
- Алфавит SCin-кода[^]

Отношение *sc-адрес** определяется как взаимно однозначное соответствие, первым компонентом каждой ориентированной пары которого является некоторый элемент sc-памяти, соответствующей некоторому sc-элементу, а вторым компонентом является sc-адрес этого элемента *sc-памяти*. То есть каждый элемент *sc-памяти* имеет уникальный *sc-адрес* как некоторый идентификатор, с помощью которого определяется *的独特性* элемента *sc-памяти*.

Каждый элемент sc-памяти в текущей реализации может быть однозначно задан его адресом (*sc-адресом*), состоящим из номера sc-сегмента и номера элемента *sc-памяти* в рамках sc-сегмента. Таким образом, *sc-адрес* служит уникальными координатами элемента *sc-памяти* в рамках *Реализации sc-памяти*.

Для каждого sc-адреса можно взаимно однозначно поставить в соответствие некоторый хэш, полученный в результате применения специальной хэш-функции над этим sc-адресом. Хэш является неотрицательным целым числом и является результатом преобразования номера sc-сегмента sc-памяти *si*, в котором располагается sc-элемент, и номера этого sc-элемента sc-памяти *ei* в рамках этого сегмента *si*. В рамках sc-памяти используется единственная хеш-функция для получения хеша sc-адреса sc-элемента и задаётся как $f(si, ei) = si \ll 16 \vee ei \wedge 0xffff$, где операция \ll – операция логического битового сдвига влево левого аргумента на количество единиц, заданное правым аргументом, относительно этой операции, операция \vee – операция логического ИЛИ, операция \wedge – операция логического И, число $0xffff$ – число 65535, представленное в шестнадцатеричном виде и обозначающее максимальное количество sc-элементов в одном сегменте sc-памяти. Числовое выражение *sc-адреса* является 32-битовым целым числом.

sc-адрес

- \coloneqq [адрес элемента sc-памяти, соответствующего заданному sc-элементу, в рамках текущего состояния реализации sc-памяти в составе программной модели sc-памяти]
- \in 32-битовое целое число

В рамках любой *реализации sc-памяти* должен существовать набор *синтаксических и семантических классов элементов sc-памяти* (меток), которые:

- задают тип элемента на уровне платформы и не имеют соответствующей sc-дуги принадлежности (а точнее – базовой sc-дуги), явно хранимой в рамках sc-памяти (ее наличие подразумевается, однако она не хранится явно, поскольку это приведет к бесконечному увеличению числа sc-элементов, которые необходимо хранить в sc-памяти);
- могут быть представлены в виде параметров соответствующих элементов sc-памяти, то есть множеством таких элементов, каждый из которых имеет "метку выраженную некоторым числовым значением";
- могут уточнять тип элементов sc-памяти с той степенью детализации, которая необходима чтобы, например, при совершении операции поиска с помощью таких классов элементов можно было легко определить класс конкретного из них.

С этой целью, в SCin-коде выделяется базовая синтаксическая классификация его элементов. Для того чтобы представлять и хранить любые конструкции SC-кода достаточно иметь только два базовых класса элементов sc-памяти, при этом остальные классы элементов sc-памяти можно добавить в расширенной версии SCin-кода и тем самым дореализовать необходимую логику на уровне конкретной реализации sc-памяти.

Синтаксическая классификация элементов SCin-кода

\supseteq
{}

Элемент sc-памяти, соответствующий sc-элементу

- \Rightarrow разбиение*:
 - {• элемент sc-памяти, соответствующий sc-узлу
 - элемент sc-памяти, соответствующий sc-дуге
 - элемент sc-памяти, имеющий нулевой sc-адрес
- }
- }

Стоит отметить, что все классы элементов sc-памяти, входящие в состав синтаксической классификации элементов SCin-кода, являются синтаксически выделяемыми классами элементов SCin-кода.

Несмотря на то, что отношение *sc-адрес** позволяет полностью описать связи элементов sc-памяти ostis-системы, для спецификации представления конструкций SC-кода внутри памяти ostis-системы только одного отношения *sc-адрес** не всегда достаточно, чтобы полностью точно и ясно указывать связи между элементами sc-памяти, соответствующими sc-элементам этих конструкций. Поэтому на практике при описании представления конструкций SC-кода внутри памяти ostis-системы необходимо использовать более частные отношения этого базового отношения, например, такие, как *sc-адрес элемента sc-памяти, соответствующего выходящей sc-дуге из заданного sc-элемента**, *sc-адрес элемента sc-памяти, соответствующего входящей sc-дуге в заданный sc-элемент**, *sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-дуги**.

*sc-адрес**

⇒ разбиение*:

- {• *sc-адрес элемента sc-памяти, соответствующего выходящей sc-дуге из заданного sc-элемента**
- *sc-адрес элемента sc-памяти, соответствующего входящей sc-дуге в заданный sc-элемент**
- *sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-дуги**

}

Отношение *sc-адрес элемента sc-памяти, соответствующего выходящей sc-дуге из заданного sc-элемента** определяется как бинарное ориентированное отношение, первым компонентом каждой ориентированной пары которого является некоторый элемент sc-памяти, соответствующей некоторому sc-элементу, из которого данная sc-дуга выходит, а вторым компонентом является sc-адрес этой выходящей sc-дуги. Частными видами этого отношения являются отношение *sc-адреса элемента sc-памяти, соответствующего начальной выходящей sc-дуге из заданного sc-элемента**, отношение *sc-адреса элемента sc-памяти, соответствующего следующей выходящей sc-дуге из заданного sc-элемента** и отношение *sc-адрес элемента sc-памяти, соответствующего предыдущей выходящей sc-дуге из заданного sc-элемента**.

*sc-адрес элемента sc-памяти, соответствующего выходящей sc-дуге из заданного sc-элемента**

⇒ разбиение*:

- {• *sc-адрес элемента sc-памяти, соответствующего начальной выходящей sc-дуге из заданного sc-элемента**
- *sc-адрес элемента sc-памяти, соответствующего следующей выходящей sc-дуге из заданного sc-элемента**
- *sc-адрес элемента sc-памяти, соответствующего предыдущей выходящей sc-дуге из заданного sc-элемента**

}

Отношение *sc-адрес элемента sc-памяти, соответствующего входящей sc-дуге в заданный sc-элемент** определяется как бинарное ориентированное отношение, первым компонентом каждой ориентированной пары которого является некоторый элемент sc-памяти, соответствующей некоторому sc-элементу, в который данная sc-дуга входит, а вторым компонентом является sc-адрес этой входящей sc-дуги. Частными видами этого отношения являются отношение *sc-адрес элемента sc-памяти, соответствующего начальной входящей sc-дуге в заданный sc-элемент**, отношение *sc-адрес элемента sc-памяти, соответствующего следующей входящей sc-дуге в заданный sc-элемент** и отношение *sc-адрес элемента sc-памяти, соответствующего предыдущей входящей sc-дуге в заданный sc-элемент**.

*sc-адрес элемента sc-памяти, соответствующего входящей sc-дуге в заданный sc-элемент**

⇒ разбиение*:

- {• *sc-адрес элемента sc-памяти, соответствующего начальной входящей sc-дуге в заданный sc-элемент**
- *sc-адрес элемента sc-памяти, соответствующего следующей входящей sc-дуге в заданный sc-элемент**
- *sc-адрес элемента sc-памяти, соответствующего предыдущей входящей sc-дуге в заданный sc-элемент**

}

Отношение *sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-дуги** определяется как бинарное ориентированное отношение, первым компонентом каждой ориентированной пары которого является некоторый элемент sc-памяти, соответствующий некоторому sc-элементу, являющейся sc-дугой, а вторым компонентом является sc-адрес некоторого инцидентного ей sc-элемента. Частными видами этого отношения являются отношение *sc-адреса элемента sc-памяти, соответствующего начальному sc-элементу sc-дуги** и отношение *sc-адрес элемента sc-памяти, соответствующего конечному sc-элементу sc-дуги**.

*sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-дуги**

⇒ разбиение*:

- {• sc-адрес элемента sc-памяти, соответствующего начальному sc-элементу sc-дуги*
 - sc-адрес элемента sc-памяти, соответствующего конечному sc-элементу sc-дуги*
- }

Sc-адрес никак не учитывается при обработке базы знаний на семантическом уровне и необходим только для обеспечения доступа к соответствующей структуре данных, хранящейся в линейной памяти на уровне *Реализации sc-памяти ostis-платформы*.

В общем случае sc-адрес элемента sc-памяти, соответствующего заданному sc-элементу, может меняться, например, при пересборке базы знаний из исходных текстов и последующем перезапуске системы. При этом sc-адрес элемента sc-памяти, соответствующего заданному sc-элементу, непосредственно в процессе работы системы в текущей реализации меняться не может. Для простоты будем говорить "sc-адрес sc-элемента", имея в виду sc-адрес элемента sc-памяти, однозначно соответствующего данному sc-элементу.

На синтаксические конструкции SCin-кода, кроме ограничения самого SC-кода, накладываются дополнительные ограничения:

- Для каждого элемента sc-памяти, соответствующего sc-элементу взаимно однозначно ставится в соответствие его sc-адрес.
- Для каждого элемента sc-памяти, соответствующего sc-узлу, существует одна и только одна пара отношения sc-адреса элемента sc-памяти, соответствующего начальной выходящей sc-дуге из заданного sc-элемента* и одна и только одна пара отношения sc-адреса элемента sc-памяти, соответствующего начальной входящей sc-дуге в заданный sc-элемент*.
- Для каждого элемента sc-памяти, соответствующего выходящей sc-дуге из заданного sc-элемента (элемента sc-памяти, соответствующего входящей sc-дуге в заданный sc-элемент), существует не более чем одна пара отношения sc-адреса элемента sc-памяти, соответствующего следующей выходящей sc-дуге из заданного sc-элемента* (sc-адреса элемента sc-памяти, соответствующего следующей входящей sc-дуге в заданный sc-элемент*) и не более чем одна пара отношения sc-адреса элемента sc-памяти, соответствующего предыдущей выходящей sc-дуге из заданного sc-элемента* (sc-адреса элемента sc-памяти, соответствующего предыдущей входящей sc-дуге в заданный sc-элемент*).
- Для каждого элемента sc-памяти, соответствующего sc-дуге, который является вторым компонентом каждой пары отношения sc-адреса элемента sc-памяти, соответствующего начальной выходящей sc-дуге из заданного sc-элемента* (sc-адреса элемента sc-памяти, соответствующего начальной входящей sc-дуге в заданный sc-элемент*) существует только и только одна пара sc-адреса элемента sc-памяти, соответствующего следующей выходящей sc-дуге из заданного sc-элемента* (sc-адреса элемента sc-памяти, соответствующего следующей входящей sc-дуге в заданный sc-элемент*).

Пункт 6.3.3.3. Денотационная семантика SCin-кода

Согласно вышесказанному, для каждого класса sc-элементов должна существовать программная модель класса элементов sc-памяти, которая удовлетворяет всем перечисленным требованиям. Поэтому важно, чтобы Алфавит SCin-кода изначально был полон, чтобы погрузить не только sc-конструкции Ядра SC-кода, но и его расширенной версии. Для этого разработаны семантические классы элементов sc-памяти, спецификация которых представляется в виде *Семантической классификации элементов SCin-кода*.

Семантическая классификация элементов SCin-кода

⇒
{

Элемент sc-памяти, соответствующий sc-элементу

⇒ разбиение*:

Типология элементов sc-памяти по признаку константности^

- = {• элемент sc-памяти, соответствующий sc-константе
 - элемент sc-памяти, соответствующий sc-переменной
 - элемент sc-памяти, соответствующий sc-метапеременной
}

⇒ разбиение*:

Типология элементов sc-памяти по признаку постоянности^

- = {• элемент sc-памяти, соответствующий постоянному sc-элементу
 - элемент sc-памяти, соответствующий временному sc-элементу
}

⇒ разбиение*:

Типология элементов sc-памяти по признаку доступности[^]

- := [класс уровня доступа к элементу sc-памяти]
- = {• элемент sc-памяти, соответствующий sc-элементу, на котором разрешено право чтения
 - элемент sc-памяти, соответствующий sc-элементу, на котором разрешено право записи
}

⇒ включение*:

элемент sc-памяти, соответствующий внутреннему файлу ostis-системы

элемент sc-памяти, соответствующий sc-узлу общего вида

⇒ разбиение*:

Структурная типология элементов sc-памяти, соответствующих sc-узлам[^]

- = {• элемент sc-памяти, соответствующий sc-узлу, обозначающему небинарную sc-связку
 - элемент sc-памяти, соответствующий sc-классу
 - элемент sc-памяти, соответствующий sc-узлу, обозначающему класс классов
 - элемент sc-памяти, соответствующий sc-структуре
 - элемент sc-памяти, соответствующий sc-узлу, обозначающему ролевое отношение
 - элемент sc-памяти, соответствующий sc-узлу, обозначающему неролевое отношение
 - элемент sc-памяти, соответствующий sc-узлу, обозначающему первичную сущность
}

⇒ разбиение*:

Структурная типология элементов sc-памяти, соответствующих sc-дугам[^]

- = {• элемент sc-памяти, соответствующий sc-дуге принадлежности
 - элемент sc-памяти, соответствующий sc-дуге общего вида
}

⇒ разбиение*:

Типология элементов sc-памяти, соответствующих sc-дугам принадлежности, по типу обозначаемой принадлежности[^]

- = {• элемент sc-памяти, соответствующий sc-дуге позитивной принадлежности
 - элемент sc-памяти, соответствующий sc-дуге нечёткой принадлежности
 - элемент sc-памяти, соответствующий sc-дуге негативной принадлежности
}

}

Все семантически и синтаксически выделяемые классы элементов sc-памяти, а также всевозможные подклассы этих классов являются экземплярами (элементами) sc-класса.

Все перечисленные классы на уровне программной реализации выражаются в виде битов в битовой строке, описываемой в каждом элементе sc-памяти.

В текущий момент sc-ребра хранятся так же, как sc-дуги, то есть имеют начальный и конечный sc-элементы, отличие заключается только в *синтаксическом типе sc-элемента*. Это приводит к ряду неудобств при обработке, но sc-ребра используются в настоящем времени достаточно редко.

Спецификация SCin-кода является объединением спецификаций его элементов. Для каждого элемента, связей между элементами и их свойствами накладываются ограничения в виде синтаксических правил, описанных выше.

элемент sc-памяти, соответствующий sc-элементу

⇒ спецификация*:

- {}
 - ▷ сужение отношения по первому домену(спецификация знака*, элемент sc-памяти, соответствующий sc-узлу)*
 - ▷ сужение отношения по второму домену(спецификация знака*, элемент sc-памяти, соответствующий sc-дуге)*

Каждый элемент sc-памяти, соответствующий некоторому sc-элементу, описывается его синтаксическим типом (меткой), а также независимо от типа указывается sc-адрес первой входящей в данный sc-элемент sc-дуги и первой выходящей из данного sc-элемента sc-дуги (могут быть пустыми, если таких sc-дуг нет). Оставшиеся байты в зависимости от типа соответствующего sc-элемента (sc-узел или sc-дуга) могут использоваться для хранения спецификации sc-дуги. Также *sc-адрес первой sc-дуги, выходящей из данного sc-элемента** и *sc-адрес первой sc-дуги, входящей в данный sc-элемент** в общем случае могут отсутствовать (быть нулевыми, "пустыми"), но размер sc-элемента в байтах останется тем же.

элемент sc-памяти, соответствующий sc-узлу

⇒ спецификация*:

- {• класс элемента sc-памяти, соответствующего sc-узлу
 - класс уровня доступа к элементу sc-памяти
 - sc-адрес*
 - sc-адрес первой sc-дуги, выходящей из данного sc-элемента*
 - sc-адрес первой sc-дуги, входящей в данный sc-элемент*
- }

элемент sc-памяти, соответствующий sc-дуге

⇒ спецификация*:

- {• класс элемента sc-памяти, соответствующего sc-дуге
 - класс уровня доступа к элементу sc-памяти
 - sc-адрес*
 - sc-адрес элемента sc-памяти, соответствующего начальному sc-элементу sc-дуги*
 - sc-адрес элемента sc-памяти, соответствующего конечному sc-элементу sc-дуги*
 - sc-адрес элемента sc-памяти, соответствующего начальной выходящей sc-дуге из заданного sc-элемента*
 - sc-адрес элемента sc-памяти, соответствующего начальной входящей sc-дуге в заданный sc-элемент*
 - sc-адрес элемента sc-памяти, соответствующего следующей выходящей sc-дуге из заданного sc-элемента*
 - sc-адрес элемента sc-памяти, соответствующего следующей входящей sc-дуге в заданный sc-элемент*
 - sc-адрес элемента sc-памяти, соответствующего предыдущей выходящей sc-дуге из заданного sc-элемента*
 - sc-адрес элемента sc-памяти, соответствующего предыдущей входящей sc-дуге в заданный sc-элемент*
- }

В текущей Реализации sc-памяти классы уровня доступа используются для того, чтобы обеспечить возможность ограничения доступа некоторых процессов в sc-памяти к некоторым элементам, хранимым в sc-памяти. Каждый элемент sc-памяти принадлежит один из двух классов: классу элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право чтения и классу элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право записи. Каждая из которых выражается числом от 0 до 255.

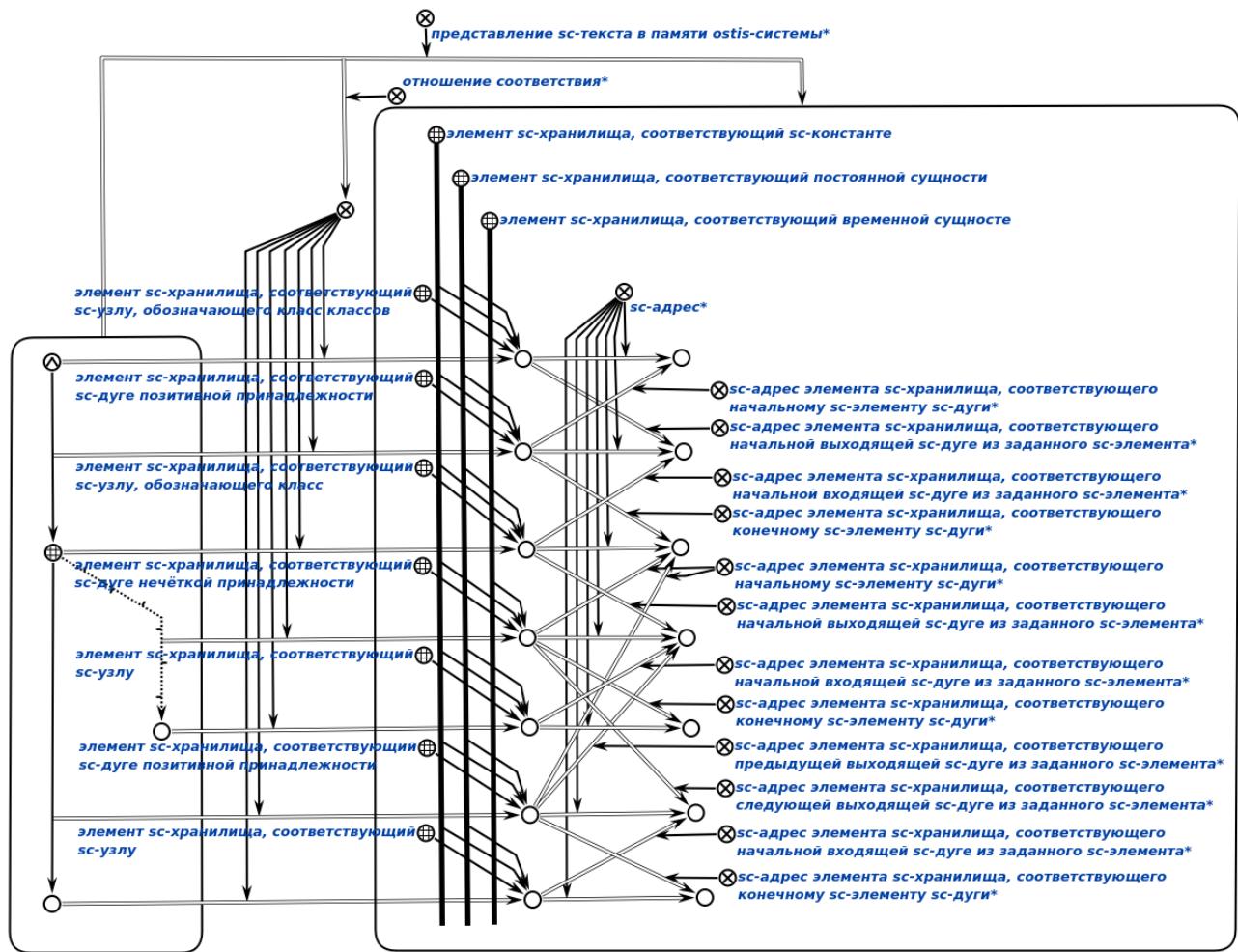
Таким образом нулевое значение числовых выражений класса элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право чтения и класса элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право записи означает, что любой процесс может получить неограниченный доступ к данному элементу sc-памяти.

Пункт 6.3.3.4. Описание процесса и пример трансляции sc-текста в sc-память ostis-платформы

Погрузка sc-конструкции в память ostis-системы означает трансляцию каждого sc-элемента этой sc-конструкции и связей инцидентности между этими sc-элементами в память ostis-системы, то есть трансляцию синтаксической структуры sc-конструкции в соответствующее представление внутри памяти ostis-системы. В общем случае, алгоритм погрузки любой произвольной sc-конструкции в память ostis-системы состоит из следующих этапов:

- Выделение sc-узлов и внутренних файлов sc-конструкции и сохранение в соответствующие ячейки памяти ostis-системы;
- Выделение всех свободных sc-коннекторов (т.е. sc-коннекторов, у которых начальным и конечным sc-элементом не является другой sc-коннектор), сохранение всех sc-коннекторов в соответствующие ячейки памяти ostis-системы и установление связей между начальным и конечным sc-элементами этих sc-коннекторов;
- Возврат в пункт 2, если остались непогруженные sc-коннекторы;
- Погрузка содержимого всех внутренних файлов ostis-системы в её файловую память.

На рисунке [Пример спецификации представления конструкции SC-кода в памяти ostis-системы](#) изображён пример спецификации представления sc-конструкции в памяти ostis-системы. Здесь каждому элементу sc-элементу данной sc-конструкции ставится в соответствие sc-элемент, обозначающий элемент sc-памяти. Для каждого sc-элемента, обозначающего элемент sc-памяти некоторого sc-элемента заданной sc-конструкции описана своя денотационная семантика: связи между элементами sc-памяти и синтаксические и семантические классы элементов.



= Пример спецификации представления конструкции SC-кода в памяти ostis-системы

Пункт 6.3.3.5. Достоинства и недостатки текущего варианта Реализации sc-памяти и языка внутреннего представления sc-конструкций в sc-памяти ostis-платформы

Данная модель представления в памяти системы синтаксических и семантических классов sc-элементов в виде синтаксических и семантических классов элементов sc-памяти, которые соответствуют первым, обладает рядом преимуществ:

- Синтаксические и семантические классы элементов sc-памяти могут комбинироваться между собой для получения более частных классов. С точки зрения программной реализации такая комбинация может быть представлена операцией побитового сложения значений соответствующих числовых выражений классов элементов sc-памяти (здесь, в спецификации на SC-коде это можно сделать с помощью пересечения соответствующих классов). Так, например, побитовое сложение числовых выражений классов элементов sc-памяти, соответствующих sc-узлу и sc-константе в результате образуют новый класс элементов sc-памяти – элемент sc-памяти, соответствующий константному sc-узлу.
- Числовые выражения некоторых классов могут совпадать. Это сделано для уменьшения размера элемента sc-памяти за счет уменьшения максимального размера числового выражения класса этих элементов. Конфликт в данном случае не возникает, поскольку такие классы не могут комбинироваться, например элемент sc-памяти, соответствующий sc-узлу ролевого отношения и элемент sc-памяти, соответствующий sc-дуге нечёткой принадлежности.
- Важно отметить, что каждому из выделенных классов элементов (кроме классов, получаемых путем комбинации других классов) однозначно соответствует порядковый номер бита в линейной памяти, что можно заметить, глядя на соответствующие числовые выражения этих классов. Это означает, что классы элементов не включаются друг в друга (хотя в спецификации это и не так), например, указание принадлежности к классу элементов sc-памяти, соответствующих sc-дуге позитивной принадлежности не означает автоматическое указание принадлежности элементов sc-памяти, соответствующих sc-дуге принадлежности. На уровне реализации, это позволяет сделать операции комбинирования и сравнения меток более эффективными.

С точки зрения программной реализации, структура данных для хранения sc-узла и sc-дуги остается оставаться та же, но в ней меняется список полей (компонентов). Кроме того, как можно заметить каждый элемент sc-памяти (в том числе, элемент *sc-памяти, соответствующий sc-дуге*) не хранит список sc-адресов связанных с ним sc-элементов, а хранит sc-адреса одной выходящей и одной входящей дуги, каждая из которых в свою очередь хранит sc-адреса следующей и предыдущей дуг в списке исходящих и входящих sc-дуг для соответствующих элементов. Всё перечисленное позволяет:

- сделать размер такой структуры фиксированным (в настоящее время 36 байт) и не зависящим от синтаксического типа хранимого sc-элемента;
- обеспечить возможность работы с sc-элементами без учета их синтаксического типа в случаях, когда это необходимо (например, при реализации поисковых запросов вида “Какие sc-элементы являются элементами данного множества”, “Какие sc-элементы непосредственно связаны с данным sc-элементом” и т.д.);
- обеспечить возможность доступа к элементу *sc-памяти* за константное время;
- обеспечить возможность помещения элемента *sc-памяти* в процессорный кэш, что в свою очередь, позволяет ускорить обработку sc-конструкций;

§ 6.3.4. Реализация файловой памяти в Программной платформе *ostis*-систем

⇒ подраздел*:

- Пункт 6.3.4.1. Формальная спецификация sc-языка внутреннего представления конструкций, не принадлежащих SC-коду, в файловой памяти *ostis*-платформы
- Пункт 6.3.4.2. Синтаксис *SCfin*-кода
- Пункт 6.3.4.3. Денотационная семантика *SCfin*-кода
- Пункт 6.3.4.4. Описание процесса и пример трансляции внешних информационных конструкций в файловую память *ostis*-платформы
- Пункт 6.3.4.5. Достоинства и недостатки Реализации файловой памяти *ostis*-платформы и sc-языка внутреннего представления внешних информационных конструкций в файловой памяти *ostis*-платформы

⇒ ключевой знак*:

- Реализация файловой памяти *ostis*-платформы
- файловая память *ostis*-платформы
- *SCfin*-код
 - := [scfin-текст]
- Алфавит *SCfin*-кода^Δ
- элемент файловой памяти *ostis*-платформы
- Синтаксис *SCfin*-кода
- Денотационная семантика *SCfin*-кода

SC-код хоть и является универсальным средством для представления любых видов знаний, но не всегда есть необходимость погружать что-либо в графодинамическую память *ostis*-системы, по крайней мере, на ранних стадиях развития *ostis*-платформы. Это может быть объяснено и тем, что информационные конструкции, не принадлежащие SC-коду, достаточно сложны в понимании для неподготовленного пользователя *ostis*-системы. Для решения таких проблем на уровне Алфавита *SC-кода*^Δ вводится дополнительный элемент – *внутренний файл ostis-системы*. С помощью *внутренних файлов ostis-системы* можно представлять, хранить, обрабатывать и визуализировать информационные конструкции, не принадлежащие SC-коду.

Поэтому при реализации sc-памяти необходимо учитывать необходимость хранения информационных конструкций, не принадлежащих SC-коду, с помощью SC-кода. Кроме собственно *sc-памяти* Реализация *sc-памяти ostis-платформы* включает также Реализацию файловой памяти *ostis*-платформы, предназначенную для хранения содержимого *внутренних файлов ostis-систем*, то есть внешние информационные конструкции (линейные тексты, изображения, видео и так далее).

За весь период развития Программного варианта реализации *ostis*-платформы было достаточно много попыток реализовать полно функциональное и быстрое файловую память на базе популярных баз данных. Однако из всех этих решений не было учтены потенциальные проблемы при реализации поисково-навигационной подсистемы Программного варианта реализации *ostis*-платформы. Сейчас файловая память реализована своими средствами, в качестве структур данных для хранения информационных конструкций, не принадлежащих SC-коду, используются префиксные деревья *Bayer R..PrefiB-1977art* и линейные списки.

Выбор аргументируется тем, что:

- префиксные структуры достаточно просты в понимании и минимальны в своём синтаксисе;
- с помощью префиксных структур достаточно удобно хранить и обрабатывать связи типа "ключ-значение";

- доступ к значению по ключу происходит в худшем случае за длину этого ключа ***tsuruta2022cPrefixTree, Belazzougui D..FastPSiLSwA-2010art***;
- за счёт того, что префиксы склеиваются, выходит сильный выигрыш в использовании памяти.

Реализация файловой памяти *ostis*-платформы

- ∈ реализация файловой памяти на основе префиксного дерева
- ⇐ программная модель*:
 - файловая память *ostis*-платформы
- ∈ атомарный многократно используемый компонент *ostis*-систем
- ∈ зависимый многократно используемый компонент *ostis*-систем
- ⇒ зависимости компонента*:
 - {• Библиотека методов и структур данных *GLib*
 - }
- ⇒ язык представления методов*:
 - C
- ⇒ внутренний язык*:
 - *SCfin*-код

Пункт 6.3.4.1. Формальная спецификация sc-языка внутреннего представления конструкций, не принадлежащих SC-коду, в файловой памяти *ostis*-платформы

Как в случае и с sc-памятью, необходимо описать sc-язык представления информационных конструкций, не принадлежащих SC-коду, внутри файловой памяти *ostis*-платформы. Такой sc-язык будем называть sc-языком внутреннего представления информационных конструкций, не принадлежащих SC-коду, или, кратко, *SCfin*-кодом (*Semantic Code file interior*). Файловая память текстов, не принадлежащих SC-коду (как абстрактную модель, по которой можно реализовывать соответствующее ей файловую память *ostis*-платформы), можно рассматривать как подмножество *scfin*-текста.

***SCfin*-код**

- := [Semantic Code file interior]
- := [Sc-язык внутреннего смыслового представления информационных конструкций, не принадлежащих SC-коду, внутри файловой памяти *ostis*-платформы]
- := [метаязык описания представления информационных конструкций, не принадлежащих SC-коду, внутри файловой памяти *ostis*-платформы]
- ⇒ часто используемый неосновной внешний идентификатор sc-элемента*:
 - [*scfin*-текст]
 - ∈ имя нарицательное
- ∈ абстрактный язык
- ∈ метаязык
- ∈ sc-язык
- ⊂ SC-код
- ⊃ файловая память *ostis*-платформы

Пункт 6.3.4.2. Синтаксис *SCfin*-кода

Синтаксис *SCfin*-кода задается: (1) Алфавитом *SCfin*-кода, (2) отношением порядка последовательность в линейном тексте*.

Алфавит *SCfin*-кода[^]

- := [синтаксический тип элемента файловой памяти *ostis*-платформы]
- := [Множество типов элементов файловой памяти *ostis*-платформы]
- ⇐ алфавит*:
 - SCfin*-код
- = {• элемент файловой памяти *ostis*-платформы, соответствующий подстроке текста линейного языка}
- }

Алфавит SCfin-кода^λ состоит из одного синтаксически выделяемого типа элементов файловой памяти – *элемента файловой памяти ostis-платформы, соответствующий подстроке текста линейного языка*.

элемент файловой памяти ostis-платформы, соответствующий подстроке текста линейного языка

\in sc-элемент

\coloneqq [элемент файловой памяти ostis-платформы]

\coloneqq [ячейка файловой памяти ostis-платформы]

\coloneqq [образ подстроки информационной конструкции, не принадлежащей SC-коду, в рамках файловой памяти ostis-платформы]

Отношение *последовательности в линейном тексте** определяется как бинарное ориентированное отношение порядка, компонентами каждой ориентированной пары которого являются элементы файловой памяти ostis-платформы, соответствующие некоторым подстрокам линейного текста, в результате конкатенации которых образуется подстрока, принадлежащая этому же линейному тексту.

Синтаксис SCfin-кода достаточно прост, поскольку информационные конструкции на нём задаются при помощи Алфавита SCfin-кода, мощность которого равна 1, и единственного отношения инцидентности *последовательности в линейном тексте**. Иерархии синтаксических элементов как таковые не выделяются, поскольку в этом нет необходимости.

На уровне реализации важно выделить семантические классы элементов *файловой памяти ostis-платформы, соответствующий подстроке текста линейного языка*, которые обозначают некоторую префиксную или постфиксную часть целой информационной конструкции.

Пункт 6.3.4.3. Денотационная семантика SCfin-кода

Семантическая классификация элементов SCfin-кода

\supseteq

{

элемент файловой памяти ostis-платформы, соответствующий текста линейного языка

\Rightarrow разбиение*:

Типология элементов по месту расположения подстроки в линейном тексте

- = {• *элемент файловой памяти ostis-платформы, соответствующий префиксной подстроке текста линейного языка*
- *элемент файловой памяти ostis-платформы, соответствующий постфиксной подстроке текста линейного языка*

}

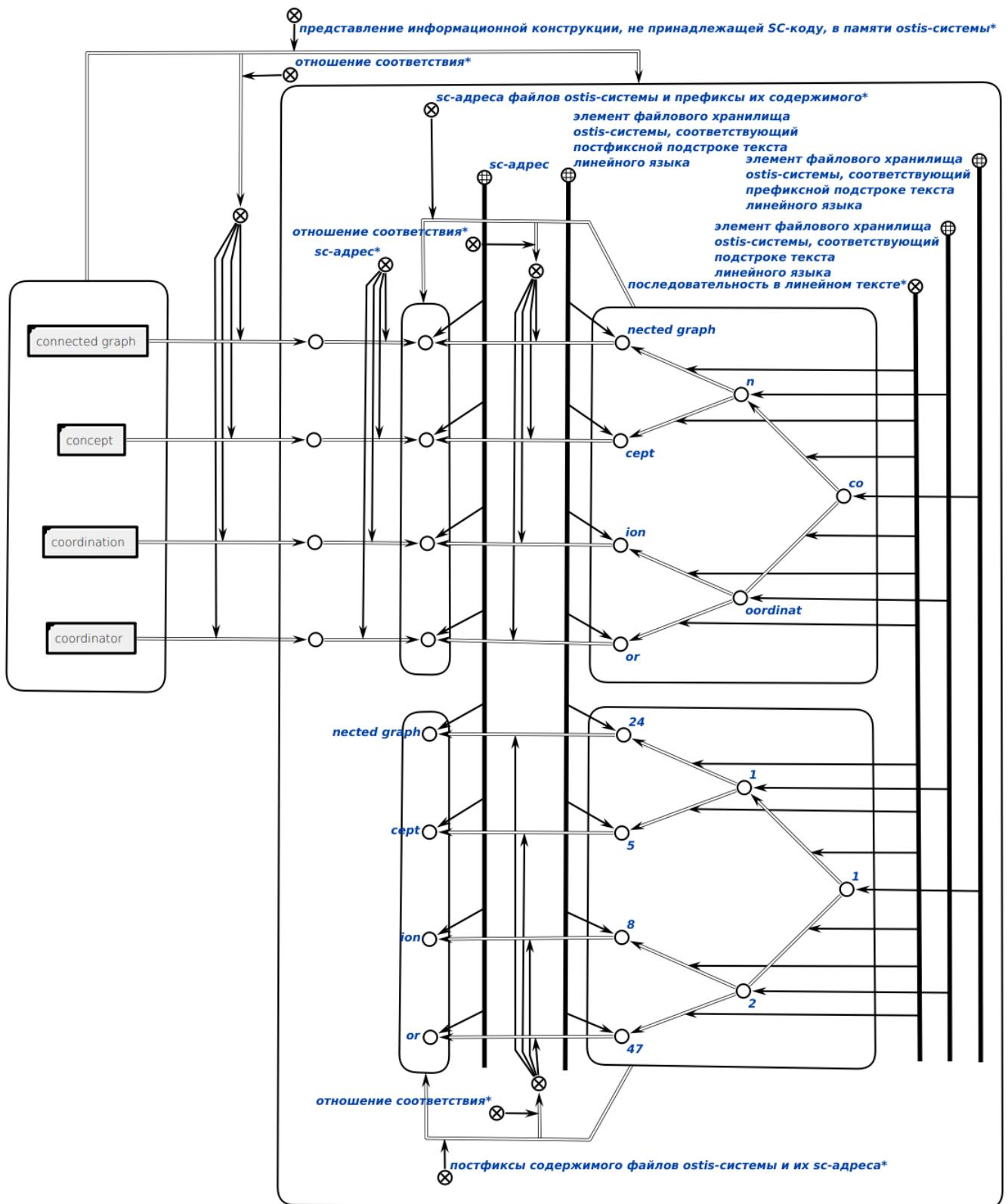
}

На SCfin-коде достаточно просто задавать информационные конструкции любых линейных текстов. Однако с точки зрения реализуемой модели sc-памяти есть необходимость задавать не столько форму информационных конструкций, не принадлежащих SC-коду, внутри файловой памяти ostis-платформы, сколько связи между этими внешними информационными конструкциями файлами ostis-системы, являющихся знаками SC-кода. Одновременно должны быть реализованы на уровне sc-памяти и метод получения файлов ostis-системы, которые содержат заданную внешнюю информационную конструкцию, и методы получения внешних информационных конструкций из заданных файлов ostis-системы.

Пункт 6.3.4.4. Описание процесса и пример трансляции внешних информационных конструкций в файловую память ostis-платформы

На рисунке [Пример спецификации представления информационных конструкций, не принадлежащих SC-коду, в памяти ostis-системы](#) показано представление информационных конструкций, не принадлежащих SC-коду и соответствия между файлами ostis-системы и информационными конструкциями. С помощью отношения *множество sc-адресов файлов ostis-системы по префиксам их содержимого** задаётся бинарная ориентированная пара, первым компонентом которой является префиксная структура, элементами которой являются подстроки внешних информационных конструкций, а вторым компонентом – множество соответствующих им sc-адресов файлов ostis-системы, а с помощью отношения *множество постфиксов содержимого файлов ostis-системы по их*

*sc-адресам** задаётся бинарная ориентированная пара, первым компонентом которой является префиксная структура, элементами которой являются подстроки sc-адресов файлов ostis-системы, представленных в строковом виде, а вторым компонентом – множество соответствующих им постфиксов внешних информационных конструкций префиксной структуры, являющейся первым компонентом каждой пары отношения множества *sc-адресов файлов ostis-системы по префиксам их содержимого**.



= Пример спецификации представления информационных конструкций, не принадлежащих SC-коду, в памяти ostis-системы

Пункт 6.3.4.5. Достоинства и недостатки Реализации файловой памяти ostis-платформы и sc-языка внутреннего представления внешних информационных конструкций в файловой памяти ostis-платформы

Используемая *Реализация файловой памяти ostis-платформы* полностью оправдывает себя при взаимодействии с системой. Благодаря использованию префиксных структур асимптотические сложности метода получения множества внешних информационных конструкций из заданных файлов ostis-системы и метода получения множества файлов ostis-системы по заданным внешним информационным конструкциям линейны, так как зависит от длины заданной строки и структуры префиксного дерева.

Среди общих недостатков данной *Реализации файловой памяти ostis-платформы* можно выделить следующие:

- Информационные конструкции, не принадлежащие SC-коду, пока полностью хранятся в оперативной памяти компьютерного устройства, на котором развернута платформа. Данную проблему можно решить, если в оперативной памяти хранить только первые знаки подстрок информационных конструкций, а остальные части этих подстрок хранить на уровне файлового системы.
- На данный момент информационно-поисковая подсистема реализована не полностью. *Реализация файловой памяти ostis-платформы* позволяет быстро решать задачи поиска внешних информационных конструкций по их префиксным подстрокам, однако не позволяет быстро решать задачи поиска информационных конструкций по любой подстроке, даже для которого задан некоторый шаблон-образец.

Описанные проблемы будут решены в рамках будущей версии *Программного варианта реализации ostis-платформы*.

§ 6.3.5. Реализация подсистемы взаимодействия с sc-памятью на основе языка JSON

⇒ подраздел*:

- Пункт 6.3.5.1. Формальная спецификация sc-языка представления сообщений между подсистемами Программного варианта реализации ostis-платформы. SC-JSON-код
- Пункт 6.3.5.2. Синтаксис SC-JSON-кода
- Пункт 6.3.5.3. Грамматика SC-JSON-кода
- Пункт 6.3.5.4. Серверная система на основе *WebSocket* и *JSON*, обеспечивающая сетевой доступ к sc-памяти

⇒ ключевой знак*:

- Реализация подсистемы взаимодействия с sc-памятью на основе языка *JSON*
- Серверная система на основе *WebSocket* и *JSON*, обеспечивающая сетевой доступ к sc-памяти
- Реализация клиентской системы на языке программирования *Python*
- Реализация клиентской системы на языке программирования *TypeScript*
- Реализация клиентской системы на языке программирования *C#*
- Реализация клиентской системы на языке программирования *Java*
- SC-JSON-код
- команда на SC-JSON-коде
- ответ на команду на SC-JSON-коде

В общем случае подсистема взаимодействия ostis-платформы с внешней средой может быть реализована по-разному. Так, например, до момента реализации текущей подсистемы ранее существовал её аналог на языке программирования *Python*, который использовал протокол *HTTP* и бинарное представление команд и ответов. Поэтому можно говорить о большом разнообразии таких подсистем, которые будут составлять множество всевозможных *Реализаций подсистемы взаимодействия с внешней средой с использованием сетевых языков*.

Данная *Реализация подсистемы взаимодействия с sc-памятью на основе языка JSON* позволяет ostis-системам взаимодействовать с системами из внешней среды на основе общепринятого транспортного формата передачи данных *JSON* и предоставляет API для доступа к sc-памяти платформы интерпретации sc-моделей.

Реализация подсистемы взаимодействия с внешней средой с использованием сетевых языков

⇒ декомпозиция программной системы*:

- {• Реализация подсистемы взаимодействия с внешней средой с использованием сетевых языков на основе языка *JSON*

}

Реализация подсистемы взаимодействия с sc-памятью на основе языка JSON

:= [Подсистема взаимодействия с sc-памятью на основе формата JSON]
 ∈ неатомарный многократно используемый компонент ostis-систем
 ∈ зависимый многократно используемый компонент ostis-систем
 ∈ клиент-серверная система
 ⇒ используемый язык представления методов*:

- C
- C++
- Python
- TypeScript
- C#
- Java

 ⇒ используемый язык*:

- SC-JSON-код

 ⇒ декомпозиция программной системы*:
 {• Серверная система на основе *Websocket* и *JSON*, обеспечивающая сетевой доступ к sc-памяти
 {}
 = {• Реализация клиентской системы на языке программирования Python

- Реализация клиентской системы на языке программирования TypeScript
- Реализация клиентской системы на языке программирования C#
- Реализация клиентской системы на языке программирования Java

}
 }

Взаимодействие с sc-памятью обеспечивается с помощью передачи информации на SC-JSON-коде и ведётся, с одной стороны, между сервером, являющимся частью ostis-системы, написанным на том же языке реализации этой ostis-системы и имеющим доступ к её sc-памяти, и с другой стороны множеством клиентов, которым известно о наличии сервера в пределах сети их использования. С помощью подсистемы взаимодействия с sc-памятью на основе языка JSON можно взаимодействовать с ostis-системой на таком же множестве возможных операций, как и в случае, если бы взаимодействие происходило (непосредственно) напрямую, на том же языке реализации платформы интерпретации sc-моделей компьютерных систем. При этом результат работы отличается только скоростью обработки информации. Взаимодействие программной модели sc-памяти с внешними ресурсами может осуществляться посредством специализированного программного интерфейса (API), однако этот вариант неудобен в большинстве случаев, поскольку:

- поддерживается только для очень ограниченного набора языков программирования;
- требует того, чтобы клиентское приложение, обращающееся к программной модели sc-памяти, фактически составляло с ней единое целое, таким образом исключается возможность построения распределенного коллектива ostis-систем;
- как следствие предыдущего пункта, исключается возможность параллельной работы с sc-памятью нескольких клиентских приложений.

Для того, чтобы обеспечить возможность удаленного доступа к sc-памяти не учитывая при этом языки программирования, с помощью которых реализовано конкретное клиентское приложение, было принято решение о реализации возможности доступа к sc-памяти с использованием универсального языка, не зависящего от средств реализации того или иного компонента или системы.

Среди эффективных протоколов, используемых при реализации клиент-серверных систем, стоит отметить протоколы прикладного уровня стека TCP/IP – протоколы HTTP и WebSocket [weuckets_overview](#), [naik2020study](#). Целесообразным является использование протокола WebSocket, это объясняется следующими причинами:

- WebSocket выгодно использовать в веб-ориентированных системах, в которых данные, отправляемые сервером, представляются или сохраняются на стороне клиента. В WebSocket данные постоянно передаются через одно и то же открытое соединение, поэтому коммуникация по протоколу WebSocket быстрее чем коммуникация по HTTP [Tomasseti M.aAotPoWiVPLaL-2021art](#), [weuckets_overview](#). Это очень важно с точки зрения проектирования Экосистемы OSTIS, которые могут состоять из десятков тысяч различного вида ostis-систем.
- Поскольку в основе ostis-систем лежит идея агентно-ориентированной обработки знаний (асинхронной обработки), а память таких систем должна быть одновременно распределённой и общей, то необходимо, чтобы каждая из них (в частности, самостоятельная ostis-система) могла общаться с другими ostis-системами. Причём такое общение может и должно происходить на условиях инициирования событий в памяти этих систем. Отсюда следует однозначный вывод, что протокол HTTP не может быть использован в передовых интеллектуальных системах нового поколения по причине односторонности создаваемого им соединения.

В качестве языка коммуникации систем был разработан строковый язык на базе языка JSON [pezoa2016foundations](#), [Marrs T.Json aWPDIftW-2017bk](#) – SC-JSON-код. Его выбор объясняется гибкостью задания отношений между описываемыми им объектами.

Пункт 6.3.5.1. Формальная спецификация sc-языка представления сообщений между подсистемами Программного варианта реализации ostis-платформы. SC-JSON-код

Как говорилось ранее, подсистемы в рамках реализуемого программного варианта специализированной платформы общаются с помощью языка внешнего представления знаний SC-JSON-код. Данный язык является строковым, то есть линейным, и легкообратаемым, поскольку существует большое количество средств для обработки его надъязыка JSON.

SC-JSON-код

:= [Semantic JSON-code]
 := [Semantic JavaScript Object Notation code]
 := [Язык внешнего смыслового представления знаний на основе языка JSON]
 ⇒ часто используемый неосновной внешний идентификатор sc-элемента*: [sc-json-текст]
 ∈ имя нарицательное
 ∈ абстрактный язык
 ∈ линейный язык
 ⊂ JSON

Пункт 6.3.5.2. Синтаксис SC-JSON-кода

Синтаксис SC-JSON-кода задается: (1) Алфавитом SC-JSON-кода, (2) Грамматикой SC-JSON-кода. В алфавите SC-JSON-кода выделяется базовая синтаксическая классификация его элементов.

Синтаксическая классификация элементов SC-JSON-кода

▷= {

SC-JSON-код

⇐ семейство подмножеств*: sc-json-предложение
 ⊂ json-список json-пар
 ⇐ семейство подмножеств*: sc-json-пара*:
 ⇐ декартово произведение*: (• sc-json-строка
 • sc-json-объект
)
 ⇒ разбиение*: {• команда на SC-JSON-коде
 • ответ на команду на SC-JSON-коде
 }

sc-json-объект

⇒ разбиение*: {• sc-json-список
 • sc-json-пара
 • sc-json-литерал
 ⇒ разбиение*: {• sc-json-строка
 • sc-json-число
 }
 }
 }

Алфавит SC-JSON-кода[^] представляет собой множество всех возможных символов в SC-JSON-коде. Поскольку SC-JSON-код является линейным строковым языком представления знаний, то его алфавит включает объединение алфавитов всех языков, тексты на которых могут представлять внешние идентификаторы и/или содержимое файлов

ostis-системы, множество всех цифр и множество всех других специальных символов. Последовательности знаков алфавита могут образовывать sc-json ключевые слова, *sc-json-пары*, *sc-json-предложения* из *sc-json-пар* и *sc-json-тексты* из *sc-json-предложений*. При этом конструкции на SC-JSON-коде строятся по следующим синтаксическим правилам:

- Каждое правило *Грамматики SC-JSON-кода* описывает корректный с точки зрения *Синтаксиса SC-JSON-кода* порядок sc-json-объектов в sc-json-предложении. Совокупность правил *Грамматики SC-JSON-кода* описывает корректный с точки зрения *Синтаксиса SC-JSON-кода* порядок sc-json-предложений в sc-json-тексте. Каждое sc-json-предложение является sc-json-списком, состоящим из sc-json-пар и представляет собой команду или ответ на эту команду.
 - Каждая *команда* (*ответ на команду*) на *SC-JSON-коде* состоит из заголовка, включающего sc-json-пары описания самой команды (ответа на команду), и сообщения, различного для каждого класса команд (ответов на команды). Сообщение *команды* (*ответа на команду*) на *SC-JSON-коде* обычно представляет собой список sc-json-объектов и может не ограничиваться по мощности.
 - Каждая sc-json-пара состоит из двух элементов: ключевого слова и некоторого другого sc-json-объекта, ассоциируемого с этим ключевым словом. Набор ключевых слов в sc-json-парах определяется конкретным классом *команд* (*ответов на команды*) на *SC-JSON-коде*. Sc-json-пара начинается знаком открывающейся фигурной скобки "{" и заканчивается знаком закрывающейся фигурной скобки "}". Ключевое слово и sc-json-объект, ассоциируемый с ним, разделяются при помощи знака двоеточия ":".
 - Sc-json-строки, записанные в sc-json-текстах, начинаются и заканчиваются знаком двух ковычек “ ”.
 - Sc-json-списки, состоящие не из sc-json-пар, начинаются знаком открывающейся квадратной скобки "[" и заканчиваются знаком закрывающейся квадратной скобки "]". Sc-json-объекты в sc-json-списках разделяются запятыми ",".

Пункт 6.3.5.3. Грамматика SC-JSON-кода

Грамматика SC-JSON-кода представляет собой множество всех возможных правил, используемых при построении команд и ответов на них на SC-JSON-коде. Каждой команде *SC-JSON-кода* однозначно соответствует правило грамматики *SC-JSON-кода*. Правила *Грамматики SC-JSON-кода* позволяют правильно представлять команды на SC-JSON-коде. Каждое правило грамматики *SC-JSON-кода* представляется в виде правила на *Языке описания грамматик ANTLR* и его интерпретации на естественном языке.

Грамматика SC-JSON-кода

- ⇒ **ключевой sc-элемент'**:
Правило, задающее синтаксис команд на SC-JSON-коде
 ⇐ **синтаксическое правило***:
 команда на SC-JSON-коде
 - ⇒ **ключевой sc-элемент'**:
Правило, задающее синтаксис ответов на команды на SC-JSON-коде
 ⇐ **синтаксическое правило***:
 ответ на команду на SC-JSON-коде
 - ⇒ **Правило, задающее синтаксис команды создания sc-элементов**
 ⇐ **синтаксическое правило***:
 команда создания sc-элементов
 - ⇒ **Правило, задающее синтаксис ответа на команду создания sc-элементов**
 ⇐ **синтаксическое правило***:
 ответ на команду создания sc-элементов

Правило, задающее синтаксис команд на SC-JSON-коде представлено на рисунке [Описание Правила, задающего синтаксис команд на SC-JSON-коде](#). Класс команд на SC-JSON-коде включает команду создания sc-элементов, команду получения соответствующих типов sc-элементов, команду удаления sc-элементов, команду обработки ключевых sc-элементов, команду обработки содержимого файлов ostis-системы, команду поиска sc-конструкций, изоморфных заданному sc-шаблону, команду генерации sc-конструкции, изоморфной заданному sc-шаблону, и команду обработки sc-событий. В команду на SC-JSON-коде включаются идентификатор этой команды, тип и сообщение.

Правило, задающее синтаксис *ответа на команду* на SC-JSON-коде описывает синтаксис ответов на команды, описываемых предыдущим правилом. Класс *ответов на команды* на SC-JSON-коде включает *ответ на команду создания sc-элементов*, *ответ на команду получения соответствующих типов sc-элементов*, *ответ на команду удаления sc-элементов*, *ответ на команду обработки ключевых sc-элементов*, *ответ на команду обработки содержимого файлов ostis-системы*, *ответ на команду поиска sc-конструкций*, *изоморфных заданному sc-шаблону*,

```

sc_json_command
: '{'
  "'id'" '! NUMBER !'
  sc_json_command_type_and_payload
}'
;

sc_json_command_type_and_payload
: sc_json_command_create_elements
| sc_json_command_check_elements
| sc_json_command_delete_elements
| sc_json_command_handle_keynodes
| sc_json_command_handle_link_contents
| sc_json_command_search_template
| sc_json_command_generate_template
| sc_json_command_handle_events
';

```

= *Описание Правила, задающего синтаксис команд на SC-JSON-коде*

ответ на команду генерации sc-конструкции, изоморфной заданному sc-шаблону, и ответ на команду обработки sc-событий. В *ответ на команду на SC-JSON-коде* включаются идентификатор соответствующей команды, статус обработки ответа и ответное сообщение [Описание Правила, задающего синтаксис ответов на команды SC-JSON-кода](#).

```

sc_json_command_answer
: '{'
  "'id'" '! NUMBER !'
  "'status'" '! BOOL !'
  sc_json_command_answer_payload
}'
;

sc_json_command_answer_payload
: sc_json_command_answer_create_elements
| sc_json_command_answer_check_elements
| sc_json_command_answer_delete_elements
| sc_json_command_answer_handle_keynodes
| sc_json_command_answer_handle_link_contents
| sc_json_command_answer_search_template
| sc_json_command_answer_generate_template
| sc_json_command_answer_handle_events
';

```

= *Описание Правила, задающего синтаксис ответов на команды SC-JSON-кода*

В *сообщении команды создания sc-элементов* представляется список описаний создаваемых sc-элементов. Такими sc-элементами могут быть sc-узел, sc-дуга, sc-ребро или файл ostis-системы. Тип sc-элемента указывается в паре с ключевым словом "el": для sc-узла sc-json-тип элемента представляется как "node", для sc-дуги и sc-ребра – "edge", для файла ostis-системы – "link". Метки типов sc-элементов уточняются в соответствующих им описаниях в *сообщении команды* в паре с ключевым словом "type". Если создаваемым sc-элементом является файл ostis-системы, то дополнительно указывается содержимое этого файла ostis-системы в паре с ключевым словом "content", если создаваемым sc-элементом является sc-дуга или sc-ребро, то указываются описания sc-элементов, из которых они выходят, и sc-элементов, в которые они входят. Описание таких sc-элементов состоят из двух пар: первая пара указывает на способ ассоциации с sc-элементом и представляется как "addr" или "idtf" или "ref" в паре с ключевым словом "type", вторая пара – то, по чёму происходит ассоциация с этим sc-элементом: его хешу, системному

идентификатору или номеру в массиве создаваемых sc-элементов – в паре с ключевым словом "value" [Описание Правила, задающего синтаксис команды создания sc-элементов](#).

Сообщением *ответа на команду создания sc-элементов* является список хэшей созданных sc-элементов, соответствующих описаниям *команды создания sc-элементов* со статусом 1, в случае успешной обработки команды [Описание Правила, задающего синтаксис ответа на команду создания sc-элементов](#).

Множество *команд на SC-JSON-коде* легко расширяемо засчёт гибкости и функциональности языка JSON. Множество *ответов на команды на SC-JSON-коде* также легко расширяемо вместе с расширением *команд на SC-JSON-коде*.

Детальное описание синтаксиса команд и ответов на эти команды, а также их примеры можно найти в Стандарте **OSTIS Standard2021**.

Пункт 6.3.5.4. Серверная система на основе Websocket и JSON, обеспечивающая сетевой доступ к sc-памяти

Серверная система на основе Websocket и JSON, обеспечивающая сетевой доступ к sc-памяти, представляет собой интерпретатор команд и ответов на них SC-JSON-кода на программное представление sc-конструкций в sc-памяти при помощи *Библиотеки программных компонентов для обработки json-текстов JSON for Modern C++* и *Библиотеки кросс-платформенных программных компонентов для реализации серверных приложений на основе Websocket WebSocket++*, а также обеспечивается комплексным тестовым покрытием посредством программных фреймворков Google Tests и Google Benchmark Tests. *Библиотека программных компонентов для обработки json-текстов JSON for Modern C++* имеет богатый, удобный и быстродействующий функционал, необходимый для реализации подобных компонентов ostis-систем, а *Библиотека кросс-платформенных программных компонентов для реализации серверных приложений на основе Websocket WebSocket++* позволяет элегантно проектировать серверные системы без использования избыточных зависимостей и решений. Настройка программного компонента осуществляется с помощью *Программного компонента настройки программных компонентов ostis-систем* и скриптов языков CMake и Bash.

Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к sc-памяти

- := [Реализация системы, работающей по принципам Websocket и предоставляющая параллельно-асинхронный многоклиентский доступ к sc-памяти платформы интерпретации sc-моделей при помощи SC-JSON-кода]
- := [sc-json-сервер]
- ⇒ часто используемый неосновной внешний идентификатор sc-элемента*:
 - [sc-сервер]
- ∈ автоматный многократно используемый компонент ostis-систем
- ∈ зависимый многократно используемый компонент ostis-систем
- ⇒ используемый язык представления методов*:
 - C
 - C++
- ⇒ используемый язык*:
 - SC-JSON-код
- ⇒ зависимости компонента*:
 - {• Библиотека программных компонентов для обработки json-текстов JSON for Modern C++
 - Библиотека кросс-платформенных программных компонентов для реализации серверных приложений на основе Websocket WebSocket++
 - Программный компонент настройки программных компонентов ostis-систем версия
 - Реализация sc-памяти

Стоит отметить, что текущая *Реализация серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к sc-памяти*, не является первой в своём роде и заменяет предыдущую ее реализацию, написанную на языке Python. Причиной такой замены состоит в следующем:

- предыдущая *Реализация серверной системы на основе Websocket, обеспечивающей доступ к sc-памяти* при помощи команд SC-JSON-кода, реализованная на языке программирования Python, зависит от библиотеки Boost Python, предоставляемой сообществом по развитию и колаборации языков C++ и Python. Дело в том, что такое решение требует поддержки механизма интерпретации программного кода на языке Python на языке C++, что является избыточным и необоснованным, поскольку большая часть программного кода *Программного варианта реализации ostis-платформы* реализована на языках C и C++. Новая реализация

```

sc_json_command_create_elements
: "type": "create_elements",
  "payload": [
    "[(
      {
        "el": "node",
        "type": SC_NODE_TYPE,
        "content": NUMBER_CONTENT
          | STRING_CONTENT
      },
      {
        "el": "link",
        "type": SC_LINK_TYPE,
        "content": NUMBER_CONTENT
          | STRING_CONTENT
      },
      {
        "el": "edge",
        "type": SC_EDGE_TYPE,
        "src": (
          {
            "type": "ref",
            "value": NUMBER
          },
          {
            "type": "addr",
            "value": SC_ADDR_HASH
          },
          {
            "type": "idtf",
            "value": SC_NODE_IDTF
          }
        )
      },
      "trg": (
        {
          "type": "ref",
          "value": NUMBER
        },
        {
          "type": "addr",
          "value": SC_ADDR_HASH
        },
        {
          "type": "idtf",
          "value": SC_NODE_IDTF
        }
      )
    ),
    scs_text
  ]*];
}

```

= Описание Правила, задающего синтаксис команды создания sc-элементов

```

sc_json_command_answer_create_elements
: "payload" !'
|
(SC_ADDR_HASH ',')*
|
;
;
```

= Описание Правила, задающего синтаксис ответа на команду создания sc-элементов

описываемой программной системы позволяет избавиться от использования ёмких и ресурсозатратных библиотек(например, boost-python-lib, llvm) и языка Python;

- при реализации распределённых подсистем важную роль играет скорость обработки знаний, то есть возможность быстро и своевременно отвечать на запросы пользователя. Качество доступа к sc-памяти посредством реализованной *Подсистемы взаимодействия с sc-памятью на основе языка JSON* должно быть соизмеримо с качеством доступа к sc-памяти при помощи специализированного программного интерфейса API, реализованного на том же языке программирования, что и сама система. Новая реализация позволяет повысить скорость обработки запросов *Подсистемой взаимодействия с sc-памятью на основе языка JSON*, в том числе и обработка знаний, не менее чем в 1,5 раза по сравнению с предыдущим вариантом реализации этой подсистемы.

Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к sc-памяти, обладает следующими общими характеристиками:

- С точки зрения своей модели, серверная подсистема имеет такой же специализированный программный интерфейс, как и *Реализация sc-памяти*, однако взаимодействие с ней при помощи такого интерфейса осуществляется посредством сети. Это обеспечивает возможность взаимодействия клиентских систем, реализованных на разных языках программирования, с одной общей памятью.
- Данную подсистему можно рассматривать как некоторый интерпретатор внешнего языка представления знаний SC-JSON-код, на котором могут общаться ostis-системы, реализованные на базе специализированной ostis-платформы. Каждой команде и ответу на команду этого языка соответствует обработчик (потенциально и вовсе агент), который является частью этого интерпретатора. Сам язык внешнего представления знаний SC-JSON-код независим от реализации платформы и используется только как язык внешнего представления знаний, но может быть задействован при реализации других средств и интерпретаторов sc-моделей ostis-систем.
- Реализованный программный компонент предоставляет многопользовательский асинхронный доступ к sc-памяти. В ходе тестирования sc-сервера выяснилось, что его реализация позволяет обрабатывать запросы не менее чем 1000 клиентских систем. В связи с необходимостью обеспечения параллельного доступа к sc-памяти на уровне реализации программного компонента были добавлены блоки синхронизации. Например, в реализации можно заметить очередь команд на обработку системой – вне зависимости от количества клиентских систем и того, в каком количестве они отправляют команды на обработку, все команды могут становиться в очередь. Такое решение позволяет временно обойти проблемы взаимодействия блоков синхронизации на уровне sc-памяти при обработке разных типов команд над ней (поисковых, генеративных, деструктивных и т. д.). При этом серверную систему невозможно отключить до тех пор, пока очередь команд имеет какие-нибудь необработанные команды. Также серверная система продолжает работать, если в списке идентификаторов клиентских систем остались неотключенные из них. Необходимость данных функций серверной подсистемы обуславливается необходимостью поддержки атомарности запросов, обрабатываемых системой.
- В процессе тестирования подсистемы были получена оценка её скорости обработки команд и ответов. При нагружочном тестировании использовалась тестовая клиентская система, написанная на C++ и не имеющая функционала обработки текстов SC-JSON-кода. В результате тестирования было выяснено, что при отправке 1000 различных команд: команд создания sc-элементов, команд обработки содержимого файлов ostis-системы и команд удаления sc-элементов – время, потраченное на их обработку не превышало 0,2 секунды. При этом в отдельных случаях на обработку 1000 команд создания sc-элементов уходило не более 0,14 секунды, команд удаления sc-элементов – не более 0,07 секунды, команд обработки содержимого файлов ostis-системы – не более 0,27 секунды, команд поиска sc-конструкций, изоморфных заданному sc-шаблону – не более 0,45 секунды.

Серверная система на основе Websocket и JSON, обеспечивающая сетевой доступ к sc-памяти описывает необходимый и достаточный программный интерфейс для взаимодействия с sc-памятью. В общем случае описывает функциональные возможности не только *Серверной системы на основе Websocket, обеспечивающей доступ к sc-памяти платформы интерпретации sc-моделей при помощи команд SC-JSON-кода*, но и клиентских систем взаимодействующих с ней, поскольку зачастую эти клиентские системы включают специализированный программный интерфейс, схожий с интерфейсом серверной системы, но реализованный на другом языке программирования.

§ 6.3.6. Реализация интерпретатора sc-моделей пользовательских интерфейсов

⇒ подраздел*:

- Пункт 6.3.6.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов
- Пункт 6.3.6.2. Достоинства и недостатки текущего варианта Реализации интерпретатора sc-моделей пользовательских интерфейсов

В большинстве случаев разработка пользовательского интерфейса в современных системах отнимает большую часть времени, затрачиваемого на разработку всей системы. Однако эффективность использования программной системы зависит от разрабатываемого пользовательского интерфейса Myers¹⁹⁹².

Наряду с Реализацией sc-памяти важной частью Программного варианта реализации ostis-платформы является Реализация интерпретатора sc-моделей пользовательских интерфейсов, которая предоставляет базовые средства просмотра и редактирования базы знаний пользователем, средства для навигации по базе знаний (задания вопросов к базе знаний) и может дополняться новыми компонентами в зависимости от задач, решаемых каждой конкретной ostis-системой.

Реализация интерпретатора sc-моделей пользовательских интерфейсов

∈ неатомарный многократно используемый компонент ostis-систем

∈ зависимый многократно используемый компонент ostis-систем

⇒ используемый язык представления методов*:

- JavaScript
- TypeScript
- Python
- HTML
- CSS

⇒ зависимости компонента*:

- {• Библиотека стандартных интерфейсных компонентов на языке программирования JavaScript
- Библиотека для реализации серверных приложений на языке программирования Python Tornado
- Реализация клиентской системы на языке программирования TypeScript
- Реализация клиентской системы на языке программирования Python

}

Важным принципом Реализации интерпретатора sc-моделей пользовательских интерфейсов является простота и однотипность подключения любых компонентов пользовательского интерфейса (редакторов, визуализаторов, переключателей, команд меню и т.д.). Для этого реализуется программная прослойка Sandbox, в рамках которой реализуются низкоуровневые операции взаимодействия с серверной частью и которая обеспечивает более удобный программный интерфейс для разработчиков компонентов. Текущий вариант Реализации интерпретатора sc-моделей пользовательских интерфейсов является открытым и доступен на sc-web.

Пункт 6.3.6.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов

Реализация интерпретатора sc-моделей пользовательских интерфейсов

⇒ декомпозиция программной системы*:

- {• Панель меню команд пользовательского интерфейса
- Компонент переключения языка идентификации отображаемых sc-элементов
- Компонент переключения внешнего языка визуализации знаний
- Поле поиска sc-элементов по идентификатору
- Панель отображения диалога пользователя с ostis-системой
- Панель визуализации и редактирования знаний

⇒ декомпозиция программной системы*:

- {• Визуализатор sc.n-текстов
- Визуализатор и редактор sc.g-текстов

}

}

Компонент переключения языка идентификации отображаемых sc-элементов является изображением множества имеющихся в системе естественных языков. Взаимодействие пользователя с данным компонентом пере-

ключает пользовательский интерфейс в режим общения с конкретным пользователем с использованием *основных sc-идентификаторов*, принадлежащих данному *естественному языку*. Это значит, что при изображении sc-идентификаторов sc-элементов на каком-либо языке, например, SCg-коде или SCn-коде будут использоваться *основные sc-идентификаторы*, принадлежащие данному *естественному языку*. Это касается как sc-элементов, отображаемых в рамках *Панели визуализации и редактирования знаний*, так и любых других sc-элементов, например, классов команд и даже самих *естественных языков*, изображаемых в рамках самого *Компонента переключения языка идентификации отображаемых sc-элементов*.

Компонент переключения внешнего языка визуализации знаний служит для переключения языка визуализации знаний в текущем окне, отображаемом на *Панели визуализации и редактирования знаний*. В текущей реализации в качестве таких языков по умолчанию поддерживаются SCg-код и SCn-код, а также любые другие языки, входящие во множество *внешних языков визуализации SC-кода*.

Поле поиска sc-элементов по идентификатору позволяет осуществлять поиск sc-идентификаторов, содержащих подстроку, введенную в данное поле (с учетом регистра). В результате поиска отображается список sc-идентификаторов, содержащих указанную подстроку, при взаимодействии с которыми осуществляется автоматическое задание вопроса “Что это такое?”, аргументом которого является либо для сам sc-элемент, имеющий данный sc-идентификатор (в случае, если указанный sc-идентификатор является основным или системным, и, таким образом, указанный sc-элемент может быть определен однозначно), либо для самого внутреннего файла ostis-системы, являющегося sc-идентификатором (в случае, если данный sc-идентификатор является неосновным).

Панель отображения диалога пользователя с ostis-системой отображает упорядоченный по времени список sc-элементов, являющихся знаками действий, которые инициировал пользователь в рамках диалога с ostis-системой путем взаимодействия с изображениями соответствующих классов команд (то есть, если действие было инициировано другим способом, например, путем его явного инициирования через создание дуги принадлежности множеству *иницированных действий* в sc.g-редакторе, то на данной панели оно отображено не будет). При взаимодействии пользователя с любым из изображенных знаков действий на *Панели визуализации и редактирования знаний* отображается окно, содержащее результат выполнения данного *действия* на том языке визуализации, на котором он был отображен, когда пользователь просматривал его в последний (предыдущий) раз. Таким образом, в текущей реализации данная панель может работать только в том случае, если инициированное пользователем действие предполагает явно представленный в памяти результат данного действия. В свою очередь, из этого следует, что в настоящее время данная панель, как и в целом *Реализация интерпретатора sc-моделей пользовательских интерфейсов*, позволяет работать с системой только в режиме диалога “вопрос-ответ”.

Панель визуализации и редактирования знаний отображает окна, содержащие sc-текст, представленный на некотором языке из множества *внешних языков визуализации SC-кода* и, как правило, являющийся результатом некоторого действия, инициированного пользователем. Если соответствующий визуализатор поддерживает возможность редактирования текстов соответствующего естественного языка, то он одновременно является также и редактором. При необходимости пользовательский интерфейс каждой конкретной ostis-системы может быть дополнен визуализаторами и редакторами различных внешних языков, которые в текущей версии *Реализации интерпретатора sc-моделей пользовательских интерфейсов* будут также располагаться на *Панели визуализации и редактирования знаний*. По умолчанию доступны две панели визуализации и редактирования: Визуализатор sc.n-текстов и Визуализатор и редактор sc.g-текстов.

Панель меню команд пользовательского интерфейса содержит изображения классов команд (как атомарных, так и неатомарных), имеющихся на данный момент в базе знаний и входящих в декомпозицию *Главного меню пользовательского интерфейса* (имеется в виду полная декомпозиция, которая в общем случае может включать несколько уровней неатомарных классов команд). Взаимодействие с изображением неатомарного класса команд инициирует команду изображения классов команд, входящих в декомпозицию данного неатомарного класса команд. Взаимодействие с изображением атомарного класса команд инициирует генерацию команды данного класса с ранее выбранными аргументами на основе соответствующей *обобщенной формулировки класса команд* (шаблона класса команд).

Семантические модели описанных компонентов пользовательского интерфейса более подробно представлены в работе **sadouski2022semantic**.

Пункт 6.3.6.2. Достоинства и недостатки текущего варианта Реализации интерпретатора sc-моделей пользовательских интерфейсов

Текущая реализация интерпретатора sc-моделей пользовательских интерфейсов имеет большое множество недостатков, а именно:

- Идея платформенной независимости пользовательского интерфейса (построения sc-модели пользовательского интерфейса) реализована не в полной мере. Полностью описать sc-модель пользовательского интерфейса

(включая точное размещение, размеры, дизайн компонентов, их поведение и др.) в настоящее время скорее всего окажется затруднительно из-за ограничений производительности, однако вполне возможно реализовать возможность задания вопросов ко всем компонентам интерфейса, изменить их расположение и т.д., однако эти возможности нельзя реализовать в текущей версии реализации платформы.

- Кроме того, часть интерфейса фактически работает напрямую с sc-памятью с использованием технологии WebSocket, а часть – через прослойку на базе библиотеки tornado для языка программирования Python, что приводит к дополнительным зависимостям от сторонних библиотек. В последнее время развития текущего *Программного варианта реализации ostis-платформы* данная проблема в большей мере была решена, однако всё ещё остались компоненты, реализуемые на Python.
- Часть компонентов (например, поле поиска по идентификатору) реализована сторонними средствами и практически никак не связана с sc-памятью. Это затрудняет развитие платформы. Тора sc-моделей пользовательских интерфейсов ориентирована только на ведение диалога с пользователем (в стиле вопрос пользователя – ответ системы). Не поддерживаются такие очевидно необходимые ситуации, как выполнение команды, не предлагающей ответа; возникновение ошибки или отсутствие ответа; необходимость задания вопроса системой пользователю и т.д.
- Ограничена возможность взаимодействия пользователя с системой без использования специальных элементов управления. Например, можно задать вопрос системе, нарисовав его в SCg-коде, но ответ пользователь не увидит, хотя в памяти он будет сформирован соответствующим агентом. Большая часть технологий, использованных при реализации платформы, к настоящему моменту устарела, что затрудняет развитие платформы.
- Не реализован механизм наследования при добавлении новых внешних языков. Например, добавление нового языка даже очень близкого к SCg-коду требует физического копирования кода компонента и внесение соответствующих изменений, при этом получаются два никак не связанных между собой компонента, которые начинают развиваться независимо друг от друга.
- Слабый уровень задокументированности текущей *Реализации интерпретатора sc-моделей пользовательских интерфейсов*. Представленная текущая спецификация пока только описывает ключевые моменты *Реализации интерпретатора sc-моделей пользовательских интерфейсов*, но не раскрывает их.

На основе описанных недостатков к будущей реализации предъявляются следующие требования:

- Унифицировать принципы взаимодействия всех компонентов интерфейса с *Реализацией sc-памяти*, независимо от того, к какому типу относится компонент. Например, список команд меню должен формироваться через тот же механизм, что и ответ на запрос пользователя, и команда редактирования, сформированная пользователем, и команда добавления нового фрагмента в базу знаний и т.д. Необходимо совершенствовать способы использования интерфейса для удобного и комфортного пользования.
- Унифицировать принципы взаимодействия пользователей с системой независимо от способа взаимодействия и внешнего языка. Например, должна быть возможность задания вопросов и выполнения других команд прямо через SCg/SCn интерфейс. При этом необходимо учитывать принципы редактирования базы знаний, чтобы пользователя не мог под видом задания вопроса внести новую информацию в согласованную часть базы знаний.
- Унифицировать принципы обработки событий, происходящих при взаимодействии пользователя с компонентами интерфейса – поведение кнопок и других интерактивных компонентов должно задаваться не статически сторонними средствами, а реализовываться в виде агента, который, тем не менее, может быть реализован произвольным образом (не обязательно на платформенно-независимом уровне). Любое действие, совершаемое пользователем, на логическом уровне должно трактоваться и обрабатываться как инициирование агента.
- Обеспечить возможность выполнять команды (в частности, задавать вопросы) с произвольным количеством аргументов, в том числе – без аргументов.
- Обеспечить возможность отображения ответа на вопрос по частям, если ответ очень большой и для отображения требуется много времени.
- Каждый отображаемый компонент интерфейса должен трактоваться как изображение некоторого sc-узла, описанного в базе знаний. Таким образом, пользователь должен иметь возможность задания произвольных вопросов к любым компонентам интерфейса.
- Максимально упростить и задокументировать механизм добавления новых компонентов.
- Обеспечить возможность добавления новых компонентов на основе имеющихся без создания независимых копий. Например, должна быть возможность создать компонент для языка, расширяющего язык SCg новыми примитивами, переопределить принципы размещения sc-текстов и т.д.
- Свести к минимуму зависимость от сторонних библиотек.
- Свести к минимуму использование протокола HTTP (начальная загрузка общей структуры интерфейса), обеспечить возможность равноправного двустороннего взаимодействия серверной и клиентской части.

Очевидно, что реализация большинства из приведенных требований связана не только с собственным вариантом реализации платформы, но и требует развития теории логико-семантических моделей пользовательских интерфейсов и уточнения в рамках нее общих принципов организации пользовательских интерфейсов ostis-систем. Однако, принципиальная возможность реализации таких моделей должна быть учтена в рамках реализации платформы.

Заключение к Главе 6.3. Программная платформа *ostis*-систем

В дальнейшем развитии *Программного варианта реализации ostis-платформы* важным и правильным будет:

- максимально детализировать спецификацию компонентов проектируемой *ostis*-платформы, в том числе используемых языков внешнего и внутреннего представления знаний, и чётко стратифицировать иерархию классов и отношений, используемых при описании компонентов *ostis*-платформ;
- устранить и учесть недостатки при реализации новых компонентов в проектируемой *ostis*-платформе, указать возможные варианты их реализации;
- свести зависимость компонентов *ostis*-платформы от её реализации к минимуму, то есть, по возможности, реализовать их на языке SCP (например, интерпретатор sc-моделей интерфейсов *ostis*-систем);
- оценить качество проектируемой системы и её компонентов в целом;

Часть 7.

Экосистема OSTIS

Описание к главе

Глава 7.1.

Структура и проблемы организации человеческой деятельности

⇒ *автор**:

- Голенков В.В.
- Гулякина Н.А.

⇒ *аннотация**:

[В главе рассмотрены принципы автоматизации различных областей человеческой деятельности с использованием интеллектуальных компьютерных систем нового поколения. Предлагается онтология различных видов деятельности и соответствующих технологий. Детализация указанных принципов осуществляется на примере человеческой деятельности в области Искусственного интеллекта.]

⇒ *подраздел**:

- § 7.1.1. Основные понятия, лежащие в основе формального описания структуры человеческой деятельности
- § 7.1.2. Структура и проблемы организации человеческой деятельности в области Искусственного интеллекта
- § 7.1.3. Ключевые виды и области человеческой деятельности
- § 7.1.4. Проблемы и перспективы комплексной автоматизации всевозможных видов и областей человеческой деятельности с помощью интеллектуальных компьютерных систем нового поколения

Введение в Главу 7.1.

Ключевая проблема современного уровня комплексной автоматизации *человеческой деятельности* заключается в следующем. В настоящее время осуществляется либо полная автоматизация некоторых классов действий, инициируемых соответствующими командами, либо частичная автоматизация некоторых видов *человеческой деятельности*, в рамках которой человек осуществляет управление соответствующими средствами автоматизации. При этом автоматизация решения комплексных задач, которые сводятся к нескольким частично автоматизированным подзадачам, требует "ручного" (неавтоматизированного) управления одновременно несколькими средствами автоматизации.

Принципы, лежащие в основе перехода к более высокому уровню автоматизации *человеческой деятельности* сводятся к следующему. Все средства автоматизации (все сервисы), управляемые в настоящее время людьми, переходят "под управление" *интероперабельными интеллектуальными компьютерными системами*, которые способны эффективно взаимодействовать между собой и, соответственно, способны полностью автоматизировать решение комплексных задач, требующих использования нескольких средств автоматизации (сервисов) *Yaghoobirafi K..aApprofSIiADIS-2022art; Ouksel A.M..SemaniGIS-1999art; Lanzenberger M..MakinOTKLiSW-2008art; Neiva F.W..TowarPiSC-2016art; Pohl J..Inter a tNfISaHP-2004art; Waters J..GlobalIUS-2009art*.

В рамках данной работы сначала уточним основные понятия, используемые нами для рассмотрения структуры *человеческой деятельности*, потом подробно рассмотрим структуру текущего состояния и проблемы развития *Человеческой деятельности в области Искусственного интеллекта*. Далее обобщим принципы организации и автоматизации *Человеческой в области Искусственного интеллекта* на все многообразие видов и областей *человеческой деятельности*.

§ 7.1.1. Основные понятия, лежащие в основе формального описания структуры человеческой деятельности

деятельность

:= [процесс ситуативного воздействия на некоторую динамическую систему, направленного либо на создание этой системы, либо на поддержание определённых характеристик этой системы, либо на её разрушение, либо на ее развитие (совершенствование)]

:=

[система всех действий, выполняемых некоторым индивидуальным или коллективным субъектом либо целостная подсистема таких действий, соответствующая назначению (обязанностям) этого субъекта]

следует отличать*

⊖ {• *деятельность*
• *действие*
}

⇒ *сравнение**:

[Каждому *действию* обязательно соответствует формулировка задачи, которая решается в результате выполнения этого действия. При этом *действие* может быть сложным (неатомарным), то есть представлять собой иерархическую систему поддействий, обеспечивающих решение *подзадач* исходной задачи.]

В отличие от этого каждой *деятельности* может соответствовать несколько исходных задач, не являющихся под задачами других задач, решаемых в рамках этой *деятельности*.

Пример такой *деятельности*, которая не является действием, – это деятельность, осуществляемая некоторым субъектом в рамках некоторой *предметной области* (имеется в виду *деятельность* по решению всевозможных задач самого различного вида, которые могут быть сформулированы в рамках указанной *предметной области*).]

⇒ *примечание**:

[Каждой деятельности и, соответственно, каждому действию однозначно соответствует субъект, выполняющий эту деятельность.]

человеческая деятельность

:= [система действий, совершаемых людьми или человеческими сообществами либо "вручную", либо с помощью "пассивных" инструментов (палок, веревок, лопат, топоров, ...), либо с помощью "активных" инструментов (транспортных средств, элеваторов, бензопил, ...)]

действие

:= [процесс изменения состояния некоторой динамической системы (из заданного состояния в требуемое целевое состояние), инициированный и, возможно, непосредственно осуществляемый некоторым субъектом с возможным использованием некоторых инструментов, дополнительных материалов и информационных ресурсов (в частности, методов)]

▷ *информационное действие*

:= [информационный процесс, выполняемый некоторым субъектом (в том числе процессором компьютера)]
:= [процесс изменения состояния некоторого информационного ресурса]

следует отличать*

⊖ {• *действие*

• *задача*

:= [спецификация некоторого действия, содержащая достаточную информацию для выполнения этого действия]

:= [формулировка задачи]

▷ *декларативная формулировка задачи*

▷ *процедурная формулировка задачи*

}

действие

:= [целенаправленный (осознанный) процесс воздействия некоторого субъекта на некоторый объект]

⇒ *примечание**:

[Субъект действия может быть как индивидуальным, так и коллективным. Объект воздействия может иметь сколь угодно сложную структуру и состоять из любого количества компонентов, на которые оказывается воздействие. Инициирование и выполнение действия может осуществляться разными субъектами с использованием различных вспомогательных инструментов.]

задача

:= [формулировка задачи]

:= [спецификация некоторого действия, позволяющая осуществлять поиск метода выполнения этого действия]

отношение, заданное на множестве действий*

⊖ *действие**

:= [быть действием, выполняемым для решения заданной задачи]

↔ *обратное отношение**:

*задача**

:= [быть задачей, решаемой в результате выполнения данного действия*]

отношение, заданное на множестве деятельности

Э *объект деятельности**

:= [быть объектом, над которым выполняется заданная деятельность*]

Э *продукт деятельности**

:= [быть продуктом заданной деятельности*]

Э *отрезок времени существования**

:= [быть отрезком времени существования данной темпоральной сущности*]

Э *деятельности, классы объектов которых совпадают**

Э *деятельности, выполняемые одновременно**

вид деятельности

:= [класс деятельности]

:= [класс аналогичных деятельности, которому можно поставить в соответствие общую технологию, обеспечивающую выполнение всех деятельности данного класса]

:= [класс, экземплярами (элементами) которого являются эквивалентные (похожие) деятельности, выполняемые в общем случае разными кибернетическими системами]

Э *пример':*

проектирование

:= [проектная деятельность]

:= [человеческая деятельность, продуктом которой является проектная документация некоторой создаваемой сущности]

:= [построение полной спецификации (проектной документации) некоторой создаваемой сущности]

⇒ *примечание*:*

[Продуктами деятельности для этого вида деятельности являются спецификации любых социально значимых создаваемых сущностей.]

⇒ *частный вид деятельности над подклассом объектов деятельности*:*

- *проектирование ostis-систем*

⇒ *частный вид деятельности над подклассом объектов деятельности*:*

- *проектирование обучающих ostis-систем*

- *проектирование ostis-систем медицинской диагностики*

- *проектирование микросхем*

- *проектирование зданий*

Э *пример':*

поддержка жизненного цикла

⇒ *частный вид деятельности над подклассом объектов деятельности*:*

- *поддержка жизненного цикла микросхем*

- *поддержка жизненного цикла зданий*

- *поддержка жизненного цикла ostis-систем*

⇒ *частный вид деятельности над подклассом объектов деятельности*:*

- *поддержка жизненного цикла баз знаний ostis-систем*

- *поддержка жизненного цикла решателей задач ostis-систем*

- *поддержка жизненного цикла интерфейсов ostis-систем*

Э *подготовка кадровых ресурсов*

:= [образовательная деятельность]

Э *производство*

:= [производственная деятельность]

Э *природоохранная деятельность*

Э *строительная деятельность*

Э *здравоохранительная деятельность*

Э *административная деятельность*

Э *научно-исследовательская деятельность*

следует отличать*

Э {• *вид деятельности*

- *класс действий*

}

следует отличать*

Э {• *класс действий*

- *класс задач*
:= [множество аналогичных (похожих) формулировок конкретных задач, которые легко обобщить, заменяя некоторые константы, входящие в эти формулировки, на переменные]
 - *формулировка класса задач*
:= [обобщенная формулировка (спецификация) класса задач, "превращающаяся" в формулировку конкретной задачи при присваивании конкретных значений всем свободным переменным, входящим в эту обобщенную формулировку класса задач]
- }

отношение, заданное на множестве действий

- Э *метод**
:= [быть методом, обеспечивающим выполнение всех действий заданного класса действий, или решение всех задач заданного класса задач, или выполнение конкретного заданного действия, или решение конкретной заданной задачи*]
- Э *максимальный класс действий**
:= [быть максимальным классом действий, выполняемых с помощью заданного метода*]

метод

- := [информационная конструкция, интерпретация которой любым субъектом, принадлежащим соответствующему классу субъектов, обеспечивает выполнение любого действия, принадлежащего соответствующему классу действий]
- ▷ *методика*
:= [метод, реализуемый человеком или коллективом людей]

отношение, заданное на множестве видов деятельности

- Э *технология**
:= [быть технологией, обеспечивающей выполнение каждой деятельности, принадлежащей заданному виду деятельности*]
- Э *класс объектов деятельности**
:= [быть классом объектов деятельности для заданного вида деятельности*]
- Э *класс продуктов деятельности**
:= [быть классом продуктов деятельности для заданного вида деятельности*]
- Э *частный вид деятельности над подклассом объектов деятельности**
:= [быть технологией, обеспечивающей выполнение каждой деятельности, принадлежащей заданному виду деятельности*]
- Э *частный вид деятельности над классом компонентов объектов деятельности**
:= [частный вид деятельности, классом объектов которого является является класс компонентов объектов деятельности заданного вида деятельности*]
- Э *частный вид одновременно выполняемой деятельности**
:= [частный вид деятельности, каждая из которых выполняется одновременно (параллельно) с деятельностями, принадлежащими заданному виду деятельности*]
- Э *частный вид деятельности, выполняемой на некотором этапе**
:= [частный вид деятельности, каждая из которых выполняется в качестве одного из этапов выполнения деятельности, принадлежащих заданному виду деятельности*]
- Э *виды деятельности, классы объектов которых совпадают**
- Э *виды деятельности, выполняемые одними и теми же субъектами**

технология

- := [система знаний и навыков, обеспечивающих выполнение деятельности соответствующего вида соответствующими субъектами]
- ⇒ *примечание*:*
[В основе любой *технологии* лежит многократность – либо многократность создания похожих (аналогичных) сущностей, либо многократность выполнения похожих действий, многократность использования похожих методов (способов, методик). Очевидно, что, чем выше степень сходства (близости, конвергентности) многократно создаваемых сущностей и многократно используемых методов, чем выше их унификация, тем проще будет соответствующая *технология*.]

область выполнения*

- ▷ *область выполнения действия**
- ▷ *область выполнения класса действий**
- ▷ *область выполнения деятельности**
- ▷ *область выполнения вида деятельности**

- ▷ быть предметной областью и соответствующей онтологией возможно с некоторыми дочерними предметными областями и соответствующими онтологиями, содержащей знания, достаточные для выполнения заданного действия, или заданного класса действий, или заданной деятельности, или заданного вида деятельности*

субъект*

:= [быть субъектом, выполняющим заданное действие или выполняющим заданную деятельность*]

⇒ примечание*:

[Инициирование действия или деятельности, выполняемой субъектом, может осуществляться либо самостоятельно (по собственной инициативе), либо по инициативе (команде, заданию) другого субъекта.]

информационный ресурс

:= [социально-значимая информационная конструкция, являющаяся продуктом соответствующей человеческой деятельности, который требует не только создания, но и сопровождения (обновления, актуализации)]

▷ проектная документация

▷ научная теория

:= [продукт научно-исследовательской деятельности, являющийся строгим описанием свойств и закономерностей некоторого класса сущностей]

▷ стандарт

▷ спецификация комплекса методик соответствующей технологии

▷ спецификация инструментальных средств соответствующей технологии

проектная документация

:= [полная спецификация создаваемой сущности]

:= [продукт проектной деятельности]

:= ["цифровая" копия сущности]

:= [информационная модель (описание) сущности, обладающая достаточной полнотой (детализацией) для воспроизведения (воспроизводства) этой сущности]

§ 7.1.2. Структура и проблемы организации человеческой деятельности в области Искусственного интеллекта

В предыдущих главах мы уточнили:

- архитектуру интеллектуальных компьютерных систем нового поколения
- то, как осуществляется деятельность (функционирование) интеллектуальных компьютерных систем нового поколения
- то, как автоматизируется прикладная инженерная человеческая деятельность по проектированию интеллектуальных компьютерных систем нового поколения и поддержке всех последующих этапов их жизненного цикла.

Уточним то, как осуществляется и автоматизируется весь комплекс человеческой деятельности в области Искусственного интеллекта.

Пункт 7.1.2.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта

Рассмотрим необходимость перехода организации человеческой деятельности **Искусственного интеллекта** на принципиально новый уровень, обеспечивающий формирование рынка **семантически совместимых интеллектуальных компьютерных систем нового поколения**, разрабатываемых на основе принципиально нового комплекса **семантически совместимых технологий Искусственного интеллекта**.

Сейчас актуально исследовать не только *модели решения интеллектуальных задач* в интеллектуальных компьютерных системах различного вида, но также методологические проблемы текущего состояния *Искусственного интеллекта* в целом и пути решения этих проблем.

Анализ современного состояния работ в области *Искусственного интеллекта* показывает то, что указанная научно-техническая дисциплина находится в серьезном методологическом кризисе. Поэтому необходимо:

- Выявление основных причин возникновения указанного кризиса;
- Уточнение основных мер, направленных на его устранение.

Решение рассматриваемых кризисных проблем требует:

- Существенного фундаментального общесистемного переосмысливания всего того, что мы делаем в области *Искусственного интеллекта* и как мы это делаем, т.е. требует уточнения характеристик *интеллектуальных компьютерных систем*, уточнения понятия сообщества, состоящего из *интеллектуальных компьютерных систем* и взаимодействующих с ними пользователей, уточнения требований, предъявляемых к *интеллектуальным компьютерным системам*, а также уточнения методик и средств их создания и использования.
- Осознания того, что *Кибернетика*, *Информатика* и *Искусственный интеллект* – это общая фундаментальная наука, требующая комплексного подхода к построению общих формальных моделей систем, основанных на обработке информации (*кибернетических систем*), путём *конвергенции и интеграции* формальных моделей различных компонентов этих систем *Yankovskaya_2017; Палагин А.В..ПроблТиРИ-2013ст*. Таким образом, современный этап развития *Искусственного интеллекта* – это переход от накопленного к текущему моменту многообразия моделей решения различного вида задач к преобразованию этого многообразия в стройную систему *семантически совместимых моделей*;
- Осознания того, что сейчас требуется не расширять многообразие точек зрения, а учиться их согласовывать, обеспечивать их *семантическую совместимость*, совершенствуя соответствующие методы.

Обсуждая современную проблематику *конвергенции* различных моделей в области *Искусственного интеллекта* и построения интегрированных *гибридных моделей*, уместно вспомнить «фантастический рассказ Д.А. Поспелова «Соприкосновение», посвященный контакту различных миров. В нем главный герой популярно излагает свою теорию *концептуальных разломов* <...>. Эта теория напоминает историю долгого периода *дифференциации* наук, когда различные научные дисциплины развивались независимо, словно параллельные миры, лишь изредка соприкасаясь друг с другом, а отдельные ученые, получая все более узкую специализацию, мало что знали о достижениях даже своих «близких собратьев». К счастью, в последние годы все чаще и чаще возникают новые области контакта между отдельными дисциплинами, происходит взаимопроникновение идей, установление *аналогий* между полученными результатами и тенденциями развития. Во многом это объясняется появлением и широким внедрением во все сферы жизни общества передовых информационных и коммуникационных технологий <...>. Современные технологии опираются на достижения многих научно-технических дисциплин, среди которых на первый план выходят синтетические науки нового поколения – науки об искусственном».

⇐ цитата*:

Тарасов В.Б.оМногоСкИО-2002кн/с.13

Анализируя современное состояние работ в области *Искусственного интеллекта* (ИИ), следует констатировать то, что *концептуальный разлом* между различными направлениями *Искусственного интеллекта* является очевидным фактом. Это подтверждается следующей цитатой из книги В.Б. Тарасова *Тарасов В.Б.оМногоСкИО-2002кн* «вновь, как и на заре ИИ, актуальными становятся формирование единых методологических основ ИИ, разработка теоретических проблем создания интеллектуальных систем новых поколений, развитие нетрадиционных аппаратно-программных средств. Здесь большие перспективы связаны с использованием идей и принципов синергетики в ИИ. Сам термин "синергетика" происходит от слова "синергия", означающего совместное действие, сотрудничество. По мнению "отца синергетики" Г.Хакена, такое название вполне подходит для современной теории сложных самоорганизующихся систем по двум причинам: а) исследуются совместные действия многих элементов развивающейся системы; б) для отыскания общих принципов самоорганизации требуется объединение усилий представителей различных дисциплин».

⇐ цитата*:

Тарасов В.Б.оМногоСкИО-2002кн/с.14

Для того, чтобы убедиться в наличии *концептуального разлома* между различными направлениями *Искусственного интеллекта*, достаточно просто перечислить основные направления работы конференций по тематике *Искусственного интеллекта*, обращая внимание на то, что многие из них развиваются независимо от других:

- синергетические модели самоорганизации интеллектуальных компьютерных систем;
- гибридные интеллектуальные компьютерные системы;
- коллаборативные интеллектуальные компьютерные системы;
- мягкие вычисления, интеллектуальные вычисления;
- моделирование не-факторов;
- неклассические, многозначные, модальные, псевдофизические, индуктивные, нечеткие логики и приближенные рассуждения, логические программы;
- нечеткие множества, отношения, графы, алгоритмы;
- функциональные программы, нечеткие алгоритмы, генетические алгоритмы, продукционные модели;
- нейросетевые модели;
- параллельные асинхронные модели децентрализованного решения задач;
- обработка сигналов;
- мультисенсорная конвергенция, сенсо-моторная координация;
- модели ситуационного управления.

Преодоление *концептуального разлома* между различными направлениями исследований в области *Искусственного интеллекта* – это, своего рода "прыжок" через "концептуальную пропасть", который требует особой концентрации усилий. Через пропасть нельзя перепрыгнуть двумя прыжками.

Если кратко охарактеризовать **текущее состояние** всего комплекса работ в области *Искусственного интеллекта*, то это – **иллюзия благополучия**. Происходит активное локальное развитие самых различных направлений *Искусственного интеллекта* (неклассические логики, формальные онтологии, искусственные нейронные сети, машинное обучение, мягкие вычисления, многоагентные системы и др.), но комплексного повышения уровня интеллекта современных интеллектуальных компьютерных систем не происходит. Для этого прежде всего требуется сближение и **интеграция всех** направлений *Искусственного интеллекта* и соответствующее построение *Общей формальной теории интеллектуальных компьютерных систем*, а также превращение современного многообразия инструментальных средств (frameworks) разработки различных компонентов интеллектуальных компьютерных систем в единую **Технологию комплексного проектирования и поддержки всего жизненного цикла интеллектуальных компьютерных систем**, гарантирующую совместимость всех разрабатываемых компонентов интеллектуальных компьютерных систем, а также совместимость самих интеллектуальных компьютерных систем как самостоятельных субъектов (агентов, акторов), взаимодействующих между собой в рамках комплексных систем автоматизации сложных видов коллективной *человеческой деятельности* (умных домов, умных больниц, умных школ, умных производственных предприятий, умных городов и т.д.). Таким образом, эпиграфом текущего состояния работ в области *Искусственного интеллекта* является известное высказывание из Экклезиаста: "Время разбрасывать камни и время собирать камни – всему своё время".

«К сожалению, в современных дискуссиях по теме ИИ (Искусственного интеллекта) научные споры часто подменяются завышенными ожиданиями от скорого внедрения ИИ и значительным сужением темы ИИ, которая оказалась сведена лишь к *машинному обучению* на основе *искусственных нейронных сетей*. <...> При этом за бортом Национальной стратегии пока остались *онтологии*, *базы знаний*, *методы рассуждений и принятия решений*, *методы синтеза и анализа сложных конструкций*, умные кибер-физические системы, цифровые двойники, автономные системы, системы анализа как "больших", так и "малых" данных. <...>

Признавая всю важность *машинного обучения* на базе *искусственных нейронных сетей*, научные и практические результаты мирового уровня следует искать на стыке разных дисциплин в **конвергенции** различных технологий ИИ и **интеграции** полипредметных знаний. В этой связи формализация знаний в виде *онтологий* и *баз знаний* в рамках *Semantic Web* рассматривается как одно из фундаментальных направлений для создания ИИ. Действительно, какой же может быть *интеллект* без использования знаний современных учебников, на основе чего ИИ будет понимать *контекст ситуации*, делать *выводы и принимать решения*? <...>

Еще одной ключевой сферой ИИ, не нашедшей отражения в Российской стратегии по ИИ, является *распределенное принятие решений*, которое все больше становится коллективным для стремительно развивающихся систем умного Интернета вещей и автономных систем управления, начиная с беспилотных автомобилей, самолетов, кораблей и т.д.

Компанией Гартнер 2020 год был объявлен годом "автономных вещей", которые по мнению компании уже прошли большую эволюцию от "цифровых" к "умным". Ожидается, что на следующем этапе автономные вещи, обладающие собственным ИИ, "заговорят" друг с другом и в научную повестку войдут вопросы *семантической интероперабельности* систем ИИ, которые будут не только обмениваться данными, но и вести переговоры для согласования решений. Дорожная карта научных исследований по ИИ США в качестве ключевых выделяет такие направления, как *связность* систем *Искусственного интеллекта* (Integrated Intelligence) и их *осмыслившее взаимодействие* (Meaningful Interaction), наряду с разными видами *самообучения* в системах (Self-Aware Learning).»

⇐ цитата*:

Баринов И.И.. ФормиСРКпИИ-2021см/с. 264-265

Ключевой причиной методологических проблем современного состояния *Искусственного интеллекта* и серьёзным вызовом для специалистов в этой области является проклятие *Вавилонского столпотворения Illiadis2019*, которое преследует нас на всех уровнях:

- на уровне внутренней организации *решения задач* в интеллектуальных компьютерных системах;
- на уровне взаимодействия интеллектуальных компьютерных систем как между собой, так и с пользователями;
- на уровне взаимодействия учёных, работающих в области *Искусственного интеллекта*, что препятствует созданию *Общей формальной теории и стандарта интеллектуальных компьютерных систем*, а также *Технологии комплексного проектирования и поддержки всего жизненного цикла интеллектуальных компьютерных систем*;
- на уровне взаимодействия между учёными, инженерами, разрабатывающими прикладные интеллектуальные компьютерные системы, преподавателями ВУЗов, которые готовят специалистов в области *Искусственного интеллекта*, а также студентами, магистрантами и аспирантами.

Сложность разрабатываемых в настоящее время *интеллектуальных компьютерных систем* и технологий Искусственного интеллекта достигла такого уровня, что для их разработки требуются не просто большие творческие коллективы, но и существенное повышение квалификации и качества этих коллективов. Как известно, квалификация коллектива разработчиков определяется не только квалификацией его членов, но также эффективностью и атмосферой их взаимодействия. Известно также, что качество любой технической системы является отражением качества того коллектива, который эту систему разработал. Может ли коллектив достаточно квалифицированных специалистов, многие из которых не обладают высоким уровнем интероперабельности, разработать интеллектуальную компьютерную систему с высоким уровнем интероперабельности, а тем более технологию комплексной поддержки всего жизненного цикла интеллектуальных компьютерных систем такого уровня? Очевидный ответ на этот вопрос и очевидная сложность создания работоспособных творческих коллективов указывают на основной вызов, адресованный специалистам в области Искусственного интеллекта в настоящее время. Таким образом, требования, предъявляемые к интеллектуальным компьютерным системам нового поколения и определяющие их способность к индивидуальному и коллективному решению комплексных сложных задач, должны предъявляться и к разработчикам этих систем, а также к разработчикам любых других сложных объектов, поскольку все сложные виды и области человеческой деятельности являются коллективными и творческими.

Создание быстро развивающегося рынка семантически совместимых *интеллектуальных компьютерных систем* – это основная цель, адресованная специалистам в области Искусственного интеллекта, требующая преодоления **Вавилонского столпотворения** во всех его проявлениях, формирования высокой культуры договороспособности и унифицированной, согласованной формы представления коллективно накапливаемых, совершенствуемых и используемых знаний. Ученые, работающие в области *Искусственного интеллекта*, должны обеспечить **конвергенцию** результатов различных направлений *Искусственного интеллекта* и построить *Общую формальную теорию интеллектуальных компьютерных систем*, а также *Комплексную технологию проектирования семантически совместимых интеллектуальных компьютерных систем*, включающую соответствующие стандарты *интеллектуальных компьютерных систем* и их компонентов. Инженеры, разрабатывающие прикладные интеллектуальные компьютерные системы, должны сотрудничать с учеными и участвовать в развитии *Комплексной технологии проектирования семантически совместимых интеллектуальных компьютерных систем*, и поддержки всех последующих этапов жизненного цикла этих систем.

Разобщенность различных направлений исследований в области *Искусственного интеллекта* является главным препятствием создания *Комплексной технологии проектирования семантически совместимых интеллектуальных компьютерных систем*, а также *Технологии комплексной поддержки* всех последующих этапов жизненного цикла *интеллектуальных компьютерных систем*.

Пункт 7.1.2.2. Структура деятельности в области Искусственного интеллекта

Для того, чтобы рассмотреть проблемы дальнейшего развития *деятельности* в области *Искусственного интеллекта* и, в частности, проблемы комплексной автоматизации этой *деятельности*, необходимо уточнить структуру указанной *деятельности*.

Человеческая деятельность в области *Искусственного интеллекта* направлена на исследование и создание *интеллектуальных компьютерных систем* различного вида и различного назначения. Объектами исследования в области *Искусственного интеллекта* являются:

- *индивидуальные интеллектуальные компьютерные системы* (в частности, когнитивные агенты);
- *многоагентные интеллектуальные компьютерные системы* (в частности, сообщества, состоящие из *индивидуальных интеллектуальных компьютерных систем*);
- *человеко-машинные сообщества*, состоящие из *интеллектуальных компьютерных систем* и их пользователей.

Основными целями человеческой деятельности в области *Искусственного интеллекта* являются:

- Построение формальной теории *интеллектуальных компьютерных систем* (искусственных интеллектуальных систем);
- Создание технологий (методик и средств), обеспечивающих *проектирование, реализацию, сопровождение и эксплуатацию интеллектуальных компьютерных систем*;
- Переход на принципиально новый уровень комплексной автоматизации всевозможных видов человеческой деятельности, который основан на массовом применение *интеллектуальных компьютерных систем* и который предполагает:
 - не только наличие *интеллектуальных компьютерных систем*, способных понимать друг друга и согласовывать свою деятельность,
 - но и учёт общей структуры *человеческой деятельности*, осуществляющей в условиях нового уровня её автоматизации (деятельности smart-общества), которая должна быть "понятна" используемым *интеллек-*

туальными компьютерными системами и которая потребует существенного переосмыслиния современной организации человеческой деятельности.

Искусственный интеллект как область человеческой деятельности включает в себя следующие направления деятельности:

- **Научно-исследовательскую деятельность в области Искусственного интеллекта,** в процессе которой осуществляется конкуренция различных точек зрения и подходов к построению формальных моделей различных компонентов интеллектуальных компьютерных систем. Конечной целью такой деятельности является постоянно развивающаяся *Общая теория интеллектуальных компьютерных систем*, объектами исследования которой являются *интеллектуальные компьютерные системы* и их формальные логико-семантические модели, включающие в себя формальные модели различного вида знаний, входящих в состав баз знаний интеллектуальных компьютерных систем, а также различные модели решения задач (логические модели различного вида, нейросетевые, генетические, производственные, функциональные и др.).
- **Разработку Стандарта интеллектуальных компьютерных систем**, включающую в себя перманентную эволюцию этого стандарта и поддержку целостности каждой его версии. Текущая версия *Стандарта интеллектуальных компьютерных систем* – это согласованная (общепризнанная) на текущий момент часть *Общей теории интеллектуальных компьютерных систем*.
- **Разработку технологии проектирования интеллектуальных компьютерных систем**, которая включает в себя семейство методик проектирования, а также методов и средств автоматизации проектирования различных компонентов интеллектуальных компьютерных систем и интеллектуальных компьютерных систем в целом. Результатом проектирования интеллектуальных компьютерных систем является полная формальная логико-семантическая модель этой системы.
- **Разработку технологии реализации спроектированных интеллектуальных компьютерных систем**, а также технологий эксплуатации и сопровождения интеллектуальных компьютерных систем. В основе технологии реализации (производства) спроектированных интеллектуальных компьютерных систем лежит *универсальный интерпретатор формальных логико-семантических моделей интеллектуальных компьютерных систем*, являющихся результатом проектирования указанных систем. Указанный универсальный интерпретатор может быть реализован либо в виде *программной системы* на современных компьютерах, либо в виде *универсального компьютера нового поколения*, ориентированного на интерпретацию формальных логико-семантических моделей интеллектуальных компьютерных систем.
- **Прикладную инженерную деятельность в области Искусственного интеллекта**, т.е. непосредственное проектирование, реализацию и сопровождение, включающее в себя обновление (реинжиниринг), осуществляющееся в ходе эксплуатации, конкретных интеллектуальных компьютерных систем.
- **Учебную деятельность в области Искусственного интеллекта**, направленную на подготовку специалистов области Искусственного интеллекта и на перманентное повышение квалификации действующих специалистов в этой области. Без эффективной организации учебной деятельности в области Искусственного интеллекта быстрый прогресс в этой области невозможен. Непосредственное включение учебной деятельности в общую структуру человеческой деятельности в области Искусственного интеллекта обусловлено следующими обстоятельствами:
 - необходимостью глубокой *конвергенции* между различными направлениями и видами деятельности в области Искусственного интеллекта и соответствующей спецификой требований, предъявляемых к специалистам в этой области – каждый такой специалист должен быть достаточно компетентен и в научно-исследовательской деятельности в области Искусственного интеллекта, и в разработке технологий (методик и средств) проектирования интеллектуальных компьютерных систем, и в разработке технологий воспроизведения (реализации) спроектированных интеллектуальных компьютерных систем, а также технологий их эксплуатации и сопровождения, и в прикладной инженерной деятельности в области Искусственного интеллекта;
 - высокими темпами эволюции результатов в области Искусственного интеллекта, что делает необходимой организацию обучения соответствующих специалистов путем их непосредственного подключения не к учебным (упрощенным) проектам, а к реальным проектам, реализуемым в текущий момент. Иначе подготовленные специалисты будут иметь квалификацию "вчерашнего дня";
 - существенным расширением объемов работ в области Искусственного интеллекта и острой необходимостью массовой подготовки соответствующих специалистов.

Сложность *Подготовки молодых специалистов в области Искусственного интеллекта* заключается не только в высокой степени наукоемкости этой области, но и в том, что формирование у них соответствующих знаний и навыков осуществляется в условиях быстрого морального старения текущего состояния технологий Искусственного интеллекта, существенные изменения в которых происходят за время обучения студентов и магистрантов. Поэтому надо учить не текущему уровню развития Искусственного интеллекта, а тому уровню развития, который будет достигнут через пять и более лет.

При подготовке молодых специалистов в области *Искусственного интеллекта* необходимо формировать у них:

- культуру формализации (математическую культуру);
- системную культуру (в частности, умение осуществлять качественную стратификацию сложных динамических систем);
- технологическую культуру (в частности, умение отличать то, что следует унифицировать и то, унификация чего ограничивает направление эволюции заданного класса сложных систем);
- технологическую дисциплину;
- культуру коллективного творчества (в частности, первоначальную *интероперабельность*);
- высокую *познавательную активность* и мотивацию;
- умение сочетать индивидуальную творческую свободу и самостоятельность с обеспечением совместимости своих результатов с результатами коллег, то есть сочетать свободу в создании (порождении) новых смыслов при согласованности (совместимости) форм их представления – о понятиях, терминах и синтаксисе не спорят, а договариваются.
- *Организационную деятельность в области Искусственного интеллекта*, направленную на создание инфраструктуры для качественного выполнения всех остальных видов деятельности в области *Искусственного интеллекта*, а именно:
 - для обеспечения глубокой *конвергенции* между различными направлениями и видами деятельности в области *Искусственного интеллекта* и, в частности, между теорией, технологиями и инженерной практикой в этой области;
 - для обеспечения баланса между тактикой и стратегией в развитии деятельности в области *Искусственного интеллекта* как ключевой основы существенного повышения уровня автоматизации всевозможных видов человеческой деятельности и перехода к *smart-обществу*.

Рассмотренная декомпозиция человеческой деятельности в области *Искусственного интеллекта* по видам деятельности не является традиционным признаком декомпозиции научно-технических дисциплин. Обычно декомпозиция научно-технических дисциплин осуществляется по содержательным направлениям, которые соответствуют декомпозиции технических систем, исследуемых и разрабатываемых в рамках этих научно-технических дисциплин, т.е. соответствуют выделению в этих технических системах различного вида компонентов. Для *Искусственного интеллекта* такими направлениями являются:

- исследование и разработка формальных моделей и языков представления знаний;
- исследование и разработка баз знаний;
- исследование и разработка логических моделей обработки знаний;
- исследование и разработка искусственных нейронных сетей;
- исследование и разработка подсистем компьютерного зрения;
- исследование и разработка подсистем обработки естественно-языковых текстов (синтаксический анализ, понимание, синтез);
- и многие другие.

Важность декомпозиции *Искусственного интеллекта* по видам деятельности определяется тем, что выделение различных видов деятельности позволяет четко ставить задачу на разработку средств автоматизации этих видов деятельности.

Приведем общую структуру Человеческой деятельности в области *Искусственного интеллекта*.

Человеческая деятельность в области Искусственного интеллекта

- := [Искусственный интеллект (как научно-техническая дисциплина)]
- Є научно-техническая дисциплина
- := [Искусственный интеллект (как научно-техническая дисциплина)]
- := [Человеческая деятельность в Предметной области интеллектуальных компьютерных систем]
- Є деятельность
- ⇒ декомпозиция*:
 - {• Интегральная деятельность по поддержке жизненного цикла всевозможных интеллектуальных компьютерных систем
 - ⇒ декомпозиция*:
 - поддержка жизненного цикла интеллектуальных компьютерных систем
 - Є вид деятельности
 - ▷ поддержка жизненного цикла *ostis*-систем
 - ⇒ обобщенная декомпозиция*:
 - {• проектирование интеллектуальных компьютерных систем
 - производство интеллектуальных компьютерных систем
 - начальное обучение интеллектуальных компьютерных систем

- мониторинг качества интеллектуальных компьютерных систем
 - восстановление требуемого уровня качества интеллектуальных компьютерных систем
 - реинжиниринг интеллектуальных компьютерных систем
 - обеспечение безопасности интеллектуальных компьютерных систем
 - эксплуатация интеллектуальных компьютерных систем конечными пользователями
}
- Поддержка жизненного цикла Общей теории интеллектуальных компьютерных систем
 - ∈ научно-исследовательская деятельность
 - Поддержка жизненного цикла Стандарта интеллектуальных компьютерных систем
 - ∈ стандартизация
 - ⇒ часть*:
 - Поддержка жизненного цикла Стандарта ostis-систем
 - Поддержка жизненного цикла Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем
 - ∈ поддержка жизненного цикла технологий
 - := [создание и сопровождение технологий]
 - ⇒ часть*:
 - Поддержка жизненного цикла Технологии OSTIS
 - Поддержка жизненного цикла кадровых ресурсов для Человеческой деятельности в области Искусственного интеллекта
 - Поддержка жизненного цикла системы комплексной организации взаимодействия между всеми направлениями Человеческой деятельности в области Искусственного интеллекта
 - ∈ поддержка жизненного цикла метасистем комплексного управления поддержкой и обеспечением поддержки жизненного цикла сущностей соответствующего класса
}

Технология поддержки жизненного цикла интеллектуальных компьютерных систем

- ⇒ вид деятельности*:
 - поддержка жизненного цикла интеллектуальных компьютерных систем
- ⇒ декомпозиция*:
 - Технология проектирования интеллектуальных компьютерных систем
 - ⇒ вид деятельности*:
 - проектирование интеллектуальных компьютерных систем
 - Технология производства интеллектуальных компьютерных систем
 - ⇒ вид деятельности*:
 - производство интеллектуальных компьютерных систем
 - Технология начального обучения интеллектуальных компьютерных систем (адаптации к конкретной деятельности)
 - ⇒ вид деятельности*:
 - начальное обучение интеллектуальных компьютерных систем
 - Технология мониторинга качества интеллектуальных компьютерных систем
 - ⇒ вид деятельности*:
 - мониторинг качества интеллектуальных компьютерных систем
 - := [плановое тестирование и диагностика интеллектуальных компьютерных систем]
 - Технология восстановления требуемого уровня качества интеллектуальных компьютерных систем в ходе их эксплуатации
 - := [Технология выявления и исправления потенциально опасных ситуаций и событий в деятельности интеллектуальных компьютерных систем (ошибок, противоречий ...)]
 - ⇒ вид деятельности*:
 - восстановление требуемого уровня качества интеллектуальных компьютерных систем
 - Технология реинжиниринга интеллектуальных компьютерных систем
 - := [Технология совершенствования, модернизации, обновления интеллектуальных компьютерных систем]
 - ⇒ вид деятельности*:
 - реинжиниринг интеллектуальных компьютерных систем
 - Технология обеспечения безопасности интеллектуальных компьютерных систем
 - ⇒ вид деятельности*:
 - обеспечение безопасности интеллектуальных компьютерных систем
 - Технология эксплуатации интеллектуальных компьютерных систем конечными пользователями
 - ⇒ вид деятельности*:
 - эксплуатация интеллектуальных компьютерных систем конечными пользователями
}

Пункт 7.1.2.3. Текущее состояние и современные проблемы Искусственного интеллекта

Рассмотрим в каких направлениях должна происходить эволюция (повышение качества) деятельности в области *Искусственного интеллекта*, а также эволюция продуктов этой деятельности.

Подпункт 7.1.2.3.1 Текущее состояние и современные проблемы научно-исследовательской деятельности в области Искусственного интеллекта

В настоящее время научные исследования в области *Искусственного интеллекта* активно развиваются по широкому спектру различных направлений (*модели представления знаний*, различного вида логики – дедуктивные, индуктивные, абдуктивные, четкие, нечеткие, различного вида *искусственные нейронные сети*, машинное обучение, принятие решений, целеполагание, планирование поведения, ситуационное поведение, многоагентные системы, компьютерное зрение, распознавание, интеллектуальный анализ данных, мягкие вычисления и многое другое).

Однако:

- Отсутствует согласованность систем *понятий* в разных направлениях *Искусственного интеллекта* и, как следствие, отсутствует *семантическая совместимость* и *конвергенция* этих направлений, в результате чего существенно затруднена работа в направлении построения *Общей теории интеллектуальных систем* с высоким уровнем формализации. Существование и продолжающееся увеличение "высоты барьеров" между различными направлениями исследований в области *Искусственного интеллекта* проявляется в том, что специалист, работающий в рамках какого-либо направления *Искусственного интеллекта*, посещая заседания "не своей" секции на конференции по *Искусственному интеллекту*, мало что там может понять и, соответственно, извлечь полезного для себя;
- Отсутствует мотивация и осознание острой необходимости в указанной *конвергенции* между различными направлениями *Искусственного интеллекта*;
- Отсутствует реальное движение в направлении построения *Общей теории интеллектуальных систем*, поскольку отсутствует соответствующая мотивация и осознание острой практической необходимости в этом;
- Отсутствует строгое и согласованное уточнение понятия *интеллектуальной компьютерные системы*. До сих пор для этого используется Тест Тьюринга. Поверхностная трактовка Теста Тьюринга породила различные имитации интеллекта в стиле "светского" разговора или разговора на "завалинке". На самом деле должна учитываться содержательная, целевая установка диалога, в рамках которого интеллект *интеллектуальной компьютерной системы* определяется как её нетривиальный вклад в коллективное решение некоторой интеллектуальной (творческой) задачи.

Подпункт 7.1.2.3.2 Текущее состояние Стандарта интеллектуальных компьютерных систем

В настоящее время необходимость унификации и стандартизации *интеллектуальных компьютерных систем* не осознана, что существенно тормозит создание *комплексной технологии Искусственного интеллекта*.

Подпункт 7.1.2.3.3 Текущее состояние и современные проблемы развития технологии проектирования интеллектуальных компьютерных систем

Современная *технология Искусственного интеллекта* представляет собой целое семейство всевозможных частных технологий, ориентированных на разработку различного вида компонентов *интеллектуальных компьютерных систем*, реализующих самые различные модели представления и обработки информации, а также ориентированных на разработку различных классов *интеллектуальных компьютерных систем*.

Однако:

- Высока трудоемкость разработки *интеллектуальных компьютерных систем*;
- Необходима высокая квалификация разработчиков;
- Современные *технологии Искусственного интеллекта* принципиально не обеспечивают разработки таких *интеллектуальных компьютерных систем*, в которых устраняются недостатки современных *интеллектуальных компьютерных систем* и, в частности, обеспечивается достаточно высокий уровень интероперабельности;
- Совместимость технологий *проектирования различных классов интеллектуальных компьютерных систем Искусственного интеллекта* практически отсутствует и, как следствие, не обеспечивается *семантическая совместимость* и взаимодействие разрабатываемых *интеллектуальных компьютерных систем*, поэтому системная интеграция *интеллектуальных компьютерных систем* осуществляется вручную;
- Отсутствует *комплексная технология проектирования интеллектуальных компьютерных систем*;

- Нет совместимости между существующими частными технологиями проектирования различных компонентов интеллектуальных компьютерных систем (баз знаний, решателей задач, интеллектуальных интерфейсов). Есть инструментальные средства по разработке компонентов, но "склеивать" (соединять, интегрировать) разработанные компоненты надо вручную, поскольку нет комплексных инструментальных средств, позволяющих разрабатывать интеллектуальные компьютерные системы в целом.

Подпункт 7.1.2.3.4 Текущее состояние и современные проблемы развития технологии реализации спроектированных интеллектуальных компьютерных систем, а также их эксплуатации и сопровождения

Был сделан целый ряд попыток разработки *компьютеров нового поколения*, ориентированных на использование в интеллектуальных компьютерных системах. Но все они оказались неудачными, так как не были ориентированы на всё многообразие *моделей решения задач* в интеллектуальных компьютерных системах. В этом смысле они не были *универсальными компьютерами для интеллектуальных компьютерных систем*.

Разрабатываемые интеллектуальные компьютерные системы могут использовать самые различные комбинации *моделей решения интеллектуальных задач* (логических моделей, соответствующих различного вида логикам, нейросетевых моделей различного вида, моделей целеполагания, синтеза планов, моделей управления сложными объектами, моделей понимания и синтеза текстов естественного языка и т.д.). Однако, современные традиционные (фон-неймановские) компьютеры не в состоянии достаточно производительно интерпретировать всё многообразие указанных *моделей решения задач*. При этом разработка специализированных *компьютеров*, ориентированных на интерпретацию какой-либо одной *модели решения задач* (нейросетевой модели или какой-либо логической модели) проблему не решает, так как в *интеллектуальной компьютерной системе* необходимо использовать сразу несколько разных *моделей решения задач*, причём в различных сочетаниях.

В настоящее время отсутствует комплексный подход к технологическому обеспечению всех этапов жизненного цикла интеллектуальных компьютерных систем – не только к поддержке проектирования и реализации (сборки, производства) интеллектуальных компьютерных систем, но также и к технологической поддержке сопровождения, реинжиниринга и эксплуатации интеллектуальных компьютерных систем.

Семантическая недружественность *пользовательского интерфейса* и отсутствие встроенных интеллектуальных справочных систем, позволяющих запрашивать информацию об элементах интерфейса и возможностях системы, приводят к низкой эффективности эксплуатации всех возможностей *интеллектуальной компьютерной системы*.

Подпункт 7.1.2.3.5 Текущее состояние и современные проблемы прикладной инженерной деятельности в области Искусственного интеллекта

Накоплен достаточно большой опыт разработки интеллектуальных компьютерных систем самого различного назначения – систем автоматизации медицинской диагностики, а также диагностики сложных технических систем, интеллектуальных обучающих, информационно-справочных и help-систем, систем естественно-языкового общения, интеллектуальных компьютерных персональных ассистентов, интеллектуальных корпоративных систем, интеллектуальных систем ситуационного управления различного рода сложными объектами, систем интеллектуального анализа больших данных, систем технического зрения и анализа сцен, интеллектуальных порталов знаний, интеллектуальных систем автоматизации проектирования различного вида сложных объектов, интеллектуальных систем автоматизации подготовки к производству спроектированной продукции различного вида, интеллектуальных автоматизированных систем управления производства различного вида продуктов, а также многих других интеллектуальных компьютерных систем.

Однако:

- Уровень эффективности практического использования научных результатов в области Искусственного интеллекта явно не соответствует современному уровню развития самих этих научных результатов. Для того чтобы повысить уровень эффективности практического использования указанных научных результатов, необходимы совместные усилия и ученых, создающих новые модели решения интеллектуальных задач, и разработчиков технологий проектирования и реализации интеллектуальных компьютерных систем, и разработчиков прикладных интеллектуальных компьютерных систем.
- Отсутствует четкая систематизация многообразия интеллектуальных компьютерных систем, соответствующая систематизации автоматизируемых видов человеческой деятельности;
- Отсутствует конвергенция интеллектуальных компьютерных систем, обеспечивающих автоматизацию областей человеческой деятельности, принадлежащих одному и тому же виду человеческой деятельности;
- Отсутствует семантическая совместимость (семантическая унификация, взаимопонимание) между интеллектуальными компьютерными системами, основной причиной чего является отсутствие согласованной системы общих используемых понятий;

- Анализ проблем комплексной автоматизации всех видов человеческой деятельности убеждает в том, что дальнейшая автоматизация человеческой деятельности требует не только повышения уровня интеллекта соответствующих интеллектуальных компьютерных систем, но и к существенному повышению уровня их способности:
 - устанавливать свою семантическую совместимость (взаимопонимание) как с другими компьютерными системами, так и со своими пользователями;
 - поддерживать эту семантическую совместимость в процессе собственной эволюции, а также эволюции пользователей и других компьютерных систем;
 - координировать свою деятельность с пользователями и другими компьютерными системами при коллективном решении различных задач;
 - участвовать в распределении работ (подзадач) при коллективном решении различных задач.

Важно подчеркнуть то, что реализация вышеперечисленных способностей создаст возможность для существенной и даже полной автоматизации системной интеграции компьютерных систем в комплексы взаимодействующих интеллектуальных компьютерных систем и автоматизации реинжиниринга таких комплексов. Такая автоматизация системной интеграции и её реинжиниринга:

- даст возможность комплексам компьютерных систем самостоятельно адаптироваться к решению новых задач;
- существенно повысит эффективность эксплуатации таких комплексов компьютерных систем, так как реинжиниринг системной интеграции компьютерных систем, входящих в такой комплекс, часто востребован (например, при реконструкции предприятий);
- существенно сокращает число ошибок по сравнению с "ручным" (неавтоматизированным) выполнением системной интеграции и её реинжиниринга, которые, к тому же, требует высокой квалификации.

Таким образом следующий этап повышения уровня автоматизации человеческой деятельности настоятельно требует создания таких интеллектуальных компьютерных систем, которые могли бы сами (без системного интегратора) объединяться для совместного решения сложных задач.

Подпункт 7.1.2.3.6 Текущее состояние и современные проблемы учебной деятельности в области Искусственного интеллекта

Многие ведущие университеты осуществляют подготовку специалистов в области Искусственного интеллекта. При этом необходимо отметить следующие особенности и проблемы текущего состояния этой деятельности:

- Поскольку деятельность в области Искусственного интеллекта сочетает в себе и высокую степень научности и высокую степень сложности инженерных работ, подготовка специалистов в этой области требует одновременного формирования у них как научно-исследовательских навыков и знаний, так и инженерно-практических навыков и знаний, а также системной и технологической культуры и стиля мышления. С точки зрения методики и психологии обучения сочетание фундаментальной научной и инженерно-практической подготовки специалистов является достаточно сложной педагогической задачей;
- Отсутствует семантическая совместимость между различными учебными дисциплинами, что приводит к "мозаичности" восприятия информации;
- Отсутствует системный подход к подготовке молодых специалистов в области Искусственного интеллекта;
- Нет персонификации обучения, а также установки на выявление, раскрытие и развитие индивидуальных способностей;
- Отсутствует целенаправленное формирование мотивации к творчеству;
- Нет формирования навыков работы в реальных коллективах разработчиков. Отсутствует адаптация к реальной практической деятельности;
- Любая современная технология (в том числе и технология Искусственного интеллекта) должна иметь высокие темпы своего развития, поскольку без этого невозможно поддерживать высокий уровень её конкурентоспособности. Но для быстро развивающейся технологии требуется:
 - не просто высокая квалификация кадров, использующих и развивающих технологию;
 - но и высокие темпы повышения уровня этой квалификации, так как без этого невозможно эффективно использовать и развивать быстро меняющуюся технологию.

Из этого следует, что учебная деятельность в области Искусственного интеллекта и соответствующая ей технология должна быть не просто важной частью деятельности в области Искусственного интеллекта, а частью, глубоко интегрированной во все остальные виды деятельности в области Искусственного интеллекта. Так, например, каждая интеллектуальная компьютерная система должна быть ориентирована не только на обслуживание своих конечных пользователей, не только на организацию целенаправленного взаимодействия со своими разработчиками, которые постоянно совершенствуют эту систему, и не только на обеспечение минимального "порога вхождения" для новых конечных пользователей и разработчиков, но и на организацию постоянного и персонифицированного повышения квалификации каждого конечного пользователя и разработчика в

условиях постоянных изменений, вносимых в указанную *интеллектуальную компьютерную систему*. Для этого эксплуатируемая *интеллектуальная компьютерная система* должна "знать", что в ней изменилось, на что она способна и как эти способности инициировать (содержание и форма, соответствующих пользовательских команд).

Когда мы говорим о *конвергенции и интеграции* в области *Искусственного интеллекта*, речь идет не только о конвергенции между *интеллектуальными компьютерными системами*, но также и между различными *видами* и областями *человеческой деятельности*. Таким образом, *учебная деятельность*, направленная на подготовку специалистов в области *Искусственного интеллекта*, органически входит в состав *деятельности в области Искусственного интеллекта*, а важнейшим направлением повышения эффективности этой деятельности является её *конвергенция и интеграция* с другими видами *деятельности в области Искусственного интеллекта*.

Подпункт 7.1.2.3.7 Текущее состояние и современные проблемы организационной деятельности в области Искусственного интеллекта

Острая потребность в существенном повышении уровня автоматизации в самых различных областях *человеческой деятельности* (в промышленности, медицине, транспорте, образовании, строительстве и во многих других), а также современные результаты в развитии *технологий Искусственного интеллекта* привели к существенному расширению работ по созданию *прикладных интеллектуальных компьютерных систем* и к появлению большого количества коммерческих организаций, ориентированных на разработку таких приложений.

Однако:

- Не так просто обеспечить баланс тактических и стратегических направлений развития всех видов деятельности в области *Искусственного интеллекта* (научно-исследовательской деятельности, развития технологии проектирования и производства интеллектуальных компьютерных систем, разработки прикладных систем, образовательной деятельности), а также баланс между всеми перечисленными *видами деятельности*;
- В настоящее время отсутствует глубокая *конвергенция* различных *видов деятельности* в области *Искусственного интеллекта* (в первую очередь, конвергенция развития технологий *Искусственного интеллекта* и разработки различных прикладных *интеллектуальных компьютерных систем*), что существенно затрудняет развитие каждого из этих видов деятельности и, в частности, существенно затрудняет при разработке интеллектуальных решателей задач интеграцию различных моделей решения задач (например, логических моделей, нейросетевых моделей, моделей обработки текстов естественных языков, моделей обработки сигналов – аудиосигналов, изображений);
- Высокий уровень научности работ в области *Искусственного интеллекта* предъявляет особые требования к квалификации сотрудников и к их способности работать в составе *творческих коллективов*.

Пункт 7.1.2.4. Ключевые задачи и методологические проблемы современного этапа развития Искусственного интеллекта

К числу **ключевых задач** современного этапа развития *Искусственного интеллекта* следует отнести:

- Построение *Общей формальной теории интеллектуальных компьютерных систем*, в рамках которой была бы обеспечена совместимость всех направлений *Искусственного интеллекта*, всех моделей представления знаний, всех моделей решения задач, всех компонентов *интеллектуальных компьютерных систем*. Это предполагает:
 - Уточнение требований, предъявляемых к *интеллектуальным компьютерным системам нового поколения* – уточнение свойств *интеллектуальных компьютерных систем*, определяющих высокий уровень их *интеллекта*;
 - *Конвергенцию и интеграцию* всевозможных видов знаний и всевозможных моделей решения задач в рамках каждой *интеллектуальной компьютерной системы*.
- Создание *инфраструктуры*, обеспечивающей интенсивное перманентное развитие *Общей формальной теории интеллектуальных компьютерных систем* в самых различных направлениях, гарантирующее сохранение логико-семантической целостности этой теории и совместимости всех направлений ее развития;
- На основе *Общей формальной теории интеллектуальных компьютерных систем* построение *Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*, обладающих высоким уровнем интероперабельности и совместимости;
- Создание *инфраструктуры*, обеспечивающей интенсивное перманентное развитие *Комплексной технологии разработки и эксплуатации интеллектуальных компьютерных систем нового поколения* в самых различных направлениях, гарантирующее сохранение целостности этой *технологии* и совместимости всех направлений ее развития;

- Разработку **компьютеров нового поколения**, ориентированных на высокопроизводительную интерпретацию логико-семантических моделей интеллектуальных компьютерных систем нового поколения;
- Создание **Глобальной экосистемы интеллектуальных компьютерных систем нового поколения**, ориентированной на комплексную автоматизацию различных видов человеческой деятельности.

Эпицентром современных методологических проблем развития *человеческой деятельности* в области *Искусственного интеллекта* является **конвергенция** и глубокая интеграция всех видов, направлений и результатов этой деятельности. Уровень взаимосвязи, взаимодействия и **конвергенции** между различными видами и направлениями деятельности в области *Искусственного интеллекта* в настоящее время явно недостаточен. Это приводит к тому, что каждая из них развивается обособленно, независимо от других. Речь идет о **конвергенции** между такими направлениями *Искусственного интеллекта*, как представление знаний, решение интеллектуальных задач, интеллектуальное поведение, понимание и др., а также между такими *видами человеческой деятельности в области Искусственного интеллекта*, как научные исследования, разработка технологий, разработка приложений, образование, бизнес. Почему на фоне уже достаточно длительного интенсивного развития научных исследований в области *Искусственного интеллекта* до сих пор не создан рынок интеллектуальных компьютерных систем и комплексная технология *Искусственного интеллекта*, обеспечивающая разработку широкого спектра интеллектуальных компьютерных систем самого различного назначения и доступной широкому контингенту инженеров. Потому что сочетание высокого уровня научности и прагматизма этой проблемы требует для ее решения принципиально нового подхода к организации взаимодействия ученых, работающих в области *Искусственного интеллекта*, разработчиков средств автоматизации проектирования интеллектуальных компьютерных систем, разработчиков средств реализации интеллектуальных компьютерных систем, включая средства аппаратной поддержки интеллектуальных компьютерных систем, разработчиков прикладных интеллектуальных компьютерных систем. Такое целенаправленное взаимодействие должно осуществляться как в рамках каждой из этих форм деятельности в области *Искусственного интеллекта*, так и между ними. Таким образом, основной тенденцией дальнейшего развития теоретических и практических работ в области *Искусственного интеллекта* является **конвергенция** как самых разных видов (форм и направлений) *человеческой деятельности* в области *Искусственного интеллекта*, так и самых разных продуктов (результатов) этой деятельности. Необходимо ликвидировать барьеры между различными видами и продуктами деятельности в области *Искусственного интеллекта* в целях обеспечения их совместимости и интегрируемости.

Конвергенция разрабатываемых интеллектуальных компьютерных систем преобразует набор индивидуальных (автономных) интеллектуальных компьютерных систем различного назначения в коллектив активно взаимодействующих интеллектуальных компьютерных систем для совместного (коллективного) решения сложных (комплексных) задач и для перманентной поддержки совместимости между всеми интеллектуальными компьютерными системами, входящими в коллектив, в процессе индивидуальной эволюции каждой из этих систем.

Конвергенция конкретных искусственных сущностей (например, технических систем) есть стремление к их унификации (в частности, к стандартизации), т.е. стремление к минимизации многообразия форм решения аналогичных практических задач – стремление к тому, чтобы все, что можно сделать одинаково, делалось одинаково, но без ущерба требуемого качества. Последнее очень важно, так как безграмотная стандартизация может привести к существенному торможению прогресса. Ограничение многообразия форм не должно приводить к ограничению содержания, возможностей. Образно говоря, "словам должно быть тесно, а мыслям–свободно".

Методологически **конвергенция** искусственно создаваемых сущностей (артефактов) сводится (1) к выявлению (обнаружению) принципиальных сходств между этими сущностями, которые часто весьма закамуфлированы и их трудно "увидеть" и (2) к реализации обнаруженных сходств одинаковым образом (в одинаковой форме, в одинаковом "синтаксисе"). Образно говоря, от "семантической" (смысловой) эквивалентности требуется перейти и к "синтаксической" эквивалентности. Кстати, в этом как раз и заключается суть *смыслового представления информации*, целью которого является создание такой языковой среды (смылового пространства), в рамках которого (1) семантически эквивалентные информационные конструкции полностью совпадали бы, а (2) **конвергенция** информационных конструкций сводилась бы к выявлению изоморфных фрагментов этих конструкций.

К числу общих методологических проблем современного этапа развития *Искусственного интеллекта* можно отнести:

- Отсутствие массового осознания того, что создание рынка интеллектуальных компьютерных систем нового поколения, обладающих *семантической совместимостью* и высоким уровнем *интероперабельности*, а также создание комплексов (экосистем), состоящих из таких интеллектуальных компьютерных систем и обеспечивающих автоматизацию различных видов человеческой деятельности, невозможно, если коллективы разработчиков таких систем и комплексов существенно не повысят уровень *социализации* **всех** своих сотрудников. Уровень качества коллектива разработчиков, т.е. уровень квалификации сотрудников и уровень согласованности их деятельности, должен превышать уровень качества систем, разрабатываемых этим коллективом. Особое значение рассматриваемая проблема согласованности деятельности специалистов в области *Искусственного интеллекта* имеет для построения *Общей формальной теории интеллектуальных*

компьютерных систем нового поколения, а также Комплексной технологии разработки и эксплуатации интеллектуальных компьютерных систем нового поколения;

- Далеко не всеми учеными, работающими в области Искусственного интеллекта, принимается прагматичность, практическая направленность Искусственного интеллекта;
- Не всеми принимается необходимость **конвергенции** различных направлений Искусственного интеллекта и необходимость их интеграции в целях построения *Общей формальной теории интеллектуальных компьютерных систем*;
- Не всеми принимается необходимость **конвергенции** различных видов деятельности в области Искусственного интеллекта;
- Важным препятствием для **конвергенции** результатов научно-технической деятельности является сформировавшийся в науке и технике акцент на выявлении не сходств, а отличий. Чтобы убедиться в этом достаточно обратить внимание на то, что уровень научных результатов оценивается научной новизной, которая может имитироваться новизной не по существу, а по форме представления (например, с помощью новых понятий или даже новых терминов). Результаты в технике, например, в патентах также оцениваются отличиями от предшествующих технических решений. Но для **конвергенции** нужны другие акценты – не поиск отличий, а выявление неочевидных сходств и превращение их в очевидные сходства, представленные в одинаковой форме;
- Нет движения к построению *Комплексной технологии проектирования, реализации, сопровождения, реинжиниринга и эксплуатации интеллектуальных компьютерных систем*. Речь идет о комплексном подходе к технологическому обеспечению всех этапов жизненного цикла интеллектуальных компьютерных систем;
- Нет активного развития работ по созданию *Глобальной экосистемы интеллектуальных компьютерных систем нового поколения*;
- В основе современной организации и автоматизации человеческой деятельности лежит "Вавилонское столпотворение" постоянно расширяемого многообразия языков. Имеются в виду не только *естественные языки*, но и *формальные языки*, направленные на точное представление знаний различного вида. Многообразие различных *специализированных языков* пронизывает всю человеческую деятельность – во многих областях *человеческой деятельности* для решения различных видов задач, для разработки различных *моделей решения задач* создаются *специализированные языки*. Примером этого является многообразие языков программирования. *Специализированные языки* могут и должны появляться, но только как *подъязыки* более общих языков, синтаксис каждого из которых совпадает с *синтаксисом* всех соответствующих ему *подъязыков*. При этом в рамках *Общей формальной теории интеллектуальных компьютерных систем* должен быть выделен один *универсальный формальный язык* – язык-ядро, по отношению к которому все остальные используемые *формальные языки* являются *подъязыками*. *Денотационная семантика* указанного *универсального формального языка* должна задаваться соответствующей *формальной онтологией* максимально высокого уровня. Иначе о какой **конвергенции** и интеграции знаний, о какой *семантической совместимости* компьютерных систем можно вести речь.

В основе предлагаемой организации человеческой деятельности в области Искусственного интеллекта лежат следующие положения:

- **комплексная конвергенция** – как "вертикальная" конвергенция между различными видами деятельности в области Искусственного интеллекта, так и "горизонтальная" конвергенция в рамках каждого из этих видов деятельности, соответствующая различным компонентам или различным классам интеллектуальных компьютерных систем – базам знаний, решателям задач, различным моделям решения задач, различным видам интерфейсов (зрительным, аудио, естественно-языковым), робототехническим интеллектуальным компьютерным системам, интеллектуальным обучающим системам, интеллектуальным автоматизированным системам управления, интеллектуальным системам автоматизации проектирования и т.д.);
- "**горизонтальная**" конвергенция в рамках каждого вида человеческой деятельности в области Искусственного интеллекта включает в себя:
 - конвергенцию в рамках *научно-исследовательской деятельности* в области Искусственного интеллекта, означающую переход от независимого развития различных направлений Искусственного интеллекта к общей теории интеллектуальных компьютерных систем;
 - конвергенцию в рамках развития *технологий Искусственного интеллекта*, означающую переход от независимого развития частных технологий к созданию единого комплекса семантически совместимых частных технологий;
 - конвергенцию в рамках *инженерной деятельности* в области Искусственного интеллекта, означающую переход от практики независимой разработки различных прикладных интеллектуальных компьютерных систем к разработке комплекса (экосистемы) интероперабельных интеллектуальных компьютерных систем;
 - конвергенцию в рамках *учебной деятельности* в области Искусственного интеллекта, обозначающую переход от изучения отдельных учебных дисциплин к формированию у молодых специалистов целостной картины текущего состояния Искусственного интеллекта и проблемных направлений дальнейшего развития;

- конвергенцию в рамках общей организационной деятельности в области Искусственного интеллекта, переход от отдельных вышеперечисленных видов деятельности в области Искусственного интеллекта к единому комплексу всех этих видов деятельности и обеспечивающую конвергенцию и интеграцию указанных видов деятельности в области Искусственного интеллекта, что существенно повысит их качество, поскольку каждый из этих видов деятельности находится в сильной зависимости от всех остальных;
- организация разработки и перманентного развития предлагаемой технологии в виде **открытого международного проекта**, предоставляющего:
 - свободный доступ к использованию текущей версии разрабатываемой технологии;
 - возможность каждому желающему войти в состав коллектива разработчиков этой технологии;
- **поэтапность** процесса формирования рынка семантически совместимых и активно взаимодействующих между собой интеллектуальных компьютерных систем нового поколения, начальными этапами которого являются:
 - разработка логико-семантических моделей (баз знаний) нескольких прикладных интеллектуальных компьютерных систем нового поколения;
 - программная реализация на современных компьютерах платформы интерпретации логико-семантических моделей интеллектуальных компьютерных систем нового поколения;
 - установка каждой разработанной логико-семантической модели прикладной интеллектуальной компьютерной системы на разработанную программную платформу интерпретации таких моделей с последующим тестированием и реинжинирингом каждой такой модели;
 - разработка и перманентное совершенствование логико-семантической модели (базы знаний) интеллектуальной компьютерной метасистемы, которая содержит (1) описание стандарта интеллектуальных компьютерных систем нового поколения, (2) библиотеку многократно используемых (в различных интеллектуальных компьютерных системах) знаний различного вида и, в частности, различных методов решения задач, (3) методы проектирования и средства поддержки проектирования различных видов компонентов интеллектуальных компьютерных систем (компонентов баз знаний, решателей задач, интерфейсов);
 - разработка ассоциативного семантического компьютера в качестве аппаратной реализации платформы интерпретации логико-семантических моделей интеллектуальных компьютерных систем нового поколения;
 - перенос разработанных логико-семантических моделей интеллектуальных компьютерных систем нового поколения на новые, более эффективные варианты реализации платформы интерпретации этих моделей;
 - развитие рынка интеллектуальных компьютерных систем нового поколения в виде Глобальной экосистемы, состоящей из активно взаимодействующих таких систем и ориентированной на комплексную автоматизацию всех видов человеческой деятельности;
 - создание **рынка знаний** на основе Глобальной экосистемы интеллектуальных компьютерных систем нового поколения;
 - автоматизация реинжиниринга эксплуатируемых интеллектуальных компьютерных систем нового поколения в направлении приведения их в соответствие с новыми версиями стандарта интеллектуальных компьютерных систем путем автоматической замены устаревших компонентов в этих системах на текущие версии этих компонентов.

Следует особо подчеркнуть, что **ключевым фактором решения рассматриваемых методологических проблем** в области Искусственного интеллекта являются различные направления **конвергенции и интеграции**, обеспечивающие переход к **интеллектуальным компьютерным системам нового поколения**, к соответствующей технологии комплексной поддержки их жизненного цикла и к существенному повышению уровня автоматизации всего комплекса человеческой деятельности:

- конвергенция и интеграция различных моделей представления и обработки информации в интеллектуальных компьютерных системах нового поколения
 - конвергенция и интеграция различных видов знаний в базах знаний интеллектуальных компьютерных систем нового поколения
 - конвергенция и интеграция различных моделей решения задач
 - конвергенция и интеграция различных видов интерфейсов интеллектуальных компьютерных систем нового поколения
- конвергенция и интеграция различных направлений Искусственного интеллекта в целях построения Общей формальной теории интеллектуальных компьютерных систем нового поколения
- конвергенция и интеграция технологий проектирования различных компонентов интеллектуальных компьютерных систем нового поколения в целях построения комплексной Технологии проектирования интеллектуальных компьютерных систем нового поколения

- *конвергенция и интеграция технологий поддержки различных этапов жизненного цикла интеллектуальных компьютерных систем нового поколения в целях построения Технологии комплексной поддержки всех этапов жизненного цикла интеллектуальных компьютерных систем нового поколения*
- *конвергенция и интеграция различных видов человеческой деятельности в области Искусственного интеллекта (научно-исследовательской деятельности, развития технологического комплекса, прикладной инженерии, образовательной деятельности) для повышения уровня согласованности и координации этих видов деятельности, а также для повышения уровня их комплексной автоматизации с помощью семантически совместимых интеллектуальных компьютерных систем нового поколения*
- *конвергенция и интеграция самых различных видов и областей человеческой деятельности, а также средств комплексной автоматизации этой деятельности с помощью интеллектуальных компьютерных систем нового поколения*

Конечным практическим результатом человеческой деятельности в области Искусственного интеллекта является:

- Реорганизация и комплексная автоматизация человеческой деятельности в области Искусственного интеллекта с помощью интеллектуальных компьютерных систем нового поколения;
- Поэтапное создание глобальной сети эффективно взаимодействующих интеллектуальных компьютерных систем нового поколения, обеспечивающих комплексную автоматизацию всевозможных видов и областей человеческой деятельности.

Переход от современных интеллектуальных компьютерных систем к интеллектуальным компьютерным системам нового поколения и к соответствующей комплексной технологии не требует от специалистов в области Искусственного интеллекта изменения сферы их научных интересов. От них требуется только преодолеть синдром “Вавилонского столпотворения”, оформляя свои научные результаты как часть общего коллективного продукта.

Проблемы текущего этапа развития Искусственного интеллекта, направленного на создание Общей теории и технологии интеллектуальных компьютерных систем нового поколения, требуют фундаментального комплексного междисциплинарного подхода и принципиально новой организации научно-технической деятельности.

Пункт 7.1.2.5. Комплексная автоматизация человеческой деятельности в области Искусственного интеллекта с помощью интеллектуальных компьютерных систем нового поколения

В рамках Технологии OSTIS поддержка жизненного цикла интеллектуальных компьютерных систем нового поколения (*ostis-систем*) осуществляется на основе Метасистемы OSTIS, которая относится к классу *ostis-систем* и фактически является формой реализации указанной Технологии. Автоматизация поддержки жизненного цикла *ostis-систем* осуществляется как в форме инструментального обслуживания инженерной деятельности (в частности, Метасистема OSTIS является системой автоматизации проектирования *ostis-систем*), так и в форме информационного обслуживания указанной деятельности. Для этого база знаний Метасистема OSTIS содержит:

- текущее состояние полного текста Стандарта *ostis-систем*;
- Текущее состояние Библиотеки многократно используемых компонентов *ostis-систем*;
- Используемые и реализуемые инженерами методики поддержки жизненного цикла *ostis-систем*;
- Документацию инструментальных средств, инженерами для поддержки жизненного цикла *ostis-систем*.

Кроме всего этого, Метасистема OSTIS:

- Обеспечивает автоматизацию Поддержки жизненного цикла Стандарта *ostis-систем*, то есть обеспечивает организацию взаимодействия между авторами этого Стандарта, направленного на перманентное его развитие;
- Обеспечивает автоматизацию Поддержки жизненного цикла Технологии OSTIS, которая сводится к поддержке жизненного цикла основной части базы знаний Метасистемы OSTIS, которая является полной документацией текущего состояния Технологии OSTIS.

Автоматизация других направлений Человеческой деятельности в области Искусственного интеллекта также может осуществляться с помощью *ostis-систем*, семантически совместимых и взаимодействующих с Метасистемой OSTIS в рамках Экосистемы OSTIS.

§ 7.1.3. Ключевые виды и области человеческой деятельности

§ 7.1.4. Проблемы и перспективы комплексной автоматизации всевозможных видов и областей человеческой деятельности с помощью интеллектуальных компьютерных систем нового поколения

Выше было рассмотрено то, как осуществляется и автоматизируется с помощью интеллектуальных компьютерных систем нового поколения весь комплекс Человеческой деятельности в области Искусственного интеллекта. Сейчас обобщим это и рассмотрим принципы организации и комплексной автоматизации человеческой деятельности в целом, т.е. автоматизации самых различных видов и областей человеческой деятельности.

Пункт 7.1.4.1. Общие принципы систематизации человеческой деятельности и ее комплексной автоматизации с помощью интеллектуальных компьютерных систем нового поколения

Опыт комплексной организации, структуризации и автоматизации человеческой деятельности в области Искусственного интеллекта (в области создания и сопровождения интеллектуальных компьютерных систем) можно обобщить и для других областей человеческой деятельности. Это обусловлено следующими причинами:

- Во-первых, потому, что человеческая деятельность, направленная на поддержку всего жизненного цикла интеллектуальных компьютерных систем нового поколения, является частной областью деятельности по отношению к виду человеческой деятельности, направленному на (обеспечивающему) поддержку всего жизненного цикла любой искусственной (искусственно создаваемой) сущности (любого артефакта). В зависимости от сложности искусственно создаваемой сущности, уровень сложности человеческой деятельности, направленной на поддержку жизненного цикла этой сущности, может быть самым различным, но общая структура этой деятельности, соответствующая различным этапам жизненного цикла искусственно создаваемых сущностей, а также необходимым направлениям обеспечения этой инженерной деятельности является однаковой для искусственных сущностей различных классов. К указанным направлениям обеспечения поддержки жизненного цикла искусственных сущностей относятся:
 - научно-исследовательская деятельность, направленная на изучение искусственных сущностей соответствующего класса;
 - разработка стандарта искусственных сущностей указанного класса;
 - разработка технологии поддержки искусственных сущностей указанного класса;
 - подготовка кадров, способных осуществлять поддержку жизненного цикла искусственных сущностей указанного класса, т.е. способных эффективно использовать указанную выше технологию;
 - подготовка кадров, способных участвовать в указанной выше научно-исследовательской деятельности;
 - подготовка кадров, способных участвовать в разработке стандарта искусственных сущностей заданного класса;
 - подготовка кадров, способных участвовать в разработке и развитии указанной выше технологии;
 - организационное обеспечение всего комплекса работ по развитию и использованию указанной технологии.
- Во-вторых, потому, что многие сложные технические системы фактически становятся интеллектуальными компьютерными системами (в т.ч. распределенными) с различными наборами сенсорных и эффекторных подсистем – интеллектуальными автомобилями с автопилотом и автоштурманом, интеллектуальными заводами-автоматами, умыми домами, умыми городами и т.п.
- В-третьих, потому, что характер деятельности интеллектуальных компьютерных систем нового поколения и характер деятельности каждого человека и каждой организации по сути мало чем отличаются, поскольку интеллектуальные компьютерные системы нового поколения становятся равноправными партнерами (субъектами) человеческой деятельности, т.к. уровень их самостоятельности, ответственности, интероперабельности и интеллектуальности приближается к соответствующим качествам естественных субъектов человеческой деятельности (физическими лицами, юридическими лицами, подразделениями крупных организаций, неформальными организациями).

Итак, структуризацию человеческой деятельности в области Искусственного интеллекта на основе понятий вида деятельности, области деятельности, продукта деятельности (объекта деятельности) можно легко обобщить для всех научно-технических дисциплин, что дает возможность рассматривать автоматизацию деятельности в рамках всех научно-технических дисциплин с общих позиций, т.к. автоматизация различных видов деятельности в рамках различных научно-технических дисциплин может выглядеть аналогичным образом, а иногда может быть реализована с помощью одной и той же интеллектуальной компьютерной системы. Так например, любая

интеллектуальная компьютерная система автоматизации проектирования технических систем заданного вида может быть построена на основе интеллектуальной компьютерной системы автоматизации проектирования и реинжиниринга баз знаний, поскольку результатом проектирования любой технической системы является формальная модель (описание, спецификация, документация) этой технической системы, обладающая достаточно полнотой для воспроизведения (реализации) этой системы.

На текущем этапе развития Искусственного интеллекта необходимо переходить от автоматизации отдельных видов человеческой деятельности к интегрированной автоматизации всего комплекса человеческой деятельности, к созданию и постоянной эволюции всей Глобальной экосистемы интеллектуальных компьютерных систем, самостоятельно взаимодействующих как между собой, так и с людьми, автоматизацию деятельности которых они осуществляют, а также с современными компьютерными системами, не являющимися интеллектуальными системами. При этом надо помнить, что основные "накладные" расходы, основные проблемы, возникают на "стыках" при интеграции различных технических решений. Разработчик каждой подсистемы должен гарантировать отсутствие указанных "накладных" расходов. При этом необходимо подчеркнуть, что следует ориентироваться не столько на создание эффективной Глобальной экосистемы интеллектуальных компьютерных систем, сколько на создание эффективных методик и средств, направленных на *перманентную эволюцию* такой экосистемы.

Методика комплексной автоматизации человеческой деятельности включает в себя следующие этапы:

- Построение общей **структуре человеческой деятельности**, в основе которой лежит иерархия человеческой деятельности по видам деятельности и продуктам деятельности с четкой фиксацией различного вида связей между различными компонентами этой структуры.
- Формализация различных видов человеческой деятельности.
- Разработка **технологии**, обеспечивающей максимально возможную автоматизацию этой деятельности с помощью интеллектуальных компьютерных систем нового поколения.
- Обеспечение максимально возможной **конвергенции** различных видов деятельности, что позволит сократить многообразие средств автоматизации (т.е. соответствующих интеллектуальных компьютерных систем нового поколения).
- Обеспечение максимально возможной **конвергенции технологий** выполнения одного и того же вида деятельности для разных объектов деятельности (конвергенции технологий проектирования объектов различных классов, конвергенции технологий мониторинга, профилактики и диагностики для агентов различных классов и т.д.) и, тем самым, обеспечить **конвергенцию** соответствующих средств автоматизации, построенных на основе интеллектуальных компьютерных систем нового поколения.

Пункт 7.1.4.2. Многообразие видов человеческой деятельности и связей между ними

Базовым видом человеческой деятельности можно считать **поддержку жизненного цикла** различных сущностей.

Классом объектов деятельности для этого вида деятельности является класс всевозможных социально значимых объектов, на которые имеет смысл воздействовать, поддержку жизненного цикла которых целесообразно осуществлять.

поддержка жизненного цикла

:= [поддержка жизненного цикла социально значимых сущностей]

∈ вид деятельности

⇒ частный вид деятельности, выполняемой на некотором этапе*:

- проектирование
- производство
- начальное обучение
:= [настройка]
- мониторинг качества
:= [плановое обследование и диагностика]
- восстановление требуемого уровня качества
:= [ремонт, лечение]
- реинжиниринг
:= [обновление, совершенствование]
- обеспечение безопасности
- использование
:= [эксплуатация, употребление]

⇒ частный вид деятельности над подклассом объектов деятельности*:

- научно-исследовательская деятельность
:= [поддержка жизненного цикла научных теорий]

- ⇒ *класс объектов деятельности**:
научная теория
- стандартизация
:= [поддержка жизненного цикла стандартов]
- ⇒ *класс объектов деятельности**:
стандарт
- поддержка жизненного цикла технологий
⇒ *класс объектов деятельности**:
технология
- образовательная деятельность
:= [учебная деятельность]
:= [поддержка жизненного цикла кадровых ресурсов]
⇒ *класс объектов деятельности**:
кадровый ресурс
- поддержка жизненного цикла метасистем комплексного управления поддержкой и обеспечение поддержки жизненного цикла сущностей соответствующих классов
⇒ *класс объектов деятельности**:
метасистема комплексного управления поддержкой и обеспечением поддержки жизненного цикла сущностей соответствующих классов

Когда выше рассматривалась общая структура человеческой деятельности путем обобщения структуры **Человеческой деятельности в области Искусственного интеллекта**, мы:

- ввели понятие *вида деятельности*
- в качестве "исходной точки" обобщения выбрали такой вид деятельности, как поддержка жизненного цикла интеллектуальных компьютерных систем
- далее расширяли класс *объектов деятельности* выбранного вида деятельности,
 - переходя от класса интеллектуальных компьютерных систем к классу всевозможных искусственных материальных сущностей
 - объединяя класс искусственных материальных сущностей с классом естественных материальных сущностей (материальных сущностей естественного происхождения), а также с классом естественно-искусственных материальных сущностей (либо естественных искусственно модифицированных материальных сущностей, либо гибридных естественно-искусственных материальных сущностей, имеющих компоненты как естественного так и искусственного происхождения);
 - объединяя класс материальных сущностей с классом информационных ресурсов, т.е. социально значимых информационных конструкций (документов), являющихся продуктами соответствующих действий или деятельности (*научными теориями, стандартами, базами знаний, методами, проектными документациями* соответствующих создаваемых объектов)
 - объединяя класс материальных сущностей информационных ресурсов с классом материально-информационных объектов, которым, в частности, относятся различные технологии.

Таким образом, поддержка жизненного цикла различных социально значимых объектов является особым видом человеческой деятельности. Во-первых, эффективность Человеческой деятельности в целом зависит (1) от длительности социально полезной (активной) фазы жизненного цикла используемых объектов и (2) от объема затрат общества на поддержание необходимых социально полезных свойств используемых объектов. Во-вторых, характер и технология поддержки жизненного цикла разных видов социально значимых объектов могут существенно отличаться друг от друга. Так, например, существенно отличается организация поддержки жизненного цикла автомобилей, традиционных компьютерных систем различного назначения, современных интеллектуальных компьютерных систем, интероперабельных интеллектуальных компьютерных систем, людей, предприятий, домов, различных юридических лиц, населенных пунктов и др. При этом типология социально значимых объектов, жизненный цикл которых должен поддерживаться, включает в себя самые разнообразные классы объектов - искусственно создаваемые материальные информационные продукты человеческой деятельности, всех людей, всевозможные социальные сообщества и предприятия. Многообразие типов социально значимых объектов порождает многообразие соответствующих им технологий, что усложняет комплексную автоматизацию человеческой деятельности в целом.

Тем не менее, заметим, что видов человеческой деятельности значительно меньше, чем областей человеческой деятельности. Это в определенной степени обусловлено тем, что видов связей между сущностями (относительных понятий) значительно меньше, чем классов различных сущностей. Данное обстоятельство указывает на то, что в основе движения в направление глобальной автоматизации деятельности общества должна лежать ориентация на грамотную систематизацию видов человеческой деятельности, и на их максимально глубокую конвергенцию (как внутри каждого вида деятельности, так и между различными видами). Благодаря этому искусственно привносимое многообразие средств автоматизации человеческой деятельности может быть сведено к минимуму.

следует отличать

- Э { • научно-исследовательская деятельность
 := [поддержка жизненного цикла научных теорий]
 • стандартизация
 := [разработка и развитие стандартов]
 := [поддержка жизненного цикла стандартов]
 • поддержка жизненного цикла технологий
}

Научно-исследовательская деятельность направлена на *изучение сущностей заданного класса*, на изучение принципов, лежащих в основе их структуры и функционирования. В рамках этого вида деятельности важна новизна и конкуренция идей и подходов, важно соотношение между структурой (архитектурой) организацией функционирования исследуемых *сущностей* и общими характеристиками (параметрами) качества этих сущностей, общими предъявляемыми к ним требованиями. Продуктом рассматриваемой деятельности является *Общая теория сущностей заданного класса*, которая отражает множественность и даже противоречивость разных точек зрения и важнейшим направлением развития (эволюции) которой является *ближение (конвергенция)* различных точек зрения и обеспечение совместимости и непротиворечивости между ними. В основе *научно-исследовательской деятельности* лежит конкуренция точек зрения, принципиальная новизна идей и верифицированных результатов, направленных на выявление и обоснование неочевидных свойств и закономерностей соответствующей *предметной области*, на разработку методов решения различных *классов задач*, решаемых в рамках этой *предметной области*. Цель *научно-исследовательской деятельности* и требуемая детализация вырабатываемых знаний об объектах исследований соответствующих *предметных областей*.

В отличие от *научно-исследовательской деятельности* в основе разработки *стандарта* создаваемых сущностей и разработки соответствующей *технологии* поддержки их жизненного цикла лежит согласование различных точек зрения (поиск консенсуса) и максимально возможное их упрощение (соблюдение принципа Бритвы Оккама). Необходимость такой методологической установки обусловлена массовым характером *человеческой деятельности* по созданию и *поддержке жизненного цикла* соответствующего класса сущностей и необходимостью вовлечения в эту деятельность людей с *разной* (в т.ч. и достаточно низкой) квалификацией. В процессе разработки *стандарта сущностей заданного класса* важна не конкуренция различных точек зрения, а их *конвергенция, семантическая совместимость* и глубокая интеграция. Каждый *стандарт искусственных сущностей заданного класса* – это согласованная на текущий момент точка зрения (консенсус) о структуре, функционировании, свойствах и закономерностях искусственных сущностей заданного класса, согласованная (общепризнанная) часть *Общей теории искусственных сущностей заданного класса*, доступная для понимания широкому контингенту практиков (инженеров), которые проектируют, производят и поддерживают весь жизненный цикл конкретных *искусственных сущностей заданного класса*.

При создании и *поддержке жизненного цикла технологий* должны учитываться ряд требований, предъявляемых к любым технологиям:

- комплексность – максимально возможное покрытие всех задач, которые должны решаться с помощью *технологии* (как минимум всех этапов жизненного цикла)
- максимально возможная простота в использовании *технологии* (необходимая полнота документации, интеллектуальная help-поддержка, отсутствие лишней информации, которая не является необходимой для использования *технологии*, наличие богатой и систематизированной библиотеки типовых многократно используемых решений)

Общество – это иерархическая система взаимодействующих индивидуальных и коллективных субъектов, каждый из которых:

- Производит либо часть социально значимой продукции, производимой коллективным субъектом, в состав которого входит данный субъект, либо целостный социально-значимый продукт (производимый товар), потребляемый другими внешними субъектами или оказывает некоторую услугу другому субъекту, направленную на обеспечение жизнедеятельности и совершенствование этого другого субъекта.
- Потребляет продукцию, произведенную другими субъектами, необходимую для производства собственной продукции (сырец и оборудование), а также необходимую для обеспечения своей жизнедеятельности.
- Потребляет услуги, оказываемые другими субъектами необходимые для производства собственной продукции или услуг, а также необходимые для совершенствования своей деятельности.

Основными направлениями автоматизации всего комплекса *человеческой деятельности* являются:

- автоматизация социально полезной профессиональной деятельности всех субъектов деятельности (как индивидуальных субъектов – всех физических лиц, так и всевозможных коллективных – корпоративных субъектов, в т.ч. юридических лиц)
- автоматизация обеспечения (создания) комфортных условий для всех субъектов деятельности общества на основе мониторинга деятельности и конкретного (адаптированного) содействия эволюции каждого субъекта с учетом его непосредственных потребностей и проблем.

Организация взаимодействий каждого субъекта с внешней средой должна осуществляться как со стороны этого субъекта, так и со стороны указанной внешней среды (т.е. со стороны общества). Общество должно повернуться "лицом" к каждому субъекту и не бросать его на произвол судьбы. В настоящее время создание (обеспечение) условий субъектов деятельности общества отдано на откуп каждого такого субъекта. Общество в лице специально предназначенных для этого других субъектов оказывает услуги и снабжает товарами по заказу (по инициативе) нуждающегося в этом субъекта. Таким образом, ответственность за развитие каждого субъекта деятельности ложится исключительно на "плечи" этого субъекта. Поддержка общества носит общий характер и никак не учитывает особенности текущего положения каждого субъекта.

Важнейшей причиной, препятствующей дальнейшему повышению общего уровня автоматизации человеческой деятельности является то, что автоматизация различных областей человеческой деятельности осуществляется локально.

На современном этапе применения интеллектуальных компьютерных систем основной проблемой является не автоматизация локальных видов и областей человеческой деятельности, а автоматизация комплексных процессов человеческой деятельности, требующая *интеграции* в априори непредсказуемых комбинациях самых различных информационных ресурсов и самых различных автоматизированных сервисов, реализуемых в виде специализированных интеллектуальных компьютерных систем.

Локальность автоматизации человеческой деятельности приводит к тому, что вся человеческая деятельность приобретает облик "архипелага", состоящего из хорошо автоматизированных "островов", но соединяемых между собой "вручную". Это "ручное" не автоматизируемое соединение указанных "островов" полностью зависит от человеческого фактора и квалификации соответствующих исполнителей.

Указанное "ручное" соединение некоторого множества семантически близких автоматизированных областей человеческой деятельности можно автоматизировать, но делать это надо очень грамотно на высоком уровне системной культуры и на фундаментальной основе общей теории человеческой деятельности.

Еще одна важная причина, препятствующая дальнейшему повышению общего уровня автоматизации общества заключается в том, что автоматизация различных областей человеческой деятельности осуществляется без выявления и глубокого анализа сходства некоторых видов деятельности в разных областях и соответственно без сближения, *конвергенции и унификации* этих видов деятельности.

Важнейшим направлением повышения уровня автоматизации человеческой деятельности является переход к автоматизации все более и более комплексных (крупных) видов и областей человеческой деятельности например, от автоматизации деятельности различных предприятий, организаций, хозяйственных служб к автоматизации деятельности города в целом).

Автоматизации комплексных видов человеческой деятельности требует создания комплекса активно взаимодействующих компьютерных систем, каждая из которых обеспечивает автоматизацию соответствующего частного вида человеческой деятельности, входящего в состав автоматизируемого комплексного вида деятельности. При этом число уровней иерархии автоматизируемых видов человеческой деятельности ничем не ограничивается. Очевидно, что уровень автоматизации комплексных видов человеческой деятельности определяется:

- уровнем конвергенции (сближения, совместимости) соответствующих частных видов деятельности;
- качеством интеграции этих частных видов деятельности;
- уровнем конвергенции компьютерных систем, обеспечивающих автоматизацию указанных частных видов деятельности;
- качеством взаимодействия этих компьютерных систем т.е. уровнем интероперабельности этих систем).

[Уровень эволюции общества во многом зависит от уровня автоматизации человеческой деятельности, от уровня развития соответствующих технологий такой автоматизации. Но эта зависимость выглядит значительно сложнее чем, кажется на первый взгляд, особенно, если речь идет об автоматизации не физической, интеллектуальной человеческой деятельности (как индивидуальной, так и коллективной). Безграмотная, а тем более социально безответственная или злонамеренная автоматизация информационной деятельности общества способны нанести огромный ущерб его развитию. Такая безграмотность и безответственность, например, приводит к таким побочным факторам, как компьютерная зависимость, виртуализация окружающей среды, поверхностный характер мышления, снижение познавательной мотивации и активности и многое другое.]

⇒ *следовательно**:

- [необходимо существенно повысить уровень социальной ответственности у разработчиков компьютерных систем и соответствующих технологий.]
- [Опасность от безграмотного, социально безответственного и тем более злонамеренного внедрения интеллектуальных компьютерных систем нового поколения может иметь для человечества летальный характер.]

человеческое общество

⇒ *направления развития**:

[Если рассматривать *общество как многоагентную систему*, состоящую из самостоятельных интеллектуальных агентов, то, очевидно, что важнейшими факторами, определяющими повышение качества (уровня развития) *общества* являются:

- повышение эффективности использования опыта, накопленного *обществом*, эффективности использования человечеством знаний и навыков
- повышение темпов приобретения, накопления и систематизации эффективно используемых человечеством знаний и навыков.

Решение указанных проблем становится вполне возможным, если для этого использовать *интеллектуальные компьютерные системы нового поколения*, с помощью которых накапливаемые человечеством знания и навыки будут организованы как систематизированная распределенная библиотека многократно используемых информационных ресурсов (знаний и навыков).]

⇒ следовательно*:

[Систематизация и автоматизация многократного использования накапливаемых человечеством информационных ресурсов требует их конвергенции, глубокой интеграции и формализации. Особое место в этом процессе занимает математика, как основа систематизации и формализации знаний и навыков на уровне формальных онтологий верхнего уровня.]

Заключение к Главе 7.1.

Благодаря тому, что *интеллектуальные компьютерные системы нового поколения* становятся самостоятельными и активными субъектами *человеческой деятельности* в достаточной степени равноправными людям (естественным индивидуальным субъектам человеческой деятельности), характер и, соответственно, уровень автоматизации *человеческой деятельности* существенно меняется – снимается необходимость управлять средствами автоматизации, поскольку такое "ручное" управление заменяется распределением обязанностей и ответственности между людьми и *интеллектуальными компьютерными системами нового поколения*.

Если автоматизация любого вида в любой области *человеческой деятельности* будет осуществляться с помощью *интеллектуальных компьютерных систем нового поколения* и если *интеллектуальные компьютерные системы нового поколения*, обеспечивающие автоматизацию разных видов и областей *человеческой деятельности*, будут содержательно взаимодействовать между собой, то общий уровень автоматизации *человеческой деятельности* существенно возрастет благодаря тому, что отпадет необходимость вручную координировать использование различных средств автоматизации.

Эффективность и трудоемкость автоматизации различных видов и областей *человеческой деятельности* будет существенно определяться степенью **конвергенции** между различными видами и областями *человеческой деятельности*. Необходимо построить иерархическую модель *человеческой деятельности*, в рамках которой должна быть проведена грамотная систематизация и стратификация всех видов и областей *человеческой деятельности*, направленная против излишнего эклектического многообразия. Таким образом, прежде, чем осуществлять комплексную автоматизацию *человеческой деятельности* с помощью *интеллектуальных компьютерных систем нового поколения*, необходимо с позиций общей теории систем переосмыслить организацию этой деятельности. В противном случае автоматизация беспорядка приведет к еще большему беспорядку.

Особо подчеркнем то, что многие из рассмотренных нами проблем текущего состояния и направлений дальнейшего развития *Человеческой деятельности в области Искусственного интеллекта* аналогичны проблемам и тенденциям развития многих других научно-технических дисциплин. Следовательно, подходы к решению этих проблем могут носить междисциплинарный характер.

Время каждого человека является главным невосполнимым ресурсом общества и тратить его надо не на рутинную поддержку жизненного цикла всевозможных социально значимых объектов, а на комплексное развитие соответствующих *технологий*. Автоматизация *человеческой деятельности* с помощью глобальной системы интероперабельных семантически совместимых и активно взаимодействующих *интеллектуальных компьютерных систем* в самых разных областях *человеческой деятельности* позволит существенно сократить время каждого человека на выполнение рутинной, легко автоматизируемой деятельности. Человеческая деятельность должна стать ориентированной на максимально возможную самореализацию, раскрытие творческого потенциала каждого человека, направленного на ускорение темпов повышения уровня интеллекта всего общества.

Создание *Глобальной экосистемы интеллектуальных компьютерных систем нового поколения*, предполагает:

- Построение формальной модели *человеческой деятельности*;
- Переход от эклектичного построения сложных *интеллектуальных компьютерных систем*, использующих различные виды знаний и различные виды моделей решения задач, к их глубокой интеграции и унифика-

ции, когда одинаковые модели представления и модели обработки знаний реализуется в разных системах и подсистемах одинаково;

- Сокращение дистанции между современным уровнем *теории интеллектуальных компьютерных систем* и практики их разработки;
- Разработку грамотной тактики и стратегии переходного периода, в рамках которого современные *интеллектуальные компьютерные системы* должны постепенно заменяться на *интеллектуальные компьютерные системы нового поколения*, которые должны эффективно взаимодействовать не только между собой, но и с хорошо зарекомендовавшими себя современными информационными ресурсами и сервисами.

Глава 7.2.

Метасистема OSTIS

⇒ *авторы**:

- Голенков В.В.
- Шункевич Д.В.
- Банцевич К.А.
- Загорский А.Г

⇒ *аннотация**:

[Данная глава посвящена рассмотрению подхода к автоматизации процессов создания, развития и применения стандартов на основе Технологии OSTIS. Также в главе сформулированы основные принципы стандартизации интеллектуальных компьютерных систем, методов и средств их проектирования в рамках предлагаемого подхода.]

⇒ *подраздел**:

- § 7.2.1. Структура, назначение, особенности и достоинства Метасистемы OSTIS
- § 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS

⇒ *ключевое понятие**:

- Метасистема OSTIS
- Стандарт Технологии OSTIS

⇒ *библиографическая ссылка**:

- ...

Введение в Главу 7.2.

В основе каждой развитой сферы человеческой деятельности лежит ряд стандартов, формально описывающих различные ее аспекты – систему понятий (включая терминологию), типологию и последовательность действий, выполняемых в процессе применения соответствующих методов и средств.

Стандарты в самых различных областях являются важнейшим видом знаний, главной целью которых является обеспечение совместимости различных видов деятельности. Несмотря на развитие информационных технологий, в настоящее время подавляющее большинство стандартов представлено либо в виде традиционных линейных документов, либо в виде web-ресурсов содержащих набор статических страниц, связанных гиперссылками. Для того чтобы стандарты выполняли свою главную функцию, они должны постоянно совершенствоваться.

Текущее оформление стандартов имеет ряд недостатков, которые мешают эффективному и грамотному использованию стандартов в различных областях:

- дублирование информации в рамках документа, описывающего стандарт;
- трудоемкость сопровождения самого стандарта, обусловленная в том числе дублированием информации, в частности, трудоемкость изменения терминологии;
- проблема интернационализации стандарта – фактически перевод стандарта на несколько языков приводит к необходимости поддержки и согласования независимых версий стандарта на разных языках;
- неудобство применения стандарта, в частности, трудоемкость поиска необходимой информации. Как следствие – трудоемкость изучения стандарта;
- несогласованность формы различных стандартов между собой, как следствие – трудоемкость автоматизации процессов развития и применения стандартов;
- трудоемкость автоматизации проверки соответствия объектов или процессов требованиям того или иного стандарта;
- и другие.

Перечисленные проблемы связаны в основном с формой представления стандартов.

Задачей любого стандарта в общем случае является описание согласованной системы понятий (и соответствующих терминов), бизнес-процессов, правил и других закономерностей, способов решения определенных классов задач и т.д. Для формального описания информации такого рода с успехом применяются онтологии. Более того,

в настоящее время в ряде областей вместо разработки стандарта в виде традиционного документа разрабатывается соответствующая онтология. Такой подход дает очевидные преимущества в плане автоматизации процессов согласования и использования стандартов.

Однако, актуальной остается проблема, связанная не с формой, а с сутью (семантикой) стандартов – проблема несогласованности системы понятий и терминов между различными стандартами, которая актуальна даже для стандартов в рамках одной и той же сферы деятельности.

В настоящее время *Информатика* преодолевает важнейший этап своего развития – переход от информатики данных (data science) к информатике знаний (knowledge science), где акцентируется внимание на семантических аспектах представления и обработки знаний.

Без фундаментального анализа такого перехода невозможно решить многие проблемы, связанные с управлением знаниями, экономикой знаний, с семантической совместимостью интеллектуальных компьютерных систем.

С семантической точки зрения каждый стандарт есть иерархическая онтология, уточняющих структуру и систем понятий соответствующих им предметных областей, которая описывает структуру и функционирование либо некоторого класса технических или иных искусственных систем, либо некоторого класса организаций, либо некоторого вида деятельности.

Наиболее перспективным подходом к решению перечисленных проблем является преобразование каждого конкретного стандарта в базу знаний, в основе которой лежит набор онтологий, соответствующих данному стандарту. Такой подход позволяет в значительной мере автоматизировать процессы развития стандарта и его применения.

В рамках *Технологии OSTIS* данный подход используется при построении *Стандарта OSTIS*.

Предлагаемый *Стандарт OSTIS* оформлен в виде семейства разделов базы знаний специальной интеллектуальной компьютерной *Метасистемы OSTIS* (Intelligent MetaSystem for ostis-systems) (см. [MetacOSTIS-2022ЭЛ](#)), которая построена по *Технологии OSTIS* и представляет собой постоянно совершенствуемый интеллектуальный портал научно-технических знаний, который поддерживает перманентную эволюцию *Стандарта OSTIS*, а также разработку различных *ostis*-систем (интеллектуальных компьютерных систем, построенных по *Технологии OSTIS*).

§ 7.2.1. Структура, назначение, особенности и достоинства Метасистемы OSTIS

Метасистема OSTIS – интеллектуальная компьютерная система, обеспечивающая

- комплексную информационную поддержку всех этапов жизненного цикла интеллектуальных компьютеров нового поколения;
- автоматизацию проектирования всех компонентов интеллектуальных компьютерных систем нового поколения;
- комплексную автоматизацию всех этапов жизненного цикла интеллектуальных компьютерных систем нового поколения.

Формой реализации представленной *Метасистемы OSTIS* является *Технология OSTIS*.

Пункт 7.2.1.1. Структура Метасистемы OSTIS

Пункт 7.2.1.2. Назначение Метасистемы OSTIS

Описываемая *Метасистема OSTIS* является:

- системой информационной и инструментальной поддержки всех этапов жизненного цикла и.к.с. нового поколения (*ostis*-систем) самого различного назначения;
- порталом знаний по *Технологии OSTIS*, обеспечивающим:
 - координацию работ по развитию *Технологии OSTIS*;
 - автоматизацию анализа качества *Стандарта OSTIS*.

Т.е. *Метасистема OSTIS* является системой управления Проектом создания и развития *Стандарта OSTIS*.

Важнейшим направлением *Метасистемы OSTIS* и, соответственно, важнейшим направлением применения *Стандарта OSTIS* является использование их в качестве комплексного интегрированного компьютерного учебного пособия по специальности "Искусственный интеллект". Для этого устанавливается связь между разделами *Стандарта OSTIS* и программами различных учебных дисциплин указанной специальности. Важно подчеркнуть при этом: *Стандарт OSTIS* содержит достаточно полный сравнительный анализ с различными альтернативными подходами, т.е. ни в коем случае не ограничивается рассмотрением только Технологией OSTIS.

Пункт 7.2.1.3. Особенности Метасистемы OSTIS

Пункт 7.2.1.4. Достоинства Метасистемы OSTIS

§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS

Стандарт Технологии OSTIS представляет собой Документацию *Технологии OSTIS*, которая представлена в виде основной части базы знаний специальной интеллектуальной компьютерной системы, предназначеннной для комплексной поддержки жизненного цикла семантически совместимых интеллектуальных компьютерных систем нового поколения (*Метасистемы OSTIS*).

Стандарт OSTIS

- := [Документация *Технологии OSTIS*]
- := [Документация Открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем]
- := [Описание *Технологии OSTIS* (Open Semantic Technology for Intelligent Systems), представленная в виде семейства разделов базы знаний специальной *ostis*-системы (системы, построенной по *Технологии OSTIS*) на внутреннем языке *ostis*-систем и обладающее достаточной полнотой для использования этой *технологии разработчиками интеллектуальных компьютерных систем*]
- := [Полное описание текущего состояния *Технологии OSTIS*, представленное в виде семейства разделов базы знаний, построенной по *Технологии OSTIS*]
- := [Семейство разделов базы знаний *Метасистемы OSTIS*, которое предназначено для комплексной поддержки онтологического проектирования семантически совместимых гибридных интеллектуальных компьютерных систем]
- ∈ семейство разделов базы знаний
 - := [семейство разделов внутреннего представления базы знаний *ostis*-системы – интеллектуальной компьютерной системы, построенной по *Технологии OSTIS*]
 - := [Достаточно полная формальная Документация текущей версии *Технологии OSTIS*, представленная либо в виде основной части базы знаний *Метасистемы OSTIS*, либо в виде внешнего формального представления этой базы знаний]
 - := [Основная часть базы знаний *Метасистемы OSTIS*, описывающая текущую версию *Технологии OSTIS*]
 - := [Формальный текст, объектом описания которого является *Технология OSTIS*, т.е. текст, являющий достаточно полным описанием текущего состояния *Технологии OSTIS*]
 - := [Документация *Технологии OSTIS*, полностью отражающая текущее состояние *Технологии OSTIS* и представленная соответствующим семейством разделов базы знаний специальной *ostis*-системы, которая ориентирована на поддержку проектирования, производства, эксплуатации и эволюции (реинжиринга) *ostis*-систем, а также на поддержку эволюции самой *Технологии OSTIS* и которая названа нами *Метасистемой OSTIS*]
 - := [Семейство разделов, в состав которого входят все разделы *Стандарта OSTIS*]
- ⇒ основной sc-идентификатор*:
 - [Стандарт OSTIS]
 - ⇐ сокращение*:
 - [Стандарт *Технологии OSTIS*]
 - ⇐ сокращение*:
 - [Стандарт Открытой технологии комплексной поддержки жизненного цикла семантически совместимых интеллектуальных компьютерных систем нового поколения]

Следует подчеркнуть, что *Стандарт OSTIS* – это не описание некоторого состояния *Технологии OSTIS*, а динамическая информационная модель процесса эволюции этой *технологии*.

В рамках *Стандарта OSTIS* вводится оглавление, система ключевых знаков. Также строго регламентированы:

- требования, предъявляемые к *Стандарту OSTIS*;
- правила построения *Стандарта OSTIS*;
- направления развития *Стандарта OSTIS*.

Это позволяет воспринимать пользователю *Стандарт OSTIS*, как целостный понятный текст. Также избежать возможных противоречий.

Пункт 7.2.2.1. Оглавление Стандарта OSTIS

Одним из составляющих *Стандарта OSTIS* является *Оглавление Стандарта OSTIS*.

Оглавление Стандарта OSTIS

:= пояснение*:

[Иерархический перечень разделов, входящих в состав *Стандарта OSTIS*, с дополнительной спецификацией некоторых разделов]

⇒ примечание*:

[Существенно подчеркнуть, что иерархия разделов *Стандарта OSTIS* не означает то, что *разделы* более низкого уровня иерархии входят в состав (являются частями) соответствующих разделов более высокого уровня. Связь между *разделами* разных уровней иерархии означает то, что *раздел* более низкого уровня иерархии является *дочерним* разделом по отношению к соответствующему *разделу* более высокого уровня, т.е. *разделом*, который наследует свойства указанного *раздела* более высокого уровня.

В отличие от этого каждая *часть Стандарта OSTIS*, а также сам *Стандарт OSTIS* является *семейством разделов* (совокупность разделов), входящих в ее состав.]

⇒ примечание*:

[Описание логико-семантических связей каждого раздела *Стандарта OSTIS* с другими разделами *Стандарта OSTIS* приводится в рамках *титульной спецификации* каждого раздела.]

Пункт 7.2.2.2. Общая структура Стандарта OSTIS

Основной текст *Стандарта OSTIS* состоит из следующих частей:

Стандарт OSTIS

⇒ декомпозиция*:

Структура Стандарта OSTIS верхнего уровня

= ⟨• *Часть 1 Стандарта OSTIS*.

Введение в интеллектуальные компьютерные системы нового поколения

:= [Анализ текущего состояния технологий Искусственного интеллекта и постановка задачи на создание комплекса совместимых технологий искусственного интеллекта, обеспечивающего поддержку всего жизненного цикла интеллектуальных компьютерных систем нового поколения и названного нами Технологии OSTIS]

• *Собственно документация Технологии OSTIS*

⇒ **декомпозиция*:**

⟨• *Стандарт ostis-систем*

:= [Стандарт интеллектуальных компьютерных систем нового поколения, построенных по Технологии OSTIS]

:= [Формальная теория ostis-систем]

:= [Формальные структурно-функциональные и логико-семантические модели ostis-систем]

⇒ **декомпозиция*:**

⟨• *Часть 2 Стандарта OSTIS*.

Смысловое представление и онтологическая систематизация знаний в и.к.с.

нового поколения.

:= [Стандарт представления информации в ostis-системах]

:= [Модели представления знаний и баз знаний в ostis-системах]

• *Часть 3 Стандарта OSTIS.*

Многоагентные решатели задач и.к.с. нового поколения

:= [Стандарт процессов и методов обработки информации в ostis-системах]

:= [Модели обработки знаний в ostis-системах (логические, продукционные, функциональные, нейросетевые, процедурные и непроцедурные, четкие и нечеткие)]

• *Часть 4 Стандарта OSTIS.*

Онтологические модели интерфейсов и.к.с. нового поколения

:= [Стандарт информационных ресурсов и моделей решения интерфейсных задач в ostis-системах]

}

• *Стандарт методов и средств поддержки жизненного цикла ostis-систем*

:= [Стандарт бизнес-процессов и методик, автоматически реализуемых процессов и методов, информационных средств и инструментальных средств, используемых для поддержки жизненного цикла ostis-систем]

⇒ **декомпозиция*:**

⟨• *Часть 5 Стандарта OSTIS.*

Методы и средства проектирования и.к.с. нового поколения
 := [Методики, методы и средства проектирования баз знаний, решателей задач и интерфейсов ostis-систем]

- *Часть 6 Стандарта OSTIS.*

Платформы реализации и.к.с. нового поколения
 := [Методы и средства реализации ostis-систем (на основе программных платформ и специально созданных для этого компьютеров)]

- *Часть 7 Стандарта OSTIS.*

Методы и средства реинжиниринга и эксплуатации и.к.с. нового поколения
 := [Методы и средства эксплуатации ostis-систем конечными пользователями, а также их сопровождения (поддержки работоспособности) и реинжиниринга (обновления, модернизации)]

}

}

- *Часть 8 Стандарта OSTIS.*

Экосистема и.к.с. нового поколения и их пользователей
 := [Описание продуктов, создаваемых с помощью Технологии OSTIS, основными их которых является Экосистема OSTIS, семантически совместимых и активно взаимодействующих ostis-систем и их пользователей]

- *Библиография OSTIS*
 := [Спецификация библиографических источников, семантически близких Технологии OSTIS, в контексте их сравнительного анализа со Стандартом OSTIS]

}

Пункт 7.2.2.3. Ключевые знаки Стандарта OSTIS

Система ключевых знаков Стандарта OSTIS упорядочена в точном соответствии с Оглавлением Стандарта OSTIS и является уточнением указанного Оглавления путем перечисления и пояснения ключевых сущностей, описываемых в разделах Стандарта, и, в первую очередь тех сущностей, которые указываются в идентификаторах (названиях) разделов Стандарта OSTIS.

Система ключевых знаков Стандарта OSTIS является целостным дополнением к Оглавлению Стандарта OSTIS, поскольку:

- иерархия и последовательность ключевых знаков четко соответствуют иерархии и последовательности разделов стандарта;
- система ключевых знаков Стандарта OSTIS, как и его Оглавление, воспринимается (читаться) как целостный понятный текст.

Пункт 7.2.2.4. Назначение Стандарта OSTIS

Поскольку Стандарт OSTIS является неотъемлемой частью Метасистемы OSTIS (основной частью ее базы знаний), основным назначением Стандарта OSTIS является обеспечение максимально эффективной реализации того, для чего предназначена Метасистема OSTIS.

Стандарт OSTIS рассматривается как результат конвергенции и интеграции всевозможных направлений Искусственного интеллекта, что позволяет студентам и магистрантам сформировать целостное представление о тематике Искусственного интеллекта, а не мозаичное представление в виде множества дисциплин (направлений), связи между которыми подробно и тем более формально не рассматриваются.

Стандарт OSTIS перманентно и достаточно быстро эволюционирует. За время обучения студентов и магистрантов происходит весьма существенные изменения текущей версии Стандарта OSTIS.

Студенты и магистранты активно вовлекаются в процесс эволюции Стандарта OSTIS, это обеспечивает:

- формирование необходимого уровня их квалификации в условиях быстрого морального старения того, чему их уже научили;
- формирование необходимых навыков, позволяющих им в процессе реальной профессиональной деятельности быстро адаптироваться к новым условиям этой деятельности и, в частности, к новым версиям соответствующих технологий.

Пункт 7.2.2.5. Аналоги Стандарта OSTIS

Аналогами *Стандарта OSTIS* можно считать:

- любую серьезную попытку систематизации результатов, полученных в области Искусственного интеллекта к текущему моменту:
 - учебник, достаточно полно отражающий текущее состояние *Искусственного интеллекта*;
 - справочник, содержащий достаточно полную информацию о текущем состоянии *Искусственного интеллекта*.
- любую попытку перехода от частных формальных моделей различных компонентов и.к.с. общей (объединенной, интегрированной) формальной модели и.к.с в целом – к общей теории и.к.с.
- любую унификацию технических решений, устранения многообразия форм технических решений при разработке и.к.с.
- первые попытки разработки стандартов *интеллектуальных компьютерных систем*, а также *технологий Искусственного интеллекта*, которые, чаще, всего, ограничиваются построением систем соответствующих понятий.

Пункт 7.2.2.6. Особенности Стандарта OSTIS

Стандарт OSTIS – это не просто систематизация современного состояния результатов в области *Искусственного интеллекта*, это систематизация, представленная в виде общей комплексной формальной модели *интеллектуальных компьютерных систем* и комплексной формальной модели поддержки их жизненного цикла. Более того, текст *Стандарта OSTIS* представляет собой основную часть базы знаний специальной *интеллектуальной метасистемы*, которая ориентирована:

- на поддержку разработки *интеллектуальных компьютерных систем* различного назначения;
- на поддержку эволюции *Стандарта OSTIS*;
- на поддержку подготовки специалистов в области *Искусственного интеллекта*.

Стандарт OSTIS – это динамический текст, перманентно отражающий новые научно-технические результаты, получаемые в области *Искусственного интеллекта* в рамках *Общей теории интеллектуальных компьютерных систем* и *Общей комплексной технологии разработки интеллектуальных компьютерных систем*. Здесь важной является оперативность фиксации новых научно-технических результатов, т.е. минимизация отрезка времени между моментом получения новых результатов и моментом интеграции описания этих результатов в состав *Стандарта OSTIS*. В перспективе авторы новых научно-технических результатов в области *Искусственного интеллекта* будут заинтересованы лично публиковать (интегрировать) свои результаты в состав *Стандарта OSTIS*, т.е. становиться соавторами *Стандарта OSTIS*, чтобы обеспечить необходимую оперативность такой публикации и отсутствие искажений своих результатов. Динамичность *Стандарта OSTIS* и достаточная оперативность интеграции в его состав новых научно-технических результатов в области *Искусственного интеллекта* делает *Стандарт OSTIS* всегда актуальным и никогда морально устаревшим.

В рамках *Стандарта OSTIS* нет противопоставления между научно-технической информацией, добываемой в области *Искусственного интеллекта*, и учебно-методической информацией, используемой для подготовки и самоподготовки специалистов в области Искусственного интеллекта. Информация о том, чему учить, должна быть "переплетена", интегрирована с информацией о том, как учить.

Важно заметить, что *Стандарт OSTIS* в отличие от остальных стандартов является структурированным формальным текстом, который может быть непосредственно использован не только разработчиками *интеллектуальных компьютерных систем*, но также и интеллектуальными компьютерными системами, осуществляющими автоматизацию проектирования разрабатываемых *интеллектуальных компьютерных систем* и поддержку последующих этапов их жизненного цикла. Таким образом, разработка *Стандарта OSTIS* является неотъемлемой частью разработки комплекса средств информационной и инструментальной поддержки всего жизненного цикла *интеллектуальных компьютерных систем*, а указанные средства поддержки жизненного цикла *интеллектуальных компьютерных систем* становятся равноправными партнерами в процессе создания, эксплуатации и сопровождения *интеллектуальных компьютерных систем* благодаря своему осознанию (пониманию) того, что такое *интеллектуальные компьютерные системы* и их жизненный цикл.

Содержательно *Стандарт OSTIS* охватывает не только описание моделей разрабатываемых *интеллектуальных компьютерных систем*, но также и описание методик, автоматизируемых методов и инструментальных средств поддержки (автоматизации) всех этапов жизненного цикла разрабатываемых интеллектуальных компьютерных систем.

Проект развития Стандарта OSTIS ориентирован на высокие темпы эволюции Стандарта OSTIS благодаря автоматизации управления этим проектом с помощью Метасистемы OSTIS, которая является полноправным участником этого проекта.

Построение и структуризация текста Стандарта OSTIS ориентированы на максимально возможное снижение языкового и когнитивного барьера для начинающих его пользователей. Для этой цели используются (1) различного рода естественно-языковые примечания и комментарии, имеющие соответствующие семантические связи с поясняемыми сущностями, а также (2) различного рода дидактические знания, указывающие на различные аналогии, различия, примеры, принципы, лежащие в основе описываемых сущностей и т.п.

Пункт 7.2.2.7. Пользователь Стандарта OSTIS

Рассмотрим целевую аудиторию Стандарта OSTIS.

Стандарт OSTIS

⇒ класс пользователей*:

пользователь Стандарта OSTIS

:= [целевая аудитория Стандарта OSTIS]

⇒ разбиение*:

{• разработчик ostis-системы

 ▷ разработчик Метасистемы OSTIS

 ⇒ разбиение*:

 {• разработчик Стандарта OSTIS

 • разработчик решателя задач Метасистемы OSTIS

 • разработчик пользовательского интерфейса Метасистемы OSTIS

 }

 ▷ разработчик базы знаний ostis-системы

 ▷ разработчик Стандарта OSTIS

 ▷ разработчик решателя задач ostis-системы

 ▷ разработчик решателя задач Метасистемы OSTIS

 ▷ разработчик интерфейса ostis-системы

 ▷ разработчик пользовательского интерфейса Метасистемы OSTIS

 ▷ разработчик платформ реализации ostis-систем

 • потенциальный разработчик ostis-системы

 • специалист в области Искусственного интеллекта, желающий интегрировать свои результаты в состав общей теории и.к.с. нового поколения и соответствующей комплексной технологии

 • студент или магистрант специальности "Искусственный интеллект" либо другой смежной специальности, желающий приобрести практический опыт в разработке прикладных и.к.с. нового поколения или в разработке соответствующей комплексной технологии

}

Пункт 7.2.2.8. Авторский коллектив Стандарта OSTIS

Для обеспечения перманентной эволюции существует ряд требований, ориентированный на авторов Стандарта OSTIS.

Авторы Стандарта OSTIS должны:

- Отслеживать и изучать новые публикации по тематике, рассматриваемой в Стандарте OSTIS. Близкими источниками для этого являются:
 - выпуски журналов;
 - материалы конференций:
 - организуемых Консорциумом ЗWC;
 - по интеграции различных направлений ИИ;
 - стандарты в области ИИ;
 - публикации, рассматривающие:
 - формальные онтологии;
 - онтологии верхнего уровня;
 - семантические сети;

- графы знаний;
 - графовые базы данных и графовые СУБД;
 - смысловое представление знаний;
 - конвергенцию различных направлений ИИ.
- Фиксировать результаты изучения новых публикаций по тематике, близкой Стандарту OSTIS, в Библиографии OSTIS, а также в основном тексте Стандарта OSTIS в виде соответствующих ссылок, цитат, сравнительного анализа.
 - Отслеживать текущее состояние всего текста Стандарта OSTIS, формировать предложения, направленные на развитие Стандарта OSTIS и на повышение темпов этого развития. Активно участвовать в обсуждении проблем развития Технологии OSTIS.
 - Максимально возможным образом увязывать персональную работу над Стандартом OSTIS с другими формами деятельности – научной, учебной, прикладной.
 - Указывать авторство своих предложений по дополнению и/или корректировке текущего текста Стандарта OSTIS.
 - Участвовать в рецензировании и согласовании предложений, представленных другими авторами Стандарта OSTIS.

Большой объем работ по созданию и развитию *Стандарта OSTIS* и, соответственно, *Технологии OSTIS*, комплексный характер этих работ, в которых необходима глубокая *конвергенции и интеграции* различных направлений *Искусственного интеллекта* предъявляют к *Авторскому коллективу Стандарта OSTIS* высокие требования по уровню мотивации, по уровню качества творческой атмосферы, по уровню *интероперабельности* всех членов коллектива, т.е. по уровню способности быстро и качественно согласовывать персональные точки зрения.

Поскольку Проект создания и развития Стандарта OSTIS является открытым, Членом Авторского коллектива Стандарта OSTIS может стать любой желающий, соблюдающий Правила организации взаимодействия членов Авторского коллектива Стандарта OSTIS, разделяющий цели и задачи разработки такого Стандарта.

Выделяются следующие ключевые пункты *Правил организации взаимодействия членов Авторского коллектива Стандарта OSTIS*:

- Коллегиально формировать тактические и стратегические направления развития Стандарта OSTIS и, соответственно, Технологии OSTIS;
- Коллегиально распределять задачи по реализации утвержденных направлений развития Стандарта OSTIS с учетом (1) научных интересов, квалификации и возможности каждого члена Авторского коллектива, (2) приоритета задач и при достаточно полном охвате всех приоритетных задач.

Пункт 7.2.2.9. Редакционная коллегия Стандарта OSTIS

В рамках *Авторского коллектива Стандарта OSTIS* выделяется также *Редакционная коллегия Стандарта OSTIS*.

Редакционная коллегия Стандарта OSTIS – часть *Авторского коллектива Стандарта OSTIS*, являющаяся центром коллегиального принятия решений по основным направлениям развития Стандарта OSTIS и, соответственно, Технологии OSTIS, по уточнению соответствующих приоритетов и сроков. *Редакционная коллегия Стандарта OSTIS* также несет ответственность за формирование и реализацию стратегических направлений развития Стандарта OSTIS и, в частности, за подбор и назначение *ответственных исполнителей разделов Стандарта OSTIS*.

Основными направлениями деятельности *Редакционной коллегии Стандарта OSTIS* являются:

- Обеспечение целостности и повышения качества постоянно развивающейся (совершенствуемой) Технологии OSTIS, а также достаточно точное описание (документирование) каждой текущей версии этой технологии.
- Обеспечение чёткого контроля совместимости версий Технологии OSTIS в целом, а также версий различных компонентов этой технологии.
- Постоянное уточнение степени важности различных направлений развития Технологии OSTIS для каждого текущего момента.
- Формирование и постоянное уточнение плана тактического и стратегического развития самой Технологии OSTIS, а также полной документации этой Технологии в виде *Стандарта OSTIS*. Подчеркнем при этом, что указанная документация является неотъемлемой частью *Технологии OSTIS*.

Стандарт OSTIS

⇒ *редакционная коллегия**:

Редакционная коллегия Стандарта OSTIS

:= [Редколлегия Стандарта OSTIS]

:=

[Рабочий орган, обеспечивающий организацию коллективного творческого процесса по развитию *Стандарта OSTIS*, совмещенного с учебно-методическим обеспечением подготовки соответствующих специалистов.]

:= [Редакционная коллегия, несущая ответственность за качество перманентно эволюционируемого *Стандарта OSTIS*.]

⇒ *обязанности**:

- [Обеспечение развития *Стандарта OSTIS*, совмещенного (интегрированного) с комплексным учебно-методическим обеспечением подготовки специалистов в области Искусственного интеллекта]

- [Обеспечение корректности (непротиворечивости), системности, целостности, полноты всех разрабатываемых материалов *Стандарта OSTIS* и, в том числе, учебно-методического обеспечения подготовки специалистов в области Искусственного интеллекта.]

- [Кординация деятельности авторов разработки очередной (следующей) версии *Стандарта OSTIS*.]

- [В частности, Редколлегия *Стандарта OSTIS* может осуществлять распределение работ по построению следующей версии *Стандарта OSTIS* с четкой привязкой ответственных лиц *Стандарта OSTIS* к соответствующим разделам *Стандарта OSTIS*.]

⇒ *примечание**:

[Очень важная структура, определяющая научно-технический уровень, авторитет и репутацию всей Технологии *OSTIS* в глазах международной научно-технической общественности.]

⇒ *примечание**:

[Каждый член Редакционной коллегии *Стандарта OSTIS* должен быть активным членом Авторского коллектива *Стандарта OSTIS*.]

Пункт 7.2.2.10. Требования, предъявляемые к Стандарту OSTIS

Стандарт OSTIS должен соответствовать следующим требованиям:

- вводимые понятия (в т.ч. дидактические отношения) должны быть четко пояснены и/или определены в соответствующем разделе Стандарта *OSTIS*;
- доступность понимания текста *Стандарта OSTIS* для читателей;
- обеспечение возможности поэтапной формализации информации, начиная от ея-текстов, которые могут быть в последствии записаны на формальном языке;
- независимость от естественных языков (только на уровне внешних идентификаторов (имен) sc-элементов);
- четкая логико-семантическая спецификация каждой предметной области, рассматриваемой в Стандарте *OSTIS*. Названная спецификация должна отражать как внутреннюю структуру предметной области (роли ее ключевых элементов), так и связи с другими предметными областями;
- конвергенция, ("бесшовная") интеграция различных видов знаний, описывающих самые различные сущности, к числу которых, в частности относятся и сами знания всевозможного вида;
- целостность, полнота, связность:
 - отсутствие информационных дыр;
 - достаточно полная спецификация всех сущностей;
 - согласованность основных идентификаторов (терминов), отсутствие синонимов и омонимов.
- отсутствие информационных излишеств и информационного мусора;
- четкая семантическая стратификация – каждый фрагмент базы знаний должен иметь свою семантическую "полочку" (никакого дублирования);
- строгая логическая последовательность текста (все используемые сущности должны быть введены либо в заданной предметной области, либо в предметной области более высокого уровня);
- унификация стилистики – текст не должен вызывать трудностей для его понимания;
- богатая библиография и сравнительный анализ;
- четкое соблюдение и совершенствование правил идентификации и спецификации описываемых сущностей;
- достаточно подробная спецификация каждого вводимого понятия в соответствующей предметной области.

Пункт 7.2.2.11. Правила построения Стандарта OSTIS

В рамках разработки *Стандарта OSTIS* выделяются *Общие правила построения Стандарта OSTIS* и *Частные правила построения Стандарта OSTIS*.

Общие правила построения Стандарта OSTIS

:= [принципы, лежащие в основе структуризации и оформления Стандарта OSTIS]

Рассмотрим основные положения:

- Основной формой представления *Стандарта OSTIS* как полной документации текущего состояния *Технологии OSTIS* является *внутреннее представление* основной части базы знаний специальной интеллектуальной компьютерной *Метасистемы OSTIS*, обеспечивающей использование и эволюцию (перманентное совершенствование) *Технологии OSTIS*. Такое представление *Стандарта OSTIS* обеспечивает эффективную семантическую навигацию по содержанию *Стандарта OSTIS* и возможность задавать *Метасистеме OSTIS* широкий спектр нетривиальных вопросов о самых различных деталях и тонкостях *Технологии OSTIS*;
- Кроме представления *Стандарта OSTIS* на внутреннем языке *представления знаний* используется также внешняя форма представления *Стандарта OSTIS* на *внешнем языке представления знаний*. При этом указанное внешнее представление *Стандарта OSTIS* должно быть структурировано и оформлено так, чтобы читатель мог достаточно легко "вручную" найти в этом тексте практически любую интересующую его *информацию*. В качестве *формального языка* внешнего представления *Стандарта OSTIS* используется *SCn-код*;
- *Стандарт OSTIS* имеет онтологическую структуризацию, т.е. представляет собой иерархическую систему связанных между собой *формальных предметных областей* и соответствующих им *формальных онтологий*. Благодаря этому обеспечивается высокий уровень стратифицированности *Стандарта OSTIS*;
- Каждому *понятию*, используемому в *Стандарте OSTIS*, соответствует свое место в рамках этого Стандарта, своя *предметная область* и соответствующая ей *онтология*, где это *понятие* подробно рассматривается (исследуется), где концентрируется вся основная информация об этом *понятии*, о различных его свойствах.
- В состав *Стандарта OSTIS* входят также файлы информационных конструкций, не являющихся конструкциями *SC-кода* (в том числе и sc-текстов, принадлежащих различным естественным языкам). Такие файлы позволяют формально описывать в базе знаний синтаксис и семантику различных внешних языков, а также позволяют включать в состав базы знаний различного рода пояснения, примечания, адресуемые непосредственно пользователям и помогающие им в понимании формального текста базы знаний;
- С семантической точки зрения *Стандарт OSTIS* представляет собой иерархическую систему формальных моделей *предметных областей* и соответствующих им *формальных онтологий*;
- С семантической точки зрения *Стандарт OSTIS* представляет собой большую *рафинированную семантическую сеть*, которая, соответственно, имеет нелинейный характер и которая включает в себя знаки любых видов описываемых сущностей(материалных сущностей, абстрактных сущностей, понятий, связей, структур) и, соответственно этому, содержит связи между всеми этими видами сущностей(в частности, связи между связями, связи между структурами);
- *Стандарт OSTIS* представляет собой иерархическую систему *предметных областей* и соответствующих им *онтологий*, специфицирующих эти *предметные области*. Каждая из *предметных областей* описывает соответствующие *классы объектов исследования* с максимально возможной степенью детализации, определяемой набором *отношений и параметров*, заданных на *классах объектов исследования*. На множестве *предметных областей*, задано отношение *дочерняя предметная область**, которое указывает направление наследования свойств объектов исследования, рассматриваемых в разных *предметных областях*;
- Каждый раздел *Стандарта OSTIS* может содержать те *знания*, которые входят в состав той *предметной области и онтологии*, которая либо полностью представлена указанным разделом, либо представлена частично в виде спецификации одного или нескольких конкретных объектов исследования;
- недопустима синонимия и омонимия основных sc-идентификаторов в рамках каждого семейства;
- Спецификация каждой предметной области и каждого раздела должна иметь достаточную степень полноты. Как минимум, для каждой предметной области должна быть указана роль каждого используемого в ней понятия;
- В состав *Стандарта OSTIS* входят также файлы информационных конструкций, не являющихся конструкциями *SC-кода* (в том числе и sc-текстов, принадлежащих различным естественным языкам). Такие файлы позволяют формально описывать в базе знаний синтаксис и семантику различных внешних языков, а также позволяют включать в состав базы знаний различного рода пояснения, примечания, адресуемые непосредственно пользователям и помогающие им в понимании формального текста базы знаний;
- Непосредственно сам *Стандарт OSTIS* представляет собой внутреннее *смысловое представление* основной части базы знаний *Метасистемы OSTIS* на внутреннем смысловом языке *ostis-систем* (этот язык назван нами *SC-кодом - Semantic Computer Code*).

Кроме *Общих правил построения Стандарта OSTIS* в *Стандарте OSTIS* приводятся описания различных частных (специализированных) правил построения (оформления) различных видов фрагментов *Стандарта OSTIS*.

К таким видам фрагментов относятся следующие:

- *sc-идентификатор*
 - := [внешний идентификатор внутреннего знака (*sc-элемента*) входящего в состав базы знаний *ostis-системы*]
 - := [*информационная конструкция* (чаще всего это строка символов), обеспечивающая однозначную идентификацию соответствующей сущности, описываемой в базах знаний *ostis-систем*, и являющейся, чаще всего, именем (термином), соответствующим описываемой сущности, именем, обозначающим эту сущность во внешних текстах *ostis-систем*]
- *sc-спецификация*
 - := [семантическая окрестность]
 - := [семантическая окрестность соответствующего *sc-элемента* (внутреннего знака, хранимого в памяти *ostis-системы* в составе её базы знаний, представленной на внутреннем языке *ostis-систем*.)]
 - := [семантическая окрестность некоторого *sc-элемента*, хранимого в *sc-памяти*, в рамках текущего состояния этой *sc-памяти*]
- *sc-конструкция \ sc-спецификация*
 - := [*sc-конструкция* (конструкция SC-кода – внутреннего языка *ostis-систем*), не являющаяся *sc-спецификацией*]
- *файл ostis-системы \ sc-идентификатор*
 - := [*файл ostis-системы*, не являющийся *sc-идентификатором*]

Важно также отметить, что к числу частных правил построения *sc-конструкций* относятся *Правила построения баз знаний ostis-систем*. Эти правила направлены на обеспечение целостности баз знаний *ostis-систем*, на обеспечение (1) востребованности (нужности) знаний, входящих в состав каждой базы знаний, и (2) целостности самой базы знаний, т.е. достаточности знаний, входящих в состав каждой базы знаний для эффективного функционирования соответствующей *ostis-системы*.

Пункт 7.2.2.12. Направления развития Стандарта OSTIS

Стандарт OSTIS

⇒ общие направления развития*:

- {• [Включить в Стандарт OSTIS достаточно подробные правила построения (оформления) *sc-идентификаторов* и *sc-спецификаций* различного вида сущностей, а также различного вида файлов *ostis-систем*]
 - [На каждом этапе четко распределять работу по развитию различных разделов Стандарта OSTIS]
 - [Все инструментальные средства, входящие в состав Технологии OSTIS, должны быть достаточно детально описаны (специфицированы) в виде соответствующих онтологических моделей, имеющих четкую семантическую связь с соответствующими онтологиями и смежными предметными областями, входящими в состав Стандарта OSTIS]
 - [Доработать структуру и содержание титульной спецификации Стандарта OSTIS]
 - [Доработать титульные спецификации всех разделов Стандарта OSTIS]
 - [Обеспечить достаточную полноту *sc-спецификации* всех рассматриваемых (описываемых, обозначаемых *sc-элементами*) сущностей]
 - [Существенно расширить Библиографию OSTIS]
 - [Постоянно отслеживать синонимию/омонимию *sc-идентификаторов*]
 - [Постоянно совершенствовать контроль качества выполнения работ по развитию Стандарта OSTIS]
 - [Необходимо постоянно проводить анализ публикаций других авторов по вопросам, близким тематике Стандарта OSTIS, и фиксировать в Стандарте OSTIS результаты сравнительного анализа точки зрения, представленной в Стандарте OSTIS с точками зрения иных авторов путем включения в Стандарт OSTIS спецификаций соответствующих библиографических источников с полезными цитатами, используемых в них терминов по сравнению с терминологией Стандарта OSTIS и т.д.]
 - [Все *sc-идентификаторы*, входящие в состав *sc.g-текстов*, *sc.n-текстов* и различных иллюстраций должны иметь одинаковый шрифт и размер. За исключением, возможно размера *sc-идентификаторов* в *sc.g-текстах*]
- }

Пункт 7.2.2.13. Достоинства Стандарта OSTIS

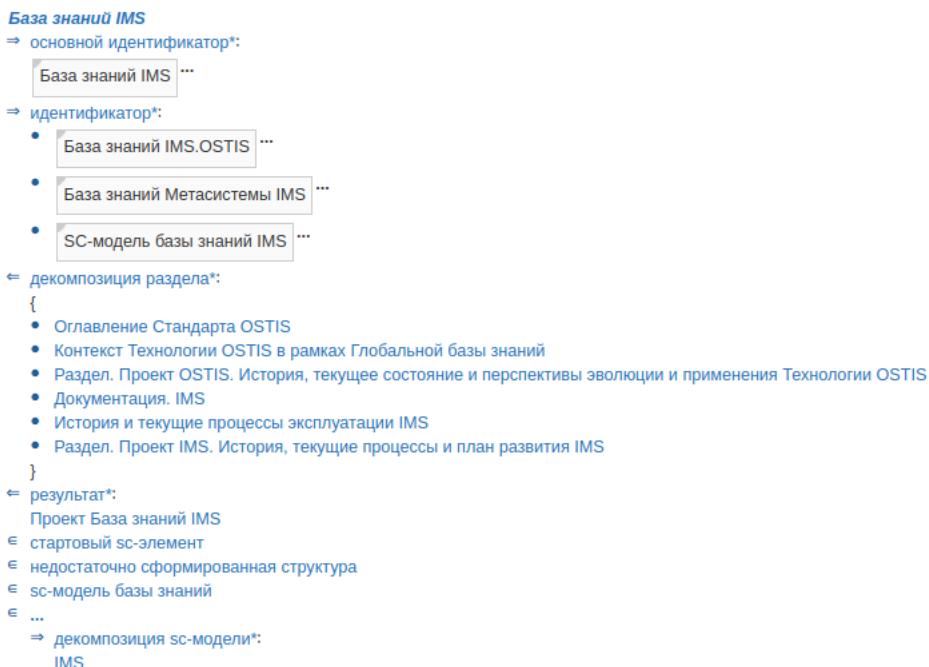
Стандарт OSTIS является примером перехода к принципиально новой форме представления и публикации научно-технической информации, результатов исследовательской деятельности – не просто к форме электронного документа, а к форме семантически структурированного электронного документа, являющегося частью базы знаний по соответствующей научно-технической дисциплине. Это существенно повышает эффективность использования накапливаемой человеком научно-технической информации, поскольку пользователь этой информации может не только ее просматривать (читать), но и взаимодействовать с и.к.с., которая становится партнером в использовании нужной ему информации.

Проект создания и развития Стандарта OSTIS является прообразом принципиально нового подхода к организации научно-технической деятельности в рамках каждой научной дисциплины. Эта деятельность реализуется в форме открытого проекта, направленного на развитие базы знаний интеллектуального портала знаний по соответствующей научно-технической дисциплине. Такой уровень формализации научно-технической информации, который понятен не только специалистам, но и интеллектуальным компьютерным системам существенно повышает эффективность и расширяет области использования этой информации в интеллектуальных компьютерных системах. Так, например, интеллектуальный портал знаний по какой-либо технической дисциплине естественным образом становится интеллектуальной системой автоматизации проектирования технических систем соответствующего класса.

Также важным достоинством является факт того, что *Стандарт OSTIS* является прототипом учебных пособий нового поколения, имеющих четкую логико-семантическую структуризацию и стратификацию учебного материала, а также теоретико-множественную и логическую классификацию всех используемых понятий. Следовательно, использование *Стандарта OSTIS* не только как основы интеллектуальной системы автоматизации комплексной поддержки жизненного цикла и.к.с. нового поколения, но и в качестве комплексного учебного пособия по подготовке молодых специалистов в области Искусственного интеллекта, является прообразом широкого использования различных интеллектуальных порталов научно-технических знаний в качестве комплексных учебных пособий для подготовки молодых специалистов по соответствующим специальностям. Это существенно повысит качество образования, которое не должно отставать от развития соответствующих научно-технических направлений, а должно становиться неотъемлемой составляющей этого развития.

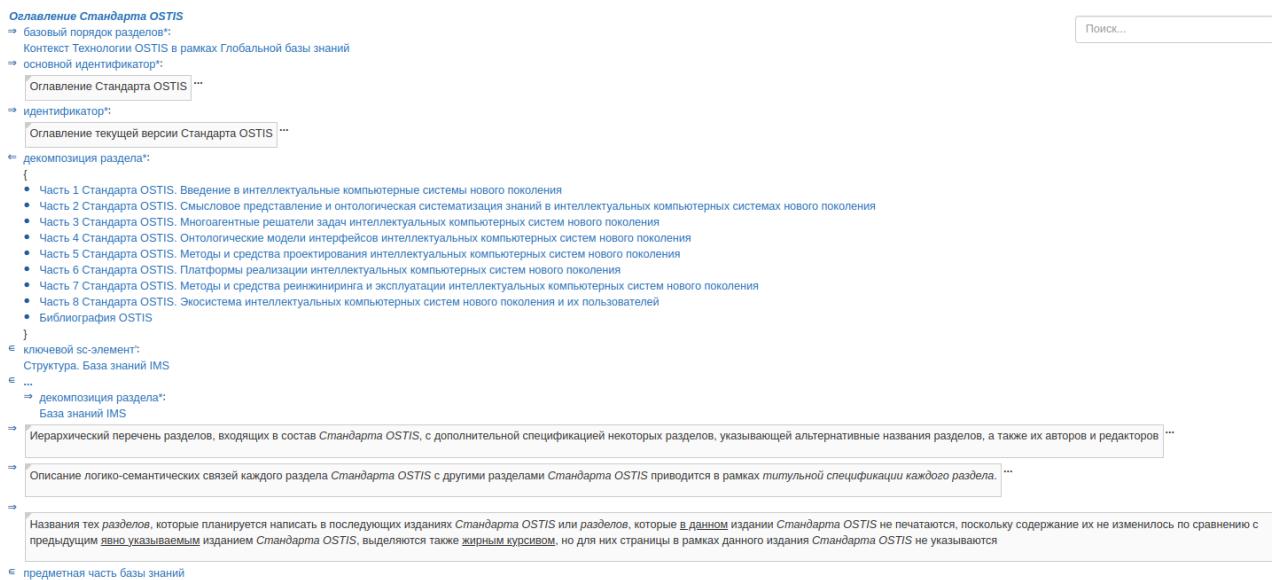
Пункт 7.2.2.14. Стандарт OSTIS как основная часть базы знаний Метасистемы OSTIS

Стандарт OSTIS является основной частью базы знаний Метасистемы OSTIS, описывающей текущую версию Технологии OSTIS, что продемонстрировано на рисунке *Стартовая страница базы знаний Метасистемы OSTIS*.



= *Стартовая страница базы знаний Метасистемы OSTIS*

Предлагаемое представление *Стандарта OSTIS* обеспечивает эффективную семантическую навигацию по содержанию *Стандарта OSTIS*, поскольку перейдя к соответствующему разделу *Метасистемы OSTIS*, как представлено на рисунке *Оглавление Стандарта OSTIS*, можно увидеть текущее оглавление *Стандарта OSTIS*.



= Оглавление Стандарта OSTIS

Пользователю предоставляется возможность перейти к любой интересующей его теме, как показано на рисунке *Навигация по Оглавлению Стандарта OSTIS* и задавать *Метасистеме OSTIS* широкий спектр нетривиальных вопросов о самых различных деталях и тонкостях *Технологии OSTIS*, как представлено на рисунке *Функция вопросов Метасистемы OSTIS* и получать ответы на заданные вопросы, как представлено на рисунке *Функция ответов Метасистемы OSTIS*.

Предметная область структур

⇒ основной идентификатор*:

Предметная область структур ...

⇒ частная предметная область по классу первичных элементов*:

Предметная область знаний

⇐ частная предметная область по классу первичных элементов*:

Предметная область sc-элементов

Є ключевой sc-элемент*:

Предметная область и онтология структур

Э максимальный класс объектов исследования*:

структура

Э немаксимальный класс объектов исследования*:

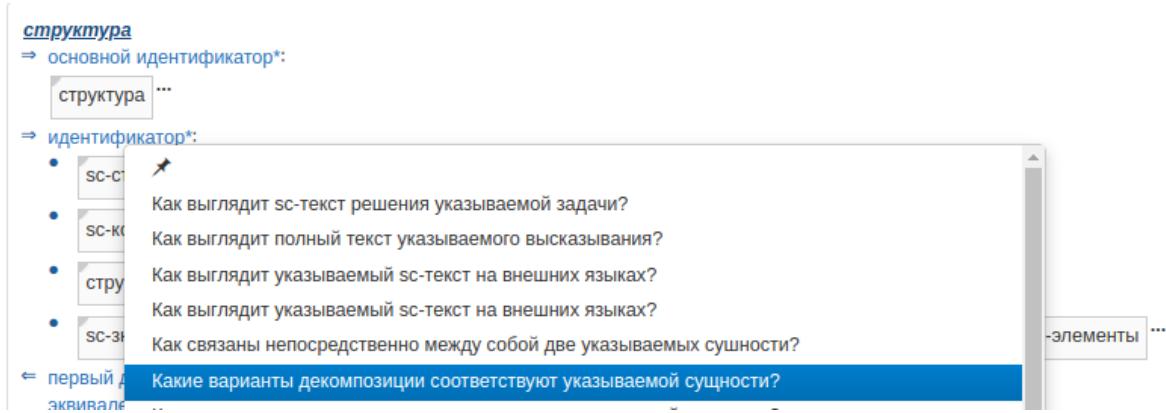
- связная структура
- несвязная структура
- тривиальная структура
- нетривиальная структура
- структура второго уровня
- семантический уровень структурного элемента
- количество семантических уровней элементов структуры

Э исследуемое отношение*:

- элемент структуры'
- непредставленное множество'
- полностью представленное множество'
- частично представленное множество'
- элемент структуры, не являющийся множеством'
- максимальное множество'
- немаксимальное множество'
- первичный элемент'
- вторичный элемент'
- элемент второго уровня'
- метасвязь'
- полиморфность*
- полиморфизм*
- гомоморфность*
- гомоморфизм*
- изоморфность*
- изоморфизм*
- автоморфность*
- автоморфизм*
- аналогичность структур*
- сходство*
- различие*
- первичная синтаксическая структура sc-текста*

Є предметная область

= Навигация по Оглавлению Стандарта OSTIS



= *Функция вопросов Метасистемы OSTIS*

структура

⇐ разбиение*:

- {
 - несформированная структура
 - сформированная структура}
- {
 - структура, имеющая средний уровень сформированности
 - достаточно сформированная структура
 - недостаточно сформированная структура}
- {
 - несвязная структура
 - связная структура}
- {
 - нетривиальная структура
 - тривиальная структура}
- {
 - sc-конструкция нестандартного вида
 - sc-конструкция стандартного вида}

= *Функция ответов Метасистемы OSTIS*

По умолчанию ответы системы пользователю отображаются в SCn-коде, который является гипертекстовым вариантом внешнего отображения текстов SC-кода и может читаться как линейный текст.

Заключение к Главе 7.2.

Для реализации кооперативного целенаправленного и адаптивного взаимодействия интеллектуальных компьютерных систем в рамках автоматически формируемых коллективов интеллектуальных компьютерных систем необходима семантическая совместимость интеллектуальных компьютерных систем, а это, в свою очередь, требует унификации интеллектуальных компьютерных систем. Унификация интеллектуальной компьютерной системы возможна только на основе общей формальной теории интеллектуальных компьютерных систем и соответствующего ей **стандарта интеллектуальных компьютерных систем**, а для этого необходима глубокая конвергенция различных направлений исследований в области искусственного интеллекта.

Поскольку результатом развития искусственного интеллекта как научной дисциплины является перманентная эволюция общей теории интеллектуальных компьютерных систем и соответствующего стандарта интеллектуальных компьютерных систем, для повышения темпов развития искусственного интеллекта и, соответственно, технологии разработки интеллектуальных компьютерных систем необходимо создание портала научно-технических знаний по искусственному интеллекту, обеспечивающего координацию деятельности специалистов, а также согласование и интеграцию результатов этой деятельности.

В главе рассмотрен подход к автоматизации процессов создания, развития и применения стандартов на основе Технологии OSTIS. На основе *Стандарта Технологии OSTIS* рассмотрены основные принципы, лежащие в основе предлагаемого подхода к стандартизации.

Предложенный в данной главе подход позволяет обеспечить не только возможность автоматизации процессов создания, согласования и развития стандартов, но и позволяет значительно повысить эффективность процессов применения стандарта, как в ручном, так и в автоматическом режиме.

Глава 7.3.

Структура Экосистемы OSTIS

⇒ *автор**:

- Загорский А.С.
- Голенков В.В.
- Шункевич Д.В.

⇒ *аннотация**:

[Рассмотрена архитектура экосистемы интеллектуальных компьютерных систем на основе Технологии OSTIS.

Уточнена формальная трактовка таких понятий как ostis-система, ostis-сообщество, выделена типология ostis-систем, что в совокупности позволяет определить структуру Экосистемы OSTIS.]

⇒ *подраздел**:

- § 7.3.1. Иерархическая система взаимодействующих ostis-сообществ
- § 7.3.2. Автоматизация человеческой деятельности в области Искусственного интеллекта, осуществляемая в рамках Экосистемы OSTIS
- § 7.3.3. Принципы автоматизации различных видов и областей человеческой деятельности в рамках Экосистемы OSTIS
- § 7.3.4. Семантически совместимые интеллектуальные ostis-порталы научных знаний
- § 7.3.5. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения
- § 7.3.6. Персональные ostis-ассистенты пользователей

Понятие архетипа сети сводится к объединению множества автономных объектов друг с другом. Эти объекты сильно связаны друг с другом, и при этом не имеют какого-либо центра. Таким образом, они образуют децентрализованную сеть, в которой отсутствует единый центр управления.

Можно выделить следующие характеристики такой системы:

- отсутствие или слабовыраженность централизованного управления;
- автономная природа участников, объектов такой сети;
- сильная связность участников такой сети друг с другом;
- влияние участников такой сети друг на друга нелинейно и достаточно сложно.

У таких распределённых искусственных систем выделяются как преимущества (высокий уровень адаптивности, устойчивости, связности), так и недостатки (неоптимальность, неуправляемость, непредсказуемость поведения). Наиболее подходящим примером реализованной технологии на основе концепции сети является интернет.

Архетип сети удобно использовать для отображения сложных процессов, взаимозависимость компонентов, экономических, социальных, экологических процессов, процессов коммуникации. В таких процессах нет начала или конца, всё является центром. Сеть является единственной топологией, способной к безграничному расширению или самостоятельному обучению, остальные топологии имеют собственные ограничения. "The Atom is the icon of 20th century science. The symbol of science for the next century is the dynamical Net".

Концепция экосистемы стала популярным способом описания взаимодействия самостоятельно действующих систем во внешней среде. Экосистемы имеют две отличительные характеристики по сравнению с другими концепциями сотрудничества: взаимодополняемость и взаимозависимость присутствуют одновременно, и система не полностью иерархически контролируется.

При традиционных подходах к решению проблемы формирования экосистемы возникают проблемы, связанные с низким уровнем интероперабельности таких систем. Зачастую каждая из систем будет иметь свой специализированный программный интерфейс и формат данных для общения с ней, что ведёт к дополнительным расходам на устранение недостатков таких проблем. Поддержка жизненного цикла, модификация уже существующих систем может накладывать дополнительные временные и ресурсные затраты.

Ключевым направлением повышения уровня интеллекта индивидуальных интеллектуальных кибернетических систем — это переход от абсолютно независимых друг от друга индивидуальных интеллектуальных кибернетических систем к их универсальным сообществам, т.е. к многоагентным системам, самостоятельными агентами которых являются указанные индивидуальные интеллектуальные кибернетические системы. В рамках таких систем обеспечивается возможность коммуникации каждого агента с каждым, а также обеспечивается возможность формирования специализированных коллективов для коллективного решения сложных коллективных задач. Реа-

лизация указанного универсального сообщества интероперабельных интеллектуальных кибернетических систем осуществляется в виде Глобальной Экосистемы OSTIS.

§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ

⇒ *ключевое понятие**:

- ...

⇒ *ключевое знание**:

- ...

⇒ *подраздел**:

- *Пункт 7.3.1.1. Формальная модель Экосистемы интеллектуальных компьютерных систем нового поколения*
- *Пункт 7.3.1.2. Поддержка совместимости между ostis-системами, входящими в состав Экосистемы OSTIS*
- *Пункт 7.3.1.3. Описание структуры Экосистемы OSTIS*

Пункт 7.3.1.1. Формальная модель Экосистемы интеллектуальных компьютерных систем нового поколения

Экосистема OSTIS — Социотехническая экосистема, представляющая собой коллектив взаимодействующих семантических компьютерных систем и осуществляющая перманентную поддержку эволюции и семантической совместимости всех входящих в нее систем, на протяжении всего их жизненного цикла.

Интеллектуальная компьютерная система, которая построена в соответствии с требованиями и стандартами Технологии OSTIS, определяется как ostis-система. Это обеспечивает существенное развитие целого ряда свойств этой компьютерной системы, позволяющих значительно повысить уровень интеллекта этой системы (и, прежде всего, ее уровень обучаемости и уровень социализации).

Экосистема OSTIS представляет собой коллектив взаимодействующих:

- самих ostis-систем;
- пользователей указанных ostis-систем (как конечных пользователей, так и разработчиков);
- некоторых компьютерных систем, не являющихся ostis-системами, но рассматриваемых ими в качестве дополнительных информационных ресурсов или сервисов.

Участники коллектива Экосистемы OSTIS характеризуются как:

- семантически совместимые;
- постоянно эволюционирующие индивидуально;
- постоянно поддерживающие свою совместимость с другими участниками в ходе своей индивидуальной эволюции;
- способные децентрализованно координировать свою деятельность.

Цель Экосистемы OSTIS — обеспечить постоянную поддержку совместимости компьютерных систем, входящих в Экосистему как на этапе их разработки, так и в ходе их эксплуатации. Проблема заключается в том, что в ходе эксплуатации систем, входящих в Экосистему OSTIS, они могут изменяться из-за чего совместимость может нарушаться. Задачами Экосистемы OSTIS являются:

- оперативное внедрение всех согласованных изменений стандарта ostis-систем (в том числе, и изменений систем используемых понятий и соответствующих им терминов);
- перманентная поддержка высокого уровня взаимопонимания всех систем, входящих в Экосистему OSTIS, и всех их пользователей;
- корпоративное решение различных сложных задач, требующих координации деятельности нескольких ostis-систем, а также, возможно, некоторых пользователей.

Экосистема OSTIS – это переход от самостоятельных ostis-систем к коллективам самостоятельных ostis-систем, т.е. к распределенным ostis-системам.

ostis-система

⇒ *разбиение**:

- {• *самостоятельная ostis-система*

- *встроенная ostis-система*
 - *коллектив ostis-систем*
- }

Пункт 7.3.1.2. Поддержка совместимости между *ostis*-системами, входящими в состав Экосистемы OSTIS

Каждая система, входящая в состав Экосистемы OSTIS, должна:

- интенсивно, активно и целенаправленно обучаться, как с помощью учителей-разработчиков, так и самостоятельно;
- сообщать всем другим системам о предлагаемых или окончательно утвержденных изменениях в онтологиях и, в частности, в наборе используемых понятий;
- принимать от других *ostis*-систем предложения об изменениях в онтологиях, в том числе в наборе используемых понятий, для согласования или утверждения этих предложений;
- реализовывать утвержденные изменения в онтологиях, хранимых в ее базе знаний;
- способствовать поддержанию высокого уровня семантической совместимости не только с другими *ostis*-системами, входящими в Экосистему OSTIS, но и со своими пользователями (обучать их, информировать их об изменениях в онтологиях).

К самостоятельным *ostis*-системам, входящим в состав Экосистемы OSTIS, предъявляются особые требования:

- они должны обладать всеми необходимыми знаниями и навыками для обмена сообщениями и целенаправленной организации взаимодействия с другими *ostis*-системами, входящими в Экосистему OSTIS;
- в условиях постоянного изменения и эволюции *ostis*-систем, входящих в Экосистему OSTIS, каждая из них должна сама следить за состоянием своей совместимости (согласованности) со всеми остальными *ostis*-системами, т.е. должна самостоятельно поддерживать эту совместимость, согласовывая с другими *ostis*-системами все требующие согласования изменения, происходящие у себя и в других системах.

По назначению *ostis*-системы, входящие в Экосистему OSTIS, могут быть:

- ассистентами конкретных пользователей или конкретных пользовательских коллективов;
- типовыми встраиваемыми подсистемами *ostis*-систем;
- системами информационной и инструментальной поддержки проектирования различных компонентов и различных классов *ostis*-систем;
- системами информационной и инструментальной поддержки проектирования или производства различных классов технических и других искусственно создаваемых систем;
- порталами знаний по самым различным научным дисциплинам;
- системами автоматизации управления различными сложными объектами (производственными предприятиями, учебными заведениями, кафедрами вузов, конкретными обучаемыми);
- интеллектуальными справочными и help-системами;
- интеллектуальными робототехническими системами.

Для обеспечения высокой эффективности эксплуатации и высоких темпов эволюции Экосистемы OSTIS необходимо постоянно повышать уровень информационной совместимости (уровень взаимопонимания) не только между компьютерными системами, входящими в состав Экосистемы OSTIS, но также между этими системами и их пользователями.

Одним из направлений обеспечения такой совместимости является стремление к тому, чтобы база знаний (картина мира) каждого пользователя стала частью (фрагментом) Объединенной базы знаний Экосистемы OSTIS. Это значит, что каждый пользователь должен знать, как устроена структура каждой научно-технической дисциплины (объекты исследования, предметы исследования, определения, закономерности и т.д.), как могут быть связаны между собой различные дисциплины.

Поддержка совместимости Экосистемы OSTIS с ее пользователями осуществляется следующим образом:

- в каждую *ostis*-систему включаются встроенные *ostis*-системы, ориентированные
 - на перманентный мониторинг деятельности конечных пользователей и разработчиков этой *ostis*-системы,
 - на анализ качества и, в первую очередь, корректности этой деятельности,
 - на повышение квалификации пользователей (персонифицированное обучение);
- в состав Экосистемы OSTIS включаются *ostis*-системы, специально предназначенные для обучения пользователей Экосистемы OSTIS базовым общепризнанным знаниям и навыкам решения соответствующих классов задач.

Экосистеме OSTIS ставится в соответствие ее объединенная база знаний, которая представляет собой виртуальное объединение баз знаний всех *ostis*-систем, входящих в состав Экосистемы OSTIS. Качество этой базы знаний (пол-

нота, непротиворечивость, чистота) является постоянной заботой всех самостоятельных ostis-систем, входящих в состав Экосистемы OSTIS.

Пункт 7.3.1.3. Описание структуры Экосистемы OSTIS

Субъект, входящий в состав Экосистемы OSTIS, является агентом Экосистемы OSTIS.

агент Экосистемы OSTIS

⇒ разбиение*:

{• индивидуальная ostis-система Экосистемы OSTIS

⇒ разбиение*:

{• самостоятельная ostis-система Экосистемы OSTIS

• встроенная ostis-система Экосистемы OSTIS

}

• ostis-сообщество

⇒ разбиение*:

{• простое ostis-сообщество

• иерархическое ostis-сообщество

}

• пользователь Экосистемы OSTIS

}

Понятие ostis-сообщества представляет собой не только коллектив ostis-систем, но также определенный коллектив людей (пользователей и разработчиков соответствующих ostis-систем). ostis-сообщество является устойчивым фрагментом Экосистемы OSTIS, обеспечивающим комплексную автоматизацию определенной части коллективной человеческой деятельности и перманентное повышение ее эффективности. Иерархическим ostis-сообществом называется такое ostis-сообщество, по крайней мере одним из членов которого является некоторое другое ostis-сообщество.

Правила поведения агентов Экосистемы OSTIS:

- Согласовывать денотационную семантику всех используемых знаков (в первую очередь понятий);
- Согласовывать терминологию, устранивать противоречия и информационные дыры;
- Постоянно бороться с синонимией и омонимией как на уровне sc-элементов (внутренних знаков), так и на уровне соответствующих им терминов и прочих внешних идентификаторов (внешних обозначений);
- Каждый агент Экосистемы OSTIS по своей инициативе может стать членом любого ostis-сообщества Экосистемы OSTIS после соответствующей регистрации;

Все правила поведения агентов Экосистемы OSTIS должны соблюдаться не только ostis-системами, которые являются агентами Экосистемы OSTIS, но и людьми, являющимися ими. Корректное поведение ostis-систем как агентов Экосистемы OSTIS значительно проще обеспечить, чем корректное поведение людей в качестве таких агентов. Поведение пользователей (естественных агентов) Экосистемы OSTIS необходимо внимательно мониторить и контролировать, постоянно способствуя повышению уровня их квалификации как агентов Экосистемы OSTIS, а также повышению уровня их мотивации, целенаправленности и самореализации.

Экосистема OSTIS является максимальным иерархическим ostis-сообществом, обеспечивающим комплексную автоматизацию всех видов человеческой деятельности. Оно не может входить в состав какого-либо другого ostis-сообщества. Принципы, лежащие в основе Экосистемы OSTIS:

- Экосистема OSTIS представляет собой сеть ostis-сообществ;
- Каждому ostis-сообществу взаимно однозначно соответствует корпоративная ostis-система этого ostis-сообщества;
- Каждое ostis-сообщество может входить в состав любого другого ostis-сообщества по своей инициативе. Формально это означает, что корпоративная ostis-система первого ostis-сообщества является членом другого ostis-сообщества;
- Каждому специалисту, входящему в состав Экосистемы OSTIS ставится во взаимооднозначное соответствие его персональный ostis-ассистент, который трактуется как корпоративная ostis-система вырожденного ostis-сообщества, состоящего из одного человека.

В Экосистеме OSTIS можно выделить следующие уровни иерархии:

- индивидуальные компьютерные системы (индивидуальные ostis-системы и люди, являющиеся конечными пользователями ostis-систем);
- иерархическая система ostis-сообществ, членами каждого из которых могут быть индивидуальные ostis-системы, люди, а также другие ostis-сообщества;

- Максимальное ostis-сообщество Экосистемы OSTIS, не являющееся членом никакого другого ostis-сообщества, входящего в состав Экосистемы OSTIS.

Качество Экосистемы OSTIS во многом определяется эффективностью взаимодействия каждой ostis-системы (в том числе каждого ostis-сообщества), каждого человека со своей внешней средой, а также качеством и чистотой самой внешней среды. Потому основной целью Экосистемы OSTIS является повышение качества информационной внешней среды для всех субъектов, входящих в состав Экосистемы OSTIS. Иными словами, Экосистема OSTIS обеспечивает поддержку Информационной экологии человеческого общества.

ostis-сообщество

⇒ разбиение*:

- {• минимальное ostis-сообщество
- коллектив ostis-систем

}

Каждой персоне, входящей в состав Экосистемы OSTIS взаимно однозначно соответствует его личный (персональный) ассистент в виде персонального ostis-ассистента. Таким образом, количество персональных ostis-ассистентов, входящих в состав Экосистемы OSTIS, совпадает с числом персон, входящих в состав Экосистемы OSTIS. Пример персон и соответствующих им персональных ostis-ассистентов приведён на рисунке ??.

Коллектив, состоящий из персоны и соответствующего ей персонального ostis-ассистента, фактически является минимальным ostis-сообществом. Пример минимальных ostis-сообществ приведён на рисунке ??.

Поскольку формально в не минимальные ostis-сообщества входят не персоны, а соответствующие им персональные ostis-ассистенты, все ostis-сообщества, кроме минимальных ostis-сообществ, являются коллективами ostis-систем.

Корпоративная ostis-система есть центральная ostis-система, осуществляющая координацию, организацию, а также поддержку эволюции деятельности членов соответствующего ostis-сообщества. Корпоративная ostis-система является представителем соответствующего ostis-сообщества в других ostis-сообществах, членом которых оно является. Пример корпоративной ostis-системы сообщества представлен на рисунке ??

Основным назначением Корпоративной системы Экосистемы OSTIS является организация общего взаимодействия при выполнении самых различных видов и областей человеческой деятельности, которые могут быть либо полностью автоматизированными, либо частично автоматизированными, либо вообще неавтоматизированными. Из этого следует, что база знаний Корпоративной системы Экосистемы OSTIS должна содержать Общую формальную теорию человеческой деятельности, включающей в себя типологию видов и областей человеческой деятельности, а также общую методологию этой деятельности.

Деятельность в области Искусственного интеллекта, осуществляемая на основе Технологии OSTIS

⇒ основной продукт*:

Экосистема OSTIS

⇒ подпроект*:

- Проект Метасистемы OSTIS
- Проект программной реализации абстрактной sc-машины
- Проект разработки универсального sc-компьютера

Продуктом человеческой Деятельности в области Искусственного интеллекта, осуществляемой на основе Технологии OSTIS, является не просто множество ostis-систем различного назначения, а Экосистема, состоящая из взаимодействующих ostis-систем и их пользователей. Типология ostis-систем, являющихся агентом Экосистемы OSTIS, представлена ниже.

ostis-система, являющаяся агентом Экосистемы OSTIS

- ▷ персональный ostis-ассистент
- ▷ корпоративная ostis-система
- ▷ ostis-портал научно-технических знаний
- ▷ ostis-система автоматизации проектирования
- ▷ ostis-система автоматизации производства
- ▷ ostis-система автоматизации образовательной деятельности
 - ▷ обучающаяся ostis-система
 - ▷ корпоративная ostis-система виртуальной кафедры
- ▷ ostis-система автоматизации бизнес-деятельности
- ▷ ostis-система автоматизации управления
 - ▷ ostis-система управления проектами соответствующего вида
 - ▷ ostis-система сенсомоторной координации при выполнении определенного вида сложных действий во внешней среде

- *ostis-система управления самостоятельным перемещением*
- *робота по пересеченной местности*

§ 7.3.2. Автоматизация человеческой деятельности в области Искусственного интеллекта, осуществляемая в рамках Экосистемы OSTIS

- ⇒ *ключевое понятие**:
- ...
- ⇒ *ключевое знание**:
- ...
- ⇒ *библиографическая ссылка**:
- ...

Экосистема OSTIS представляет собой саморазвивающуюся сеть ostis-систем, которая обеспечивает комплексную автоматизацию всевозможных видов и областей человеческой деятельности. Особое место среди ostis-систем, входящих в состав Экосистемы OSTIS, занимают корпоративные ostis-системы, через которое осуществляется координация и эволюция деятельности некоторых групп ostis-систем и их пользователей. Основная цель корпоративных ostis-систем – локализовать базы знаний указанных групп ostis-систем, перевести их из статуса виртуальных в статус реальных и автоматизировать их эволюцию.

Экосистема OSTIS является следующим этапом развития человеческого общества, обеспечивающий существенное повышение уровня общественного (коллективного) интеллекта путем преобразования человеческого общества в экосистему, состоящую из людей и семантически совместимых интеллектуальных систем. Экосистема OSTIS — предлагаемый подход к реализации smart-общества или Общества 5.0, построенного на основе Технологии OSTIS.

Сверхзадачей Экосистемы OSTIS является не просто комплексная автоматизация всех видов человеческой деятельности (разумеется, только тех видов деятельности, автоматизация которых целесообразна), но и существенное повышение уровня интеллекта различных человеческих (точнее человеко-машинных) сообществ и всего человеческого общества в целом.

§ 7.3.3. Принципы автоматизации различных видов и областей человеческой деятельности в рамках Экосистемы OSTIS

- ⇒ *ключевое понятие**:
- ...
- ⇒ *ключевое знание**:
- ...
- ⇒ *библиографическая ссылка**:
- ...

§ 7.3.4. Семантически совместимые интеллектуальные ostis-порталы научных знаний

- ⇒ *ключевое понятие**:
- ...
- ⇒ *ключевое знание**:
- ...
- ⇒ *библиографическая ссылка**:
- ...

Без Общей формальной теории интеллектуальных систем невозможно построить набор методов и средств, обеспечивающий комплексную поддержку разработки интеллектуальных компьютерных систем различного назначения и с различным набором навыков, которыми могут обладать интеллектуальные компьютерные системы, но необязательно каждая из них. При этом важно не просто построить Общую теорию интеллектуальных систем и довести ее до строгого формального уровня, но также довести представление такой формальной теории до уровня базы знаний соответствующего портала научных знаний.

Целями интеллектуального портала научных знаний являются:

- ускорение погружения каждого человека в новые для него научные области при постоянном сохранении общей целостной картины Мира (образовательная цель);
- фиксация в систематизированном виде новых научных результатов так, чтобы все основные связи новых результатов с известными были четко обозначены;
- автоматизация координации работ по рецензированию новых результатов;
- автоматизация анализа текущего состояния базы знаний.

Создание интеллектуальных порталов научных знаний, обеспечивающих повышение темпов интеграции и согласования различных точек зрения, – это способ существенного повышения темпов эволюции научно-технической деятельности. Совместимые порталы научных знаний, реализованные в виде ostis-систем, входящих в Экосистему OSTIS, являются основой новых принципов организации научной деятельности, в которой результатами этой деятельности являются не статьи, монографии, отчеты и другие научно-технические документы, а фрагменты глобальной базы знаний, разработчиками которых являются свободно формируемые научные коллективы, состоящие из специалистов в соответствующих научных дисциплинах. С помощью порталов научных знаний осуществляется как координация процесса рецензирования новой научно-технической информации, поступающей от научных работников в базы знаний этих порталов, так и процесс согласования различных точек зрения ученых (в частности, введению и семантической корректировке понятий, а также введению и корректировке терминов, соответствующих различным сущностям).

Реализация семейства семантически совместимых порталов научных знаний в виде совместимых ostis-систем, входящих в состав Экосистемы OSTIS, предполагает разработку иерархической системы семантически согласованных формальных онтологий, соответствующих различным научно-техническим дисциплинам, с четко заданным наследованием свойств описываемых сущностей и с четко заданными междисциплинарными связями, которые определяются связями между соответствующими формальными онтологиями и специфицируемыми ими предметными областями.

Примером портала научных знаний, построенного в виде ostis-системы является Метасистема OSTIS, содержащая все известные на текущий момент знания и навыки, входящие в состав Технологии OSTIS.

§ 7.3.5. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения

⇒ *ключевое понятие**:

- ...

⇒ *ключевое знание**:

- ...

⇒ *библиографическая ссылка**:

- ...

Корпоративная ostis-система позволяет отслеживать, анализировать и постепенно автоматизировать все процессы обработки данных в рамках ostis-сообщества. Такая система действует по следующим принципам:

- интеллектуальные подсистемы (агенты) упорядочивают структуру данных таким образом, что актуальная информация всегда доступна, а устаревшая информация автоматически архивируется или удаляется в соответствии с законами о хранении и защите данных;
- запросы к системе выполняются в виде простых инструкций, система помогает менеджерам вводить необходимую информацию, осуществляет частичную или полную автоматизацию обновления информации из баз данных, доступных через Интернет;
- интеллектуальные подсистемы выполняют структуризацию и классификацию документов и информации для принятия быстрых и правильных решений, автоматически обрабатывает документы и доступные базы данных для отбора ключевой информации, необходимой в данный момент и в будущем;
- существующее системное окружение на предприятии может быть легко подключено к системе через открытые интерфейсы, вся информация остается доступной;

- все ключевые системы данных синхронизируются с основной системой, данные постоянно сравниваются друг с другом, чтобы избежать потерь;
- вся информация доступна в базе знаний, которая является источником данных для рабочих процессов, отчетности и комплексных проверок;

Таким образом, предлагаемая платформа позволяет представить всю информацию об ostis-сообществе единым целостным образом. Достоинствами внедрения предложенной системы являются:

- помочь сбора и оценки информации без преднамеренных искажений или ошибок, связанных с человеческим фактором;
- предоставление возможности полного контроля своих данных;
- система предоставляет только высококачественные, достоверные и актуальные данные;
- цифровое представление всех процессов сообщества обеспечивает интегрированную обработку информации внутри сообщества, что дает полную прозрачность управления, облегчает доступ ко всей информации и ее анализ;
- Благодаря поддержке интеллектуальных подсистем все необходимые данные из документов, процессов и внешних источников могут быть извлечены, структурированы и грамотно оценены.

Корпоративные ostis-системы могут быть применены в различных областях: медицина и здравоохранение, образовательная деятельность широкого профиля, страховой бизнес, промышленная деятельность, административная деятельность, недвижимость, транспорт и т.д.

§ 7.3.6. Персональные ostis-ассистенты пользователей

⇒ *ключевое понятие**:

- ...

⇒ *ключевое знание**:

- ...

⇒ *библиографическая ссылка**:

- ...

Общество должно повернуться "лицом" к каждому человеку, нести ответственность и действительно способствовать каждому человеку персонально, адаптируясь к его особенностям, обеспечить:

- максимальный уровень физического здоровья, активности и максимальное долголетие;
- максимальный уровень физического комфорта, личного пространства, материального потребления;
- максимальный уровень социального, административно-правового комфорта.

Для этого должен осуществляться:

- персональный мониторинг каждой личности по всем направлениям;
- диагностика и устранение нежелательных отклонений;
- оказание своевременной персональной помощи в уточнении направлений дальнейшей эволюции каждой личности.

Необходимо перейти от оказания услуг в решении различных проблем по инициативе самих лиц, столкнувшихся с этими проблемами, к своевременному обнаружению возможности возникновения этих проблем и к соответствующей профилактике. Это возможно только при наличии четкой системной организации персонального мониторинга.

Клиент не обязан знать множество сервисов, из которых он должен выбирать подходящий ему функционал. Для клиента комплекс семантически совместимых сервисов должен располагаться "за кадром". Следовательно, все используемые клиентом информационные ресурсы и сервисы должны быть семантически совместимы. Выбор подходящего для пользователя ресурса или сервиса должен производить его персональный ассистент.

Персональный ассистент должен учитывать, что роли клиентов могут меняться, расширяться, как и его интересы и цели. При этом, все персональные ассистенты должны быть семантически совместимыми с целью понимания друг друга, а также обладать способностью самостоятельно взаимодействовать в рамках различных корпоративных систем, представляя интересы своих клиентов.

Персональный ostis-ассистент есть ostis-система, являющаяся персональным ассистентом пользователя в рамках Экосистемы OSTIS. Такая система предоставляет возможности:

- анализа деятельности пользователя и формирования рекомендаций по ее оптимизации;
- адаптации под настроение пользователя, его личностные качества, общую окружающую обстановку, задачи, которые чаще всего решает пользователь;

- перманентного обучения самого ассистента в процессе решения новых задач, при этом обучаемость потенциально не ограничена;
- вести диалог с пользователем на естественном языке, в том числе в речевой форме;
- отвечать на вопросы различных классов, при этом если системе что-то не понятно, то она сама может задавать встречные вопросы;
- автономного получения информации от всей окружающей среды, а не только от пользователя (в текстовой или речевой форме).

При этом система может как анализировать доступные информационные источники (например, в интернете), так и анализировать окружающий ее физический мир, например, окружающие предметы или внешний вид пользователя.

Достоинства персонального ostis-ассистента:

- пользователю нет необходимости хранить разную информацию в разной форме в разных местах, вся информация хранится в единой базе знаний компактно и без дублирований;
- благодаря неограниченной обучаемости ассистенты могут потенциально автоматизировать практически любую деятельность, а не только самую рутинную;
- благодаря базе знаний, ее структуризации и средствам поиска информации в базе знаний пользователь может получить более точную информацию более быстро.

Персональные ассистенты имеют самое различное назначение и могут быть использованы для самых различных категорий пользователей (пациент, юридическое обслуживание, административное обслуживание, покупатель, потребитель различных услуг).

Глава 7.4.

Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами

⇒ *автор**:

- Загорский А.С.
- Коршунов Р.А.
- Таранчук В.Б.
- Шункевич Д.В.

⇒ *аннотация**:

[***]

⇒ *подраздел**:

- § 7.4.1. Общие принципы интеграции Экосистемы OSTIS с современными сервисами и информационными ресурсами
- § 7.4.2. Принципы интеграции Экосистемы OSTIS с разнородными сервисами
- § 7.4.3. Принципы интеграции Экосистемы OSTIS со структуризованными информационными ресурсами
- § 7.4.4. Интеграция инструментов компьютерной алгебры в приложения OSTIS

Очень важно проектировать не только саму Экосистему OSTIS как форму реализации Общества 5.0, но и процесс поэтапного перехода от современной глобальной сети компьютерных систем к глобальной сети ostis-систем (т.е. к Экосистеме OSTIS).

В рамках такого переходного периода ostis-системы могут выполнять роль системных интеграторов различных ресурсов и сервисов, реализованных современными компьютерными системами, поскольку уровень интеллекта ostis-систем позволяет им с любой степенью детализации специфицировать интегрируемые компьютерные системы и, следовательно, достаточно адекватно "понимать что знает и/или умеет каждая из них. Также ostis-системы способны достаточно качественно координировать деятельность стороннего ресурса и сервиса и обеспечивать "релевантный" поиск нужного ресурса и сервиса. Кроме того системы могут выполнять роль интеллектуальных help-систем – помощников и консультантов по вопросам эффективной эксплуатации различных компьютерных систем со сложными функциональными возможностями, имеющими пользовательский интерфейс с нетривиальной семантикой и использующимися в сложных предметных областях. Такие интеллектуальные help-системы можно сделать интеллектуальными посредниками между соответствующими компьютерными системами их пользователями.

На первых этапах перехода к Обществу 5.0 нет необходимости преобразовывать в ostis-системы все современные системы автоматизации некоторых видов и областей человеческой деятельности. Однако, ostis-системы должны взять на себя координационно-связующую роль благодаря высокому уровню своей интероперабельности. ostis-системы должны научиться либо выполнять миссию активной интероперабельной надстройки над различными современными средствами автоматизации, либо ставить перед современными средствами автоматизации выполнимые для них задачи, обеспечивая их непосредственное участие в решении сложных комплексных задач и организуя управление взаимодействием различных средств автоматизации в процессе коллективного решения сложных комплексных задач.

§ 7.4.1. Общие принципы интеграции Экосистемы OSTIS с современными сервисами и информационными ресурсами

⇒ *ключевое понятие**:

- ...

⇒ *ключевое знание**:

- ...

⇒ *библиографическая ссылка**:

- ...

§ 7.4.2. Принципы интеграции Экосистемы OSTIS с разнородными сервисами

⇒ *ключевое понятие**:

- ...

⇒ *ключевое знание**:

- ...

⇒ *библиографическая ссылка**:

- ...

Под сервисом понимается программно реализованный механизм преобразования данных в соответствии с заданной функцией. Функциональность приложения может быть абсолютно любой: вывести прогноз погоды в заданном городе, увеличить размер изображения, синтезировать речь текстового сообщения, и т.д. Зачастую такое приложение может предоставить программный интерфейс (API – Application Programming Interface), который можно использовать с определённым форматом входов, которым будут соответствовать определённые форматы выходов.

Основной проблемой интеграции функциональных сервисов при формировании цифровой экосистемы из множества взаимодействующих сервисов является различие форматов данных, с которыми работают участники этой экосистемы. Два сервиса, предполагающих обработку данных из одной предметной области (ПрО), с высокой вероятностью будут иметь различный формат данных. Проблема согласования формата данных различных сервисов значительно усложняет разработку самих сервисов и ведёт к увеличению временных затрат.

Под интеграцией Экосистемы OSTIS с сервисом следует понимать возможность использовать функционал сервиса для изменения внутреннего состояния базы знаний системы.

В рамках Экосистемы OSTIS выделены несколько уровней интеграции.

Под полной интеграцией будет подразумеваться исполнение функции сервиса на платформонезависимом уровне и использованием языка SCP. То есть, задача интеграции такого сервиса сводится к выделению алгоритма обработки графовой конструкции и его реализации в рамках базы знаний системы. В результате такой интеграции отпадает надобность вовсе использовать сторонний сервис.

Частичная интеграция означает изменение состояния базы знаний системы на этапах исполнения функции сервиса. Степень глубины интеграции может различаться. В некоторых случаях сервис может обращаться к базе знаний для получения дополнительной информации или для записи промежуточных результатов. В простейшем случае изменение базы знаний может происходить единожды, после получения результата отработки функции сервиса.

Таким образом, минимальными требованиями для интеграции сервиса являются:

- сформировать спецификацию входной конструкции в базе знаний системы;
- сформировать спецификацию выходной конструкции в базе знаний системы;
- реализовать агент обработки знаний, который преобразует конструкцию базы знаний в необходимые данные для запуска сервиса, а также погрузит результаты работы сервиса в базу знаний системы в соответствии со спецификацией.

В рамках задачи формирования спецификаций необходимо выделить ключевые понятия, необходимых для формирования запросов к функциональному сервису, а также для погружения в базу знаний результатов.

Обобщённый алгоритм агента обработки знаний с использованием стороннего сервиса выглядит следующим образом:

- извлечение из базы знаний необходимых структур знаний;
- преобразование извлечённых знаний в формат, необходимый для подачи на вход сервиса;
- отправка запроса и ожидание ответа сервиса;
- формирование конструкции знаний по полученным данным;
- погружение новых знаний в базу знаний интеллектуальной системы.

Рассмотрим процесс внедрения такого агента с точки зрения архитектуры Экосистемы.

Одним из вариантов внедрение агента является его подключение в рамках уже существующей, основной ostis-системы. В таком случае все зависимости для установки сервиса будут также являться и зависимостями основной ostis-системы. Архитектурным паттерном такой системы будет являться монолитная архитектура.

Преимуществом такого варианта является простота реализации и внедрения. Хорошим вариантом использования такой интеграции является отсутствия возможности и необходимости изменить процесс выполнения функции сервиса, или же когда обращение к сервису производится по сети. Так могут быть интегрированы и сервисы получения знаний из внешних источников (получение прогноза погоды, обработка статистической информации, и т.д.), и функциональные сервисы (обработка аудиоинформации и погружение результатов обработки в базу знаний, получение синтаксического анализа предложения).

Альтернативным вариантом внедрения является реализация отдельной ostis-системы, в рамках которой будет интегрирована функция сервиса, что характеризует переход к распределённым взаимодействующим ostis-системам. Архитектурным паттерном такой системы будет являться микросервисная архитектура.

Преимуществом такого варианта интеграции является распределённость, децентрализованность и доступность нового функционала. Система выходит за рамки технических ограничений, функционал может быть распределён на различном аппаратном обеспечении. Полученный функционал может быть использован различными ostis-системами в рамках Экосистемы для достижения своих целей. Минусами такой системы является сложность разработки, а также увеличение временных затрат на общение систем друг с другом по сетевым протоколам.

Одним из вариантов использования является интеграция сервиса, которому для успешного выполнения своей функции необходимо взаимодействовать с базой знаний системы. Примерами такой интеграции могут служить сервисы считывания эмоционального состояния пользователя, или же помощи принятия решения на основе актуальных знаний.

§ 7.4.3. Принципы интеграции Экосистемы OSTIS со структуризованными информационными ресурсами

⇒ *ключевое понятие**:

- ...

⇒ *ключевое знание**:

- ...

⇒ *библиографическая ссылка**:

- ...

§ 7.4.4. Интеграция инструментов компьютерной алгебры в приложения OSTIS

⇒ *ключевое понятие**:

- ...

⇒ *ключевое знание**:

- ...

⇒ *библиографическая ссылка**:

- ...

Глава 7.5.

Автоматизация образовательной деятельности в рамках Экосистемы OSTIS

Гулякина Н.А.

Козлова Е.И.

Гракова Н.В.

Оськин А.Ф.

Головатый А.И.

Самодумкин С.А.

Ли В.

⇒ *аннотация**:

[Аннотация к главе.]

§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем

Организация образовательной деятельности сегодня во многом определяет уровень развития любого государства и общества. Поэтому вполне объясним большой интерес к применению телекоммуникационных и компьютерных технологий в целях повышения эффективности этой деятельности. В этой связи особую актуальность приобретает такое направление исследований из области искусственного интеллекта, как интеллектуальные обучающие системы и автоматизация образовательной деятельности. Интеллектуальные обучающие системы (ИОС) должны стать частью комплекса подготовки специалиста. В то же время, такие системы можно эффективно использовать и в процессе обеспечения повышения квалификации, при осуществлении непрерывного образования. При этом одной из особенностей разработки интеллектуальных обучающих систем становится то, что пользователями таких систем будут и неспециалисты в области компьютерных и телекоммуникационных технологий, люди, существенно разные и по возрасту, и по уровню знаний в той или иной предметной области. Это является стимулом для развития технологий искусственного интеллекта в различных прикладных областях деятельности человека для реализации возможности построения ИОС для подготовки специалистов разных профессиональных направлений. Наиболее перспективным с точки зрения разработки и внедрения ИОС в учебный процесс ВУЗов видится использование экосистемы OSTIS для формирования как элементов обучающих и образовательных систем, так и их интеграции в единые комплексы.

В связи с развитием средств компьютерной поддержки процесса обучения и создания автоматизированных обучающих систем и систем дистанционного обучения, появилась острая необходимость в интеллектуализации всего процесса обучения, так как традиционные системы уже не в силах удовлетворить всех потребностей, как учащихся, так и преподавателей. Это произошло отчасти и потому, что автоматизированные обучающие системы предполагали лишь наличие разветвленной системы ссылок, предлагая обучаемому самому, вне зависимости от его уровня знаний, выбирать путь для дальнейшего обучения. Эффективность обучения определяется не только выбором среди обучения, но и формами представления знаний. Именно форма представления знаний играет важную роль в развитии дидактического принципа “интеллектуальной наглядности” среди обучения. К числу основных задач, направленных на интеллектуализацию компьютерных средств обучения (КСО) можно отнести ориентацию на семантическое представление используемых знаний (учебного материала, знаний об обучаемых, знаний о методах и технологиях обучения, знаний об учебных задачах, знаний об учебных лабораторных работах). Семантическое представление знаний позволяет, например, обеспечить достаточно эффективную компьютерную реализацию такой формы обучения, как консультация обучаемого по заданному предмету.

Современные КСО должны обеспечивать структуризацию, систематизацию и интеграцию учебного материала, возможность навигации по семантическому пространству учебного материала и ассоциативный доступ к любому его фрагменту, а также адекватное и наглядное представление обучаемому семантической структуры этого материала. Необходимо сокращать время разработки КСО, используя технологии параллельного проектирования, в основе которой лежит разбиение учебного материала соответствующей учебной дисциплины на достаточно самостоятельные разделы, с последующей интеграцией в единую интегрированную интеллектуальную обучающую систему по всей дисциплине.

Главной задачей любой обучающей системы является предоставление пользователю информации об изучаемой дисциплине в понятном и наглядном виде. Решение этой задачи возлагается на электронный учебник, который

является неотъемлемой частью любой компьютерной системы обучения. Электронный учебник представляет собой компьютерное средство обучения, ориентированное на самостоятельную работу обучаемого и обеспечивающее:

- хранение учебного материала в электронном виде;
- отображение учебного материала обучаемому;
- возможность навигации по учебному материалу;
- редактирование учебного материала (для разработчиков электронного учебника).

Однако на современном этапе, когда объемы информации стремительно возрастают, появляется необходимость в разработке таких электронных учебников, которые бы обеспечивали:

- возможность семантической структуризации учебного материала;
- интеграцию различных форм представления учебного материала;
- ассоциативный доступ к фрагментам учебного материала;
- возможность интеграции учебной информации с учебно-методической информацией;
- возможность интеграции самих электронных учебников.

Одним из подходов к решению задачи представления и обработки знаний в КСО является использование графодинамических моделей обработки знаний, представленных однородными семантическими сетями с базовой теоретико-множественной интерпретацией. Особенностями таких моделей являются:

- приспособленность к распределенной, асинхронной и параллельной обработке знаний;
- наглядная визуализация сложноструктурированных знаний и метазнаний;
- интеграция семантического представления знаний с любыми другими формами представления информации;
- интегрируемость различных механизмов обработки знаний.

Семантический электронный учебник является тем КСО, которое имеет более развитые средства отображения семантической структуры предметной области с соответствующими навигационными возможностями.

Существенное отличие семантического электронного учебника от традиционного электронного учебника состоит в представлении учебного материала на семантическом уровне. В результате этого, с практической точки зрения, усиливаются возможности электронного учебника как обучающей среды, появляется возможность эффективного самостоятельного изучения предмета.

семантический электронный учебник – это электронный учебник, в основе которого лежит представление учебного материала в виде гипермейдийной семантической сети [4] и обеспечивается возможность семантической навигации и ассоциативного доступа к любому фрагменту учебного материала.

Учебная база знаний семантического электронного учебника представляет собой гипермейдийную семантическую сеть, структурирующую учебный материал. В виде гипермейдийной семантической сети представляется вся учебная информация, такая как, тематические планы обучения, содержательная структура учебного материала, непосредственно сам учебный материал. Основным преимуществом такой формы представления учебной информации является, во-первых, приспособленность семантических сетей для представления неформальных предложений естественного языка, во-вторых, обеспечение наглядности такого представлениями.

Формирование учебных материалов семантического электронного учебника происходит путем формализации знаний предметной области и описания их на соответствующих языках представления знаний, а также с использованием различных традиционных технологий представления информации. Технология разработки гипермейдийных семантических сетей является результатом интеграции традиционных гипермейдийных технологий, мультимедиа-технологий, а также технологий, связанных с представлением структуры знаний в виде семантических сетей.

Основным языком представления знаний для семантического электронного учебника является базовый семантический язык SC (Semantic Code), который является ядром открытого семейства графовых языков представления знаний, построенных на теоретико-множественной основе. Главным достоинством языка SC является то, что он представляет собой удобную основу для создания целого семейства языков, имеющих различное назначение и легко интегрируемых друг с другом.

Для навигации по семантическому пространству учебного и учебно-методического материала семантического электронного учебника разработаны следующие операции:

- навигационно-поисковые операции для баз знаний, представленных на языке SC;
- операции поиска заданного фрагмента учебного материала;
- операции поиска понятий;
- операции поиска учебных вопросов и задач.

При интеграции семантических электронных учебников решаются следующие проблемы:

- поиск противоречий;
- выявление пар синонимичных элементов;
- локализация предположительно синонимичных пар элементов.

Семантические электронные учебники представляют собой принципиально новый тип электронных учебников, в которых явно (в визуальной форме) представлена семантическая структура учебного материала. В семантиче-

ских электронных учебниках полностью сохраняется преемственность с традиционными формами представления учебного материала. К достоинствам семантических учебников можно отнести:

- обеспечение семантической структуризации учебного материала;
- возможность семантической навигации по учебному материалу с помощью навигационно-поисковой графо-динамической ассоциативной машины;
- простые механизмы интеграции семантических электронных учебников, в основе которых лежит явное описание междисциплинарных связей, позволяет разрабатывать электронные учебники по комплексу смежных учебных дисциплин.

Этапы преобразования традиционного учебника в семантический электронный учебник:

1 этап. Описание структуры исходного учебного материала и библиографических атрибутов.

2 этап. Разбиение текста традиционного учебника на семантически элементарные фрагменты с указанием последовательности этих фрагментов в исходном тексте.

3 этап. Установление семантической типологии выделенных элементарных фрагментов текста.

4 этап. Выделение в указанных текстовых фрагментах ключевых понятий и формирование соответствующих им sc-узлов, а также указание связей соответствующих фрагментов с указанными sc-узлами.

5 этап. Перевод на язык SC указанных выделенных фрагментов исходного учебного материала. Установление связей семантической эквивалентности между исходными текстовыми фрагментами и их формализованной записью на языке SC.

6 этап. Построение определений или пояснений указанных выше ключевых понятий (если таковые отсутствуют в учебнике), а также ключевых узлов, которые введены для описания структуры предметной области на русском языке и языке SC, с установлением связи семантической эквивалентности текстов между ними. Кроме того, на данном этапе для выделенного набора понятий и отношений предметной области необходимо отобрать и сформулировать их основные определения, комментарии к ним, расшифровать их семантику и выделить группы семантически связанных элементов предметной области.

7 этап. Построение теоретико-множественной классификационной схемы выделенных понятий.

8 этап. Указание синонимов и омонимов выделенных понятий.

9 этап. Описание наиболее важных соотношений между указанными понятиями (кроме указанной выше теоретико-множественной классификации понятий).

Технологически семантический электронный учебник можно построить полностью сохраняя традиционный учебник, лишь надстраивая над ним соответствующую семантическую сеть. Это актуально на сегодняшний день, поскольку существует большое количество качественного учебно-методического материала, представленного в традиционной форме.

§ 7.5.2. Автоматизация среднего образования с помощью ostis-систем

Современный человек в информационном обществе обязан уметь адаптироваться к быстро меняющимся информационным потокам. Формирование таких навыков – главная задача учебных организаций, к которым в современных условиях предъявляются все более высокие требования.

Дальнейшее развитие образования невозможно без совершенствования методов и средств его информатизации. Как и раньше существуют проблемы развития мотивированного отношения к обучению, формирования навыков самообучения, несогласованности учебных материалов. Для преодоления этих проблем существует острая необходимость в применении технологий искусственного интеллекта в процессе обучения, так как традиционные компьютерные системы обучения уже не в силах удовлетворить всем требованиям, как со стороны учащихся, так и со стороны преподавателей.

Очевидной становится проблема нехватки времени на образование. Для того, чтобы получить хотя бы базовые знания, человеку приходится обучаться в системах среднего и высшего образования в течение 11-18 лет, не считая дошкольного и последипломного образования. Кроме этого, темпы развития современного общества и, в частности, различных технологий, показывают, что каждому человеку для того, чтобы сохранять профессиональную пригодность и быть полноценным членом общества необходимо постоянно развиваться и обучаться.

Предлагаются следующие этапы работ по реализации предлагаемого комплексного инновационного проекта

Проект 1. Разработка семантических электронных учебников по основным дисциплинам школьного образования, имеющих средства редактирования, верификации, интеграции баз знаний, а также средства навигации по базе знаний.

Проект 2. Разработка интеллектуальных решателей задач для каждого семантического электронного учебника.

Проект 3. Разработка специальных средств пользовательских интерфейсов для каждого семантического электронного учебника.

Проект 4. Построение на базе разработанных семантических учебников интеллектуальных обучающих систем, осуществляющих управление обучением на основе индивидуальных особенностей обучаемого.

Проект 5. Разработка интегрированного комплекса обучающих систем, обеспечивающего комплексное обучение, соответствующее среднему образованию.

Проект 6. Разработка семантического ассоциативного компьютера (с не фон-неймановской архитектурой), ориентированного на обработку семантических сетей и обеспечивающего аппаратную поддержку разработанного комплекса обучающих систем.

Рассмотрим реализацию указанного подхода к разработке ИОС на примере ОИС по Геометрии Евклида.

Основой любой интеллектуальной системы является база знаний. Это наиболее динамичный компонент, который меняется в течение всего жизненного цикла. Поэтому сопровождение интеллектуальных систем — серьезная проблема.

Наиболее важный параметр базы знаний — качество содержащихся знаний. Лучшие базы знаний включают самую релевантную и актуальную информацию, имеют совершенные системы поиска информации и тщательно продуманную структуру и формат знаний. Поэтому стадия концептуального анализа или структурирования знаний традиционно является “узким местом” в жизненном цикле разработки интеллектуальных систем. Из этого следует, что разработчиками баз знаний может быть ограниченный круг специалистов, что не позволяет массово разрабатывать качественные интеллектуальные системы.

Интеллектуальная справочная система по геометрии разработана на основании технологии проектирования интеллектуальных систем OSTIS (Open Semantic Technology for Intelligent Systems).

Применим методику проектирования баз знаний, основанную на технологии OSTIS, при проектировании базы знаний интеллектуальной справочной системы по геометрии.

Согласно методике, первым этапом проектирования баз знаний является разработка первой версии тестового сборника предполагает выделение семантически полного набора вопросов, ответы на которые должны содержаться в стартовой версии базы знаний. Для системы по геометрии были выделены следующие классы вопросов: запросы определений; запросы основных свойств заданного объекта; запрос доказательств высказываний; запросы минимального высказывания (минимального фрагмента базы знаний), описывающего семантически значимую связь между всеми объектами заданного множества объектов; запросы пар высказываний, описывающих отличающиеся свойства заданных двух объектов и др.

На следующем этапе проектирования необходимо записать ответы на выделенные вопросы, тем самым будет формироваться стартовая версия базы знаний. В процессе записи ответов на вопросы на формальном языке SCg выделяются ключевые узлы описываемой предметной области. К ключевым узлам, являющимися классами объектов исследования геометрии, относятся следующие ключевые узлы: геометрическая фигура, точка, отрезок, луч, линия, плоскость, многоугольник, треугольник, четырехугольник и др. К ключевым узлам, являющимися отношениями и составляющими предмет исследования, относятся: параллельность, перпендикулярность, пересечение, конгруэнтность, сторона, внутренний угол, лежать между, лежать против, вписанность и др.

На следующем этапе проектирования каждый выделенный ключевой узел геометрии анализируется на предмет его свойств и описывается в псевдоестественной форме с использованием специализированного языка SCn-кода (Semantic Code natural). Данная форма представления является промежуточной между представлением на естественном языке и формальном, и в дальнейшем она будет использоваться в качестве исходных данных для автоматической трансляции конструкций в формальное представление.

База знаний по геометрии на SCn-коде представляет собой семантически структурированный гипертекст. Все понятия из предметной области Геометрии имеют связи между собой, связи реализуются в виде ссылок, что позволяет переходить от понятия к понятию по определенной связи, указываемой при помощи ключевого слова SCn-кода.

В зависимости от типа описываемого понятия, статьи описания делятся на различные виды. В интеллектуальной справочной системе по геометрии были выделены следующие типы статей описания:

формальная теория (пример описываемой сущности – *Геометрия Евклида*)

- синонимичные названия теории;
- объект исследования;
- предмет исследования;
- декомпозиция на разделы;
- надтеория;

- система аксиом, лежащих в основе теории;
- неопределяемые понятия;
- противоречивые теории.

персона (пример описываемой сущности – *Евклид*)

- годы жизни;
- место рождения и жизни;
- объект, автором чего персона является;
- учителя и ученики персоны;
- биография.

понятие, не являющееся отношением (примеры понятий – *отрезок, треугольник, многоугольник*)

- синонимы понятия;
- классификация понятия по различным признакам;
- надклассы понятия и подклассы понятия;
- отношения, заданные на понятии;
- определение, пояснение понятия;
- понятия, на основе которых определяется указанное понятие;
- основные утверждения, описывающие свойства понятия;
- аналоги понятия и антиподы понятия.

отношение (примеры понятий – *внутренний угол*, площадь*, периметр*, граничная точка*, быть между**)

- синонимичные названия отношения;
- область определения отношения;
- схема отношения;
- домены отношения;
- классы отношения по признакам классификации:
 - арность
 - ориентированность
 - наличие кратных связок и встречных связок
 - наличие мультисвязок
 - транзитивность, рефлексивность
- подклассы отношения;
- аналоги отношения;
- утверждения, описывающие свойства данного отношения или его подмножеств.

высказывание (пример описываемой сущности – *Теорема Пифагора*)

- в состав какой теории входит высказывание;
- логический статус (определение, аксиома, теорема);
- формулировка высказывания;
- логический тип (существование, несуществование, единственность, всеобщность, эквивалентность, необходимость и достаточность)
- на основании каких высказываний доказывается указанное высказывание;
- доказательство.

конкретный объект исследования (пример – *Треугольник ABC*)

- идентификаторы объекта;
- рисунок;
- описание связей с другими фигурами и точками (стороны, вершины, биссектрисы, медианы, высоты, граница, точки пересечения и др.);
- числовые характеристики (периметр, площадь, величина углов, величина сторон и др.);
- типология по различным признакам.

Таким образом, в базе знаний хранится большое многообразие видов знаний – классы геометрических фигур (планарные фигуры, линейные фигуры, фигуры, имеющие граничные точки), конкретные фигуры (треугольник, трапеция, окружность), классы отношений, конкретные отношения, утверждения различного типа (аксиомы, теоремы, утверждения определяющего типа), доказательства утверждений (доказательства теорем, лемм), алгоритмы решения задач (в т.ч. геометрических построений), изображения, иллюстрирующие различные геометрические чертежи, видео, флеш, иллюстрирующие решение задач и различные геометрические построения. Возможность представления знаний различного вида позволяет описывать информацию в базе знаний с различных ракурсов, что в свою очередь переводит базу знаний на другой, более высокий интеллектуальный уровень.

Технология проектирования интеллектуальных решателей задач основана на задачно-ориентированной методологии. В связи с этим проектирование системы операций состоит из четырех основных этапов:

- создание тестового сборника задач, которые решаются в рамках исследуемой предметной области;

- определение списка операций, которые будут использоваться при решении задач из тестового сборника;
- создание семантической спецификации каждой из операций;
- реализация и отладка операций.

Такая методика проектирования операций позволяет создавать предметно независимые операции для решения конкретных прикладных задач из исследуемой предметной области. Спроектированные таким образом операции являются переносимыми ip-компонентами (компонентами интеллектуальной собственности, от англ. intellectual property) и могут быть использованы в других интеллектуальных справочных системах.

Опишем этапы проектирования системы операций для интеллектуального решателя задач по геометрии на примере конкретной прикладной задачи.

Пусть дан некоторый плоский треугольник с известными длинами сторон. Необходимо найти величину площади заданного треугольника. С точки зрения геометрии задача является довольно тривиальной, однако ее решение будет достаточно непростым, если предположить, что пользователь не определился, что он хочет найти в треугольнике: площадь, углы, периметр или другие характеристики. При помощи sc-технологии [2] (sc – semantic code), основанной на семантических сетях, можно универсально представить знания о данном треугольнике таким образом, чтобы все задачи, перечисленные выше, решались с помощью одного и того же набора операций.

Тестовый сборник задач будет состоять из единственной задачи, описанной выше.

Определим те операции, которые будут использованы при решении задачи о нахождении площади:

- операция поиска в базе знаний информации о значении искомой величины (площади) данного треугольника;
- операция поиска в базе знаний формулы, позволяющей по имеющейся информации найти значение искомой величины (площади);
- операция, осуществляющая прямой логический вывод (modus ponens);
- операция, которая осуществляет унификацию найденной (сгенерированной) формулы с учетом заданных параметров;
- операция, вычисляющая значение арифметического выражения;
- элементарные арифметические операции (сложение (вычитание), умножение (деление), возведение в степень (извлечение корня) и т.д.).

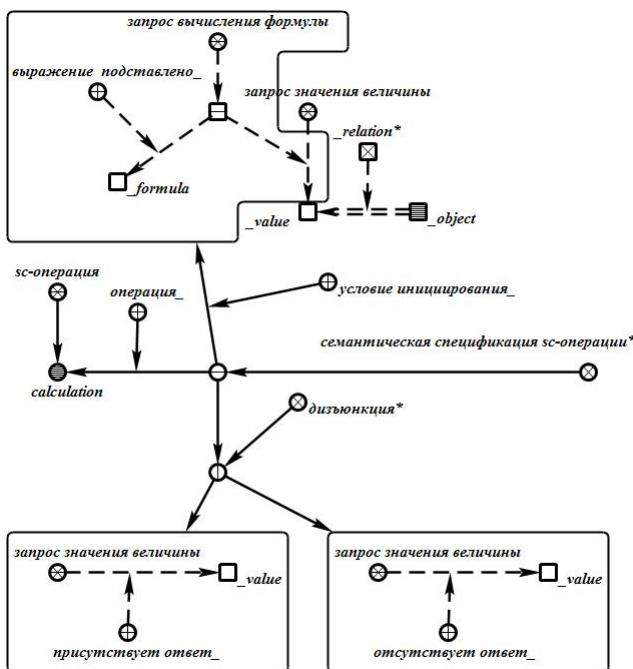
Далее опишем часть третьего этапа проектирования на примере операции вычисления значения формулы.

Пример семантической спецификации операции

Семантическая спецификация операции – это sc-конструкция, которая описывает интерфейс проектируемой операции (условие инициирования, название операции, возможные результаты работы).

Условием инициирования для этой операции является факт появления в памяти sc-дуги, выходящей из узла “запрос вычисления формулы”. После появления в памяти этой дуги вызывается операция “calculation”. В результате работы этой операции либо вычисляется значение некоторой искомой величины, либо генерируются знания о том, что ответ не найден. Отрицательный результат работы этой операции может быть использован другими операциями для того, чтобы попытаться каким-либо другим способом вычислить значение искомой величины.

Завершающим этапом является реализация и тестирование спроектированной операции.



= Семантическая спецификация операции вычисления формулы

Этап реализации также можно разбить на два шага:

- Разработка алгоритма операции
- Реализация программы на подходящем языке. Предпочтительно использовать специально адаптированный язык SCP (semantic code programming).

SCP – специальный язык программирования, предназначенный для обработки семантических сетей[3].

Опишем алгоритм работы операции:

1. Проверяем корректность структуры, которая представляет собой запрос.
2. Из запроса получаем информацию об искомой величине.
3. Находим узел, обозначающий искомую величину в формуле.
4. Находим в формуле все связки отношений “сложение*”, “произведение*” и “возведение в степень*”.
5. Просматриваем все связки, найденные в шаге 4.
 1. Если связка принадлежит классу отношения “сложение*”, то инициируем операцию сложения.
 2. Если связка принадлежит классу отношения “произведение*”, то инициируем операцию умножения.
 3. Если связка принадлежит классу отношения “возведение в степень*”, то инициируем операцию возведения в степень.
 4. Если количество выполняемых операций превысило определенный пользователем предел, то генерируем факт отсутствия ответа и завершаем работу операции.
6. Если значение искомой величины не посчитано, то переходим к шагу 5.
7. Генерируем факт присутствия ответа.

Пример протокола решения задачи

Опишем протокол решения вышеописанной задачи. Протокол отражает последовательность запуска и выполнения операций, текущие условия их запуска, а также результаты выполнения каждой из операций.

1. В памяти появляется конструкция вида:



= Запрос значении площади треугольника ABC

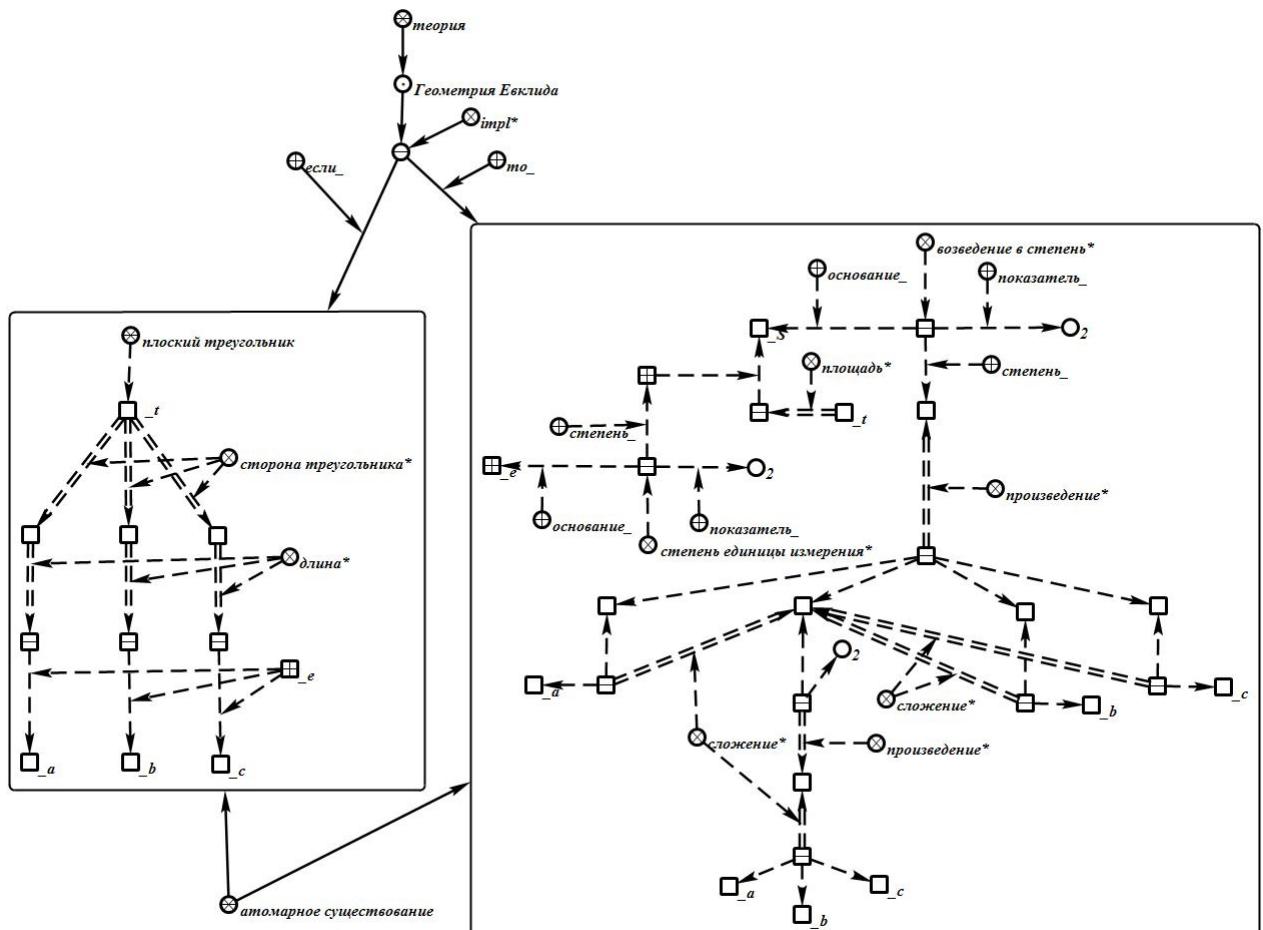
Запускаются операции `find_value` и `find_formula`, при этом выполнение `find_formula` завершается, т.к. конструкция не полностью соответствует условиям запуска. `find_value` проверяет наличие значения указанной величины и, не найдя его, заявляет о его отсутствии:



= Результат работы операции `find_value`

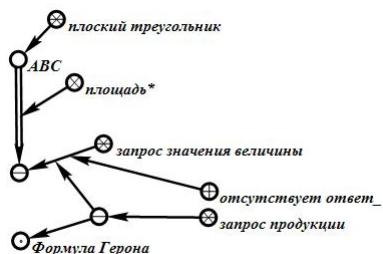
2. Снова запускаются операции `find_value` и `find_formula`, при этом выполнение `find_value` завершается, т.к. конструкция не полностью соответствует условиям запуска.

Операция `find_formula` производит поиск в базе знаний подходящей формулы, которая позволила бы вычислить значение требуемой величины у указанного объекта. Под формулой понимается некоторое импликативное логическое высказывание, справедливо для произвольного класса объектов, которому принадлежит рассматриваемый объект (в данном случае – треугольник), в посылке которого описаны требуемые для вычисления значения, а в заключении – собственно арифметическое выражение, вычисление которого приводит к получению требуемого результата. В данном случае также используется сокращенная форма высказывания о всеобщности, т.е. по умолчанию квантором всеобщности связываются те переменные, которые присутствуют в обеих частях высказывания.



= Пример формулы – формула Герона для вычисления площади треугольника

При просмотре каждой из формул производится проверка на соответствие предполагаемого результата желаемому и проверка наличия в базе знаний всей необходимой информации. Например, в данном случае проверяется тот факт, что формула Герона позволяет вычислить именно площадь (а не периметр) треугольника (а не четырехугольника или круга). Далее проверяется тот факт, что в базе присутствуют длины всех трех сторон данного треугольника, что позволит воспользоваться именно формулой Герона. В противном случае перебор формул продолжается. В данном случае перебор формул заканчивается на формуле Герона. В памяти генерируется запрос на подстановку конкретных значений в формулу:



= Запрос на унификацию формулы

3. Запускается операция `find_value_production`, задачей которой является унификация предложенной формулы константами из базы знаний. Операция использует посылку формулы для поиска значений, соответствующих именно указанному объекту (в данном случае – длин треугольника ABC, а не какого-либо другого треугольника). После этого отбираются значения переменных, связанных квантором всеобщности, и производится генерация арифметического выражения с подстановкой в него значений связанных переменных из формулы. Результатом работы является константное арифметическое выражение, требующее вычисления, о чем сообщается путем генерации соответствующей конструкции.
4. Далее запускается операция `calculation`, алгоритм и условия срабатывания которой подробно рассмотрены выше.

В результате последовательного выполнения указанного набора операций у указанного объекта явно указывается значение требуемого параметра.

§ 7.5.3. Автоматизация высшего образования с помощью ostis-систем

Можно выделить три основных направления интеллектуализации учебного процесса, соответствующих трем уровням учебной деятельности.

Во-первых, это – самообучение на уровне одной дисциплины. В предположении, что обучаемый положительно мотивирован, процесс обучения строится таким образом, чтобы предоставить ему максимальную свободу, помогая быстро ориентироваться в незнакомой предметной области. В связи с этим учебный материал должен быть так структурирован, чтобы его изучение было максимально удобным и, следовательно, эффективным. Здесь требуется совместная кропотливая работа эксперта-предметника и эксперта-педагога. В настоящее время актуальной является проблема повышения степени наглядности, когнитивности учебной информации электронного учебника с целью повышения самостоятельной познавательной деятельности обучаемого. Для решения этой задачи предлагается семантический электронный учебник, который представляет собой интерактивный интеллектуальный самоучитель по некоторой предметной области, содержащий подробные методические рекомендации по ее изучению и предназначенный для мотивированного, самостоятельного и активного пользователя, желающего овладеть знаниями по соответствующей дисциплине.

Во-вторых, это – управление обучением на уровне отдельной дисциплины. В связи с повышением сложности и информационной насыщенности компьютерных средств обучения возникает необходимость в осуществлении управления обучением и процессом взаимодействия с пользователем. Поскольку обучающая система становится более сложной и многофункциональной и предназначена для различных категорий пользователей, то требуется адаптация к индивидуальным особенностям и обстоятельствам для каждого конкретного пользователя. Способность обучающей системы адаптироваться к пользователю является одним из показателей ее эффективности и, как следствие, интеллектуальности. Интеллектуальные обучающие системы представляет собой сложную иерархическую систему, состоящую из совокупности взаимодействующих между собой подсистем, каждая из которых решает определенный класс задач. В качестве базового компонента интеллектуальных обучающих систем используется семантический электронный учебник.

В-третьих, это – управление учебной деятельностью на уровне специальности. Учебная организация и процесс обучения – это не просто совокупность автоматизированных и интеллектуальных обучающих систем по определенным дисциплинам, обладающих средствами мультимедиа, гибкими стратегиями обучения, подсистемами адаптации к пользователю и т.д. Для эффективного использования всех этих средств необходима инфраструктура, в которой осуществляется обработка информации, взаимодействие пользователей и подсистем, совместное решение задач, в которое вовлекаются как пользователи, так и подсистемы.

§ 7.5.4. Семантические модели и средства контроля знаний пользователей в ostis-системах

§ 7.5.5. Дидактические знания

Глава 7.6.

Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов

Андрушевич А.А.
Войтешенко И.С.

⇒ *аннотация**:

[В данной главе описан оригинальный подход к построению межотраслевой экосистемы интернета вещей и приложений умного дома через её семантическое представление на базе технологии OSTIS. Полученные результаты в будущем позволят повысить эффективность компонентного подхода к разработке приложений в интернете вещей, а также обеспечить возможность автоматической синхронизации различных версий компонентов, повышая их совместимость и согласованность.]

Введение в Главу 7.6.

Многоагентная и ситуационная (контекстная) обработка нашла широкое применение в приложениях интернета вещей, например в умном доме **Andrushevich2010**. Однако, несмотря на значительный прогресс последних лет в развитии отраслевых коммуникационных систем (ОКС) и автоматизированных систем управления (АСУ), по-прежнему актуальными остаются следующие проблемы:

- разнородность стандартов и технологий получения, хранения, обработки, передачи данных в приложениях интернета вещей;
- ограниченная отраслью функциональность ОКС;
- низкая адаптивность, совместимость и масштабируемость приложений.

Таким образом, приходится констатировать отсутствие единого комплексного подхода к проектированию экосистемы межотраслевого универсального интернета вещей (ИВ).

§ 7.6.1. Анализ существующих подходов к созданию умных домов

Отметим, что концепция повсеместного межотраслевого контекстно-зависимого безопасного интернета вещей быстро набирает популярность благодаря активно развивающейся конвергенции пакетов технологий, системных функций, платформ и услуг. Яркими примерами механизмов такой конвергенции являются становление и развитие таких общих системных функций в интернете вещей, как базовые службы / сервисы контекстной осведомленности, геолокации и информационной безопасности. Различные отрасли приложений интернета вещей также зачастую реализуют и используют схожие функциональные блоки, включая аутентификацию и авторизацию пользователей, гео-временную локализацию, обработку событий, контекстную осведомленность и многие другие. Например, научным сообществом уже опубликованы **de_paola_context-awareness_2016; kamilaris_geospatial_2018; dobrescu_context-aware_2019** результаты разработки базовых технологий и алгоритмов для получения и предоставления информации о гео-временном местоположении и контексте. На самом деле, видение повсеместного всепроникающего интернета вещей включает в себя не только множество технологий, но и динамические изменения окружающей среды и виртуальное информационное окружение (пространство) пользователей. Сбор и использование контекстной информации выходит за рамки обычных приложений с замкнутым циклом, которые используют только простую информацию о местоположении. Например, универсальная гео-временная информация успешно модернизирует приложения интернета вещей в таких отраслях, как дом, офис, промышленность, торговля, транспорт, здравоохранение, города.

§ 7.6.2. Предлагаемый подход к созданию умных домов

В рамках данной работы предлагается взять за основу общезвестные компоненты и библиотеки технологии OSTIS Голенков В.В.[СтандOTOPPiЭССГИКС-2021кн](#) для формализации описания предметной области экосистемы интернета вещей. Системы, разрабатываемые на основе Технологии OSTIS, названы *ostis-системами*. В основе Технологии OSTIS лежит универсальный способ смыслового представления (кодирования) информации в памяти интеллектуальных компьютерных систем, названный *SC-кодом*. Таким образом может быть достигнута лучшая конвергенция разнородных стандартов и технологий интернета вещей, которая способствует межотраслевой интеграции ОКС и АСУ в единую экосистему.

Итак, дадим следующие определения рассматриваемой предметной области интернета вещей в *SC-коде*:

интернет вещей

- := [Множество физических объектов, подключенных к интернету и обменивающихся данными. Термин «интернет вещей» был впервые употреблен в 1999 году Кевином Эштоном, предпринимателем и соучредителем центра Auto-ID Labs (распределённая исследовательская группа в области радиочастотной идентификации и новых сенсорных технологий) при Массачусетском технологическом институте МИТ]
- := [Концепция сети передачи данных между физическими объектами («вещами»), оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой]

Веб Вещей

- := [Глобальная сеть веб-страниц, автоматически генерированных и автоматически считываемых встроенными в физические объекты (парковка, тротуарная плитка, окна, двери, детская игрушка, посуда, одежда, почва и т.д.) вычислительными устройствами. Для Веба Вещей характерна конвергенция и интеграция с технологиями искусственного интеллекта]
- := [Подход от W3C по использованию веб-технологий в интернете вещей с целью устранения фрагментации в стандартах разработки интернета вещей]

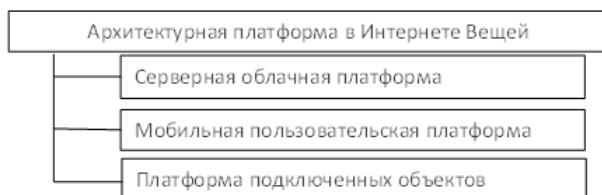
Контекстно-зависимая информационная система

- := [Информационная система, использующая понятие контекста для предоставления имеющей отношение персонифицированной динамической информации и / или услуг пользователю, где степень отношения зависит от текущей среды окружения, интересов, задач, намерений и действий пользователя]

Прикладной программный интерфейс с передачей репрезентативного состояния

- := [Концепция построения распределённого приложения по типу клиент-сервер, в которой каждый запрос (REST-запрос) клиента к серверу содержит в себе исчерпывающую информацию о желаемом ответе (репрезентативном состоянии) сервера без необходимости хранения информации о состоянии клиента (клиентской сессии)]

На основе многолетнего профессионального опыта авторов можно утверждать, что организация универсального типового межотраслевого (или "горизонтального") способа технологической реализации системной архитектуры и приложений в интернете вещей может быть представлена в виде трех потоков разработки, каждый из которых посвящен реализации одной системной платформы интернета вещей. Эти 3 фундаментальные независимые от приложений платформы вместе образуют типовую архитектурную платформу интернета вещей, которая интегрируется во взаимосвязанную гетерогенную систему подсистем интернета вещей, предлагая разработчикам приложений общие функциональные возможности и оставляя разработчикам больше времени для концентрации на бизнес-логике их приложений.



= *Архитектура платформы IoT*

Серверная облачная платформа, обладающая высокими вычислительными ресурсами и большими объёмами памяти, в наибольшей степени управляет всей системой интернета вещей. *Платформа для мобильных устройств* отвечает за представление результирующей и полезной информации пользователям. *Платформа подключенных*

объектов, объединяет сенсорную информацию об окружающей среде и способна выполнять простые действия через подключенные объекты.

Опишем указанные выше платформы подробнее:

- платформа Smart Server: совокупность различных серверов для обеспечения API строительных блоков в облаке для интеграции приложений и сервисов. Данная платформа обладает высокими вычислительными возможностями, большим количеством ресурсов памяти и может взаимодействовать с любыми стандартами. Можно считать вычислительные ресурсы и возможности неограниченными. Её роль заключается в сборе, агрегировании и / или интерпретации контекстно-зависимой информации, поступающей от подключенных узлов приложений ИВ, чтобы сделать ее эффективной и имеющей смысл (потребительскую ценность) для пользователей мобильных устройств. Она также предоставляет разработчикам приложений API-интерфейсы локальных или удаленных универсальных базовых компонент, реализующих функциональность серверной стороны всех приложений, например: обработку и аналитику больших данных, сложную обработку событий, механизм локализации, фреймворк контекстного управления, профиль пользователя и модель поведения пользователей, политики безопасности, контроль доступа (включая регистрацию пользователей, аутентификацию и схему доверия). Наиболее распространёнными примерами платформы Smart Server являются Yandex.Cloud, Amazon Web Services (AWS), Microsoft Azure, Google Cloud, Cisco IoT Cloud Connect, SAP, Oracle IoT, ThingsBoard, SiteWhere, Predix IoT, Thinger.io, Ubidots;
- платформа Smart Mobile: эталонная платформа для мобильных приложений, обеспечивающая функциональность на мобильных устройствах для всех приложений ИВ. Мобильные устройства наиболее часто принадлежат конечным пользователям-людям. Общими функциями мобильных клиентов являются: общие компоненты пользовательских интерфейсов, профилирование пользователей, аутентификация и авторизация, информационная безопасность (конфиденциальность), поиск и обнаружение интеллектуальных серверов, связь с серверными компонентами, получение уведомлений от серверов, "локальные" алгоритмы добычи данных (толстый клиент). Данная платформа предоставляет приложениям ИВ возможность использовать необходимую им контекстную информацию, не беспокоясь о том, каким образом эта контекстная информация добывается. Мобильная платформа может взаимодействовать как с платформой Smart Server, так и непосредственно с некоторыми подключенными узлами ИВ. В заключение мобильная платформа ИВ отвечает за последний этап формирования потребительской ценности через мультимедийное представление / вывод последовательной и интересующей пользователей информации. Наиболее распространёнными примерами платформы Smart Mobile являются Zetta, HP Enterprise Universal, Carriots, ThingsBoard, ThingWorx, Xively. Для сокращения времени разработки мобильных приложений часто используются так называемые гибридные приложения, код которых адаптируется при помощи фреймворков сразу под несколько мобильных аппаратных платформ. Такими наиболее популярными фреймворками являются: PhoneGap, Rhodes, Appcelerator, Xamarin, Ionic, Appy Pie, Native Script.
- платформа Smart Object: интеграция различных шлюзов, сетевых концентраторов и связанных с ними технологий подключенных объектов. Подключенные объекты могут воспринимать физические данные окружающей среды (температура, давление, освещённость, влажность, вибрация и т.д.) через датчики или быть исполнительными механизмами (выключатель, электропривод, электромагнит и т.д.). Передача данных как правило происходит через маломощные коммуникационные стандарты связи. Такие устройства также могут быть оснащены "легковесной" операционной системой. Такие устройства очень разнородны и могут работать очень по-разному. Однако, все особенности реализации подключённых объектов (вещей) должны быть скрыты от других устройств, чтобы обеспечить однородный способ общения с другими компонентами системы, независимо от того, какие данные они воспринимают или как они устроены. Следовательно, единый объектный программный интерфейс является ключевым моментом для того, чтобы иметь возможность легко интегрировать различные шлюзы, обеспечивающие доступ к различным подключённым объектам через одни и те же стандарты, не требуя от разработчиков приложения ИВ знания сложности взаимосвязи объектов со шлюзом и того, какие функциональные возможности (сбор данных или приведение в действие) выполняются на этих объектах. Данная платформа обладает общими функциями, такими как: обнаружение объектов, безопасность и управление. Наиболее распространёнными примерами платформы Smart Object являются open source IoT Kaa, Arduino, Flutter, Qualcomm's IoT Development Kit, Particle.io, ESP8266, Intel Edison, Raspberry Pi, Beagle Bone..

Описываемые в данной работе предметные области разрабатывались на основе

§ 7.6.3. Многокомпонентная модель умных домов

Данный раздел посвящён теоретическому определению универсальной многокомпонентной модели типового приложения интернета вещей для платформ из предыдущего подраздела. Описание для "канонической" или "классической" архитектуры приложения интернета вещей, состоящей из облака или серверных мощностей, различных протоколов передачи данных, пользовательских устройств (ПК, ноутбуки, планшеты, телефоны, встраиваемые

пользовательские интерфейсы в любые устройства), любого рода датчики по сбоду данных и любого "устройства-исполнители-руки". Учитывая особенности и свойства трех частей горизонтальной платформы (Smart Server, Smart Mobile и Smart Object), а также характеристики приложений интернета вещей, предлагается следующее универсальное древовидное представление многокомпонентной модели архитектуры приложения в интернете вещей.

Все компоненты и параметры данной древовидной иерархической модели приведены в качестве примера и могут изменяться в конкретных приложениях ИВ.

(0) Приложение Интернета Вещей $S = A * B * C$

(1) Интеллектуальная Серверная Платформа $A = D * E$

(1.1) Контроль Доступа $D = G * H * I$

(1.1.1) Регистрация G: G1 (Пользователи), G2 (Машины)

(1.1.2) Аутентификация H: H1 (Периодическая), H2 (По Требованию)

(1.1.3) Схема доверия I: I1 (Управляемая приложениями), I2 (Управляемая политикой), I3 (Управляемая профилем), I4 (Управляемая рисками)

(1.2) Обработка Данных $E = J * K * L$

(1.2.1) Контекстное управление J: J1 (Управляемое событиями), J2 (Опрос)

(1.2.2) Обработка потока K: K1 (Оффлайн), K2 (Полу-онлайн), K3 (Онлайн)

(1.2.3) Хранение Данных L: L1 (Простое), L2 (Иерархическое), L3 (Структурированное), L4 (Динамическое)

(2) Интеллектуальная Мобильная Платформа $B = M * \Phi$

(2.1) Общие Характеристики Мобильности $M = O * P$

(2.1.1) Основной тип трафика данных O: O1 (Мультимедиа), O2 (Восприятие/считывание), O3 (Управление), O4 (Хорошо сбалансированный)

(2.1.2) Информация о местоположении P: P1 (Режим включения/выключения), P2 (На основе точности)

(2.2) Пользовательский Интерфейс $\Phi = R * F$

(2.2.1) Пользовательский интерфейс R: R1 (Один носитель), R2 (Мультимедиа), R3 (Адаптивный)

(2.2.2) Доставка контента F: F1 (На основе услуг), F2 (На основе устройств)

(3) Платформа Интеллектуального Объекта $C = Q * T$

(3.1) Общие Характеристики Узла $Q = W * V * U$

(3.1.1) Источник Питания W: W1 (Пассивный), W2 (Активный)

(3.1.2) Физическое взаимодействие V: V1 (Считывание), V2 (Действие), V3 (Комбинированное)

(3.1.3) Шифрование U: U1 (Да), U2 (Нет)

(3.2) Связность $T = X * Y * Z$

(3.2.1) Носитель X: X1 (Беспроводной), X2 (Проводной)

(3.2.2) Тип сети Y: Y1 (Сетка), Y2 (Точка-точка)

(3.2.3) Конфигурация Z: Z1 (Адаптивная), Z2 (Статическая)

На основе предложенной древовидной многокомпонентной модели экосистемы интернета вещей могут быть реализованы такие приложения интернета вещей, как умный дом, офис, промышленность, торговля, транспорт, здравоохранение, умные города.

Остановимся на популярном приложении умный дом и опишем его функции более подробно.

§ 7.6.4. Подсистемы умного дома

Среди функциональных задач, реализуемых в виде компонентов/приложений при проектировании и разработке программно-аппаратного обеспечения «умного дома» **Stojkoska2017ARO**, типичными являются:

- задача доступа к жилое помещение;
- задача наблюдения за одинокими пожилыми людьми **andrushevich_zigbeeieee_2009; biallas_living_2017;**
- задача управления освещенностью жилища;

- задача управления энергопотреблением и энергоэффективностью Zhou2016.

Таким образом, функциональная классификация приложений умного дома может быть определена с использованием SC-кода следующим образом:

Приложение умного дома

:= [обособленный программно-аппаратный комплекс, работающий согласно функциональным требованиям]

⇒ разбиение*:

- Разбиение класса приложений умного дома по функциональности*
- = {• *приложение управления физическим доступом*
 - ▷ *функциональность контроля и управления состоянием всех традиционных и эвакуационных входов и выходов помещения*
 - *приложение наблюдения за одинокими пожилыми людьми*
 - ▷ *функциональность по обеспечению бытовой безопасности одиноких пожилых людей*
 - *приложение управления освещенностью жилища*
 - ▷ *функциональность контроля и управления состоянием источников естественного и искусственного освещения*
 - *приложение управления энергопотреблением и энергоэффективностью*
 - ▷ *функциональность контроля и управления потребления всех видов ресурсов и энергии*
}

Далее более детально опишем функциональность вышеуказанных приложений умного дома.

Пункт 7.6.4.1. Подсистема доступа в жилое помещение

Ключевой задачей такой системы является идентификация людей, которые подходят к двери помещения, и корректная обработка ее результатов: если подошедший к двери человек должен иметь доступ к помещению, дверь открывается автоматически, в противоположном же случае система предлагает поговорить с жителями дома или квартиры. При этом желательно, чтобы система умела определять, находится ли кто-то из жителей дома. Если дома никого нет, то система должна сообщить посетителю, что зайти сейчас он не может.

С точки зрения устройств система оснащена камерой, а также лампами для освещения. Камера включается по датчику движения, причем также включается и освещение, если окружающая среда слишком темная для работы камеры. Система пытается распознать посетителя по снимкам с камеры.

В дополнение к данным камеры, система также имеет в своем распоряжении данные о местоположении жителей. Если по данным геолокации никого нет дома, то система отклоняет посетителей. Также для пользователя предусмотрена возможность включить такой режим вручную. Это может быть полезно, например, если в какой-то период времени жителей не должны беспокоить.

Система также должна содержать графический пользовательский интерфейс, через который можно как наблюдать за выбранными показателями и состоянием устройств, так и управлять поведением системы. Кроме того, через пользовательский интерфейс жители могут получать уведомления о том, что кто-то пытается войти в помещение.

Таким образом, можно выделить следующие функциональные требования к системе:

1. Система позволяет задать несколько профилей жителей таким образом, чтобы их можно было идентифицировать на входе.
2. При положительной идентификации система открывает дверь автоматически.
3. В состоянии по умолчанию камера и освещение выключены, однако они должны включаться по необходимости.
4. Система может определить, находится ли кто-либо из жителей дома или нет. В случае, если никого нет дома, система должна сообщить посетителю об этом. В противном случае система должна предложить поговорить с людьми внутри помещения.
5. Система также должна поддерживать режим «Не беспокоить». Поведение системы при этом соответствует случаю, когда в помещении никого нет.
6. Система должна уведомлять жителей о посетителях.

В дополнение к функциональным требованиям к системе также были разработаны следующие нефункциональные требования:

1. Система может распознавать до 10 различных профилей жителей.
2. Система должна корректно реагировать на сигналы устройств, даже если произошел разрыв соединения.

3. Допустимо также ручное управление, в частности, использование ключа для того, чтобы открыть дверь.
4. Поддерживается расширение системы большим количеством устройств.

Пункт 7.6.4.2. Подсистема наблюдения за одинокими пожилыми людьми

В условиях снижающейся рождаемости доля пожилых людей в обществе растет, в то время как доля людей трудоспособного возраста снижается. Пожилые люди часто нуждаются в помощи с заботой о здоровье, свободном перемещении и в случае нежелательных происшествий. Благодаря использованию технологий интернета вещей, такие люди могли бы жить в большей безопасности и комфорте. Сфера интернета вещей, связанная с наблюдением за пожилыми людьми, также называется AAL (ambient assisted living, проживание с фоновым сопровождением).

В **andrushevich_zigbeeieee_2009** были выделены следующие сферы применения технологий IoT:

1. информационная помощь (легкодоступность всей необходимой информации),
2. умное ситуационное поведение (среда должна узнавать типичные шаблоны поведения и предлагать соответствующую помощь),
3. предсказание нежелательных событий (распознавание таких ситуаций на основании поведенческих и физиологических показателей, а также применение превентивных мер),
4. распознавание нежелательных ситуаций и реакция на них,
5. безопасность (защита от вторжений с использованием авторизационных и аутентификационных механизмов),
6. конфиденциальность (минимизация вмешательства в личную жизнь).

Для удовлетворения указанных потребностей система может полагаться как на исторические, так и получаемые в реальном времени данные. Выделяют две группы датчиков. Данные об окружающей среде исходят от датчиков окружения, например, температуры, движения. Данные о поведении и состоянии человека исходят от носимых датчиков. В системе также могут присутствовать устройства обратной связи для визуального и голосового уведомления о различных событиях. Современные технологии беспроводной связи позволяют создавать надежную, легкую в установке и недорогую инфраструктуру для передачи данных.

Центральным для систем AAL являются сбор и хранение данных о поведении пользователя, выделение шаблонов и определение нежелательных ситуаций на основании отклонений от них. Состояние пользователя может быть определено на основе данных от нескольких датчиков, расположенных в жилище **biallas_living_2017**.

Среди нежелательных ситуаций особо отметим падения. Одним из способов распознавания падений является совместное использование носимого датчика ускорения и атмосферного давления. В случае неожиданного ускорения система дважды считывает данные о давлении, на основании которых делает вывод о положении тела человека **andrushevich_zigbeeieee_2009**.

С точки зрения моделирования таких систем можно выделить три категории показателей. К физическим аспектам относятся как условия внешней среды, так и параметры здоровья человека. В связи с непрерывной природой этой категории наиболее естественным представляется метод системной динамики в сочетании со стохастическим моделированием для учета роли случайности. С другой стороны, дискретно-событийное моделирование позволяет учесть появляющиеся события, например, падения. Наконец, спонтанное поведение пользователя наилучшим образом моделируется с помощью агентного метода.

Пункт 7.6.4.3. Подсистема для управления освещенностью жилища

Примерные функциональные требования к приложению: при входе в помещение свет загорается, среагировав на движение, а через 10 секунд после выхода из помещения свет отключается. Яркость освещения должна корректироваться с учетом уровня уличного освещения, проникающего в жилище. Кроме того, освещение с 23-00 до 6-00 утра должно работать в режиме ночника, с пониженной яркостью.

Пункт 7.6.4.4. Подсистема управления энергопотреблением и энергоэффективностью

В связи с нарастающим энергетическим кризисом и неадекватностью традиционных централизованных энергосистем новым вызовом появилась новая модель: гибридное распределенное производство энергии со значительным вкладом возобновляемых источников энергии. Такая модель также характеризуется двунаправленным потоком

информации и электричества. Существуют решения на всех этапах производства энергии, однако на стороне потребителя основным является направление интернета вещей, в частности, домашней автоматизации. Такие системы получили название HEMS (home energy management system, система управления энергопотреблением для жилых домов) **Stojkoska2017ARO**

К системам управления энергией предъявляют следующие требования **Zhou2016**:

1. Система собирает в реальном времени данные о потреблении и получении энергии, а также о состоянии устройств.
2. Система сохраняет и анализирует исторические данные.
3. Система управляет устройствами в своих рамках таким образом, чтобы обеспечить оптимальное энергопотребление.
4. Пользователь имеет возможность управления устройствами напрямую и удаленно.
5. В случае возникновения нежелательных ситуаций система предупреждает пользователя. Рассмотрим выбор метрик для оценки работы систем управления энергопотреблением.

Среда работы систем управления энергопотреблением — жилое помещение, домохозяйство, даже офисное или производственное здание — приводит к многоцелевому характеру таких систем [6], то есть к необходимости нахождения компромисса между несколькими задачами. В то время как глобальной целью всегда является повышение эффективности использования энергии, на ее достижения налагаются ограничения, в частности, по комфорту пользователей. Кроме того, формулировка конкретной задачи может различаться в зависимости от контекста: снижение выбросов, денежная экономия, балансировка нагрузки на энергосистему — вот некоторые возможные направления.

Цели управления потреблением энергии могут быть выражены через стоимость. Хотя наиболее очевидным и преобладающим фактором является цена потребляемой энергии, в общую формулу расчета также могут включаться такие факторы, как затраты на начальную установку системы, штраф за вклад в общую нагрузку, прогнозируемый износ оборудования, налог на выбросы парниковых газов **Beaudin2015**. Таким образом, система может следовать единой цели — снижение получаемых по этой формуле денежных затрат. Выбор корректного соотношения может представлять сложность, если нет устоявшихся денежных значений для некоторых целей.

Основная задача из тех, которые сложно представить в денежном эквиваленте, — это комфорт пользователей, то есть их неудобства, связанные с качеством услуг, предоставляемых при доставке энергии. Система не должна приводить к значительному изменению их образа жизни. Для оценки влияния системы на комфорт пользователей могут использоваться различные штрафные функции: ограничения по значению, отклонение, штраф за отсутствие сервиса **Beaudin2015**. Иначе говоря, если стоимость представляет собой некоторое соотношение, которое требуется минимизировать, то требования по комфорту налагают ограничения на возможные стратегии.

Описанные выше соображения могут применяться в тестировании общих подходов к оптимизации потребления энергии. Например, может потребоваться оценить выгоды от добавления того или иного устройства и сравнить их с ценой приобретения и установки, или же выбрать из нескольких разрабатываемых алгоритмов наилучший.

С точки зрения конечного пользователя к системе могут предъявляться такие требования, как доступ к историческим данным и тенденциям, возможность удаленного управления устройствами, предупреждение о нежелательных ситуациях **Zhou2016**. Конфиденциальность данных, надежность и эффективность системы также играют роль в оценке ее качества. Подобные требования скорее характерны для продуктов практической направленности, пред назначенных для прямого использования по назначению, а не исследовательских прототипов.

§ 7.6.5. Элементы технической реализации умных домов

Проектирование и программная реализация указанных компонентов/приложений осуществлялось с помощью инструмента визуального программирования Node-RED в сочетании с использованием облачных технологий, в частности, Yandex IoT Core или AWS IoT Core. Node-RED — это инструмент потокового программирования для соединения аппаратных устройств, API и онлайн сервисов. Их совместное позволяет создавать прототипы систем интернета вещей без использования реальных устройств, что позволяет проработать архитектуру системы до ее физической реализации. Одним из немаловажных преимуществ Node-RED является то, что с помощью этого инструмента возможно создать простой прототип системы в той же среде, в которой ведется или будет вестись основная разработка. Такой подход значительно упрощает разработку. Кроме того, Node-RED также предоставляет средства для простой визуализации получившейся системы, что позволяет создать прототип графического интерфейса в рамках этой же среды.

Использование приложениями облачных сервисов, к которым относятся Yandex IoT Core и AWS IoT Core, позволяет значительно уменьшить затраты на инфраструктуру системы, при этом обеспечивая лучшую масштабируемость и отказоустойчивость. Облачные технологии предлагают вычислительные ресурсы, ресурсы для хранения данных и ресурсы коммуникации.

Для передачи сообщений внутри системы в этом случае используется MQTT — сетевой протокол, использующий архитектуру издатель-подписчик и работающий поверх TCP/IP с использованием очередей (Message Queuing Telemetry Transport). Он удобен при использовании при невысокой пропускной способности сети.

Реализация прототипа системы для управления освещенностью жилища была проведена согласно описанию **LightingControlNodeRED**.

Пункт 7.6.5.1. Запуск Node-RED на виртуальной машине в Yandex Cloud

Для исполнения среды Node-RED создается виртуальная машина в облачном сервисе Yandex Compute Cloud. Этот сервис является частью Yandex Cloud и предоставляет масштабируемые вычислительные мощности для создания виртуальных машин и управления ими. Данный сервис предлагает широкий выбор настроек виртуальных машин, от различных операционных систем до тонкой настройки используемых ресурсов.

Нами была использована виртуальная машина на базе операционной системы CentOS 8. Поскольку для работы с Node-RED и для исполнения прототипа приложения не требуется больших вычислительных возможностей, ресурсы виртуальной машине были выделены минимальные.

Для доступа к виртуальной машине использовался SSH. Для этого в настройках облачных сервисов Yandex был выделен публичный IP-адрес. Доступ к виртуальной машине по SSH необходим для установки Node-RED, а разработка может вестись в обычном веб-браузере. После генерации пары ключей для SSH и задания соответствующего публичного ключа в настройках доступа к виртуальной машине к ней можно подключиться с использованием интерфейса командной строки. Для установки Node-RED был использован официальный ресурс «Linux Installers for Node-RED». Также был включен автоматический запуск Node-RED при запуске виртуальной машины. Доступ к Node-RED был настроен на порт 1880.

Пункт 7.6.5.2. Создание объектов сервиса Yandex IoT Core. Интеграция с Node-RED

Основными элементами сервиса Yandex IoT Core являются устройство и реестр. Эти объекты могут обмениваться данными и командами по протоколу MQTT. Устройство в данном сервисе представляет собой абстракцию физического устройства, а реестр является группой устройств, которые логически связаны друг с другом. Данные могут передаваться между устройством и реестром.

Чтобы добавить устройства, необходимо сначала создать реестр. Для доступа к устройствам и реестрам IoT Core предлагает задать пароль, но можно также добавить сертификат. Для данной системы создадим один реестр и три устройства: освещение, камеру и модуль геолокации. Последний нужен только для целей прототипирования и является абстракцией для реальных устройств. Датчики создавать в прототипе не нужно, поскольку эта часть реализуется через Node-RED.

Для обмена MQTT-сообщениями между устройствами и реестрами в IoT Core предусмотрен MQTT-брокер, который отвечает за получение, обработку и доставку сообщений. Node-RED поддерживает соединение с MQTT-брокером через специальные узлы MQTT Input и MQTT Output. Обычно реестрам соответствуют узлы MQTT Input, поскольку этот узел позволяет подписаться на сообщения какой-либо темы («топики») и таким образом его данные можно передать на обработку. Устройствам же чаще соответствуют узлы MQTT Output. Чтобы подключить эти узлы к объектам Yandex IoT Core, требуется указать идентификаторы, а также созданные пароли. Наконец, MQTT-узлы необходимо подписать на нужные топики, чтобы верно передавать сообщения.

Заключение Главы 7.6.

В главе рассмотрен подход к описанию предметной области в приложениях интернета вещей на примере умного дома на основе Технологии OSTIS.

Полученные результаты в будущем позволят повысить эффективность компонентного подхода к разработке приложений в интернете вещей, а также обеспечить возможность автоматической синхронизации различных версий компонентов, повышая совместимость и согласованность.

Глава 7.7.

Автоматизация производственной деятельности в рамках Экосистемы OSTIS

**Иванюк Д.С.
Таберко В.В.
Прохоренко В.А.
Смородин В.С.**

⇒ *аннотация**:

[Современные направления научно-технической деятельности в области оптимизации функционирования производства требуют разработки актуальных подходов в реализации адаптивного управления производственной деятельностью с использованием элементов искусственного интеллекта, нейросетевого моделирования и разработки интеллектуальных компьютерных систем на основе использования новых технологий.

В настоящем разделе предлагается подход к построению интеллектуальной компьютерной системы адаптации управления нового поколения в рамках Экосистемы OSTIS. Формализация компьютерной системы адаптации управления реализуется на основе онтологии предметной области «технологические процессы производства с вероятностными характеристиками», реализованной посредством применения OSTIS-систем в рамках технологии OSTIS.

В основе создания гибридной интеллектуальной системы адаптации управления лежит идея разработки математических моделей нейросетевых контроллеров, решающих задачи реализации методов и алгоритмов синтеза обратных связей по управлению технологическим циклом в зависимости от изменения параметров функционирования объекта управления в режиме реального времени, на основе использования средств программно-аппаратного сопряжения с технологическим циклом производства.]

§ 7.7.1. Адаптивное управление технологическим циклом производства на основе Технологии OSTIS

Современное направление конвергенции работ в области создания интеллектуальных систем [1] требует разработки соответствующего программного обеспечения с элементами когнитивных способностей на основе семантически совместимых технологий искусственного интеллекта. Данное направление включает в себя также создание компьютерных систем, обеспечивающих интеллектуализацию процессов принятия аналитических управленческих решений, что напрямую связано с адаптацией процессов управления сложными динамическими системами (техническими объектами) в режиме реального времени, созданием семантически совместимых баз знаний в области анализа функционирования динамических систем и оптимизации функционирования сложных технических систем на их основе посредством создания открытого программного кода интеллектуальных компьютерных систем поддержки принятия решений. В основу формализации контура управления и математических моделей объекта исследования положены результаты научных разработок авторов в области имитационного моделирования сложных технических систем [3].

Пункт 7.7.1.1. Онтология предметной области “технологические процессы с вероятностными характеристиками”

Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов

Центральными понятиями в рассматриваемой предметной области являются технологический процесс (цикл) и вероятностный технологический процесс.

технологический процесс

:= [технологический цикл]

:=

- [установленная технологическими документами последовательность взаимосвязанных действий, направленных на объект процесса с целью получения требуемого конечного результата]
- \coloneqq [множество технологических операций $\{TCO_{ij}\}$, где $i, j = \overline{1, N}$, в совокупности с используемыми ими ресурсами]

вероятностный технологический процесс

- \coloneqq [ВТП]
- \subset технологический процесс
- \coloneqq [технологический процесс с вероятностными параметрами функционирования]
- \coloneqq [технологический процесс с изменяющейся в ходе его реализации структурой технологического цикла]

технологическая операция

- \coloneqq [TCO]
- \subset технологический процесс
- \coloneqq [часть технологического процесса, выполняемая непрерывно на одном рабочем месте над одним или несколькими одновременно обрабатываемыми или собираемыми изделиями, одним или несколькими исполнителями]

Микротехнологическая операция

- \coloneqq [МТСО]
- \subset технологическая операция
- \coloneqq [конечная последовательность элементарных операций, составляющих в совокупности содержание технологической операции, выполняемой непрерывно на одном рабочем месте]

Традиционно рассматривают две группы технологических процессов:

- непрерывного типа
- дискретного типа

Первая группа технологических процессов обычно реализуется на производстве в режиме реального времени. Данные технологические процессы являются объектом АСУТП. Примером применения АСУТП является контроль за процессом выплавки стали, контроль за поступлением сырья в мартеновские печи и автоматической разливкой металла по формам. Вторая группа технологических процессов характеризуется графовой структурой организации технологического цикла, который реализуется в результате взаимодействия множества технологических операций $\{TCO_i\}$. Некоторые TCO_i могут состоять, в свою очередь, из множества микротехнологических операций $\{MTCO_{ij}\}$. В зависимости от способа вхождения $\{MTCO_{ij}\}$ в состав $\{TCO_i\}$ различают следующие типы технологических процессов:

- одноуровневые, в которых $\{MTCO_{ij}\}$ могут выполняться параллельно-последовательно (в соответствии с графовой структурой связей между $MTCO_{ij}$);
- иерархические, в которых основная технологическая ветвь разделяется на несколько дочерних, после чего происходит обратное слияние; при этом в составе $MTCO_{ij}$ любого уровня иерархии имеются операции «расщепления» технологических линий и операция «сбора» технологических линий;
- итеративные, в которых дочерние операции вложены в основные операции; при этом результаты выполнения операций на одном уровне вложенности используются для выполнения дочерних технологических ветвей.

Моделирование всех типов технологических процессов осуществляется либо на основе критических, либо на основе средних значений расхода ресурсов. Наиболее сложными для моделирования считаются вероятностные технологические процессы второго и третьего типов.

При выполнении $MTCO_{ij}$ осуществляется расход ресурсов технологического цикла. В зависимости от характера расхода этих ресурсов выделяют две группы процессов:

- детерминированные (ДТП), у которых расход ресурсов предприятия осуществляется по средним или максимальным значениям;
- вероятностные (ВТП), у которых одна часть ресурсов используется детерминированным способом, задаваемым списками ресурсов и их количеств, а другая часть ресурсов предприятия используется вероятностным способом, задаваемым с помощью функций вероятностей распределения значений ресурсов, необходимых для выполнения $MTCO_{ij}$.

Подпункт 7.7.1.1.2 Надежностные характеристики функционирования оборудования

Одноуровневые ВТП используют для своего выполнения устройства оборудования, которые могут иметь различные характеристики надежности функционирования. Надежностные характеристики оборудования являются вероятностными и характеризуются следующим образом: моменты отказа определяются функцией распределения

вероятностей значений времени безотказной работы устройства оборудования $F(\tau_{NOw})$, по истечении которого происходит отказ выполнения функций устройством. Отказы могут быть простыми, и после интервала времени восстановления (τ_{ROw}) его функционирование возобновляется. Некоторые отказы могут с вероятностью (P_{f1}) приводить к простой аварии, которая требует для своей ликвидации дополнительного времени (τ_{EM1w}) и дополнительной стоимости (c_{EM1w}). В этом случае имеет место авария оборудования 1-го типа. С вероятностью (P_{f2}) после обычного отказа может происходить авария 2-го типа, которая требует для своей ликвидации некоторой последовательности специальных технологических операций. Каждая такая технологическая операция может требовать дополнительного времени ее ликвидации (τ_{liq}) и дополнительной стоимости ее выполнения (c_{liq}). В итоге ликвидация аварии 2-го типа может приводить к задержке выполнения $MTCO_{ij}$ из-за отказа оборудования на величину, равную сумме этих времен $\sum \tau_{liq}$, и росту стоимости выполнения $MTCO_{ij}$ также на величину суммы стоимости $\sum c_{liq}$. После ликвидации отказов аварий 1-го и 2-го типа происходит лишь задержка времени выполнения и увеличение стоимости ее выполнения, а технологический процесс продолжает функционировать с ухудшенными временными и стоимостными характеристиками его реализации.

Устройства оборудования обладают некоторым ресурсом выполнения своих функций, который постепенно уменьшается и зависит от времени активного использования устройства. Специалисты по надежности обычно оперируют понятием «время наработки устройства на отказ» (T_{TTF}). При достижении значения времени активного использования устройства этого порогового значения вероятность отказа резко возрастает, и поэтому на практике стремятся либо переключить устройство на резервное (с временем фактического использования устройства близким к нулю), либо выполнить профилактические работы с целью восстановления ресурса времени использования устройства.

Имеются также ТП, в которых появление аварии оборудования 3-го типа приводит к отказу функционирования всего ТП. Аварии третьего типа ликвидируются аналогично ликвидации аварии 2-го типа с использованием специального оборудования и специальных бригад исполнителей. Но нормальное функционирование ТП приостанавливается.

Таким образом, проектное моделирование ВТП в силу вероятностного характера запросов ресурсов множеством $\{MTCO_{ij}\}$ и наличия отказов оборудования, природа которых также вероятностная, представляет собой сложную научную задачу.

Подпункт 7.7.1.3 Параметры (переменные) управления технологическим циклом

Некоторые $MTCO_{ij}$ в ВТП используют не только ресурсы, но могут выполнять ряд контрольных функций за изменением значений компонентов множества переменных управления ВТП U_s . При нормальном выполнении ВТП каждый компонент этого множества должен находиться в допустимых диапазонах между минимальным U_s^- и максимальным значением U_s^+ s -го компонента множества переменных управления U_s . Причем, изменения значения U_s может иметь вероятностную природу. При выполнении другой группы $MTCO_{ij}$ значения переменных управления могут корректироваться таким образом, чтобы U_s возвращалось в допустимые пределы ($U_s^- \leq U_s \leq U_s^+$). Третья группа $\{MTCO_{ij}\}$ корректирует значения компонентов U_s . Некоторые $MTCO_{ij}$ используются только для индикации состояний оборудования и ВТП.

По способу использования управляющих переменных можно разделить $\{MTCO_{ij}\}$ на следующие группы:

- индикации состояний ТП (используют минимальное число ресурсов);
- исполнительные элементы (не контролируют и не меняют компоненты U_s);
- контролирующие выход U_s за допустимые диапазоны;
- восстанавливающие значения компонентов U_s в отведенные диапазоны их изменения.

В составе ВТП контролирующие $MTCO_{ij}$ могут изменяться вероятностным образом: выход за допустимые пределы (P_{thr}) на величину (ΔU_s), которая в ту или в другую сторону может изменяться также вероятностным образом.

Подпункт 7.7.1.4 Математическая модель микротехнологической операции

Состав ресурсов, которые может использовать каждая $MTCO_{ij}$, включает десять типов:

1. Устройство оборудования индивидуального пользования
2. Устройство оборудования общего пользования
3. Ресурс индивидуального пользования
4. Ресурс общего пользования
5. Индивидуальные исполнители
6. Бригады исполнителей
7. Материалы
8. Комплектующие

9. Стоимость выполнения операций
10. Время выполнения операций

Устройства оборудования индивидуального использования ($DEVBIN_r$) могут использовать какой-либо состав операций на время ее выполнения τ_{ij} . Состав устройств общим количеством может задаваться либо персональным списком, либо $MTCO_{ij}$ требуется для ее выполнения только число таких устройств, которые свободны на момент запроса операций ресурсов первого типа ($r = 1$)

$MTCO_{ij}$ может затребовать для своего выполнения устройство оборудования общего пользования (n_2) специальным списком. На устройствах общего пользования номера r выделяет место для ij -й операции размером V_{r2} на время ее выполнения, которое после окончания операции свободно для выделения другой операции. V_{r2} является случайной величиной. Допускается конкуренция $\{MTCO_{ij}\}$ за устройства общего пользования и место на этих устройствах.

Первые два типа ресурсов, являясь устройствами оборудования, могут отказывать в функционировании при их использовании $MTCO_{ij}$.

$MTCO_{ij}$ может потребовать для своего выполнения ресурсы индивидуального пользования ($r = 3$) общим количеством n_3 и ресурсы общего пользования в количестве n_4 ($r = 4$). Ресурсы ($r = 3$) и ($r = 4$) могут задаваться либо списком, либо для случая, когда номер ресурса не имеет значения, используются первые свободные n_3 или n_4 ресурсов. От ресурсов ($r = 1$) и ($r = 2$) эти ресурсы отличаются только тем, что это абсолютно надежные устройства, которые выделяются в распоряжение $MTCO_{ij}$ на время ее выполнения, а затем снова перераспределяются между $\{MTCO_{ij}\}$, либо на принципах конкуренции их за эти ресурсы, либо специальным образом зарезервированными за операциями место на общем ресурсе ($r = 4$) также может выделяться нескольким операциям и величина запроса является случайной величиной, которая задается с помощью функции распределения $F_{6ij}(V_{r4})$.

Любая $MTCO_{ij}$ может требовать конкретное число исполнителей индивидуальных (n_5) и бригад исполнителей (n_6). Состав бригад определяется списком размера (n_6). Эти запросы являются детерминированными величинами.

Ресурсы 7-10 имеют вероятностную природу. В общем случае для выполнения операции $MTCO_{ij}$ расходуется k_1 -го типа материалы (mt_{k1}) и комплектующие k_2 -го типа (prt_{k2}). Количество таких вероятностных ресурсов равны соответственно n_{7k1} и n_{8k2} и представляют собой детерминированные величины. Расход количества материалов определяется с помощью функций распределения $F_{4k2ij}(mt)$ и расход количества комплектующих также задается с помощью соответствующих функций распределения $F_{3k2ij}(prt)$.

Для нормального выполнения операции $MTCO_{ij}$ требуется расход ресурсов $r = 9$ и $r = 10$ финансовых затрат и времени выполнения операции. Обе эти характеристики являются случайными величинами и поэтому запросы этих ресурсов задаются соответственно с помощью функций вероятности распределения их значений $F_{2ij}(C)$ и $F_{1ij}(\tau)$.

Подпункт 7.7.1.1.5 Модель функционирования устройств оборудования

Устройства оборудования используются двух типов: индивидуального использования ($DEVIN_{r1}$) номера r_1 и общего пользования ($DEVSHR_{r2}$) номера r_2 . Для устройства $DEVSHR_{r2}$ имеется ограничение места, которые разные $MTCO_{ij}$ могут одновременно использовать размером V_{0r2} , задаваемым в начале имитации. За каждый $MTCO_{ij}$ резервируется определенное место V_{r2ij} , при этом всегда проверяется, чтобы сумма используемых мест на $DEVSHR_{r2}$ была бы меньше V_{0r2} . В противном случае формируется отказ $MTCO_{ij}$ в использовании $DEVSHR_{r2}$, что приводит к ожиданию появления места в случае завершения предыдущей $MTCO_{ij}$.

Устройства оборудования обоих типов могут отказывать в функционировании. При этом они имеют надежностные характеристики вероятностной природы. Для каждого времени нахождения устройства r в состояниях задаются функции вероятностей распределения значений времени:

- безотказного функционирования $\Phi_{1r}(\tau_{NOr})$
- восстановление работоспособности после простого отказа $\Phi_{2r}(\tau_{ROr})$
- ликвидации аварии 1-го типа $\Phi_{3r}(\tau_{em1})$;
- ликвидации аварии 2-го типа $\Phi_{5r}(\tau_{em2})$

При появлении аварий время использования оборудования (а следовательно и выполнение операции) возрастает и определяется с помощью функций вероятностей распределения значений стоимости:

- ликвидации аварии 1-го типа $\Phi_{4r}(Cem1)$;
- ликвидации аварии 2-го типа $\Phi_{6r}(Cem2)$

Задаются также при появлении отказов оборудования вероятности возникновения:

- ликвидации аварии 1-го типа (P_{em1});
- ликвидации аварии 2-го типа (P_{em2})

Таким образом, время выполнения на оборудовании r номера операции $MTCO_{ij}$ равно $1 - (P_{em1} + P_{em2}) = P_{5A}$. Каждое r -е устройство оборудования обладает пороговым значением времени наработки t_{optr} .

Подпункт 7.7.1.6 Математическая модель вероятностного технологического процесса

При реализации последовательных вероятностных технологических процессов (ВТП1) только одна из $MTCO_{ij}$ выполняется в текущий момент времени. При этом она может использовать все ресурсы предприятия или же эти ресурсы заранее распределены между операциями до начала реализации ВТП1. Поэтому в последовательных ВТП используются только устройства индивидуального пользования. Кроме того, в ВТП1 имеется несколько нестандартных $MTCO_{ij}$:

- одиночного оперативного резервирования устройств оборудования в тех случаях, когда фактическое время наработки на отказ t_{n1} приблизится к критическому значению времени наработки $T_{N_{crit}}$;
- групповое резервирование устройств оборудования, если время наработки на отказ достигает критической величины одновременно у группы устройств;
- проведение профилактического ремонта всех устройств оборудования, когда процент устройств, требующих резервирования очень высок и при этом допустима остановка выполнения ВТП1;
- ликвидации аварий на устройствах для тех случаев, когда режимы резервирования и профилактики не удается заблаговременно выполнить.

система управления

\coloneqq [СУ]
 \coloneqq [строго определённый набор программно-аппаратных средств для управления подконтрольным объектом, обеспечивающий возможность сбора показаний о его состоянии и воздействий на его поведение для достижения заданных целей]

автоматизированная система управления технологическим процессом

\coloneqq [АСУТП]
 \subset система управления
 \coloneqq [комплекс технических и программных средств, обеспечивающих работу технологического оборудования в автоматическом (или автоматизированном) режиме в соответствии с выбранным критерием управления]

Вероятностный технологический процесс ВТП2 обычно выполняется с высокой скоростью. Поэтому имитационная модель не успевает вмешиваться в динамику его реализации. Для этой цели имеется система управления (СУ), которая с помощью специального оборудования управляет его функционированием. Здесь также используется только оборудование индивидуального использования. Но это оборудование может выходить из строя. Поэтому крайне важно обеспечить своевременное резервирование устройств оборудования для предотвращения отказов и аварий в ВТП2. Состав и структура СУ ВТП2 известны технологу до имитации, и обычно это логические схемы, имеющие графовую структуру.

Вероятностный технологический процесс с параллельно-последовательной организацией (ВТП3) обычно имеет довольно низкую скорость реализации. Кроме того, эксперту-технологу известна технологическая схема реализации ВТП3. Как правило, группы выполняются одновременно, однако сохраняется последовательный характер выполнения каждой $MTCO_{ij}$. Это означает, что внутри технологической схемы ВТП3 имеются устройства синхронизации типа «И», которые срабатывают по последнему во времени приходу сигналов активизации, после завершения предыдущих $MTCO_{ij}$. Эти сигналы имеют сложную структуру за счет их информационной «подкраски» признаками π_{emij} (наличие аварии на оборудовании при выполнении $MTCO_{ij}$) и π_{uij} (наличие выхода компонентов вектора управления U_k за допустимые пределы). После прихода последнего во времени сигнала на схему совпадения «И» на выходе формируется множество сигналов, активизирующих другую группу $\{MTCO_{ij}\}$. Реализация ВТП3 начинается с первой $MTCO_{11}$ и завершается последней схемой совпадения «И». Все схемы синхронизации пронумерованы от 1 до n , а $MTCO_{ij}$ имеет двойную нумерацию (i номер предыдущей схемы синхронизации, а j номер последующей схемы синхронизации). Число входов и выходов у схем синхронизации может быть любым. Структура ВТП3 определяется графом связи $MTCO_{ij}$, который задается с помощью таблицы коммутации операций в ВТП3.

В общем случае ВТП3 может использовать все ресурсы и оборудование в режиме конкуренции. Поэтому некоторые $MTCO_{ij}$ могут ожидать освобождения оборудования, а затем начинается имитация их выполнения. Связь между $MTCO_{ij}$ и собственно устройствами ВТП3 может осуществляться либо через специально выделенное оборудование индивидуального пользования, либо через любое из устройств оборудования, выделяемого $MTCO_{ij}$ на основе конкуренции. . Важной особенностью ВТП3 является возможность регулирования технологического процесса с помощью контроля за содержимым управляющих переменных и модификации значений компонентов вектора переменных управления $\{U_k\}$ с целью возврата их в отведенные для них диапазоны значений. Выполнение этой

коррекции осуществляется специальными резервными $MTCO_{ij}$, регулирующими значения $\{U_k\}$. Кроме того, ведется контроль за надежностью выполнения оборудованияем своих функций. После ликвидации аварий оборудования в технологической цепи предусмотрены резервные $MTCO_{ij}$, которые ликвидируют последствия аварий на оборудовании. Наличие аварий оборудования и задержка активизации $MTCO_{ij}$ из-за нехватки ресурсов приводит к общему увеличению времени и стоимости выполнения всего множества $MTCO_{ij}$. Поэтому важным откликом ИМ ВТПЗ является время выполнения $MTCO_{ij}$ за один цикл имитации.

Параметрами моделирования являются начальное значение количества ресурсов, предоставленное в распоряжение всех $MTCO_{ij}$, и количество устройств оборудования, с помощью которых реализуется технологический цикл. Откликами ИМ ВТПЗ являются суммарные расходы ресурсов вероятностного типа на один цикл имитации, а также коэффициенты использования ресурсов ВТПЗ η_r . Статистиками имитации являются вектора значений удельных весов времени свершения событий в технологической схеме ВТПЗ.

Если ВТПЗ является циклическим, то важными статистиками становятся среднее время цикла выполнения всего множества $MTCO_{ij}$ до завершения имитации функционирования ВТПЗ и минимально допустимый состав ресурсов и оборудования, при котором возможна его реализация.

Подпункт 7.7.1.1.7 Имитационное моделирование технологических процессов

Состав и функции операторов алгоритма имитации $MTCO_{ij}$ включают в себя:

1. Начало активизации имитации операции - Интервал времени - A_0 , Время - $t_{a_{ij}}$
2. Формирование заказа ресурсов - Интервал времени - A_1 , Время - $t_{1_{ij}}$
3. Захват свободных ресурсов на время выполнения операций - Интервал времени - A_2 , Время - $t_{2_{ij}}$
4. Ожидание выделения освободившегося ресурса - $\tau_{w_{ij}}$ - Интервал времени - A_3
5. Захват освободившегося ресурса - Интервал времени - A_4 , Время - $t_{3_{ij}}$
6. Ожидание следующего освободившегося ресурса - $\tau_{w2_{ij}}$ - Интервал времени - A_5
7. Захват последнего из затребованных устройств на время выполнения - Интервал времени - A_6 , Время - $t_{4_{ij}}$
8. Выполнение имитации операции - $\tau_{r2_{ij}}$ - Интервал времени - A_7
9. Конец выполнения операции, возврат захваченных ресурсов - Интервал времени - A_8 , Время - $t_{5_{ij}}$
10. Формирование информации о характере использования оборудования - Интервал времени - A_9 , Время - $t_{9_{ij}}$
11. Формирование информации о модификации значений компоненты вектора управления U_k - Интервал времени - A_{10} , Время - $t_{7_{ij}}$
12. Формирование адресата продолжения операции - Интервал времени - A_{11} , Время - $t_{8_{ij}}$

Надежностные характеристики функционирования устройств оборудования включают в себя:

1. Время безотказного функционирования - с распределением $\Phi_{1r}(\tau_{NO_r})$
2. Время восстановления работоспособности - с распределением $\Phi_{2r}(\tau_{RO})$
3. Вероятность появления аварии 1-го типа - с вероятностью P_{em1}
4. Время ликвидации аварии 1-го типа - с распределением $\Phi_{3r}(\tau_{em1})$
5. Стоимость ликвидации аварии 1-го типа - с распределением $\Phi_{4r}(C_{em1})$
6. Вероятность появления аварии 2-го типа - с вероятностью P_{em2}
7. Время ликвидации аварии 2-го типа - с распределением $\Phi_{5r}(\tau_{em2})$
8. Стоимость ликвидации аварии 2-го типа - с распределением $\Phi_{6r}(C_{em2})$
9. Время наработки устройства на отказ - T_{optr}

Состав и структура операторов алгоритма имитации функционирования устройств оборудования индивидуального пользования включает в себя:

1. Начало активизации устройства r - оператор B_0 - Момент времени - t_{ar}
2. Розыгрыш по функции распределения τ_{NOK} - оператор B_1 - Момент времени - t_{ar}
3. Определение типа использования оборудования - оператор B_2 - Момент времени - t_{1r}
4. Имитация нормального использования устройства длительностью - Интервал времени - τ_{ij} - оператор B_3 τ_{ij} - Момент времени - t_{1r}
5. Формирование признака $\pi_\phi := 0$ - оператор B_4 - Момент времени - t_{1r}
6. Активизация операции ij - оператор B_5 - Момент времени - t_{1r}
7. Розыгрыш по функции распределения τ_{ROK} - оператор B_6 - Момент времени - t_{1r}
8. Имитация восстановления работоспособности устройств - оператор B_7 - Интервал времени - τ_{ROr}
9. Розыгрыш по P_{em1} ситуации появления аварии 1-го типа - оператор B_8 - Момент времени - t_{2r}
10. Имитация повторения имитации - оператор B_9 - Интервал времени - τ_{ij}
11. Формирование признака $\pi_{emk} := 0$ - оператор B_{10} - Момент времени - t_{3r}
12. Розыгрыш по P_{em2} ситуации появления аварии 2-го типа - оператор B_{11} - Момент времени - t_{2r}
13. Розыгрыш по функции распределения - Интервал времени - τ_{em1} - оператор B_{13}
14. Имитация ликвидации аварии 1-го типа - оператор B_{14} - Интервал времени - τ_{em1}
15. Имитация повторения операции - оператор B_{15} - Интервал времени - τ_{ij}

16. Формирование признака $\pi_{emk} := 1$ и активизация операции - оператор B_{16} - Момент времени $-t_{4r}$
17. Розыгрыш по функции P_{AV} ситуации появления аварии 2-го типа - оператор B_{17} - Момент времени $-t_{2r}$
18. Активизация последовательности процедур ликвидации сложной аварии и остановов - оператор B_{18} - Момент времени $-t_{2r}$
19. Ожидание завершения ликвидации аварии 2-го типа - оператор B_{19} - Интервал времени - τ_{w2r}
20. Формирование признака $\pi_{emk} := 1$ и активизация операции - оператор B_{20} - Момент времени - t_{5r}

Состав и функции операторов алгоритма имитации функционирования устройств общего пользования включают в себя:

1. Активизация устройства - оператор - C_0 - Момент времени - t_{ar}
2. Определение типа использования устройства - оператор - C_1 - Момент времени - t_{ar}
3. Имитация нормального использования длительностью C_2 - Интервал времени - τ_{ij}
4. Формирование признака $\pi_{ak} := 0$ и переход - оператор - C_3 - Момент времени - t_{1r}
5. Розыгрыш по функции распределения τ_{ROk} - оператор - C_4 - Момент времени - t_{ar}
6. Имитация восстановления работоспособности устройств - оператор - C_5 - Интервал времени - τ_{ROk} - Момент времени - t_{2r}
7. Розыгрыш по P_{em1} ситуации появления аварии 1-го типа - оператор - C_6 - Момент времени - t_{2r}
8. Имитация появления аварии - оператор - C_7 - Интервал времени - τ_{ij}
9. Формирование признака $\pi_{ak} := 0$ и переход на C_{19} - оператор - C_8 - Момент времени - t_{3r}
10. Розыгрыш по P_{em2} ситуации появления аварии 2-го типа - оператор - C_9 - Момент времени - t_{3r}
11. Розыгрыш по функции распределения τ_{em2} - оператор - C_{10} - Момент времени - t_{3r}
12. Имитация ликвидации аварии 2-го типа C_{11} - Интервал времени - τ_{em2}
13. Формирование признака $\pi_{emk} := 1$ и переход на C_{19} - оператор - C_{13} - Интервал времени - τ_{ij}
14. Розыгрыш по P_{em2} ситуации появления аварии 2-го типа - оператор - C_{15} - Момент времени - t_{2r}
15. Активизация последовательности процедур ликвидации сложной аварии и остановов C_{16} - Момент времени - t_{2r}
16. Ожидание завершения ликвидации аварии 2-го типа - оператор - C_{17} - Интервал времени - τ_{w2k}
17. Формирование признака $\pi_{emr} := 1$ - оператор - C_{18} - Момент времени - t_{5r}
18. Розыгрыш по функции распределения τ_{NOk} нового значения интервала безотказного функционирования - оператор - C_{19} - Момент времени - t_{5r}
19. Активизация операции - оператор - C_{20} - Момент времени - t_{5r}

В таблице 1.2 приведены состав и функции операторов алгоритма имитации $MTCO_{ij}$. В момент активизации $MTCO_{ij}$ (t_{a1j}) выполняется оператор "начало активизации имитации" операции (A_0). С помощью оператора A_1 формируется заказ ресурсов для имитации выполнения $MTCO_{ij}$. Для этой цели используются функции распределения вероятностей их значений, приведенные в таблице 1.1. В итоге выполнения оператора формируется вектор детерминированных запросов ресурсов

$$(n_1, n_2, n_3, n_4, n_5, n_6, n_{7k1}, n_{8k2})$$

и подмножество конкретных значений вероятностных характеристик запроса ресурса

$$(\{V_{r2l}\}, \{V_{r4l}\}; \{mt_{k1}\}, \{kok2\}, c_{ijl}; \tau_{ijl})$$

Далее с помощью оператора захвата ресурсов A_2 на время выполнения операции осуществляется резервирование ресурсов, представленных множествами (1.1) и (1.2). В некоторых случаях из-за конкуренции $MTCO_{ij}$ за ресурсы может не оказаться свободных ресурсов или же места на ресурсах или устройствах. В этом случае с помощью оператора A_3 осуществляется ожидание освобождения ресурса длительностью τ_{w1ij} . По мере освобождения ресурсов осуществляется захват частично освободившихся ресурсов с помощью оператора A_4 . Далее с помощью оператора A_5 операция $MTCO_{ij}$ ожидает освобождения следующего ресурса длительностью τ_{w2ij} . С помощью оператора A_6 имитируется захват последнего из требованных ресурсов. И в момент t_{4ij} начинается с помощью оператора A_7 имитация выполнения операции. Имитируется запуск всех устройств оборудования на заказанное время выполнения операции τ_{lij} сформированное с помощью функции распределения $F_{1ij}(\tau)$. Имитация выполнения операции продолжается до тех пор, пока все устройства оборудования не завершат имитацию выполнения операции. Из-за отказов устройств оборудования и ликвидации аварий оборудования фактическое время имитации операций может быть больше, чем заказанное время имитации $\tau_{zlij} \geq \tau_{ij}$.

С помощью оператора A_7 в момент времени t_{a1j} по окончании имитации операции все захваченные $MTCO_{ij}$ ресурсы возвращаются системе. В этот момент возможно продолжение тех $MTCO_{ij}$, которые ждут освобождения ресурсов. По информации, поступившей от устройств оборудования о том, что имела место авария оборудования на хотя бы одном из устройств, формируется признак аварии ($\pi_{em} := 1$) оператором A_9 .

С помощью оператора A_{10} формируется признак модификации значений компонентов вектора переменных управления за допустимые пределы (т.е. $U_k \prec U_k^-$ или $U_k \succ U_k^+$) $\pi_u := 1$, в случае если $MTCO_{ij}$ является операцией

изменения $\{U_k\}$. Для случая, когда $MTCO_{ij}$ является резервной операцией возврата переменных управления в допустимые пределы с помощью оператора A_{10} осуществляется обратная модификация $\{U_k\}$ таким образом, чтобы $U_k \leq U_k \leq U_k^+$ в момент времени t_{7ij} . Наконец, с помощью оператора A_{11} формируется адресат продолжения операции. При этом формируется сигнал сложного вида (информационно «подкрашенный» значениями признаков π_{aij} и π_{uij}), которые согласно таблице коммутации $MTCO_{ij}$ в составе технологической схемы ВТП посылаются на одно из устройств синхронизации типа «ИЛИ» и «И». Эти устройства имеют нумерацию входов и поэтому в адресате продолжения операции внутри сигнала указывается номер и номер входа устройства синхронизации.

Возможны четыре случая имитации операции на оборудовании:

- нормальное выполнение операции на всем оборудовании длительностью τ_{lij} ;
- имеет место восстановление функций оборудования хотя бы на одном из устройств $\tau_{2ij} > \tau_{lij}$;
- имела место авария 1-го типа на одном из устройств оборудования $\tau_{3ij} > \tau_{2ij} > \tau_{lij}$;
- имела место авария 2-го типа на любом из устройств оборудования $\tau_{4ij} > \tau_{3ij} > \tau_{2ij} > \tau_{lij}$.

Таким образом, наличие отказов и аварий устройств оборудования, а также ожиданий освобождения ресурсов в ряде случаев может существенно увеличить время выполнения операций по сравнению с разыгранным по функции распределения $F_{1ij}(\tau)$ с помощью жребия 3-го типа.

Это означает, что аварии оборудования приводят только к увеличению времени и стоимости выполнения $MTCO_{ij}$ и, как следствие, к росту общего времени и стоимости выполнения ВТП.

В процессе имитации функционирования устройства оборудования фиксируется статистика фактического времени наработки устройства по формуле

$$t_{optr} := t_{optr} + \tau_{ij}$$

При достижении t_{optr} пороговой величины T_{opt} вероятность отказа устройства настолько возрастает, что при следующем использовании устройства r возможен отказ функционирования устройства, который может привести к появлению аварий на устройстве r во время выполнения очередной операции $MTCO_{ij}$.

Одним из способов предотвращения ситуации может служить либо переключение на резервные устройства, либо проведение профилактики. После проведения одной из этих операций фактическое время наработки устройства устанавливается в нуль $t_{optr} := 0$. Пороговое значение времени наработки указывается в таблице 1.3 для каждого r -го устройства оборудования.

В таблице 1.4 приведен состав и функциональное назначение операторов алгоритма имитации функционирования устройств оборудования индивидуального пользования.

В момент активации устройства номера r операцией $MTCO_{ij}$ (t_{ar}) с помощью оператора B_0 осуществляется запись в базу данных конкретного номера устройства оборудования индивидуального пользования. Используя таблицу 1.3, с помощью функции распределения $\Phi_{1r}(\tau_{NOr})$ разыгрывается конкретное значение интервала безотказного функционирования устройства τ_{lrNOK} .

Далее с помощью оператора B_2 в момент t_{2r} определяется тип использования оборудования. Если выполняется неравенство $\tau_{lij} \leq \tau_{lrNOK}$, то это означает нормальное использование устройства оборудования (без отказов). В подобном случае с помощью оператора B_4 формируется признак $\tau_{emr} := 0$, означающий отсутствие аварий.

С помощью оператора B_5 активизируется $MTCO_{ij}$, и имитация выполнения завершается с передачей информационного сигнала с признаком наличия аварии ($\pi_{emr} := 0$).

Если же $\tau_{lij} \geq \tau_{lrNOr}$, то это означает случай 2 (наличие отказа оборудования). В этом случае с помощью оператора B_6 в момент t_2 по функции распределения $\Phi_{2r}(\tau_{ROr})$ формируется значение интервала восстановления работоспособности τ_{RORi} . Затем осуществляется имитация восстановления работоспособности r -го устройства с помощью оператора B_7 . По завершении этой имитации по вероятности P_{em1r} с помощью жребия 1-го типа моделируется появление аварии с помощью оператора B_8 в момент ($t_{2r} > \tau_{ROr}$). Далее операция повторяется с помощью оператора B_9 , который имитирует ее выполнение длительностью τ_{lij} . Эта ситуации означает отсутствие аварий, поэтому с помощью оператора B_{10} формируется признак $\pi_{em} := 0$. Далее с помощью оператора B_{11} по вероятности P_{em1} разыгрывается ситуация появления аварии 1-го типа. Разыгрывается по функции распределения длительности ликвидации аварий с помощью оператора B_{13} и осуществляется имитация ликвидации аварий 1-го типа длительностью τ_{lem1} оператором B_{14} . Далее аналогично имитируется повторение операции длительностью τ_{lij} с помощью оператора B_{19} . По завершении этой имитации оператор B_{16} формирует признак «была авария» $\pi_{emr2} := 1$ и активизирует операцию $MTCO_{ij}$. В противном случае с помощью оператора B_{17} в момент t_{2r} по вероятности P_{a2} с помощью жребия 1-го типа формируется ситуация «наличие аварии» 2-го типа. С помощью оператора B_{18} активизируется начальный элемент последовательности процедур ликвидации сложной аварии. Само же устройство r_1 останавливается и с помощью оператора B_{20} имитируется ожидание завершения ликвидации аварии 2-го типа длительностью τ_{w2r} . Далее с помощью оператора B_{20} активизируется устройство оборудования типа 2. Устанавливается признак $\pi_{emr} := 1$, активизируется операция $MTCO_{ij}$ и устройство останавливается, завершая имитацию выполнения операции на устройстве в момент t_{5r} .

Может иметь место 4 случая:

- нормальное выполнение операции длительностью ($\tau_{\phi 1ij} = \tau_{ij}$) ;
- автоматическое восстановление отказа без аварии на устройстве оборудования r_1 длительностью ($\tau_{\phi 2ij} > \tau_{ij}$) ;
- ликвидация аварии 1-го типа длительностью ($\tau_{\phi 3ij} > \tau_{\phi 2ij} > \tau_{ij}$) ;
- наличие имитации аварии на оборудовании r_1 длительностью ($\tau_{\phi 4ij} \gg \tau_{ij}$) .

Оборудование общего пользования имитируется по более сложному алгоритму. Надежностные характеристики устройств оборудования общего использования задаются аналогичным образом с помощью таблицы 1.3.

В таблице 1.5 приведен состав функций операторов алгоритма имитации функционирования устройств общего пользования. Основное отличие состоит в том, что в моменты завершения интервалы безотказного функционирования начинаются с начала имитации, поскольку это устройство функционирует непрерывно. На рисунке 1.3 приведена временная диаграмма имитации выполнения алгоритма устройства оборудования общего пользования номера r_2 . Здесь также могут быть четыре случая:

- нормальное выполнение длительностью τ_{ij} ;
- наличие восстанавливаемого отказа длительностью ($\tau_{ROLr2} + 2\tau_{ij}$);
- ликвидация аварии 1-го типа длительностью ($\tau_{ROLr2} + \tau_{em1r2} + 2\tau_{ij}$);
- ликвидация аварии 2-го типа длительностью ($\tau_{ROLr2} + \tau_{O2r} + 2\tau_{ij}$).

Фактическое время выполнения операции возрастает с ростом сложности отказа оборудования. Сам же алгоритм имитации функционирования устройств общего пользования похож на алгоритм имитации функционирования устройств оборудования индивидуального пользования.

Вероятностный переход от одной к другой $MTCO_{ij}$ осуществляется элементом выбора операции. При этом он проверяет наличие критической ситуации устройств оборудования и своевременно активизирует нестандартные $MTCO_{ij}$. После ликвидации нестандартной ситуации у устройств оборудования осуществляется переход на выполнение стандартных $MTCO_{ij}$ согласно матрице вероятностей перехода $\| P_{ij} \|$, где i - номер предыдущей, а j - номер последующей $MTCO_{ij}$. Очевидно, что элемент выбора операции ресурсов не использует, поэтому это переключение осуществляется мгновенно в модельном времени. Поскольку ВТП1 может быть циклическим, то он может повторяться многократно, начиная с начальной $MTCO_{1j}$ и заканчивая последней $MTCO_{in}$.

В качестве исходной информации задаются информация, представленная таблице 1.3, а также начальное количество ресурсов, представляемые в распоряжение всех $MTCO_{ij}$. Для всех устройств оборудования указывается время наработки на отказ каждого устройства. Откликами ИМ ВТП1 являются среднее время цикла выполнения ВТП1 ($T_{\mu 1}$) и вектор коэффициентов использования ресурсов ВТП1 (η_k). Статистиками имитации являются вектора удельных весов времени и суммарной стоимости выполнения каждой ($z_r \tau_{ij}$ и $z_r c_{ij}$).

Подпункт 7.7.1.8 Формализация систем управления на основе агрегатного способа имитации

Состав и структура СУ ВТП2 известны технологу до имитации, и обычно это логические схемы, имеющие графовую структуру.

На входе СУ может быть несколько TCO_i , из которых в СУ поступают сигналы активизации ее компонентов с интенсивностями λ_i .

Анализ особенностей СУ ВТПП позволяет установить что имитационное моделирование взаимодействия элементов систем управления и компонентов оборудования вероятностных процессов производства можно осуществить на основе блок-схем синхронизации их функционирования. Элементами такой синхронизации являются:

- синхронизатор первого типа $SLAST_k$ ($k = \overline{1, N}$) взаимодействия нескольких микротехнологических операций $MTCO_{ij}$ (i и j - номера синхронизаторов соответственно на входе и выходе $MTCO_{ij}$), функционирующий по логической схеме совпадения «и»;
- синхронизатор второго типа $SFIRST_k$ ($k = \overline{1, N}$) взаимодействия нескольких $MTCO_{ij}$, функционирующий по логической схеме совпадения «или»;
- элементы $INDS_j$ СУ ВТПП являющиеся инициаторами цикла его функционирования с интенсивностью λ_j и формирования воздействий на вероятностный процесс через устройства оборудования (здесь j – номера элементов-синхронизаторов $MTCO_{ij}$, которые инициализируются данными элементами)
- элементы $INDF_i$ СУ, на которых завершаются цепочки управления выполнением функций ВТПП (где i – номер индикатора предшественника данному элементу)
- исполнитель $ISPF_{ij}$ функциональных действий микротехнологической операции $MTCO_{ij}$, инициируемый синхронизатором номера i и посылающий сигнал синхронизатора на синхронизатор номера j ;
- исполнитель $CORF_{ij}$ функциональных действий, корректирующий вектор глобальных переменных U_k при выходе за границы допустимых значений его компонентов; здесь i и j – номера элементов синхронизации соответственно на входе и выходе $CORF_{ij}$.

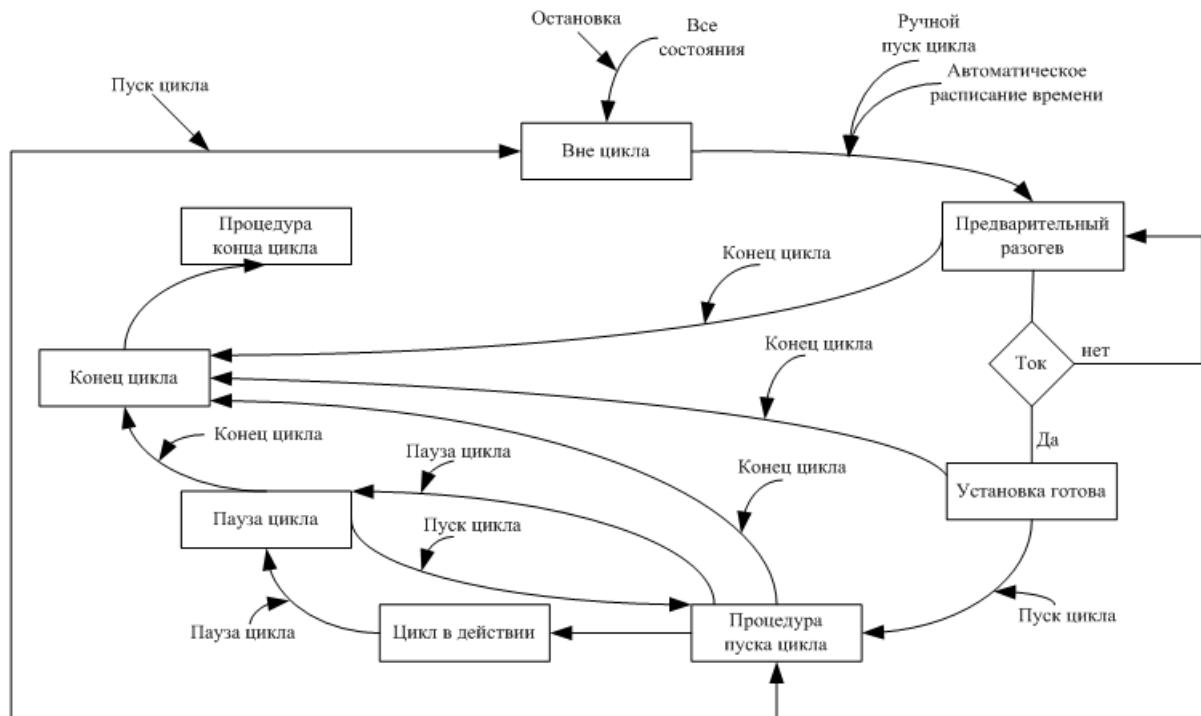
- исполнитель LIQ_{ij} функциональных действий, ликвидирующий последствия аварии оборудования ВТПП, имевшей место перед его выполнением, и приводящий СУ ВТПП в нормальное состояние (i и j – номера элементов синхронизации соответственно на входе и выходе LIQ_{ij})
- универсальный элемент-исполнитель $UNIV_{ij}$, который может одновременно корректировать значения векторов U_k и ликвидировать последствия аварий на оборудовании ВТПП;
- индикатор состояний оборудования $INDS_{ij}$ вероятностного процесса; в отличие от остальных элементов управляющих воздействий на ВТПП не посылает.

Взаимодействие исполнительных элементов с вероятностным процессом осуществляется с помощью множества устройств оборудования индивидуального ($DEVIN_{r_1}$) и общего ($DEVSHR_{r_2}$) пользования (r_1 и r_2 – номера устройств соответственно индивидуального и общего пользования). Исполнительные элементы СУ ВТПП с индексом ij представляют собой двухполюсники, на входах которых приходят сигналы из элементов синхронизации $SLAST_i$ и $SFIRST_j$, а на выходах формируются сигналы, поступающие на аналогичные элементы синхронизации с номером j . Связи между элементами СУ осуществляются с помощью комбинации сигналов Sgn_{ij} сложной структуры, каждый из которых идентифицируется следующими характеристиками:

- индексом ij , показывающим направление его передачи (от i -го к j -му исполнительному элементу);
- булевым признаком pt_{ij} прихода сигнала в момент времени t_{ij} , принимающим значение $pt_{ij} = 1$ в момент прихода; в остальные моменты модельного времени t_{mod} (для $t_{mod} \leq t_{ij}$, $pt_{ij} = 0$) элемент синхронизации находится в состоянии ожидания сигнала;
- признаком ps_{ij} типа сигнала, принимающим одно из четырех значений (00 – нормальное выполнение исполнительного элемента синхронизации; 01 – во время выполнения исполнительного элемента имела место ликвидация аварии на устройствах оборудования; 10 – произошел выход компонентов переменной управления U_j за допустимые границы диапазона значений; 11 – обе чрезвычайные ситуации имели место одновременно).

Для выполнения исполнительного элемента вероятностного процесса с индексом ij в общем случае требуются затраты 10 типов ресурсов системы (см. пункт !!!) Под нормальным выполнением исполнительного элемента понимается случай, когда во время выполнения операции управления не происходит отказов оборудования индивидуального и общего пользования.

Подпункт 7.7.1.1.9 Пример построения технологического процесса производства в рамках Industry 4.0



= Пример схемы технологического цикла производства (линия эмалирования)

Исходная информация для построения ИМ СУ ВТПП включает в себя множество параметров, характеризующих ее состав и структуру:

- количество реализаций имитационной модели (N) согласно процедуре Монте-Карло;
- множество $\{X_i\}$ запросов ресурсов ВТПП, требуемых для выполнения исполнительных элементов системы управления;
- параметры коммутации исполнительных элементов с элементами синхронизации, генераторами входных сигналов и финальным элементом функционирования;
- множество $\{REL\}$ надежностных характеристик оборудования, с помощью которого осуществляется управление функционированием ВТПП;
- множество $\{\lambda_j\}$ значений компонентов важности откликов имитации;
- вектор $\{\delta_h\}$ коэффициентов важности откликов имитации;
- вектор DEVLIST конфигурации состава оборудования, используемого в исследуемом варианте организации системы управления ВТПП.

Для заполнения этих параметров производится мониторинг выполнения функций СУ ВТПП. Результаты обработки заполненных данных аппроксимируются некоторым распределением.

Объектом моделирования является технологическая схема организации производства для изготовления элементов электроаппаратуры специального назначения, которая имеет графовую структуру и поэтому может быть формализована в рамках разработанной модели.

Пункт 7.7.1.2. Решатели задач ostis-системы адаптивного управления вероятностным технологическим процессом производства

При осуществлении процессов управления технологическим циклом производства часто возникает потребность комплексного учета многообразия факторов воздействия на технологический процесс случайных сбоев используемого оборудования, а также воздействий внешней среды, включая и человеческий фактор. В этой связи является актуальной реализация адаптивного управления процессом производства (с обратными связями по управлению) в составе технических средств управления технологическим циклом и программного обеспечения адаптации процесса управления на случайные внешние возмущения в ходе его реализации.

Адаптивное управление технологическим циклом понимается как способность системы управления адекватно реагировать на внешние возмущения и штатные управляющие воздействия изменением соответствующих параметров управления в процессе функционирования системы.

Под оптимальными управлением понимается формализованная нейронной сетью структура адаптивного управления технологическим циклом, построенная в опорных узлах вероятностной сетевой графовой структуры (СГС) или модели полумарковской сети (ПМС) в рамках заданного критерия качества.

Формализация процесса управления технологическим циклом производства с вероятностными характеристиками основана на использовании в структуре контура управления специальных сигналов и стандартных элементов, которые в дальнейшем участвуют в формировании регулирующих воздействий непосредственно на используемое оборудование либо выдачи рекомендаций по выполнению определенных действий операторами-людьми.

Данные регулирующие воздействия (исполнение данных рекомендаций) могут приводить к изменениям параметров функционирования системы. Воздействия формируются в моменты изменения состояния ТЦ.

Необходимость оперативного изменения параметров функционирования системы возникает вследствие наличия внешних факторов; человеческого фактора, существования возможности возникновения отказов оборудования и аварий.

Система управления ТП связана с оборудованием ТП посредством средств программно-аппаратного сопряжения и в процессе реализации ТП получает сигналы о его состоянии. Изменение состояния ТП приводит к изменению состояния системы управления.

Для построения контроллера системы управления могут быть использованы статистические данные функционирования ТЦ, собранные с применением соответствующих средств аппаратно-программного сопряжения, либо может быть осуществлено использование имитационного моделирования функционирования ТЦ на базе модели ТЦ, построенной в соответствии с изложенной онтологией для технологических процессов с вероятностной природой.

Подпункт 7.7.1.2.1 Проблемы адаптивного управления производственной деятельностью при разработке интеллектуальных компьютерных систем нового поколения

Современный анализ состояния разработок в области исследования управляемых производственных систем показывает, что проблема определения параметров функционирования подобных объектов исследования в режиме

реального времени возникает прежде всего при необходимости производства сложных технических изделий, требующих точности их изготовления и высокой производительности труда.

При этом в рамках решения многокритериальной задачи оптимизации управления предъявляются строгие требования к качеству и алгоритму выполнения производственного процесса, минимизации влияния человеческого фактора на качество реализации технологического цикла производства, исключению возникновения аварийных ситуаций техногенного характера. Подобная ситуация характерна для роботизированных производственных систем, работающих под управлением программно-аппаратного контроллера, который администрирует работу системы управления технологическим циклом в соответствии с заложенными программами.

Вместе с тем возникающие нештатные ситуации ввиду наличия сбоев оборудования, случайных внешних управляющих воздействий, включая и наличие человеческого фактора, приводят к отклонению параметров функционирования производственной системы от заданных, в связи с чем возникает необходимость корректировки параметров управления в режиме реального времени на основе моделей нейрорегуляторов, работающих в составе средств программного-аппаратного сопряжения с технологическим циклом производства.

Существующие специальные модели искусственного интеллекта, такие как нейронные сети, обладают уникальными свойствами, могут применяться в качестве универсальных аппроксиматоров, имеющих способность к обобщению. Эти особенности делают целесообразным применение моделей такого типа при решении сложных задач адаптивного управления.

Современное направление конвергенции работ в области создания интеллектуальных систем [golenkov mono] требует разработки соответствующего программного обеспечения с элементами когнитивных способностей на основе семантически совместимых технологий искусственного интеллекта. Данное направление включает в себя также создание компьютерных систем, обеспечивающих интеллектуализацию процессов принятия аналитических управлений решений, что напрямую связано с адаптацией процессов управления сложными динамическими системами (техническими объектами) в режиме реального времени, созданием семантически совместимых баз знаний в области анализа функционирования динамических систем и оптимизации функционирования сложных технических систем на их основе посредством создания открытого программного кода интеллектуальных компьютерных систем поддержки принятия решений.

Подпункт 7.7.1.2.2 Создание математической модели производственной системы в рамках концепции Industry 4.0

Внедрение концепции Industry 4.0 на промышленных предприятиях сопровождается построением единой онтологической модели производства, которая является ядром комплексного информационного обслуживания предприятия [savushkin1].

Основные принципы концепции Industry 4.0 [DesignPrinciples 2016]:

- **Взаимодействие.** Возможность взаимодействия устройств, датчиков, людей посредством Интернета вещей (IoT), Интернета людей (IoP), Интернета услуг (IoS).
- **Виртуализация.** Означает способность киберфизической системы контролировать физические процессы. Данные сенсоров проецируются на модель предприятия, включающую состояние всех киберфизических систем. В случае возникновения нештатной ситуации должна быть возможность уведомить оператора, предоставив ему информацию по ее устранению и обеспечению безопасности, тем самым осуществляя поддержку принятия решений персоналом.
- **Децентрализация.** Растущая потребность в штучных партиях заказных продуктов увеличивает сложность централизованного управления производством. КФС могут иметь встроенные вычислительные модули, позволяющие им принимать решения самостоятельно и переадресовывать задачу управляющей системе только в случае необходимости. Несмотря на это, необходимо обеспечить контроль качества конечного продукта и прослеживаемость, что требует централизованного управления. К примеру, необходимые шаги производственного процесса могут быть закодированы в RFID-метках, что освобождает от необходимости централизованного управления данным аспектом производства малых партий продукта.
- **Анализ и реагирование в реальном времени.** С целью управления производством необходимо, чтобы данные с сенсоров постоянно собирались и анализировались в режиме реального времени. В случае отказа одной производственной установки, можно "перепоручить" её задачу другой.
- **Ориентированность на услуги.** Услуги компаний, КФС и людей доступны в Интернете услуг и могут быть использованы другими участниками. Услуги могут предоставляться как внутри предприятия, так и другим предприятиям. КФС предоставляют свои услуги в виде веб-служб. Это позволит реализовать производство продукта путем комбинирования производственных операций в соответствии со спецификацией клиента, закодированной, например на RFID-метке.

- **Модульность.** Система должна быть гибкой, т.е. легко адаптируемой к меняющимся требованиям (например, сезонным изменениям в потреблении, изменению характеристик продукта или производства). Адаптация должна осуществляться заменой или расширением отдельных компонентов системы. Обеспечение совместимости компонентов требует наличия стандартизованных механизмов взаимодействия, позволяющих автоматически идентифицировать компоненты и включать их в интернет услуг. Полная совокупность производственной деятельности предприятия может быть описана в качестве набора технологических процессов производства с вероятностными характеристиками в соответствие с описанной онтологией данной предметной области.

Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS

В рамках решения задачи адаптивного управления в изложенной формализации может быть предложено несколько подходов:

- Управление с эталонной моделью, при котором путем обучения нейронной сети на базе существующих «оптимальных» сигналов системы управления, приводящих к формированию желаемых траекторий в фазовом пространстве системы, строится нейросетевой контроллер системы управления.
- Управление с помощью нейросетевого контроллера, сформированного через процедуру обучения с подкреплением на задаче поиска в некотором («геометрическом») смысле оптимальной траектории в фазовом пространстве системы управления в явном виде.
- Управление с помощью нейросетевого контроллера, сформированного через процедуру обучения с подкреплением на задаче неявного поиска оптимальной траектории путем максимизации определенной оценки качества управления R .
- Управление с помощью нейросетевого контроллера, сформированного через процедуру обучения с подкреплением совмещенную с предварительным подбором структуры нейронной сети эволюционными методами.

Подпункт 7.7.1.2.4 Формализация управления технологическим циклом производства на основе применения ostis-систем

В соответствие с изложенной онтологией под технологическим циклом производства подразумевается последовательность действий и операций, в результате которых осуществляется производство готовой продукции.

Функциональное взаимодействие компонентов комплекса управления и работающего в режиме реального времени технологического цикла производства осуществляется на основе непрерывного мониторинга состояния оборудования и параметров управления с помощью регистров-индикаторов и технических средств сопряжения.

С целью обеспечить возможность интеграции различных моделей (включая нейросетевые) с другими интеллектуальными системами предлагается формализация системы принятия решений на базе Технологии OSTIS.

Подпункт 7.7.1.2.5 Принципы оптимизации процессов управления технологическим циклом с использованием нейросетевого моделирования

При управлении с эталонной моделью ставится задача построения такого нейросетевого контроллера таким образом, чтобы происходило приемлемое обобщение собранных массивов данных об управлении технологическим циклом. С точки зрения теории машинного обучения в этом случае будет решаться задача обучения с учителем, при которой собранные данные будут использованы в качестве пар эталонных векторов входов и выходов контроллера.

Возможен подход при котором применяются методы обучения с подкреплением для построения эффективного контроллера системы управления. Методы обучения с подкреплением подразумевают исследовательское поведение агента в процессе построения контроллера, что может иметь позитивный эффект при решении задач со сложной структурой принятия управляющих решений.

Пусть задан некоторый функционал оценки качества управления R (качества выбора политики управления (выбора агентом (системой управления) действий) π), рассчитываемый на промежутке времени реализации вероятностного сетевого графика ТП $[0, t]$. (Данный функционал может «награждать», например, за снижение издержек производства, «штрафовать» за простой оборудования, возникновения отказов оборудования либо аварий.)

Принятие решений в этой среде может осуществляться нейронной сетью (агент под управлением нейросетевого контроллера).

Совершение агентом некоторых действий в фазовом пространстве системы управления приводит к построению некоторой траектории (в данном контексте построение управления рассматривается как построение траектории в фазовом пространстве системы управления технологического процесса производства).

Задача поиска оптимальной траектории в фазовом пространстве системы, таким образом, сводится к максимизации R (как в текущий момент, так и в будущем – т.е. к контроллеру системы управления ТЦ предъявляется требование формирование политики выбора действий π , максимизирующей оценку качества управления R).

Можно выделить две группы алгоритмов обучения с подкреплением, используемые для решения сложных задач управления:

1. “value-based” – обучение контроллера оценке функции будущего вознаграждения;
2. “policy-based” – обучение контроллера распределению действий, приводящих к выбору оптимальной политики.

Применение методов обучения с подкреплением подразумевает создание среды, в которой агент совершает действия. Агент совершает действия на основании текущего набора наблюдений за средой, а по результату совершения действий и последовавшего за ним возможного изменения состояния среды, получает численную награду.

Средой, в которой действует агент, в контексте решения задач управления технологическим циклом является система управления технологическим циклом производства, которая делает доступным для агента наблюдение за сигналами регистров о состоянии оборудования и параметров управления. На основании решений агента система принятия решений формирует запросы на изменения параметров управления.

В контексте применения подходов к построению контроллера системы управления, основанных на обучении с подкреплением могут быть предложены следующие методы к организации среды, в которой действует и обучается агент:

1. использование наблюдений за функционированием физического технологического процесса, при котором имплементированы схемы сравнения действий физической системы управления с действиями, предложенными агентом, с оценкой их соответствия текущей ситуации и допустимостью
2. использование имитационного моделирования технологического процесса производства в качестве базы для построения среды для обучения с подкреплением

Подпункт 7.7.1.2.6 Построение фазовой плоскости (фазового пространства) состояний системы адаптации управления технологическим циклом производства

Фазовое пространство состояний системы управления ТП – среда, в которой на основании наблюдаемого состояния необходимо принимать решения о выборе текущего управления.

Подпункт 7.7.1.2.7 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования

Процесс обучения нейронной сети заключается в поиске таких значений настраиваемых параметров сети (весовых коэффициентов связей между нейронами и уровней активации нейронов), чтобы она осуществляла правильное отображение. Обучение часто можно рассматривать как нелинейную оптимизационную задачу минимизации некоторой функции потерь относительно настраиваемых параметров сети. Форма этой функции определяется постановкой задачи, решаемой с применением нейронной сети.

При подходе, основанном на управлении с эталонной моделью, нейроконтроллер обучается с учителем. При этом используется функция потерь, оценивающая отклонение выходных сигналов сети от эталонных, и производится оптимизация данной функции (обычно градиентными методами).

Q-обучение является примером “value-based” алгоритма обучения с подкреплением, который может быть применен для поиска настраиваемых параметров модели при решении задач управления технологическим циклом.

Идея Q-обучения состоит в том, что агент в процессе обучения строит верную функцию Q оценки вознаграждения за следующее состояние, к которому может привести выбор некоторого управления[3].

Функция вознаграждения при этом основана на оценке качества управления контроллера. Определение функции вознаграждения играет важную роль, определяя поведение агента, формируемое при обучении. Выбор функции вознаграждения позволяет выбрать для оптимизации те критерии поведения агента, которые пользователь системы считает важными.

В качестве аппроксиматора функции вознаграждения может быть использована нейронная сеть. В этом случае задачей обучения является поиск таких значений настраиваемых параметров нейросети, при которых приближенная функция Q будет достаточно близкой к оптимальной функции Q^* при данной политике выбора действий π [6]:

для которой выполняется уравнение Беллмана:

При решении сложных практических задач агенту часто бывает недоступна полная информация о состоянии среды. В этой ситуации агент, который пользуется аппроксиматором Q , зависящим только от текущего наблюдаемого состояния среды может быть неэффективным при достаточно сложной структуре среды или наличии в ней рас-

пределенных во времени процессов, как связанных с действиями агента, так и независимыми от них. В описанной ситуации возможно использование в качестве аппроксиматора рекуррентной нейросети, обладающей внутренним состоянием[4]. LSTM-блоки, при использовании в рекуррентной сети позволяют ей аппроксимировать сложные зависимости, растянутые на длительные периоды времени[5].

В policy-based методах вместо аппроксимации числовых функций, оценивающей возможные награды получаемые в окружении агентом за совершенные действия, происходит построение напрямую функции политики выбора действий, связывающей состояния с действиями агента. Имеет место параметризации политики выбора действий настраиваемыми параметрами модели, под управлением которой функционирует агент.

Численная функция (функция награды) при этом может быть использована для оптимизации политики относительно настраиваемых параметров, но не используется для выбора действий.

Стochasticальная политика выбора действий возвращает распределение вероятностей возможных действий. Такие политики используются в частично наблюдаемых окружениях при наличии неопределенности.

Было показано, что для определенных классов задач методы, основанные на политиках сходятся быстрее чем value-based (Q-обучение), предпочтительны в пространствах выбора действий высокой размерности [4]. Гарантируется сходимость по крайней мере к локальному максимуму качества.

Политика π параметризована настраиваемыми параметрами θ .

$$\pi_\theta(a|s) = P[a|s]$$

Эта политика возвращает распределение действий a при наблюдаемом состоянии среды s .

С целью поиска настраиваемых параметров необходимо решить оптимизационную задачу максимизации функции оценки качества $J(\theta)$.

$$J(\theta) = E_{\pi_\theta}(\sum \gamma r)$$

Правила обновления настраиваемых параметров на шаге t :

$$\theta_{(t+1)} := \theta_t + \alpha \bigtriangledown J(\theta_t)$$

Согласно Policy Gradient Theorem[5]

$$\bigtriangledown E_{\pi_\theta}(r(\tau)) = E_{\pi_\theta}(r(\tau) \bigtriangledown \log \pi_{\theta(\tau)})$$

что может быть преобразовано как

$$\bigtriangledown E_{\pi_\theta}(r(\tau)) = E_{\pi_\theta}(r(\tau) (\sum_{t=1}^T \bigtriangledown \log \pi_\theta(a_t|s_t)))$$

Алгоритм REINFORCE, предназначенный для изучения агентом политики выбора действий, приводящей к максимизации кумулятивных будущих наград, может быть сформулирован следующим образом[4]:

1. Инициализировать параметры политики θ
2. Сгенерировать эпизод взаимодействия агента со средой с $\{S_i\}, \{A_i\}, \{R_i\}$ – последовательностями наблюдаемых агентом состояний среды, выборов действий, полученных наград длиной T .
3. Для каждого шага t вычислить discounted reward $G_t := \sum_{k=t+1}^T \gamma^{k-t-1} R_k$
4. Обновить параметры по правилу $\theta := \theta + \alpha \gamma^t G \bigtriangledown_\theta \ln \pi(A_t|S_t, \theta)$
5. Повторить шаги 2-4 до сходимости.

Подпункт 7.7.1.2.8 Решатели задач ostis-системы адаптивного управления технологическим циклом производства

В рамках Технологии OSTIS решатели задач строятся на базе мультиагентного подхода. В соответствии с этим подходом решатель задачи строится как набор агентов, называемых sc-agents. Все такие агенты разделяют память и могут обмениваться данными через специальные семантические структуры (sc-texts). Важно отметить, что некоторые агенты могут быть неатомарными, т.е. представленными в виде двух или более sc-агентов.

Полный решатель для задачи управления может быть представлен как разложение абстрактного неатомарного sc-агента.

abstract non-atomic sc-agent of cycle recommendation system

\Rightarrow decomposition of abstract sc-agent*:

- { • abstract sc-agent of interaction with the observation system
 - abstract sc-agent of forming recommendations
 - abstract sc-agent of forming requests
- }

1. abstract sc-agent of interaction with the observation system – предназначен для извлечения наблюдений из средств аппаратно-программного сопряжения в технологическом цикле, инициирует работу агента, ответственного за формирование рекомендаций.
2. abstract sc-agent of forming recommendations – на основании полученных наблюдений инициирует работу контроллера для получения рекомендаций по осуществлению управляющих воздействий.
3. abstract sc-agent of forming requests – на основании данных полученных от агента формирования рекомендаций, формирует запросы на изменения переменных управления ВТП посредством соответствующих программно-аппаратных средств сопряжения.

Построение контроллера, используемого в 2) может быть осуществлено на базе нейронных сетей.

Подпункт 7.7.1.2.9 Система адаптации управления

При разработке системы адаптации управления в рамках построения решателя задач ostis-системы предложенной структуры, необходимо реализовать агент для формирования рекомендаций на основании полученных наблюдений. В основе данного агента лежит контроллер, сформированный в соответствие с одним из алгоритмов, описанных выше.

Подпункт 7.7.1.2.10 Пример реализации системы адаптации управления

В качестве примера можно рассмотреть задачу поиска оптимальной стратегии обслуживания оборудования технологического цикла производства.

Решатель такой задачи может быть представлен как разложение абстрактного неатомарного sc-агента.

abstract non-atomic sc-agent of cycle maintenance recommendation system

⇒ decomposition of abstract sc-agent*:

- {• abstract sc-agent of interaction with the observation system
- abstract sc-agent of forming recommendations
- abstract sc-agent of forming maintenance requests
}

1. abstract sc-agent of interaction with the observation system – предназначен для извлечения наблюдений из средств аппаратно-программного сопряжения в технологическом цикле, инициирует работу агента, ответственного за формирование рекомендаций.
2. abstract sc-agent of forming recommendations – на основании полученных наблюдений инициирует работу нейроконтроллера для получения рекомендаций по совершению операций обслуживания.
3. abstract sc-agent of forming maintenance requests – на основании данных полученных от агента формирования рекомендаций, формирует запросы на обслуживание для соответствующих программно-аппаратных средств сопряжения.

Для построения нейроконтроллера будет использовано обучение с подкреплением. В качестве среды для обучения с подкреплением будет использоваться имитационная модель технологического цикла производства.

Для реализации имитационной модели в описанной формализации используются:

- распределения длительности безотказной работы для оборудования i -го типа $F_{ir}(t_{wf})$ в режиме $r(M_i)$;
- распределения длительности восстановления (ремонта) оборудования i -го типа после отказа $F_{if}(t_r)$;
- распределения длительности ликвидации аварии на оборудовании i -го типа $F_{ife}(t_{re})$;
- вероятности возникновения аварии при отказе i -го узла P_{ie} .

Имитационная модель функционирует в течение заданного интервала времени, перезапуская производственный цикл и, возможно, осуществляя перед этим запуском профилактические действия. Для принятия решения о проведении и содержании профилактических действий используется система управления технологическим циклом.

Система управления технологическим циклом производства делает доступным для агента наблюдение за временемем безотказной работы всех узлов M_i .

Формирование функции вознаграждения включает такие компоненты как время непрерывной работы цикла (R_{nop}), суммарный объём затрат на обслуживание и ликвидацию отказов и аварий оборудования (R_{cost}), суммарное число отказов оборудования (R_f), в том числе, приведшее к аварии (R_{fe}), суммарное число профилактик за цикл (R_{rep}). В процессе обучения модели значение функции вознаграждения формируется следующим образом во время единичного прохождения процедуры «профилактика-цикл». Каждый из компонентов награды входит в уравнение с заданным весовым коэффициентом α_i , характеризующим его значимость.

$$R = \alpha_1 R_{nop} + \alpha_2 R_{cost} + \alpha_3 R_f + \alpha_4 R_{fe} + \alpha_5 R_{rep}$$

Для управления агентом использованы нейросети на базе МСП с блоком LSTM. В случае с policy gradient используется sigmoid, выход сети представляет собой распределение вероятностей действий. Структура сети:

1. Dense x64 ReLU
2. Dense x64 ReLU
3. LSTM x32 ReLU
4. Dense x6 softmax

§ 7.7.2. Построение умных предприятий рецептурного производства с помощью ostis-систем

Пункт 7.7.2.1. Проблемы проектирования умных предприятий

Подпункт 7.7.2.1.1 Проблемы разработки систем комплексной автоматизации

Существующие средства автоматизации деятельности предприятия имеют высокую стоимость, трудны в освоении и адаптации к конкретному производству. Как правило, такие средства, с одной стороны, жёстко ориентированы на решение некоторого ограниченного класса задач, с другой стороны, разработчики стремятся сделать такого рода средства как можно более универсальными, наращивая их частными решениями, что приводит к сложности и громоздкости таких систем. Вследствие подобного подхода к наращиванию функционала существующие средства автоматизации деятельности предприятия имеют низкий уровень гибкости (возможности внесения изменений), что приводит к существенным накладным расходам при адаптации таких средств к новым требованиям. Как правило, внесение изменений в указанные средства требует вмешательства разработчиков (часто сторонних с точки зрения предприятия), что влечёт значительные временные и финансовые затраты. Как следствие указанных проблем, далеко не всякое предприятие может обеспечить высокий уровень автоматизации своей деятельности, даже в случае наличия на рынке подходящих решений. Отсутствие общих унифицированных моделей и средств построения систем автоматизации деятельности предприятия приводит к большому количеству дублирований аналогичных решений как в рамках различных предприятий, так и в рамках разных подразделений одного предприятия. При этом часто возникает ситуация, когда некоторые частные системы, решающие различные задачи в рамках одного предприятия, оказываются несовместимыми между собой, что приводит к дополнительным расходам на реализацию механизмов согласования, например, преобразование форматов данных. Отсутствие такого рода моделей препятствует дальнейшему повышению уровня автоматизации предприятия, в частности, в области автоматизации принятия решений в нештатных ситуациях, прогнозирования дальнейшего развития событий.

Подпункт 7.7.2.1.2 Требования, предъявляемые к стандартизации предприятий

Средства автоматизации предприятия должны оперативно и с минимальными затратами времени сотрудников адаптироваться к любым изменениям самого производства – к расширению или сокращению объёмов производства, изменениям номенклатуры производства, изменению используемого оборудования, изменению общей структуры производства, изменению взаимодействия с поставщиками и потребителями, к изменению нормативно-правовых актов (включая стандарты), к различного рода непредвиденным обстоятельствам. Адаптация средств автоматизации предприятия ко всем видам изменений самого предприятия и всем аспектам его взаимодействия с внешней средой требует внесения изменений в модель предприятия, полностью отражающую текущее состояние его деятельности. Средства автоматизации предприятия должны быть гибкими не только для оперативной адаптации к реконфигурации производства, но и для оперативного внесения изменений в сами средства автоматизации в направлении их постоянного совершенствования. Здесь существенным является не только снижение трудоёмкости повышения уровня автоматизации, но и поддержка высоких темпов повышения уровня автоматизации, а также чётко продуманный переходный процесс от одного уровня автоматизации к следующему, в ходе которого одновременно используется и устаревший вариант, и новый. Эксплуатация системы автоматизации предприятия текущего уровня и перманентный процесс повышения этого уровня требуют согласованного и квалифицированного взаимодействия сотрудников предприятия. Основой такого взаимодействия является хорошо структурированная, достаточно полная и оперативно актуализируемая модель предприятия, отражающая все аспекты текущего состояния структуры и деятельности предприятия, а также планы его развития. Такого рода комплексная модель называется знаниями предприятия, которыми надо управлять (добывать, хранить, модернизировать, распространять и т.д.). Повышение уровня автоматизации предприятия предполагает существенное расширение числа автоматически или автоматизировано решаемых задач, а это, в свою очередь, приводит к автоматизации решения интеллектуальных задач, т.е. к использованию технологий искусственного интеллекта. К числу интеллектуальных задач, решаемых на предприятии, можно отнести:

1. анализ производственных ситуаций (в том числе нештатных);

2. принятие решений на различных уровнях;
3. планирование поведения в сложных обстоятельствах;
4. генерация, актуализация документации;
5. обучение новых и повышение квалификации действующих сотрудников;
6. и т.д.

Для того, чтобы обеспечить широкое применение технологий искусственного интеллекта в автоматизации предприятия, все корпоративные знания предприятия должны быть записаны на формальном языке представления знаний. При этом указанный язык должен быть удобен не только для использования в интеллектуальных компьютерных системах, но и для использования всеми сотрудниками предприятия.

Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности

Анализ работ в данной области позволил сформулировать наиболее важные и общие проблемы, связанные с разработкой и применением современных стандартов в различных областях:

- Прежде всего сложность ведения самих стандартов из-за дублирования информации, особенно сложность изменения терминологии.
- Дублирующаяся информация в документации, описывающей стандарт.
- Проблемы интернационализации стандартов – перевод стандарта на несколько языков фактически требует поддержки и координации независимых версий стандарта на разных языках.
- В результате несоответствия форматов разных стандартов. В результате автоматизация процесса разработки и применения стандартов усложняется.
- Неудобство использования стандарта, особенно сложность поиска нужной информации. Как следствие, сложность изучения стандартов.
- Сложность автоматизации проверки соответствия объекта или процесса требованиям определенного стандарта.
- etc.

Эти проблемы в основном связаны с представлением стандартов. Наиболее перспективным подходом к решению этих задач является преобразование каждого конкретного стандарта в базу знаний, в основе которой лежит набор соответствующих этому стандарту онтологий. Такой подход позволяет значительно автоматизировать процессы разработки стандарта и его применения.

В качестве примера рассмотрим стандарт **ISA-88 ISA88BC-1995эл** (базовый стандарт для рецептурного производства). Хотя этот стандарт широко используется американскими и европейскими компаниями и активно внедряется на территории Республики Беларусь, он имеет ряд недостатков, перечисленных ниже. Опыт автора со стандартами ISA-88 и ISA-95 выявил следующие проблемы, связанные с версиями стандарта:

- Американская версия стандарта – *ANSI/ISA-88.00.01-2010* – была обновлена и находится в третьем издании 2010 года;
- *ISA-88.00.02-2001* – содержит структуры данных и рекомендации для языков;
- *ANSI/ISA-TR88.00.02-2015* – описывает пример реализации *ANSI/ISA-88.00.01*;
- *ISA-88.00.03-2003* – действие, описывающее использование общих рецептов сайта внутри и между компаниями;
- *ISA-TR88.0.03-1996* – показывает возможные форматы представления процедур рецепта;
- *ANSI/ISA-88.00.04-2006* – структура записей серийного производства;
- *ISA-TR88.95.01-2008* – объясняет совместное использование ISA-88 и ISA-95;
- В то же время европейская версия, утвержденная в 1997 году – *IEC 61512-1* – основана на более старой версии *ISA-88.01-1995*;
- Русская версия стандарта – *ГОСТ Р МЭК 61512-1-2016* – идентична *МЭК 61512-1*, то есть также устарела. Также вызывает ряд вопросов, связанных с не очень удачным переводом оригинальных английских терминов на русский язык.

Другим стандартом, часто используемым в контексте Индустрии 4.0, является **ISA-95 ISA95**. **ISA-95** – это отраслевой стандарт для описания систем управления высокого уровня. Его основная цель — упростить разработку таких систем, абстрагироваться от аппаратной реализации и предоставить единый интерфейс для взаимодействия со слоями ERP и MES. Состоит из следующих частей:

- *ANSI/ISA-95.00.01-2000*, Часть 1: «Модели и терминология» — состоит из стандартной терминологии и объектных моделей, которые можно использовать для определения того, какая информация подлежит обмену;
- *ANSI/ISA-95.00.02-2001*, Часть 2: «Атрибуты объектной модели» — он состоит из атрибутов для каждого объекта, определенного в части 1. Объекты и атрибуты могут использоваться для обмена информацией между различными системами, а также могут использоваться в качестве основы для реляционных баз данных;
- *ANSI/ISA-95.00.03-2005*, Часть 3: «Операционные модели управления производством» — основное внимание уделяется функциям и действиям Уровня 3 (Производство/MES);

- *ISA-95.00.04* Часть 4: «Объектные модели и атрибуты для операций управления производством». Комитет SP95 все еще разрабатывает эту часть ISA-95. Эта техническая спецификация определяет объектную модель, которая определяет обмен информацией между действиями MES (определен в Части 3 стандарта ISA-95). Модель и атрибуты Части 4 составляют основу для разработки и внедрения стандартов интерфейса, обеспечивая гибкий поток сотрудничества и обмена информацией между различными видами деятельности MES;
- *ISA-95.00.05* Часть 5: «Транзакции между бизнесом и производством (B2M транзакции)». Часть 5 стандарта ISA-95 все еще находится в разработке. Эта техническая спецификация определяет операции между рабочими местами и структурами автоматизации производства, которые могут использоваться вместе с моделями элементов частей 1 и 2. Операции объединяют и упорядочивают описанные производственные элементы и действия в рамках предыдущей части стандарта. Такие операции возникают в любом отношении внутри организации, однако внимание данной технической спецификации уделяется взаимодействию между организацией и системой управления.
- *ISA-95.00.06* Часть 6: «Транзакции между производственными операциями».
- *ISA-95.00.07* Часть 7: «Модель сервисных сообщений».
- *ISA-95.00.08* Часть 8: «Профили обмена информацией»

Модели помогают определить границы между бизнес-системами и системами управления. Они помогают ответить на вопросы о том, какие функции могут выполнять какие задачи и какой информацией необходимо обмениваться между приложениями.

Первый этап построения модели цифрового двойника требует встраивания данных на более низких уровнях производства, таких как производственные процессы и оборудование. Схема производства P&ID служит источником этих данных. Поэтому стандарт ISA 5.1 **ISA_5_1** должен работать со схемой P&ID и широко используется в системах управления наряду со стандартом ISA 88 для полного описания нижних уровней производства. Этот стандарт полезен, когда требуется ссылка на оборудование в химической, нефтяной, энергетической, кондиционирующей, металлообрабатывающей и многих других отраслях промышленности. Стандарт позволяет любому человеку с достаточным уровнем знаний о предприятии читать блок-схемы, чтобы понять, как измерять и контролировать процесс, не вдаваясь в детали приборов или знаний эксперта по приборам. Он предназначен для предоставления достаточной информации, чтобы SA5.1 Целью настоящего стандарта было установить последовательный метод наименования приборов и контрольно-измерительных систем, используемых для измерения и контроля. Для этого представлена система обозначений, включающая символы и идентификационные коды. Последним выпуском подкомитета ISA5.1 является обновленный *ISA-5.1-2022, «Символы приборов и идентификация»*.

Обучение — это простой способ достичь этих стандартов. Международное общество автоматизации (ISA) — некоммерческая профессиональная ассоциация и признанный лидер в области обучения автоматизации и управлению, занимающийся подготовкой рабочей силы к технологическим изменениям и отраслевым вызовам. Однако цена относительно высока, около 1000 долларов на человека в день. Для 2 человек это 10000\$ за обычный курс на 5 дней. Для одних стран это доступно, для других нет.

Принимаются во внимание различные установленные процедурные требования различных организаций, но это делается путем предоставления альтернативных методов символики, если это не противоречит целям стандарта. Существует множество вариантов добавления информации или упрощения символа при желании.

Эти и другие стандарты сейчас распространяются в виде документов, неудобных для автоматизированной обработки и, как отмечалось выше, сильно зависят от языка, на котором написан каждый документ.

Пункт 7.7.2.2. Подходы к проектированию и стандартизации умных предприятий рецептурного производства с помощью ostis-систем

Подпункт 7.7.2.2.1 Подходы к автоматизации предприятий

В настоящее время существует ряд подходов, ориентированных на повышение уровня автоматизации и гибкости предприятий различного рода. Рассмотрим те из них, которые оказали влияние на развитие подхода, предлагаемого в данной работе.

- **Онтологические модели предприятий.** Подход к проектированию различного рода систем на основе онтологических моделей широко используется в настоящее время [3], при этом в особую область исследований выделяют «онтологии предприятия» [2]. Суть предлагаемых подходов состоит в построении онтологий, описывающих деятельность того или иного предприятия или его подразделений. Недостатками моделей являются отсутствие унификации представления различных видов знаний предприятия, отсутствие единого подхода к выделению и формированию онтологий, отсутствие единого подхода к построению иерархии онтологий, что ограничивает возможность построения комплексной взаимосвязанной системы онтологий.

- Модели управления знаниями предприятий. Управление знаниями в организации – это систематический процесс идентификации, использования и передачи информации, знаний, которые люди могут создавать, совершенствовать и применять. Это процесс, в ходе которого организация генерирует знания, накапливает их и использует в интересах получения конкурентных преимуществ. В настоящее время управление знаниями предприятия реализуется в виде систем управления знаниями [4]. Наиболее актуальным направлением в формализации процесса накопления и управления знаниями предприятия является применение онтологического подхода к построению моделей такого рода процессов.
- Модели ситуационного управления. Термин «ситуационное управление» впервые появился в работах Д.А. Пospelova [5]. Это направление получило дальнейшее развитие [6, 7], а в ряде новых работ показано применение в реализации методов ситуационного управления онтологического подхода [8, 9]. Таким образом, модели и методы ситуационного управления могут быть использованы при построении онтологической модели предприятия с целью повышения эффективности разрабатываемых решений в конкретных производственных ситуациях.
- Многоагентные модели предприятий. В настоящее время многоагентная модель широко применяется при проектировании систем автоматизации производства на различных уровнях. Удобство такого подхода и широта его использования обусловлены схожестью многоагентной модели с реальными процессами, происходящими на предприятии. Действительно, в классической многоагентной системе под агентом понимается некий субъект, как правило, активный и способный взаимодействовать с окружающей средой [10, 11]. Будучи объединёнными в коллектизы, такие агенты способны решать задачи гораздо более сложные, чем мог бы решить один агент. К достоинствам многоагентного подхода можно отнести возможность построения на его основе распределённых многоуровневых систем. Наиболее очевидной интерпретацией такого рода модели в применении к конкретному предприятию является рассмотрение его работников как агентов, каждый из которых способен решать определённый класс задач, вынужденных координировать свои действия для достижения общей коллективной цели. С учётом иерархии структурных подразделений конкретной организации могут быть выделены и уровни иерархии агентов, соответствующие отделам или цехам.
- Модели реинжиниринга бизнес-процессов предприятий. Реинжиниринг бизнес-процессов – это фундаментальное переосмысление и радикальное перепроектирование бизнес-процессов предприятий для достижения резких, скачкообразных улучшений в основных актуальных показателях их деятельности: стоимость, качество, услуги и темпы [12]. Он базируется на понятиях будущего образа фирмы и модели бизнеса, раскрываемых в [13]. Для того, чтобы повысить эффективность реинжиниринга, необходимо обеспечить возможность построения формальных моделей, описывающих предприятие на разных уровнях детализации, и обеспечить унификацию таких моделей, их интегрируемость и иерархичность.

Основной недостаток всех приведённых выше моделей заключается в том, что ни одна из них не обладает достаточной полнотой, и для наиболее адекватного соответствия реальному предприятию его модель должна быть результатом интеграции всех этих моделей.

Подпункт 7.7.2.2.2 Предлагаемый подход к автоматизации предприятий

В основе предлагаемого подхода к решению указанных проблем лежат следующие принципы.

- Предприятие рассматривается как распределённая, интеллектуальная социотехническая система, в основе которой лежит хорошо структурированная общая база знаний предприятия.
- В рамках базы знаний предприятия интегрируются все вышеуказанные модели.
- Предприятие рассматривается как иерархическая многоагентная система. В качестве агентов выступают как сотрудники предприятия, так и программные (программно-аппаратные) агенты. Иерархичность многоагентной системы означает то, что агенты могут быть неатомарными, т.е. коллективами взаимодействующих между собой агентов, причём такая структура может быть многократно вложенной.
- Весь комплекс средств (как информационных, так и материальных), обеспечивающих деятельность предприятия, оформляется в виде интегрированной распределённой интеллектуальной системы, которую будем называть интеллектуальной корпоративной системой. Основными пользователями этой системы являются сотрудники предприятия.
- Проектирование онтологической модели предприятия сводится к проектированию онтологической модели его интеллектуальной корпоративной системы, которая далее может интерпретироваться имеющимся набором материальных ресурсов. При этом онтологическая модель предприятия является и объектом, и результатом проектирования.

Для реализации корпоративной системы предприятия предлагается использовать технологию OSTIS [14, 15]. Из этого следует:

- в качестве основы для представления знаний используется унифицированный, универсальный язык представления – SC-код;
- разработка системы сводится к разработке её модели, описанной средствами SC-кода (sc-модели), которая затем интерпретируется одной из платформ интерпретации;

- база знаний имеет иерархическую структуру, позволяющую рассматривать хранимые знания на различных уровнях детализации (прежде всего это иерархия предметных областей (ПрО) и соответствующих им онтологий [16]);
- частью технологии являются средства коллективного проектирования баз знаний, средства проектирования машин обработки знаний и их компонентов;
- модель обработки знаний основана на многоагентном подходе, позволяющем строить параллельные асинхронные машины обработки знаний, интегрировать различные частные модели обработки в рамках одной системы;
- все агенты взаимодействуют исключительно посредством общей памяти, хранящей конструкции SC-кода (sc-памяти); такой подход позволяет обеспечить гибкость системы и возможность параллельного решения различных задач [17];
- для разработки программ агентов используется внутренний параллельный язык SCP, тексты которого также представлены в SC-коде, что позволяет обеспечить платформенную независимость таких агентов.

На данном этапе работы основное внимание уделено решению задачи разработки онтологической модели базы знаний, в частности – построению набора онтологий ПрО, описывающих содержание основных стандартов. Формальное представление стандартов является основой для согласования всех ключевых аспектов деятельности предприятия и построения общей онтологической модели всего предприятия в целом и отдельных его компонентов.

Подпункт 7.7.2.2.3 Принципы формализации стандартов рецептурного производства

Основой онтологического подхода к проектированию предприятия является формализация стандартов. Каждый стандарт рассматривается как онтология соответствующей ПрО, являющаяся основой для автоматизированного решения ряда задач, включая информационное обслуживание сотрудников, формальную оценку соответствия предприятия этим стандартам и т.д. Одной из важных проблем внедрения стандарта на предприятии является возможность неоднозначной трактовки некоторых положений стандарта, а также необходимость постоянной коррекции такой трактовки с целью приближения её к смыслу оригинала. Кроме того, существуют особенности применения стандарта на каждом предприятии, необходимость актуализации используемого стандарта (т.к. любой стандарт постоянно эволюционирует), с последующим внесением изменений в структуру и организацию деятельности предприятия для обеспечения соответствия стандарту. Одним из путей решения такого рода проблем является построение его формальной семантической модели, которая могла бы одинаково интерпретироваться как компьютерной системой, так и человеком. Формальное семантическое представление стандарта позволяет без внесения каких-либо изменений в структуру такого представления дополнять его различного рода дидактической информацией (примерами, пояснениями и т.д.), которая способствует пониманию стандарта сотрудниками предприятия. Кроме того, формальное семантическое представление стандарта обеспечивает значительное упрощение внесения изменений в такое представление стандарта, которые могут быть вызваны либо уточнением трактовки этого стандарта, либо эволюцией самого стандарта (его исходного документа). Построение формальной модели стандарта сводится к построению интегрированной формальной онтологии, специфицирующей соответствующую ПрО. Для этого необходимо отобразить структуру и содержание исходного текста стандарта на иерархию ПрО и соответствующих им онтологий. Использование онтологического подхода позволяет путём добавления интеллектуальных агентов построить на его основе интеллектуальную справочную систему, предоставляющую широкий спектр информационных услуг пользователям, в том числе способную отвечать на широкий спектр вопросов, ответ на которые может быть в явном виде не представлен в тексте стандарта, или поиск его в таком тексте затруднителен.

Подпункт 7.7.2.2.4 Формализация стандарта ISA-88

Структурно исходный документ первой части стандарта ISA-88 состоит из 6 разделов:

- область применения стандарта (Scope of the standard);
- онормативные ссылки (Normative References);
- отермины и определения (Definitions);
- опроцессы серийного производства и оборудование (Batch Processes and Equipment);
- опонятия, используемые при управлении серийным производством (Batch Control Concepts);
- одействия и функции процесса управления серийным производством (Batch Control Activities and Functions)

Отобразим структуру документа на иерархию ПрО и соответствующих им онтологий. Стандарт ISA-88 в целом соответствует ПрО предприятий рецептурного производства. Первые два раздела специфицируют документ стандарта и непосредственно к описанию ПрО рецептурных производств не относятся и, соответственно, не отображаются на иерархии ПрО и онтологий. Раздел 3 соответствует терминологической онтологии и логической онтологии [16] ПрО предприятий рецептурного производства. Подраздел 4.1 соответствует ПрО процессных моделей рецептурных производств. Остальные подразделы четвёртого раздела соответствуют ПрО физических моделей рецептурных

производств. Раздел 5 соответствует ПрО моделей процедурного управления. Раздел 6 соответствует ПрО деятельности по управлению рецептурным производством. Запишем иерархию ПрО базового уровня в формальном виде на языке SCn [22]:

ПрО предприятий рецептурного производства

- С ПрО физических моделей рецептурных производств
- С ПрО процессных моделей рецептурных производств
- С ПрО моделей процедурного управления оборудованием рецептурных производств
- С ПрО деятельности по управлению рецептурным производством

Заключение

В настоящее время реализуются различные подходы при решении научных и прикладных задач по управлению роботизированными производствами в условиях наличия случайных внешних воздействий на объект управления.

В данном разделе предложена реализация процедуры синтеза обратных связей по управлению технологическим циклом производства на основе его формализации в рамках OSTIS-систем и построения моделей искусственных нейронных сетей для решения прикладных задач оптимизации управления сложными технологическими комплексами.

Полученные результаты дают возможность разработки гибридных интеллектуальных компьютерных систем, предназначенных для решения широкого класса прикладных задач адаптации управления технологическими объектами в режиме реального времени.

Глава 7.8.

Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения

Самодумкин С.А.

⇒ *аннотация**:
[Аннотация к главе.]

§ 7.8.1. Требования, предъявляемые к интеллектуальным геоинформационным системам нового поколения

§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами

§ 7.8.3. Основные компоненты формальных онтологий, используемых в геоинформационных системах

Пункт 7.8.3.1. Геосемантические элементы: местоположение, топология, близость, ориентация, динамика

Пункт 7.8.3.2. Формализация пространственных отношений в геоинформационных системах

§ 7.8.4. Стратифицированная модель информационного пространства объектов местности

§ 7.8.5. Формальная денотационная семантика языка карт

§ 7.8.6. Формальная онтология объектов местности

§ 7.8.7. Средства автоматизации проектирования интеллектуальных геоинформационных систем

Глава 7.9.

Обеспечение информационной безопасности в рамках Экосистемы OSTIS

**Чертков В.М.
Захаров В.В.
Крищенович В.А.**

⇒ *аннотация**:
[Аннотация к главе.]

Введение в Главу 7.9.

Большое разнообразие моделей обеспечения информационной безопасности, всё возрастающий объем данных, которые необходимо анализировать для обнаружения атак на информационные системы, изменчивость методов атак и динамическое изменение защищаемых информационных систем, необходимость оперативного реагирования на атаки, нечеткость критериев обнаружения атак и выбора методов и средств реагирования на них, нехватка высококвалифицированных специалистов по защите влечет за собой потребность в использовании методов искусственного интеллекта для решения задач безопасности.

§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения

Информационную безопасность интеллектуальных систем следует рассматривать с двух точек зрения:

- с точки зрения применения искусственного интеллекта в информационной безопасности;
- с точки зрения организация информационной безопасности в интеллектуальных системах.

Применения искусственного интеллекта в информационной безопасности

Следует отметить, что искусственный интеллект активно применяется для мониторинга и анализа уязвимостей безопасности в сетях передачи информации [2 Исобоев]. Система искусственного интеллекта позволяет машинам более эффективно выполнять поставленные задачи, такие как:

- визуальное восприятие, распознавание речи, принятие решений и перевод с одного языка на другой;
- обнаружение вторжений – искусственный интеллект может обнаруживать сетевые атаки, заражения вредоносным программным обеспечением и другие киберугрозы;
- кибераналитика – искусственный интеллект также используется для анализа больших данных с целью выявления закономерностей и аномалий в системе кибербезопасности организации с целью обнаружения не только известных, но и ещё неизвестных угроз;
- безопасная разработка программного обеспечения – искусственный интеллект может помочь создать более безопасное программное обеспечение, предоставляя разработчикам обратную связь в режиме реального времени.

При этом искусственный интеллект используется не только для защиты, но и для нападения, например для эмуляции акустических, видео и других образов с целью обмана механизмов аутентификации и дальнейшей имперсонации, обман проверки человек или робот капча и т.д.

В настоящее время можно определить следующие классы систем, в которых применяется искусственный интеллект:

- UEBA (User and Entity Behavior Analytics) – система анализа поведения субъектов (пользователей, программ, агентов и т.д.) на предмет обнаружения нестандартного поведения и использования их для обнаружения потенциальных угроз с использованием шаблонов угроз (паттернов);
- TIP (Threat Intelligence Platform) – платформы раннего обнаружения угроз на основе сбора и анализа информации индикаторов компрометации и реагирования на них. Применение методов машинного обучения повышает эффективность обнаружения неизвестных угроз на ранних этапах;

- EDR (Endpoint Detection and Response) – системы обнаружения атак оперативного реагирования на конечных точках компьютерной сети. Могут обнаруживать вредоносные программы, автоматически классифицировать угрозы и самостоятельно реагировать на них;
- SIEM (Security Information and EventManagement) – системы сбора и анализа информации о событиях безопасности от сетевых устройств и приложений в реальном времени и оповещения;
- NDR (Network Detection and Response) – системы обнаружения атак на сетевом уровне и оперативного реагирования на них. ИИ использует накопленную статистику и базу знаний об угрозах;
- SOAR (Security Orchestration and Automated Response) – системы, позволяющие выявлять угрозы информационной безопасности и автоматизировать реагирование на инциденты. В решениях данного типа, в отличие от SIEM-систем, ИИ помогает не только проводить анализ, но и автоматически реагировать надлежащим образом на выявленные угрозы;
- Средства защиты приложений (Application Security) – системы, позволяющие определять угрозы безопасности прикладных приложений, управлять процессом мониторинга и устранения таких угроз;
- Антифрод (Antifraud) – платформы в режиме реального времени обнаруживают угрозы в бизнес-процессах и мошеннические операции. ИИ используется для определения отклонений от идентифицированных бизнес-процессов с целью выявления вторжений или уязвимости процессов и повышает адаптивность к изменению логики и метрик бизнес-процессов.

В работе [3 Частикова] предложена методика построения нейроиммунной системы анализа инцидентов информационной безопасности, объединяющей модули сбора и хранения (сжатия) данных, модуль анализа и корреляции событий информационной безопасности и подсистемы обнаружения сетевых атак на основе сверточных нейронных сетях. Использование технологий машинного обучения в информационной безопасности создает узкие места и системные уязвимости, которые можно использовать и имеет следующие недостатки [4 Абдурахман]:

- наборы данных, которые должны быть сформированы из значительного количества входных выборок, что требует много времени и ресурсов;
- требуется огромное количество ресурсов, включая память, данные и вычислительную мощность;
- частые ложные срабатывания, которые нарушают работу и в целом снижают эффективность таких систем;
- организованные атаки на основе искусственного интеллекта (семантические вирусы).

Организация информационной безопасности в интеллектуальных системах

Определим цели обеспечения информационной безопасности систем нового поколения.

Из монографии А.В. Остроух [5] **целями обеспечения информационной безопасности традиционных интеллектуальных систем являются:**

- обеспечение конфиденциальности информации в соответствии с проведенной классификацией;
- обеспечение целостности информации на всех этапах, связанных с нею процессов (создание, обработка, хранение, передача и уничтожение) при предоставлении публичных услуг;
- обеспечение своевременной доступности информации при предоставлении публичных услуг;
- обеспечение наблюдаемости, направленной на фиксирование любой деятельности пользователей и процессов;
- обеспечение аутентичности и невозможности отказа от транзакций и действий, производимых участниками предоставления публичных услуг;
- учет всех процессов и событий, связанных с вводом, обработкой, хранением, предоставлением и уничтожением данных.

Следует отметить так как интеллектуальные системы нового поколения будут взаимодействовать с подобными себе системами понимая при этом, о чем осуществляется запрос, то цели обеспечения будут выглядеть по-другому.

Целями обеспечения информационной безопасности интеллектуальных систем нового поколения являются:

- обеспечение сохранности семантической совместимости информации;
- защита достоверности и целостности информации;
- обеспечение доступности информации на разных уровнях интеллектуальной системы;
- минимизация ущерба от событий, несущих угрозу информационной безопасности.

В настоящее время разработаны классические подходы и принципы обеспечения безопасности баз знаний (данных), интерфейсов связи (обмена информацией) между компонентами интеллектуальных систем такие, как шифрование передаваемых данных, фильтрация ненужного (избыточного) контента и политика разграничения доступа к данным.

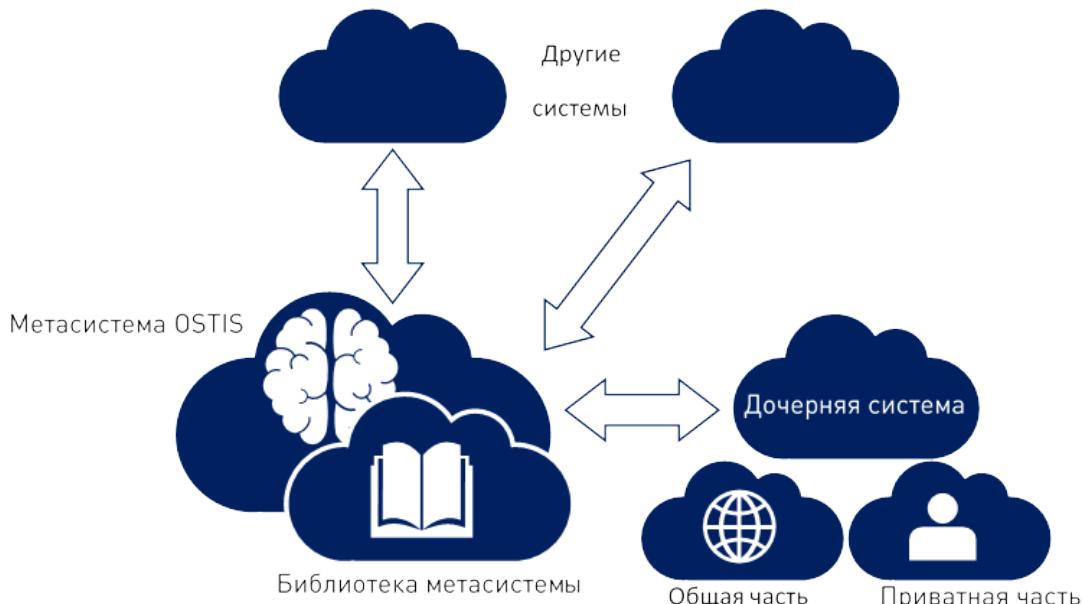
Система обеспечения информационной безопасности должна создаваться на следующих принципах:

- принцип равнoprочности – означает обеспечение защиты оборудования, программного обеспечения и системы управления от всех видов угроз;
- принцип непрерывности – предусматривает непрерывное обеспечение безопасности информационных ресурсов, ИС для непрерывного предоставления публичных услуг;
- принцип разумной достаточности – означает применение таких мер и средств защиты, которые являются разумными, рациональными и затраты на которые, не превышают стоимости последствий нарушения информационной безопасности;

- принцип комплексности – для обеспечения безопасности во всем многообразии структурных элементов, угроз и каналов несанкционированного доступа должны применяться все виды и формы защиты в полном объеме;
- принцип комплексной проверки – заключается в проведении специальных исследований и проверок, специального инженерного анализа оборудования, верификационных исследований программных средств. Должен осуществляться непрерывный мониторинг аварийных сообщений и параметров ошибок, постоянно должно выполняться тестирование аппаратного и программного оборудования, а также контроль целостности программных средств, как при загрузке программных средств, так и в процессе функционирования;
- принцип надежности – методы, средства и формы защиты должны надежно перекрывать все пути проникновения и возможные каналы утечки информации, для этого допускается дублирование средств и мер безопасности;
- принцип универсальности – меры безопасности должны перекрывать пути угроз независимо от места их возможного воздействия;
- принцип плановости – планирование должно осуществляться путем разработки детальных планов действий по обеспечению информационной защищенности всех компонент системы предоставления публичных услуг;
- принцип централизованного управления – в рамках определенной структуры должна обеспечиваться организованно-функциональная самостоятельность процесса обеспечения безопасности при предоставлении публичных услуг;
- принцип целенаправленности – необходимо защищать то, что должно защищаться в интересах конкретной цели;
- принцип активности – защитные меры обеспечения безопасности в работе процесса предоставления услуг должны претворяться в жизнь с достаточной степенью настойчивости;
- принцип квалификации обслуживающего персонала – обслуживание оборудования должно осуществляться сотрудниками, подготовленными не только в вопросах эксплуатации техники, но и в технических вопросах обеспечения безопасности информации;
- принцип ответственности – ответственность за обеспечение информационной безопасности должна быть ясно установлена, передана соответствующему персоналу и утверждена всеми участниками в рамках процесса обеспечения информационной безопасности.

§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем

Архитектура Экосистемы OSTIS представлена на рисунке *Архитектура Экосистемы OSTIS в контексте решения задач информационной безопасности*.



= *Архитектура Экосистемы OSTIS в контексте решения задач информационной безопасности*

Экосистема OSTIS представляет собой коллектив взаимодействующих:

- ostis-систем;
- пользователей ostis-систем (конечных пользователей и разработчиков);
- других компьютерных систем, не являющихся ostis-системами, но являющихся дополнительными информационными ресурсами или сервисами для них

Ядро технологии OSTIS включает следующие компоненты:

- семантическая база знаний OSTIS, которая может описывать любой вид знаний, при этом ее легко дополнять новыми видами знаний.
- решатель задач OSTIS основанный на подходе, в котором применяется множество агентов. Такой подход позволяет легко интегрировать и комбинировать любые модели решения задач.
- интерфейс OSTIS-системы, представляющий собой подсистему со своей базой знаний и решателем задач.

Представленная архитектура Экосистемы OSTIS реализует следующие основные идеи:

- все базы знаний объединяются в Глобальную базу знаний, качество которой (логичность, корректность, целостность) постоянно проверяется множеством агентов. Все проблемы описываются в единой базе знаний, и для их устранения при необходимости привлекаются специалисты;
- каждое приложение, связанное с экосистемой OSTIS, имеет доступ к последней версии всех основных компонентов OSTIS, обновление компонентов выполняется автоматически;
- каждый владелец приложения Экосистемы OSTIS может поделиться частью своих знаний платно или бесплатно.

В рассмотренной Экосистеме OSTIS требуется организация обеспечения информационной безопасности на каждом из уровней: обмен данными, права доступа к данным, аутентификация клиентов Экосистемы, шифрование данных, получение данных из открытых источников, обеспечение достоверности и целостности хранимых и передаваемых данных, контроль за нарушением связей в базе знаний.

Следует отметить, что для некоторых направления по обеспечению информационной безопасности семантических систем могут применяться методы и алгоритмы, разработанные в рамках традиционных интеллектуальных систем. Для интеллектуальных систем нового поколения можно выделить ряд аспектов, в рамках которых требуетсяработка новых алгоритмов и методов обеспечения информационной безопасности. Рассмотрим основные направления обеспечения информационной безопасности семантических интеллектуальных систем.

Ограничение информационного трафика, анализируемого интеллектуальной системой

Экспоненциальный рост объема информации, циркулирующей в информационных потоках и ресурсах в условиях вполне определенных количественных ограничений на возможности средств ее восприятия, хранения, передачи и преобразования формирует новый класс угроз информационной безопасности, характеризуемых избыточностью совокупного входящего информационного трафика интеллектуальных систем.

В результате переполнение информационных ресурсов интеллектуальной системы избыточной информацией может спровоцировать распространения искаженной (деструктивной семантической) информации. Общая методология защиты интеллектуальных систем от бесполезной информации осуществляется посредством использования аксиологических фильтров, реализующих функции численной оценки ценности поступающей информации, отбора наиболее ценной и отсеивания (фильтрации) менее ценной (бесполезной или вредной) с использованием вполне определенных критериев.

Следует также выделить в отдельную категорию угроз информационной безопасности активные средства разрушения семантики баз знаний (семантические вирусы) [6].

Политика разграничения доступа к базе знаний

Мандатная политика безопасности (MAC – mandatory access control) основывается на мандатном (принудительном) разграничении доступа, определяющемся четырьмя условиями: все субъекты и объекты системы идентифицируются; задается решетка уровней безопасности информации; каждому объекту системы присваивается уровень безопасности, определяющий важность содержащейся в нем информации; каждому субъекту системы присваивается уровень доступа, определяющий уровень доверия к нему в интеллектуальной системе. Кроме того, мандатная политика имеет более высокую степень надежности. Реализация данной политики основывается на разработанном алгоритме определения согласованных уровней безопасности всех элементов онтологии.

Так как семантические базы знаний в отличие от реляционной базы данных позволяют выполнять правила для получения логических выводов, то для обеспечения безопасности данных актуальным является разработка алгоритмов и методов, с помощью которых можно будет получать только данные, имеющие уровни безопасности меньше уровней доступа субъектов их запросивших [7].

Связность

Вся информация, хранящаяся в семантической памяти интеллектуальной системы, систематизирована в виде единой базы знаний. К такой информации относятся непосредственно обрабатываемые знания, интерпретируемые программы, формулировки решаемых задач, планы и протоколы решения задач, информация о пользователях, описание

синтаксиса и семантики внешних языков, описание пользовательского интерфейса и многое другое [8]. В информационной базе знаний между фрагментами информации (единицами информации) должна быть предусмотрена возможность установления связей различного типа. Прежде всего, эти связи могут характеризовать отношения между информационными единицами. Нарушение связей приводит к неправильному логическому выводу, либо к получению ложных знаний, либо к несовместимости знаниям в базе.

Введение семантической метрики

На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, т.е. силу ассоциативной связи между информационными единицами [11]. Его можно было бы назвать отношением релевантности для информационных единиц. Такое отношение дает возможность выделять в информационной базе знаний некоторые типовые ситуации. Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

Семантическая совместимость

Внутренняя семантическая совместимость между компонентами интеллектуальной компьютерной системы (т.е. максимально возможное введение общих, совпадающих понятий для различных фрагментов хранимой базы знаний), являющаяся формой конвергенции и глубокой интеграции внутри интеллектуальной компьютерной системы для различного вида знаний и различных моделей решения задач, что обеспечивает эффективную реализацию мультимодальности интеллектуальной компьютерной системы. Внешняя семантическая совместимость между различными интеллектуальными компьютерными системами, выражаясь не только в общности используемых понятий, но и в общности базовых знаний и являясь необходимым условием обеспечения высокого уровня социализации интеллектуальных компьютерных систем [9].

Активность

В интеллектуальной системе для актуализации тех или иных действий способствуют знания, имеющиеся в этой системе. Таким образом, выполнение активностей в интеллектуальной системе должно инициироваться текущим состоянием информационной базы знаний. Появление в базе фактов или описаний событий, установление связей может стать источником активности системы [10]. В том числе преднамеренное искажение информации и связей может стать источником преднамеренного искажения информации.

Важно отметить, что интеллектуальная информационная система нового поколения – это самостоятельный субъект, который может сам осознанно, целенаправленно и постоянно заботиться о себе, в том числе о своей собственной безопасности.

Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии

⇒ автор*:

- Голенков В. В.
- Гулякина Н. А.

Особенность текущего состояния работ в области Искусственного интеллекта

Необходимость перехода от современных *компьютерных систем* (в том числе, и от современных *интеллектуальных компьютерных систем*) к *интеллектуальным компьютерным системам нового поколения* обусловлена необходимостью перехода к автоматизации более сложных (комплексных) видов и областей деятельности требующих создания целых комплексов компьютерных систем, способных эффективно взаимодействовать между собой в процессе коллективного решения сложных задач.

Компьютерные системы, обладающие указанными способностями называют *интероперабельными компьютерными системами*. Поскольку указанные компьютерные системы не могут не иметь высокого уровня *интеллекта*, мы также называем их *интеллектуальными компьютерными системами нового поколения*. Высокий уровень *интеллекта* интероперабельным компьютерным системам необходим:

- для адекватной оценки собственной компетенции и компетенции своих партнеров;
- для обеспечения взаимопонимания, договороспособности и координации (согласованности) своих действий с действиями партнеров в ходе *коллективного решения* сложных задач в условиях возможного возникновения непредсказуемых (нештатных) обстоятельств.

Очевидно, что для создания и эксплуатации таких *интероперабельных интеллектуальных компьютерных систем* необходимо:

- разработать общую формальную теорию таких систем;
- разработать комплексную технологию проектирования и поддержки жизненного цикла этих систем;
- разработать общую формальную теорию всего многообразия видов и областей человеческой деятельности, которые целесообразно автоматизировать.

В основе предлагаемого нами подхода к построению *интероперабельных интеллектуальных компьютерных систем* лежат следующие принципы:

- смысловое представление знаний, хранимых в памяти интероперабельных интеллектуальных компьютерных систем;
- онтологическая структуризация и систематизация хранимых в памяти знаний;
- децентрализованная ситуационная агенто-ориентированная организация процесса решения задач;
- конвергенция и глубокая (диффузная) интеграция различных моделей решения задач и, как следствие гибридный характер решателей задач;
- смысловая интеграция информации поступающей из вне по разным сенсорным каналам и на разных языках в результате трансляции входной информации на общий язык внутреннего смыслового представления знаний.

*следует отличать**

- ≡ { • индивидуальная интеллектуальная компьютерная система нового поколения
• коллективная интеллектуальная компьютерная система нового поколения
 ⊂ коллектив индивидуальных интеллектуальных компьютерных систем нового поколения
 ⊂ иерархический коллектив интеллектуальных компьютерных систем нового поколения
 := [коллектив интеллектуальных компьютерных систем нового поколения, членами которого могут быть как коллективные, так и индивидуальные интеллектуальные компьютерные системы]
- }

Индивидуальные интеллектуальные компьютерные системы нового поколения имеют следующие особенности:

- Индивидуальные интеллектуальные компьютерные системы нового поколения невозможно декомпозировать на подсистемы, которые можно разрабатывать независимо друг от друга и согласовывать только входами-выходами, реализуя принцип “черного ящика”
- В индивидуальной интеллектуальной компьютерной системы нового поколения необходима конвергенция, совместимость и “осмысленное” взаимодействие самых различных видов знаний и моделей решения задач. То есть индивидуальная интеллектуальная компьютерная система нового поколения должна быть гибридной системой

Технология OSTIS

:= [Комплексная технология Искусственного интеллекта, обеспечивающая

- совместимость всех частных технологий Искусственного интеллекта,
- совместимость и интероперабельность разрабатываемых интеллектуальных компьютерных систем
- поддержку не только проектирования интеллектуальных компьютерных систем, но и всего их жизненного пути

]

⇒ предъявляемые требования*:

- универсальность — Технология OSTIS должна быть ориентирована на разработку и сопровождение интероперабельных интеллектуальных компьютерных систем любого назначения
- Технология OSTIS должна обеспечить перманентную эволюцию самой Технологии OSTIS (самой себя)

Задачи текущего этапа работ в области Искусственного интеллекта в направлении создания интероперабельных интеллектуальных компьютерных систем

Достижение указанной цели требует получения авторов на следующие вопросы:

- Какие требования предъявляются к интеллектуальным компьютерным системам, обеспечивающим указанную комплексную автоматизацию человеческой деятельности
- Почему современные интеллектуальные компьютерные системы указанным выше требованиям не удовлетворяют и, соответственно, почему необходим переход к принципиально новому поколению интеллектуальных компьютерных систем
- Какие функциональные принципы должны лежать в основе интеллектуальных компьютерных систем нового поколения;
- Какие принципы должны лежать в основе максимально автоматизируемой (!) технологии проектирования и поддержки всего жизненного цикла интеллектуальной компьютерных систем нового поколения
- Какие принципы должны лежать в основе структуры и организации различных видов и областей человеческой деятельности в условиях её комплексной и максимально возможной автоматизации с помощью интеллектуальных компьютерных систем нового поколения

К числу текущих задач по созданию теории и технологии интеллектуальных компьютерных систем нового поколения относятся:

- Разработка теории многоагентных систем, агентами в которых являются индивидуальные или коллективные интерооперабельные компьютерные системы.
- Унификация и стандартизация различных моделей представления и обработки знаний. Эффект от данной унификации будет виден не сразу. Но, если этого происходить не будет, мы никогда не приедем к эффективной комплексной автоматизации человеческой деятельности. Эффективное многообразие методов и средств автоматизации приводит только к необоснованному дублированию и повышению сложности использования и сопровождения разрабатываемых систем.
- Конвергенция и интеграция различных направлений Искусственного интеллекта.
- Различные направления Искусственного интеллекта сейчас имеют достаточно высокий уровень развития (signal processing, natural language processing (NLP), логические модели, онтологические модели, модели интеллектуальных компьютерных систем, многоагентные модели). Интеграция всех этих направлений является пусть и достаточно трудоёмкой задачей, но задачей вполне решаемой, в основе решения которой лежит согласование смежных понятий — “точек соприкосновения”. Но почему-то этого не происходит!!
- Конвергенция таких видов деятельности в области Искусственного интеллекта, как:
 - подготовка специалистов в области искусственного интеллекта;
 - инженерная деятельность по разработке прикладных интеллектуальных компьютерных систем нового поколения;

- развитие технологии проектирования и поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения;
- научно-исследовательская деятельность в области Искусственного интеллекта.
- Конвергенция деятельности в области Искусственного интеллекта с другими областями человеческой деятельности. Для развития Технологии OSTIS необходима также конвергенция Технологии OSTIS со всеми видами и областями человеческой деятельности, которые не входят в состав деятельности в области Искусственного интеллекта. То есть дальнейшее развитие Технологии OSTIS носит ярко выраженный междисциплинарный характер. Это означает что все знания, накапливаемые человеческим обществом в самых различных областях должны быть представлены в составе Глобальной базы знаний Экосистемы OSTIS (с помощью порталов научно-технических, административных и прочих знаний), должны быть чётко стратифицированы в виде иерархической системы семантически совместимых онтологий и тем самым превращены в иерархическую систему семантически совместимых компонентов баз знаний ostis-систем различного прикладного назначения.
- Обеспечение семантической совместимости интеллектуальных компьютерных систем нового поколения не только на этапе их проектирования, но и на всех последующих этапах их жизненного цикла.
- Разработка Модели коллективного поведения интероперабельных интеллектуальных компьютерных систем, то есть модели децентрализованного коллективного решения задач в рамках:
 - многоагентной системы, агенты которой являются внутренними агентами индивидуальной интеллектуальной компьютерной системы, взаимодействующими через общую память (через общую базу знаний, хранимую в одной памяти)
 - многоагентной системы, агенты которой являются интероперабельными интеллектуальными компьютерными системами, взаимодействующими через общую базу знаний, хранимую в памяти корпоративной интеллектуальной компьютерной системы или в памяти координатора деятельности временного коллектива интеллектуальной компьютерной системы
- В рамках теории коллективного решения задач следует выделить два типа задач:
 - задача, соответствующая компетенции того коллектива интеллектуальных компьютерных систем, в рамках которого эта задача инициирована
 - задача, выходящая за пределы компетенции того коллектива интеллектуальной компьютерной системы, в рамках которого эта задача инициирована. Эта задача требует формирования временного коллектива, координатором которого становится та интеллектуальная компьютерная система, в рамках которой эта задача инициировалась. Для этого необходимо найти те интеллектуальные компьютерные системы, которые в совокупности обеспечивают необходимую компетенцию.

Отметим при этом, что каждая интероперабельная интеллектуальная компьютерная система (как индивидуальная, так и коллективная) должна знать свою компетенцию для того, чтобы каждая интероперабельная интеллектуальная компьютерная система могла быстро определить, сможет или не сможет она решить ту или иную заданную (возникшую) задачу. Это, в частности, необходимо для формирования временных коллективов интеллектуальной компьютерной системы

- Разработка мощной библиотеки многократно используемых и совместимых компонентов интеллектуальной компьютерной системы нового поколения, которая обеспечивает полную автоматизацию интеграции этих компонентов в процессе сборки проектируемых систем
- Перманентное расширение библиотеки многократно используемых компонентов интероперабельных интеллектуальных компьютерных систем должно стать важнейшим направлением в самых различных областях человеческой деятельности:
 - Разработчики любой интеллектуальной компьютерной интеллектуальной системы должны декомпозировать разработанную интеллектуальную компьютерную систему на множество компонентов, вводимых в состав Библиотеки OSTIS — так, чтобы разработка любой аналогичной системы свелась к сборке компонентов из Библиотеки OSTIS.
 - Научно-техническая деятельность должна сводиться к развитию баз знаний различных интеллектуальных порталов научно-технических знаний. При этом база знаний каждого такого портала должна декомпозироваться на фрагменты, включаемые в состав Библиотеки многократно используемых компонентов баз знаний ostis-системы (для этого они соответствующим образом специфицируются), которые могут иерархически входить друг в друга.
 - Таким образом все(!) должны заботиться о расширении Библиотеки OSTIS, это приводит к существенному снижению трудоёмкости разработки новых ostis-систем в рамках Экосистемы OSTIS. При этом авторство компонентов Библиотеки OSTIS должно поощряться, что является фундаментальной основой развития рынка знаний, экономики знаний.

Если грамотно использовать Технологию OSTIS, то разработка любой ostis-системы сводится в её автоматической сборке из указываемых разработчиком компонентов этой системы. Некоторые компоненты разрабатываемой ostis-системы входят в текущее состояние Библиотеки OSTIS, а некоторые из них дополнительны разработаны. Но при этом каждый такой новый компонент является либо результатом модификации существующего компонента из

Библиотеки OSTIS, либо компонентом, который может и должен быть специфицирован и включен в Библиотеку OSTIS. Таким образом разработчик прикладной ostis-системы должен разработать не только эту систему, но и внести вклад в развитие Библиотеки OSTIS, в результате которого разрабатываемая им ostis-система может быть собрана без дополнительно разрабатываемых компонентов, а только из компонентов Библиотеки OSTIS. Если все разработчики прикладных систем будут так действовать, то темпы повышения уровня автоматизации человеческой деятельности будут существенно возрастать.

Используемые сокращения и предметный указатель OSTIS

Предлагаемый вашему вниманию предметный указатель представляет собой алфавитный перечень всех основных терминов, используемых в данной монографии, которые взаимно-однозначно соответствуют элементам рафинированной семантической сети, представляющей собой базу знаний *Метасистемы OSTIS*, основная часть которой семантически эквивалентна тексту данной монографии.

В рамках внутреннего языка *ostis-систем* (в рамках SC-кода) указанные основные термины называются *основными sc-идентификаторами* (основными внешними идентификаторами sc-элементов – элементов рафинированных семантических сетей).

В данный предметный указатель включаются как русскоязычные, так и англоязычные термины (если нет аналогичного русскоязычного), непереводимые интернациональные названия (например, названия различных программных систем, такие как *Neo4j*, *MySQL* и т.д.), а также различные используемые сокращения.

При этом для каждого основного русскоязычного термина указывается синонимичный ему основной англоязычный термин, например:

sc-идентификатор

:= [строка символов или пиктограмма, взаимно однозначно представляющая соответствующий sc-элемент, хранимый в sc-памяти]

:= [внешний идентификатор sc-элемента]

{

sc-элемент

:= [*sc-element*]

 ∈ *основной англоязычный sc-идентификатор*

 := [*основной sc-идентификатор для англоязычного режима*]

:= [*sc-элемент*]

 ∈ *основной русскоязычный sc-идентификатор*

 := [*основной sc-идентификатор для русскоязычного режима*]

}

⇒ *пояснение**:

[Для каждого ключевого термина в предметном указателе указывается его основной русскоязычный идентификатор и основной англоязычный идентификатор. При этом по умолчанию считается, что русскоязычный термин является основным русскоязычным идентификатором, а соответствующий ему англоязычный – основным англоязычным идентификатором.]

Для каждого неосновного, но часто используемого русскоязычного термина указывается синонимичный ему основной русскоязычный термин. Кроме того, для каждого основного русскоязычного термина указывается ссылка на соответствующую главу, параграф или пункт монографии, где сущность, обозначаемая этим термином, является ключевым знаком.

Абстрактная scp-машина

:= [*Abstract scp-machine*]

⇐ *ключевой знак*:*

- § 3.2.5. *Базовый язык программирования ostis-систем*

абстрактный sc-агент

:= [*abstract sc-agent*]

⇐ ключевой знак*:

- § 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты

абстрактный sc-агент, не реализуемый на Языке SCP

:= [abstract sc-agent, non-implementable in the SCP Language]

⇐ ключевой знак*:

- § 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты

абстрактный sc-агент, реализуемый на Языке SCP

:= [abstract sc-agent, implementable in the SCP Language]

⇐ ключевой знак*:

- § 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты

ассоциативный семантический компьютер

:= [associative semantic computer]

⇐ ключевой знак*:

- Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем
- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

атомарный абстрактный sc-агент

:= [atomic abstract sc-agent]

⇐ ключевой знак*:

- § 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты

базовая ostis-платформа

:= [basic ostis-platform]

⇐ ключевой знак*:

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

библиотека многократно используемых компонентов ostis-систем

:= [library of ostis-systems reusable components]

⇐ ключевой знак*:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Библиотека OSTIS

:= [OSTIS library]

⇐ ключевой знак*:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

блокировка*

:= [lock*]

⇐ ключевой знак*:

- § 3.2.4. Принципы синхронизации деятельности sc-агентов

действие в sc-памяти

:= [action in sc-memory]

⇐ ключевой знак*:

- § 3.2.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой

действие в sc-памяти, инициируемое вопросом

:= [action in sc-memory, initiated by a question]

⇐ ключевой знак*:

- § 3.2.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой

действие редактирования базы знаний

:= [action of knowledge base editing]

⇐ ключевой знак*:

- § 3.2.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой

задача, решаемая в sc-памяти

\coloneqq [*problem, solved in sc-memroy*]

\Leftarrow *ключевой знак**:

- § 3.2.2. *Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*

класс логически атомарных действий

\coloneqq [*class of logically atomic actions*]

\Leftarrow *ключевой знак**:

- § 3.2.2. *Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*

компонентное проектирование

\coloneqq [*component design*]

\Leftarrow *ключевой знак**:

- Глава 5.1. *Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

машина обработки знаний

\coloneqq [*knowledge processing machine*]

\Leftarrow *ключевой знак**:

- Глава 3.2. *Агентно-ориентированные модели гибридных решателей задач ostis-систем*

менеджер многократно используемых компонентов ostis-систем

\coloneqq [*ostis-systems reusable component manager*]

\Leftarrow *ключевой знак**:

- Глава 5.1. *Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

многократно используемый компонент ostis-систем

\coloneqq [*ostis-systems reusable component*]

\Leftarrow *ключевой знак**:

- Глава 5.1. *Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

минимальная конфигурация ostis-системы

\coloneqq [*ostis-system minimal configuration*]

\Leftarrow *ключевой знак**:

- Глава 6.1. *Универсальная модель интерпретации логико-семантических моделей ostis-систем*

неатомарный абстрактный sc-агент

\coloneqq [*non-atomic abstract sc-agent*]

\Leftarrow *ключевой знак**:

- § 3.2.3. *Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*

параметр scp-программы'

\coloneqq [*scp-program parameter'*]

\Leftarrow *ключевой знак**:

- § 3.2.5. *Базовый язык программирования ostis-систем*

*планируемая блокировка**

\coloneqq [*planned lock**]

\Leftarrow *ключевой знак**:

- § 3.2.4. *Принципы синхронизации деятельности sc-агентов*

*приоритет блокировки**

\coloneqq [*lock priority**]

\Leftarrow *ключевой знак**:

- § 3.2.4. *Принципы синхронизации деятельности sc-агентов*

программный вариант ostis-платформы

\coloneqq [*software version of ostis-platform*]

\Leftarrow *ключевой знак**:

- Глава 6.1. *Универсальная модель интерпретации логико-семантических моделей ostis-систем*

расширенная ostis-платформа

:= [extended ostis-platform]

⇐ ключевой знак*:

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

решатель задач ostis-системы

:= [problem solver of ostis-system]

⇐ ключевой знак*:

- Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

специализированная ostis-платформа

:= [specialized ostis-platform]

⇐ ключевой знак*:

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

тип блокировки

:= [lock type]

⇐ ключевой знак*:

- § 3.2.4. Принципы синхронизации деятельности sc-агентов

транзакция в sc-памяти

:= [transaction in sc-memory]

⇐ ключевой знак*:

- § 3.2.4. Принципы синхронизации деятельности sc-агентов

удаляемые sc-элементы*

:= [sc-elements to be deleted*]

⇐ ключевой знак*:

- § 3.2.4. Принципы синхронизации деятельности sc-агентов

Язык SCP

:= [SCP Language]

⇐ ключевой знак*:

- § 3.2.5. Базовый язык программирования ostis-систем

ostis-платформа

:= [ostis-platform]

⇐ ключевой знак*:

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

sc-агент

:= [sc-agent]

⇐ ключевой знак*:

- § 3.2.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты

sc-идентификатор

:= [sc-identifier]

⇐ ключевой знак*:

- Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний

SCg-код

:= [SCg-code]

⇐ ключевой знак*:

- Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний

SCn-код

:= [SCn-code]

⇐ ключевой знак*:

- Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний

SCs-код:= [*SCs-code*]<= *ключевой знак**:

- *Семейство внешних языков *ostis*-систем, близких языку внутреннего смыслового представления знаний*

[*SC-код*]:= [*SC-code*]<= *часто используемый sc-идентификатор**:*sc-структура*<= *ключевой знак**:

- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

sc-машина:= [*sc-machine*]<= *ключевой знак**:

- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей *ostis*-систем*

scp-операнд':= [*scp-operand'*]<= *ключевой знак**:

- *§ 3.2.5. Базовый язык программирования *ostis*-систем*

scp-оператор:= [*scp-operator*]<= *ключевой знак**:

- *§ 3.2.5. Базовый язык программирования *ostis*-систем*

sc-элемент:= [*sc-element*]<= *ключевой знак**:

- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

Правила оформления библиографических источников

Для добавления нового библиографического источника необходимо выполнить следующие шаги:

- Убедиться, что нужный источник еще не присутствует в файле `biblio.bib`, который находится в репозитории с исходными текстами Стандарта OSTIS. В настоящее время все библиографические источники изначально описываются в этом файле.
- Добавить в файл `biblio.bib` описание библиографического источника в соответствии с форматом описания BibTex. Более подробно про формат можно почитать на сайте <https://www.bibtex.com/g/bibtex-format/>. Для помощи в оформлении можно использовать различные бесплатные средства, например, сервис <https://www.doi2bib.org/> позволяет сгенерировать bib-описание на основе идентификатора DOI, кроме того, многие онлайн-библиотеки позволяют выгрузить описание нужного источника в формат BibTex.
- Каждому источнику в соответствии с форматом BibTex присваивается некоторое условное имя (цитатный ключ или просто ключ), по которому затем можно процитировать соответствующий источник. В рамках Стандарта OSTIS рекомендуется цитатные ключи источников в формате BibTex формировать путем транслитерации в латинский алфавит фамилии первого автора и добавления года издания источника, например:

Trudeau1993

Golenkov2011

Если при этом возникает неоднозначность, связанная с тем, что существует несколько работ того же автора в один год, то в конце ключа рекомендуется добавлять строчные латинские буквы *a*, *b*, *c* и так далее, например:

Gribova2015a

Gribova2015b

При формировании ключа для электронного источника или коллективной публикации, где невозможно выделить ключевого автора, рекомендуется формировать ключ из 1-2 английских слов или аббревиатур, позволяющих однозначно идентифицировать соответствующий источник. При использовании нескольких слов их можно соединять знаком нижнее подчеркивание, пробелы в ключах запрещены. При необходимости в конце ключа можно добавлять год издания. Например:

IMS (библиографическая ссылка на сайт Метасистемы OSTIS)

CYPHER (библиографическая ссылка на сайт с описанием языка Cypher)

AIDictionary1992 (библиографическая ссылка на Словарь по искусственно-интеллекту 1992 года издания)

Для добавленного источника необходимо описать его идентификатор, который далее будет использоваться в рамках текста Стандарта. Это делается при помощи BibTex поля `shorthand`, например (см. *Правила идентификации библиографических источников*):

```
shorthand = {Trudeau R.J.Intro tGT-1993bk}
shorthand = {Duchi J..AdaptSMfOLaSO-2011art}
shorthand = {Грибова В.В..БазовТРИСиОП-2015ст}
```

Далее этот идентификатор может использоваться как в формальном тексте, также как и идентификатор любой другой сущности, так и в рамках естественно-языкового текста. Для автоматической вставки идентификатора библиографического источника в формальный либо естественно-языковой текст используется команда

```
\scncite{<цитатный ключ>}
```

Пример исходного кода:

```
\scnheader{конвергенция\scnsupergroupsign}
\scnidtf{уровень конвергенции (близости)\scnsupergroupsign}
\scnsuperset{конвергенция кибернетических систем\scnsupergroupsign}
\begin{scnreftolist}{ключевой знак}
\scnitem{\scncite{Yankovskaya2017}}
\scnitem{\scncite{Palagin2013}}
\scnitem{\scncite{Yankovskaya2010}}
\scnitem{\scncite{Kovalchuk2011}}
\end{scnreftolist}
```

Результат компиляции:

конвергенция[^]
:= [уровень конвергенции (близости)[^]]

▷ *конвергенция кибернетических систем[^]*

⇐ *ключевой знак*:*

- *Янковская А.Е..ГибридИСЭДНИБ-2017см*
- *Палагин А.В..ПроблТиРИ-2013см*
- *Янковская А.Е.АналиДиЗнОКННиНУ-2010см*
- *Ковальчук М.В.КонвеНиТПвБ-2011см*

□ Для каждого источника крайне желательно добавить его краткую аннотацию. Это делается при помощи BibTeX поля annotation, например:

`annotation = {В этой книге представлены исследования по внедрению концептуальных основ, стратегий,`

В рамках аннотации допускается использование средств форматирования естественно-языковых текстов, принятых в рамках Стандарта OSTIS, например, выделение курсивом или полужирным курсивом.

Для вставки аннотации в формальный scn-текст используется команда

`\scnciteannotation{<цитатный ключ>}`

Пример исходного кода:

```
\scnheader{\scncite{McBride2021}}
\scnciteannotation{McBride2021}
```

Результат компиляции:

McBride R..отОнтолоАAanA-2021art

⇒ *аннотация*:*

[В работе с философской точки зрения исследуются понятия "действие" и "актор". Дается определение понятию действия как целенаправленному поведению и понятию актора как субъекту этого целенаправленного поведения.]

Библиография OSTIS

В *Библиографии OSTIS* приводится список библиографических источников в алфавитном порядке их идентификаторов, при этом вначале перечисляются русскоязычные источники, а затем — англоязычные.

Каждый библиографический источник имеет соответствующий уникальный идентификатор, а также формальную спецификацию.

Идентификаторы статей, книг и других печатных работ строятся следующим образом:

- Пишется фамилия первого автора на том языке, на котором опубликована данная работа. Затем через пробел ставятся инициал(-ы) первого автора
- Если работа опубликована с участием только одного автора, то ставится одна точка, если нескольких — две точки. Если работа представляет собой коллективную публикацию под редакцией кого-либо, то вместо фамилии автора указывается фамилия и инициалы главного редактора и вместо двух точек ставится сочетание символов “.ред.” или “.ед.” в зависимости от языка.
- Пишется первых пять букв первого слова из названия работы на том языке, на котором опубликована данная работа. Если первое слово названия работы служебное (предлог, частица, artikel и т.д.), то ставится первая строчная буква этого слова, а первых пять букв берется из следующего (первого значимого) слова.
- Перечисляются заглавные (прописные) первые буквы всех остальных слов названия работы за исключением служебных слов, таких как предлоги, частицы, артикли и т.п. Для всех служебных слов указываются строчные первые буквы. Если название содержит очень много слов, то допускается использовать только первые 5-7 слов. Если служебное слово идет сразу же после первого значимого слова, то перед соответствующей строчной буквой ставится пробел.
- Ставится дефис
- Указывается год издания работы
- Указывается 2-3 буквенный код, обозначающий тип работы, на том языке, на котором она опубликована, например:
 - *кн* или *bk* — книга
 - *ст* или *art* — статья

Идентификаторы электронных и прочих ресурсов формируются аналогичным образом, с учетом того, что опускается год издания и фамилии авторов, а также ставится буквенный код *эл* для обозначения электронного ресурса. Например, *МетасOSTIS-2022эл*, *Cypher-2022эл*.

При спецификации конкретного библиографического источника в разделе библиографии указываются:

- Идентификатор библиографического источника, составленный в соответствии с указанными выше правилами.
- Его библиографическое описание (библиографический идентификатор), составленное в соответствии с каким-либо общепринятым стандартом (ГОСТ, IEEE, ACM и т.д.).
- Указывается аннотация данного библиографического источника.
- Перечисляются ключевые знаки (понятия или конкретные сущности) для описываемого библиографического источника. При этом подразумевается, что и имя (термин), и трактовка данного ключевого знака в описываемом источнике и в рамках Стандарта OSTIS совпадают. В противном случае имя такой ключевой сущности описывается как ключевой термин.
- Перечисляются ключевые термины для описываемого библиографического источника. При этом для каждого термина желательно при возможности указывать соответствующие понятие или конкретную сущность (ключевой знак), описываемые в рамках Стандарта OSTIS, для которых данный термин может рассматриваться как синонимичный идентификатор. Примеры использования ключевых терминов и ключевых знаков можно найти в спецификации источников *Поспелов Д.А. СтудяУП-1986кн* и *Голенков В.В..оОбучеИСкОСиР-2018ст*.
- Указываются разделы Стандарта OSTIS (Монографии OSTIS), для которых данный библиографический источник является важным.

- Приводятся цитаты из данного библиографического источника. При необходимости указывается часть источника, из которой непосредственно взята цитата. Примеры указания цитат можно найти в спецификации источников *Баринов И.И.. ФормиСРКпИИ-2021см* и *Поспелов Д.А.СитуаУТП-1986кн*.
- Указывается любая другая информация об описываемом библиографическом источнике.

Для ссылки на библиографический источник в естественно-языковом или формальном тексте используется его идентификатор, составленный по рассмотренным выше правилам. Например:

- “Работа *Голенков В.В.оОбучеИСкОСиР-2018см* посвящена вопросам обучения и обучаемости в интеллектуальных системах.”
- “Основные принципы многоагентных систем рассматриваются в ряде работ (*Wooldridge M.aIntro tMS-2009bk*, *Тарасов В.Б.оМногоСкИО-2002кн*).”

Альтшуллер Г.С..НайтиИ:ВвТРИЗ-2010кн

:= стандартное библиографическое описание*:

[Г. С. Альтшуллер, *Найти идею: Введение в ТРИЗ – теорию решения изобретательских задач*, 3-е изд. М.: Альпина Паблишер, 2010]

⇒ аннотация*:

[Книга посвящена Теории решения изобретательских задач – ТРИЗ.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

Анцыферов С.С.ОценкКИС-2013art

:= стандартное библиографическое описание*:

[С. С. Анцыферов, “Оценка уровня качества интеллектуальных систем,” *Искусственный интеллект*, с. 316—323, 2013]

⇒ аннотация*:

[annotation]

⇐ библиографическая ссылка*:

- § 1.1.1. Понятие интеллектуальной кибернетической системы

Баринов И.И.. ФормиСРКпИИ-2021см

:= стандартное библиографическое описание*:

[И. Баринов и др., “Формирование стратегии развития Комитета по искусственному интеллекту в Научно-образовательном центре «Инженерия будущего»,” *Онтология проектирования*, т. 11, № 3, с. 260—293, сент. 2021. DOI: 10.18287/2223-9537-2021-11-3-260-293. url: <https://doi.org/10.18287/2223-9537-2021-11-3-260-293>]

⇒ часть*:

Баринов И.И.. ФормиСРКпИИ-2021см/с. 270

⇒ цитата*:

[Выбор многоагентных технологий объясняется тем, что в настоящее время любая сложная производственная, логистическая или другая система может быть представлена набором взаимодействий более простых систем до любого уровня детальности, что обеспечивает фрактально-рекурсивный принцип построения многоярусных систем, построенных как открытые цифровые колонии и экосистемы ИИ. В основе многоагентных технологий лежит распределенный или децентрализованный подход к решению задач, при котором динамически обновляющаяся информация в распределенной сети интеллектуальных агентов обрабатывается непосредственно у агентов вместе с локально доступной информацией от "соседей". При этом существенно сокращаются как ресурсные и временные затраты на коммуникации в сети, так и время на обработку и принятие решений в центре системы (если он все-таки есть).]

⇐ библиографическая ссылка*:

- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Батыршин И.З.ОсновОНииЛО-2001кн

:= стандартное библиографическое описание*:

[И. З. Батыршин, *Основные операции нечеткой логики и их обобщения*. Казань: Отечество, 2001]

⇒ аннотация*:

[В книге рассматриваются свойства операций конъюнкции, дизъюнкции и отрицания нечеткой логики и определяемых ими операций пересечения, объединения и дополнения нечетких множеств]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Беркинблит М.Б.НейроСЭУП-1993кн

:= стандартное библиографическое описание*:

[М. Б. Беркинблит, Нейронные сети : эксперим. учеб. пособие. М.: МИРОС : Всерос. заоч. многопредмет. шк. Рос. акад. наук, 1993]

⇒ аннотация*:

[Книга представляет собой учебник про искусственным нейронным сетям.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Борисов А.Н.ПострИСOnЗсПИК-2014ст

:= стандартное библиографическое описание*:

[А. Н. Борисов, “Построение интеллектуальных систем, основанных на знаниях, с повторным использованием компонентов,” в Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2014) : материалы Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2014, с. 97—102]

⇒ аннотация*:

[В работе рассмотрены теоретические основы и средства построения интеллектуальных систем с многократным использованием компонентов на основе методов инженерной онтологии. Приведены два основных семейства инструментальных средств: модельно-ориентированный подход и подход, использующий декомпозицию решаемой проблемы. Описаны основные языки представления онтологий, а также программные средства построения онтологий. Рассмотрены средства представления декларативных знаний и механизмы обработки процедурных знаний, а также средства взаимодействия онтологических баз знаний и декларативных правил. Проанализированы дискуссионные вопросы развития реляционных языков и онтологической семантики.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Вагин В.Н..Досто и ПВвИС-2008кн

:= стандартное библиографическое описание*:

[В. Н. Вагин и др., Достоверный и правдоподобный вывод в интеллектуальных системах, 2-е изд., испр. и доп., под ред. В. Н. Вагина и Д. А. Поспелова, ред. М.: Физматлит, 2008]

⇒ аннотация*:

[Рассматриваются методы достоверного (дедуктивного) и правдоподобного (абдуктивного, индуктивного) выводов в интеллектуальных системах различного назначения. Приводятся методы дедуктивного вывода на графовых структурах. Описываются как классические, так и немонотонные модальные логики. Приводятся основы теории аргументации и методы абдуктивного вывода.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Варшавский В.А..ОркесИБД-1984кн

:= стандартное библиографическое описание*:

[В. И. Варшавский и Д. А. Поспелов, Оркестр играет без дирижера: размышления об эволюции некоторых технических систем. Москва : Наука. Главная редакция физико-математической литературы, 1984]

⇒ аннотация*:

[Мир, создаваемый человеком в технических системах во многом похож на тот, который окружает человека в природе. И в искусственном мире техники могут происходить процессы, подобные эволюции живых организмов. Возникают колонии и сообщества технических систем, формируются «сверхорганизмы» типа муравейника, возникают «коллективы», живущие по своим законам. Авторы книги анализируют эти аналогии и рассматривают принципы построения управления в таких технических системах, которые во многом отличаются от привычных схем управления. Для чтения книги не требуется никакой специальной подготовки, хотя она обращена не только к так называемому широкому читателю, но и к специалистам, работающим в области управления и кибернетики.]

⇐ библиографическая ссылка*:

- § 1.1.1. Понятие интеллектуальной кибернетической системы

Владимиров А.Н..ПрогрКУПРАОЛВсЛВСиОМСП-2010ст

:= стандартное библиографическое описание*:

[А. Н. Владимиров и др., "Программный комплекс "УДАВ": практическая реализация активного обучаемого логического ввода с линейной вычислительной сложностью на основе миарной сети правил," в Труды НИИР : сб. науч ст., Науч.-исслед. ин-т радио, М., 2010, с. 108—116]

⇒ аннотация*:

[В статье описан программный комплекс "УДАВ".]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Волченкова Н.И.ТехноМРиЖБПВМнЯФ-1984ст

:= стандартное библиографическое описание*:

[Н. И. Волченкова, "Технология многомашинной реализации и жизнеобеспечения библиотек подпрограмм вычислительной математики на языке Фортран," 1984]

⇒ аннотация*:

[Библиотеки подпрограмм создаются и используются уже много лет, однако, и сейчас в этой области существуют еще не решенные проблемы. В качестве одной из форм организации вычислений на ЭВМ авторами предлагалось использование библиотеки подпрограмм, и уже тогда была показана эффективность такого метода программирования.]

⇐ библиографическая ссылка*:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

>>>> 2259c7b (added bibliography for Chapter 1.2)

Гладков Л.А..ГенетАУП-2006кн

:= стандартное библиографическое описание*:

[Л. А. Гладков, В. В. Курейчик и В. М. Курейчик, Генетические алгоритмы : учеб. пособие, 2-е испр. и доп. М.: Физматлит, 2006]

⇒ аннотация*:

[Книга содержит описание фундаментальных основ генетических алгоритмов и эволюционного моделирования.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Голенков В.В..БазовПТЯSCL-1996мо

:= стандартное библиографическое описание*:

[К. В. Голенков В.В., Базовые преобразования текстов языка SCL для реализации механизмов дедуктивного логического вывода. Минск: ИТК АН Беларусь, 1996]

⇒ аннотация*:

[Рассмотрены графодинамические ассоциативные модели обработки информации и соответствующие им абстрактные графодинамические ассоциативные машины, осуществляющие параллельную асинхронную обработку семантических сетей в графодинамической ассоциативной памяти. Одним из перспективных направлений применения предложенных в данной работе моделей являются интеллектуальные обучающие системы и распределенные интеллектуальные системы.]

⇐ библиографическая ссылка*:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Голенков В.В.ред.Пред и ОЗвГАМ-2001кн

:= стандартное библиографическое описание*:

[В. Голенков, О. Елисеева и др., Представление и обработка знаний в графодинамических ассоциативных машинах. Минск: БГУИР, 2001]

⇒ аннотация*:

[Рассмотрены графодинамические ассоциативные модели представления и обработки знаний в системах искусственного интеллекта. В основе этих моделей лежит, во-первых, представление всевозможных знаний в виде однородных семантических сетей, имеющих базовую теоретико-множественную интерпретацию, и, во-вторых, трактовка обработки знаний как графодинамического процесса, т.е. как процесса изменения конфигурации семантических сетей.]

⇒ ключевой знак*:

- sc-элемент
 - sc-узел
 - sc-дуга
- ⇐ библиографическая ссылка*:
- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Голенков В.В..ГрафоАМиСПОИвСИИ-2011см

- := стандартное библиографическое описание*:
- [В. В. Голенков и Н. А. Гулякина, “Графодинамические ассоциативные модели и средства параллельной обработки информации в системах искусственного интеллекта,” Доклады БГУИР, с. 92—101, 2004]
- ⇒ аннотация*:
- [Рассмотрены графодинамические ассоциативные модели обработки информации и соответствующие им абстрактные графодинамические ассоциативные машины, осуществляющие параллельную асинхронную обработку семантических сетей в графодинамической ассоциативной памяти. Одним из перспективных направлений применения предложенных в данной работе моделей являются интеллектуальные обучающие системы и распределенные интеллектуальные системы.]
- ⇐ библиографическая ссылка*:
- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Голенков В.В..оОбучеИСкОСиР-2018см

- := стандартное библиографическое описание*:
- [В. Голенков, Н. А. Гулякина и др., “От обучения интеллектуальных систем к обучению средств их разработки,” в Открытые семантические технологии проектирования интеллектуальных систем, сер. Вып. 2, Минск : БГУИР, 2018, с. 81—98]
- ⇒ аннотация*:
- [Работа содержит описание обучения интеллектуальных систем и их средств разработки]
- ⇐ библиографическая ссылка*:
- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Голенков В.В..СтандOTOPPiЭССГИКС-2021кн

- := стандартное библиографическое описание*:
- [В. Голенков, Н. Гулякина и Д. Шункевич, Стандарт открытой технологии онтологического проектирования, производства и В. Голенков, ред. Минск: Бестпринт, 2021, с. 690]
- ⇐ библиографическая ссылка*:
- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Головко В.А.НейроCOOиПУП-2001кн

- := стандартное библиографическое описание*:
- [В. А. Головко, Нейронные сети: обучение, организация и применение : учеб. пособие. М.: Журн. «Радиотехника», 2001, (Нейрокомпьютеры и применение ; кн. 4)]
- ⇒ аннотация*:
- [В книге изложены математические и алгоритмические аспекты функционирования нейронных сетей с прямыми и обратными связями; отражены вопросы самоорганизации, отказоустойчивости и реализации нейронных сетей на стационарных процессорах; большое внимание уделено применению и проектированию нейронных сетей для решения различного рода задач]
- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Головко В.А..НейроТОД-2017кн

- := стандартное библиографическое описание*:
- [В. А. Головко и В. В. Краснопрошин, Нейросетевые технологии обработки данных. Минск: Издательство БГУ, 2017, ISBN: 9789855664674]
- ⇒ аннотация*:
- [Изложены математические и алгоритмические аспекты функционирования искусственных нейронных сетей]
- ⇐ библиографическая ссылка*:
- Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах

Горбань А.Н..НейроСиПК-1996кн

- := стандартное библиографическое описание*:

[А. Н. Горбань и Д. А. Россиев, Нейронные сети на персональном компьютере. Новосибирск: Наука, 1996]

⇒ аннотация*:

[В книге описаны элементы нейронных сетей, способы их соединения и настройки для решения различных задач, набор дополнительных элементов, необходимых для создания полноценного нейрокомпьютера, и различные алгоритмы обучения нейронных сетей]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Городецкий В.И..БазовОКПААиР-2015ст

:= стандартное библиографическое описание*:

[В. И. Городецкий, В. В. Самойлов и Д. В. Троцкий, “Базовая онтология коллективного поведения автономных агентов и ее расширения,” Изв. Рос. акад. наук. Теория и системы упр., № 5, с. 102—121, 2015]

⇒ аннотация*:

[В работе рассмотрен онтологический подход к проектированию многоагентных систем.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт Пункт 3.2.1.2. ??

Грибова В.В..ПроектIACPaaSКИССОВ-2011ст

:= стандартное библиографическое описание*:

[В. В. Грибова, А. С. Клещев, Д. А. Крылов, Ф. М. Москаленко, С. В. Смагин и др., “Проект IACPaaS. Комплекс для интеллектуальных систем на основе облачных вычислений,” Искусств. интеллект и принятие решений, № 1, с. 27—35, 2011]

⇒ аннотация*:

[В данной работе описан проект IACPaaS и комплекс интеллектуальных систем на основе облачных вычислений.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Грибова В.В..БазовТРИСиОПРБЗиРЗ-2015ст

:= стандартное библиографическое описание*:

[В. В. Грибова, А. С. Клещев, Д. А. Крылов, Ф. М. Москаленко, В. А. Тимченко и др., “Базовая технология разработки интеллектуальных сервисов на облачной платформе IACPaaS. Ч. 1. Разработка базы знаний и решателя задач,” Програм. инженерия, № 12, с. 3—11, 2015]

⇒ аннотация*:

[В данной работе описана технология разработки базы знаний и решателя задач в рамках платформы IACPaaS.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Деменков Н.П.НечетУвТСУП-2005кн

:= стандартное библиографическое описание*:

[Н. П. Деменков, Нечеткое управление в технических системах : учеб. пособие. М.: Изд-во Моск. гос. техн. ун-та, 2005]

⇒ аннотация*:

[Рассматриваются вопросы, связанные с использованием теории нечеткого управления для решения неформализованных задач оптимизации в технических системах]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Евгунев Г.Б.Экспе кССОИЗ-2019ст

∈ статья

:= стандартное библиографическое описание*:

[Г. Б. Евгунев, “Экспертопедия как средство создания онтологического Интернета знаний,” Онтология проектирования, т. 9, № 3(33), с. 307—320, 2019, DOI: 10.18287/2223-9537-2019-9-3-307-320]

⇒ **аннотация*:**

[Предложена методология создания семантических вики-систем для представления знаний с использованием технологий искусственного интеллекта. Методология основана на многоагентных методах создания баз знаний и пригодна для разработки систем проектирования и управления для цифровых интеллектуальных производств. При формировании внешнего представления баз знаний могут использоваться национальные языки. Проанализированы схемы ввода знаний в компьютер. На основе технологии экспертного программирования для создания баз знаний предложена семантическая вики-система, которая может быть названа «Экспертопедией». Возможность замены словаря базы знаний позволяет обеспечить выполнение концептуального требования вики-систем: использование любых языков. В базе знаний имеются возможности добавления модулей, выбора модуля-аналога, трансляции модуля знаний на один из языков программирования, тестирования полученного результата, а также определения входимости модуля в базы знаний и удаления выбранного модуля. Эти действия могут выполняться любым носителем знаний, что позволяет реализовать вторую концептуальную основу вики-систем: обеспечение возможности независимого пополнения и корректирования содержания системы. Разработана метаонтология Экспертопедии. Подробно рассмотрены методы создания многоагентных баз знаний для проектирования и управления в машиностроении. Приведена архитектура агента САПР. Даны примеры применения многоагентных систем для создания интеллектуальных систем полуавтоматического проектирования изделий машиностроения. Описана технология экспертного программирования]

⇒ **ключевой знак*:**

- семантическая сеть

Евгунев Г.Б.Индус5.0кИИЗиИВ-2019см∈ **статья**:= **стандартное библиографическое описание*:**

[Г. Б. Евгунев, “Индустрия 5.0 как интеграция Интернета знаний и Интернета вещей,” Онтология проектирования, т. 9, № 1(31), с. 7—23, 2019, DOI: 10.18287/2223-9537-2019-9-1-7-23]

⇒ **аннотация*:**

[Предложена методология создания систем класса «Индустрия 5.0» с использованием технологий искусственного интеллекта. Методология основана на многоагентных методах создания баз знаний и пригодна для разработки систем проектирования и управления для цифровых интеллектуальных производств. Разработана интегрированная структура Интернета знаний и Интернета вещей. Проанализирован жизненный цикл изделий машиностроения и предложены методы применения Интернета знаний и Интернета вещей на различных этапах этого цикла. Приведена функциональная декомпозиция основных этапов жизненного цикла. Даны концептуальные основы Интернета знаний. Разработаны многоагентные методы создания баз знаний. Предложена метаонтология инженерных агентов. Описаны принципы построения многоагентных систем полуавтоматического проектирования изделий. Приведено описание возможностей интеллектуальных систем программирования обработки на оборудовании с ЧПУ в части формирования траектории и областей переходов. Описаны возможности интеллектуальных систем проектирования и нормирования технологических процессов. Предложено использование стандарта IDEF3 для создания метамоделей технологических процессов и модифицированных маршрутных карт для формирования баз знаний. Дано описание интеллектуальной системы оперативного управления машиностроительным производством.]

Емельянов В.В..Теори иПЭМ-2003кн:= **стандартное библиографическое описание*:**

[В. В. Емельянов, В. В. Курейчик и В. М. Курейчик, Теория и практика эволюционного моделирования. М.: Физматлит, 2003]

⇒ **аннотация*:**

[Книга содержит описание генетических и синергетических подходов, а также средства эволюционного моделирования.]

⇐ **библиографическая ссылка*:**

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Еремеев А.П.ПострРФнБТЛвСПРвУН-1997см:= **стандартное библиографическое описание*:**

[А. П. Еремеев, “Построение решающих функций на базе тернарной логики в системах принятия решений в условиях неопределенности,” Изв. Рос. акад. наук. Теория и системы упр., № 2, с. 138—143, 1997]

⇒ **аннотация*:**

[В данной работе рассматриваются проблемы существующих методов, средств и технологий построения машин обработки знаний и ставится проблема отсутствия средств, позволяющих относительно неподготовлен-

ному разработчику в удовлетворительные сроки проектировать машины обработки знаний для прикладных интеллектуальных систем различного назначения]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Ефимов Е.И.РешатИЗ-1982кн

:= стандартное библиографическое описание*:

[Е. И. Ефимов, Решатели интеллектуальных задач. М.: Наука, 1982]

⇒ аннотация*:

[Книга посвящена рассмотрению элементов теории интеллектуальных решателей и ее практических приложений]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Жукова Н.А.ОнтоЛМТДоСТО-2019ст

∈ статья

:= стандартное библиографическое описание*:

[Н. А. Жукова, “Онтологические модели трансформации данных о состоянии технических объектов,” Онтология проектирования, т. 9, № 3(3), с. 345—360, 2019, DOI: 10.18287/2223-9537-2019-9-3-345-360]

⇒ аннотация*:

[Для наблюдения за техническими объектами на них устанавливается множество датчиков, которые позволяют собирать данные о состоянии этих объектов. В зависимости от свойств собираемых данных и решаемых практических задач определяются процессы их обработки. При изменении данных или требований к результатам их обработки процессы перестраиваются. Для обеспечения эффективной обработки предложены метод и модель трансформации данных. Они предусматривают формальное описание последовательностей трансформаций в общем виде, а также их представление в виде процессов обработки. Для практической реализации метода предложено представлять трансформации в виде связанных онтологических моделей, включающих функциональную, информационную и процессно-сервисные модели. Построение моделей осуществляется с использованием правил, множество которых представляется в виде управляющей модели. Приводится онтологическая модель для трансформации данных, представленных в виде временных рядов. Структуру её классов определяют классификаторы, определённые для исходных временных рядов и их представлений, а также методов, алгоритмов и процедур их обработки и оценки результатов. Приводится пример обработки результатов измерений значений параметров давления технического объекта космического назначения при проведении технического контроля его состояния]

Кахро М.В..ИнстрСПЕСЭВМ-1988кн

:= стандартное библиографическое описание*:

[М. В. Кахро, А. П. Калья и Э. Х. Тыугу, Инstrumentальная система программирования ЕС ЭВМ (ПРИЗ). М.: Финансы и статистика, 1988]

⇒ аннотация*:

[В книге описывается оригинальная отечественная разработка – система программирования ПРИЗ. Ориентированное на создание всевозможных прикладных программ, снабжённых проблемно-ориентированными языками высокого уровня, система ПРИЗ обеспечивает существенное повышение производительности труда программистов.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Колесников А.В.ГибриИСТИР-2001кн

:= стандартное библиографическое описание*:

[А. В. Колесников, Гибридные интеллектуальные системы: Теория и технология разработки, А. М. Яшин, ред. СПб.: Изд-во СПбГТУ, 2001, ISBN: 5-7422-0187-7]

⇒ аннотация*:

[Монография содержит материалы исследований по гибридным интеллектуальным системам за период 1986–2000 гг. Предназначена для широкого круга читателей.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем
- Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

Комарцова Л.Г.Нейро-2004кн

- := стандартное библиографическое описание*:
 [Л. Г. Комарцова и А. В. Максимов, Нейрокомпьютеры. - 2-е изд. (Информатика в техническом университете).
 Москва: МГТУ им. Баумана, 2004]
- ⇒ аннотация*:
 [В учебнике изложены вопросы современной теории нейрокомпьютеров. Приведен анализ различных архитектур вычислительных устройств с параллельной организацией работы. Рассмотрен биологический аналог параллельной организации обработки информации. Большое внимание удалено разновидностям построения формальных нейронов, технологий сетей и классическим методам их обучения, методам подготовки задач для решения на нейрокомпьютерах. Приведены оригинальные результаты применения нейронных сетей для решения систем дифференциальных уравнений и степенных рядов в конструировании нейросетевых алгоритмов; обучения нейронных сетей на базе генетического алгоритма и теории аддитивного резонанса. Представлены программные системы эмуляции нейронных сетей, разработанных в Калужском филиале МГТУ им. Н.Э.Баумана. Уделено внимание аппаратной реализации нейрокомпьютеров, в том числе и на отечественной элементной базе.]
- ⇐ библиографическая ссылка*:
 • Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

Кулик Б.А.ЛогикЕР-2001кн

- := стандартное библиографическое описание*:
 [Б. А. Кулик, Логика естественных рассуждений. СПб.: Нев. диалект, 2001]
- ⇒ аннотация*:
 [Книга по индуктивному выводу]
- ⇐ библиографическая ссылка*:
 • § 1.1.2. Понятие интеллектуальной многоагентной системы
 • § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Курбатов С.С..ПрогрОдАРЗпП-2016ст

- := стандартное библиографическое описание*:
 [С. С. Курбатов, А. П. Лобзин и Г. К. Хахалин, “Программное обеспечение для автоматического решения задач по планиметрии,” в Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : матер. Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 159—164]
- ⇒ аннотация*:
 [В статье рассмотрена система автоматического решения задач по планиметрии с использованием онтологии и естественно-языкового интерфейса. В рамках системы разработано программное обеспечение, использующее онтологию при поиске решения задачи, сформулированной на предметно-ориентированном естественном языке. Для найденного решения разработана визуализация на предметном и онтологическом уровнях. Программное обеспечение включает макросы Word для работы с графикой и решатель, реализованный в программной среде онтологии]
- ⇐ библиографическая ссылка*:
 • § 1.1.2. Понятие интеллектуальной многоагентной системы
 • § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Летичевский А.А..ИнсерП-2003ст

- := стандартное библиографическое описание*:
 [А. Летичевский и др., “Инсерционное программирование,” Кибернетика и системный анализ, № 1, с. 19—32, 2003]
- ⇐ библиографическая ссылка*:
 • § 1.2.3. Принципы, лежащие в основе многоагентных моделей решателей задач интеллектуальных компьютерных систем нового поколения

Летичевский А.А..ИнсерМ-2012ст

- := стандартное библиографическое описание*:
 [Л. А. Ад., “Инсерционное моделирование,” Управляющие системы и машины, № 6, с. 3—14, 2012]
- ⇒ аннотация*:
 [Работа посвящена инсерционному программированию.]

- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - Глава 3.2. Агентно-ориентированные модели гибридных решателей задач *ostis*-систем

МетасОСТИС-2022эл

- := стандартное библиографическое описание*:
 [Метасистема OSTIS [Электронный ресурс], Режим доступа: <http://ims.ostis.net>. — Дата доступа: 04.12.2022]
- ⇒ аннотация*:
 [Официальный сайт Метасистемы OSTIS]
- ⇐ библиографическая ссылка*:
- Глава 7.2. Метасистема OSTIS

Москаленко Ф.М..ТехноРРЗИСдОБИНОРРОИ-2016ст

- := стандартное библиографическое описание*:
 [В.Ф.М.Москаленко, Технология разработки решателей задач интеллектуальных сервисов для облачной платформы IACPRA 2016, с. 39—44]
- ⇒ аннотация*:
 [***]
- ⇐ библиографическая ссылка*:
- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов *ostis*-систем

Нариньни А.С.НЕФакКВ-2004ст

- := стандартное библиографическое описание*:
 [А. Нариньни, “НЕ-факторы: краткое введение,” Новости искусственного интеллекта, № 2, с. 52—63, 2004]
- ⇒ аннотация*:
 [В статье рассмотрено понятие НЕ-факторов, дается их классификация и описание.]
- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

Подколзин А.С.КомпьюоМЛПАРиЯРЗ-2008кн

- := стандартное библиографическое описание*:
 [А. С. Подколзин, Компьютерное моделирование логических процессов: архитектура решателя и язык решателя задач. М.: Физматлит, 2008]
- ⇒ аннотация*:
 [В книге представлено описание разработанного автором пакета прикладных программ "Логическая система "Искра, обобщающего многолетний опыт компьютерного моделирования логических процессов, в результате которого возникла развитая технология обучения "решателя". В основном моделировались процессы решения математических задач]
- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Пойа Д.Матем иПР-1975кн

- := стандартное библиографическое описание*:
 [Д. Пойа, Математика и правдоподобные рассуждения, пер. с англ. И. А. Ванштейна ; под ред. С. А. Яновской, ред. М.: Наука, 1975]
- ⇒ аннотация*:
 [В книге описывается какими путями добываются новые факты в математике, с какой степенью доверия следует относиться к той или иной математической гипотезе]
- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Попов Э.В..ИскусИССОиЭС-1990кн

- := стандартное библиографическое описание*:
 [Э. В. П. [др.], ред., Искусственный интеллект справочник : в 3 кн. — М. : Радио и связь, 1990. — Кн. 1 : Системы общения и

⇒ *аннотация**:

[В данной книге описаны системы общения и экспертные системы.]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Поспелов Д.А. СитуаУТП-1986кн

:= *стандартное библиографическое описание**:

[Д. А. Поспелов, Ситуационное управление: теория и практика. М. : Наука, 1986]

⇒ *аннотация**:

[Ситуационное управление – метод управления сложными техническими и организационными системами, основанный на идеях теории искусственного интеллекта: представление знаний об объекте управления и способах управления им на уровне логико-лингвистических моделей, использование обучения и обобщения в качестве основных процедур при построении процедур управления по текущим ситуациям, использование дедуктивных систем для построения многошаговых решений.]

⇒ *ключевой термин**:

- [язык ситуационного управления]

⇒ *ключевой знак**:

- *ситуационное управление*

⇒ *цитата**:

[Фактически из-за сложности объектов управления, которыми мы занимаемся, нет надежды на то, что исходные знания о них и способах управления ими будут достаточно полны. Поэтому система управления подобного типа принципиально должна быть открытой системой. Она должна иметь возможность корректировать свои знания об объекте и методах управления им.]

⇐ *библиографическая ссылка**:

- Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем
- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Поспелов Д.А. МоделиРОАМА-1989кн

:= *стандартное библиографическое описание**:

[Д. А. Поспелов, Моделирование рассуждений: опыт анализа мыслительных актов. М.: Радио и связь, 1989]

⇒ *аннотация**:

[Нечеткий вывод]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Пратт Т. ЯзыкиПРиР-2002кн

:= *стандартное библиографическое описание**:

[Т. Пратт и М. Зелкович, Языки программирования: разработка и реализация : пер. с англ. 4-е, под общ. ред. А. Матросова, ред. СПб.: Питер : Питер прнт, 2002]

⇒ *аннотация**:

[В книге рассматриваются общие концепции разработки и реализации языков программирования, а также основы формальных грамматик и конечных автоматов - математических моделей, используемых для определения и реализации языков программирования]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Раговский А.П. ИнтелМСДВиОСО-2011ст

:= *стандартное библиографическое описание**:

[А. П. Раговский, “Интеллектуальная многоагентная система дедуктивного вывода на основе сетевой организации,” Искусств. интеллект и принятие решений, № 2, с. 73—86, 2011]

⇒ *аннотация**:

[В статье рассматриваются вопросы практической реализации методов теории многоагентных систем в организации интеллектуальной прикладной системы, которая в качестве основного механизма обработки знаний использует процедуру дедуктивного вывода. Исследуются задачи создания модели агентов, а также организа-

ции многоагентной системы. Разрабатывается механизм опосредованной централизованной коммуникации интеллектуальных агентов. Обсуждаются архитектура и принципы функционирования интеллектуальной прикладной многоагентной системы]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Рыбаков В.В.Мультиагентные временные логики и проблемы допустимости в описании агентных систем

:= стандартное библиографическое описание*:

[В. В. Рыбаков, “Мультиагентные Временные Нетранзитивные Линейные Логики, Проблема Допустимости,” Алгебра и Логика, т. 59, с. 123—141, март 2020. DOI: [10.1007/s10469-020-09581-0](https://doi.org/10.1007/s10469-020-09581-0)]

⇒ аннотация*:

[В данной статье изучается расширение временной логики – мультиагентную логику на моделях с нетранзитивным линейным временем (в некотором смысле расширение интервальной логики). Предлагаемые реляционные модели допускают пробелы в отношениях достижимости агентов – и эти отношения в принципе различны – то есть информация достижимая для одного из агентов может быть недостижима для других. Логический язык использует временные операторы UNTIL и Next (для каждого из агентов), через которые могут вводиться модальные операции возможно и необходимо. Главная изучаемая проблема для вводимой логики это проблема распознавания допустимости правил вывода. Ранее эта проблема исследовалась для логики с равномерной фиксированной длинной интервалов транзитивности. Данная работа не предполагает равномерной длины и расширяет логику индивидуальными временными операторами для различных агентов. Находится алгоритм решающий проблему допустимости в данной логике – а именно – распознающий допустимые правила вывода.]

⇐ библиографическая ссылка*:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Сапатый П.С.ЯВОЛНАкОНСдБЗ-1986ст

:= стандартное библиографическое описание*:

[С. П.С., “Язык ВОЛНА-0 как основа навигационных структур для баз знаний на основе семантических сетей,” Изв. АН СССР. Техническая кибернетика., № 5, с. 198—210, 1986]

⇒ аннотация*:

[В работе предложен язык волнового программирования ВОЛНА-0 для баз знаний на основе семантических сетей.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

Тарасов В.Б.о_{много}СкИО-2002кн

:= стандартное библиографическое описание*:

[В. Тарасов, От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. М.: Эдиториал УРСС, 2002]

⇒ аннотация*:

[Книга содержит описание теории агентов, многоагентных систем и интеллектуальных организаций. Исследованы пути создания искусственных сообществ и интеллектуальных организаций]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

Уилкс М..СоставПдЭСМ-1953кн

:= стандартное библиографическое описание*:

[Г. С. Уилкс М. Уиллер Д., Составление программ для электронных счётных машин. Издательство иностранной литературы, 1953]

⇒ аннотация*:

[В предлагаемой читателям книге Уилкса, Уилера и Гилла дается подробное описание методов составления программ для построенной в Англии электронной счетной машины "ЭДСАК"; приведены также многочисленные примеры программ для решения отдельных задач и, в частности, так называемые "стандартные подпрограммы заготовленные заранее и используемые для составления более сложных программ"]

⇐ библиографическая ссылка*:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Филипов А.А..ЕдинаОПИАД-2016ст

:= стандартное библиографическое описание*:

[А. А. Филипов и др., “Единая онтологическая платформа интеллектуального анализа данных,” в Открытые семантические Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 77—82]

⇒ аннотация*:

[В данной работе описана единая онтологическая платформа интеллектуального анализа данных.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Finn2021

:= стандартное библиографическое описание*:

[**Finn2021**]

⇒ аннотация*:

[**Finn2021**]

⇐ библиографическая ссылка*:

- § 1.1.1. Понятие интеллектуальной кибернетической системы

Щедровицкий Г.П.СхемаМСССиС-1995кн

:= стандартное библиографическое описание*:

[Г. П. Щедровицкий, Схема мыследеятельности – системно-структурное строение, смысл и содержание. М.: Шк. культ. пол., 1995, ISBN: 5-88969-001-9]

⇒ аннотация*:

[В книге рассматриваются идеи СМД-методологии, предложенной школой Г.П. Щедровицкого.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

Averin A.I..UsingPiDI-2004арт

:= стандартное библиографическое описание*:

[A. Averin и V. Vagin, “Using parallelism in deductive inference,” англ., Journal of Computer and Systems Sciences International, т. 43, с. 603—614, июль 2004]

⇒ аннотация*:

[Problems of applying parallelism in deductive inference are considered. A general classification of parallelism types is given. Two subtypes of parallelism are analyzed: parallel unification and parallelism at the clause level. An approach for representing terms of first-order predicate logic and an algorithm for parallel unification that uses this representation are proposed. An algorithm of parallel inference on graph structures is presented.]

⇐ библиографическая ссылка*:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Balaji P.G..aIntro tMAS-2010арт

:= стандартное библиографическое описание*:

[P. G. Balaji и D. Srinivasan, “An introduction to multi-agent systems,” в Innovations in multi-agent systems and applications-1, Springer, 2010, с. 1—27]

⇒ аннотация*:

[annotation]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

Blahser J..ThineAfaFTDPMBoHP-2021арт

:= стандартное библиографическое описание*:

[J. Blahser, T. Goller и M. Bohmer, “Thine — Approach for a fault tolerant distributed packet manager based on hypercore protocol,” англ., в 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, 2021. DOI: [10.1109/compsac51774.2021.00266](https://doi.org/10.1109/compsac51774.2021.00266). url: <https://doi.org/10.1109/compsac51774.2021.00266>]

⇒ аннотация*:

[Существует ряд менеджеров пакетов, которые используются почти в каждом проекте разработки программного обеспечения для интеграции сторонних пакетов в проект. Эти пакеты обычно хранятся в центральных репозиториях в облаке. Этот центральный облачный подход имеет некоторые недостатки, особенно в отно-

шении отказоустойчивости, которую мы хотим решить с помощью нашего подхода, вдохновленного мыслями о росистых вычислениях. В этой статье мы представляем полностью распределенный менеджер пакетов, который можно использовать как в облаке, так и на периферии или в гибридной среде. Подход основан на одноранговой сети, реализованной с помощью протокола Hypercore.]

⇐ библиографическая ссылка*:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Boissier O..MultiAOPwJCM-2013art

:= стандартное библиографическое описание*:

[O. Boissier и др., “Multi-agent Oriented Programming with JaCaMo,” англ., Science of Computer Programming, т. 78, № 6, с. 747—761, июнь 2013]

⇒ аннотация*:

[Статья объединяет парадигмы программирования, возникшие в результате исследований в области многоагентных систем.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Bordini R.H..ProgMASiASUJ-2007bk

:= стандартное библиографическое описание*:

[R. H. Bordini, J. F. Hubner и M. J. Wooldridge, Programming multi-agent systems in AgentSpeak using Jason, англ. Chichester: Wiley, 2007]

⇒ аннотация*:

[Книга описывает и объясняет расширение AgentSpeak, интерпретируемое платформой Jason, и показывает, как создавать мультиагентные системы с использованием платформы Jason.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Cao O.IndepBUaUtBIA-2010art

:= стандартное библиографическое описание*:

[L. Cao, “In-depth behavior understanding and use: The behavior informatics approach,” Information Sciences, т. 180, № 17, с. 3067—3085, сент. 2010. DOI: [10.1016/j.ins.2010.03.025](https://doi.org/10.1016/j.ins.2010.03.025). url: <https://doi.org/10.1016/j.ins.2010.03.025>]

⇒ аннотация*:

[В работе рассматривается применение идей бихевиоризма в информатике.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

Cao O.BehavIaNP-2014art

:= стандартное библиографическое описание*:

[L. Cao и др., “Behavior Informatics: A New Perspective,” IEEE Intelligent Systems, т. 29, № 4, с. 62—80, июль 2014. DOI: [10.1109/mis.2014.60](https://doi.org/10.1109/mis.2014.60). url: <https://doi.org/10.1109/mis.2014.60>]

⇒ аннотация*:

[В работе рассматривается применение идей бихевиоризма в информатике.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

Castillo O..RecenAoHIS-2014bk

:= стандартное библиографическое описание*:

[O. Castillo, P. Melin и J. Kacprzyk, Recent advances on hybrid intelligent systems, англ. Berlin: Springer, 2014]

⇒ аннотация*:

[В работе рассматривается построение гибридных интеллектуальных систем на основе многоагентных систем.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

- Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

Chatterjee B.. nBlockDUGwWCAB-2022art

:= стандартное библиографическое описание*:

[B. Chatterjee и др., “Non-Blocking Dynamic Unbounded Graphs with Worst-Case Amortized Bounds,” en, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. DOI: [10.4230/LIPICS.OPODIS.2021.20](https://doi.org/10.4230/LIPICS.OPODIS.2021.20). url: <https://drops.dagstuhl.de/opus/volltexte/2022/15795/>]

⇒ аннотация*:

[В работе рассмотрены основные концепции lock-free алгоритмов.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.4. Принципы синхронизации деятельности sc-агентов

Chen G.. TempoLIfFDoSSwGPD-2021art

:= стандартное библиографическое описание*:

[G. Chen, P. Wei и M. Liu, “Temporal Logic Inference for Fault Detection of Switched Systems With Gaussian Process Dynamics,” англ., *IEEE Transactions on Automation Science and Engineering*, c. 1—16, май 2021. DOI: [10.1109/TASE.2021.3074548](https://doi.org/10.1109/TASE.2021.3074548)]

⇒ аннотация*:

[In this article, we present a method for constructing the fault detector in the form of signal temporal logic (STL) formulas, which can be understood by human users and formally proven to detect faults with probabilistic satisfaction guarantees, for a class of switched nonlinear systems with partially unknown dynamics. First, the partially unknown internal dynamics are approximated by the Gaussian process with stability guarantees. Second, a novel temporal logic inference algorithm is proposed to find the fault detector, which takes advantage of the internal properties of temporal logic and searches for the optimal formula along a partially ordered direction. Moreover, the algorithm is not allowed for missing faults but allowed for false alarms during the temporal logic inference process. In addition, we simulate finitely many trajectories with Chua’s circuit and infer the temporal logic formulas with the Gaussian optimization. The results show that the proposed method can find a temporal logic formula to detect the faulty trajectory with a probability guarantee. Note to Practitioners —The method proposed in this article can be used to detect faults for switched systems with partially unknown dynamics. STL is used to describe the behaviors of the system, which acts as a classifier and detector, such that all normal behaviors of the system will satisfy the description, while the faulty behaviors will violate the description. Moreover, STL formulas can be understood by human operators, which is important for the timely response to faulty events. For example, the normal behavior of a smart grid can be described as follows: “if the smart grid is safe, it should reach 9 kV within 15 min when the voltage to region A is above 12 kV,” which can be expressed with STL. Due to the unknown dynamics, the Gaussian process regression is applied to estimate the model and the region that is robust to noises.]

⇐ библиографическая ссылка*:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Cho J.. StramMtToCBS-2019art

:= стандартное библиографическое описание*:

[J.-H. Cho и др., “STRAM: Measuring the Trustworthiness of Computer-Based Systems,” *ACM Comput. Surv.*, т. 51, № 6, февр. 2019]

⇒ аннотация*:

[annotation]

⇐ библиографическая ссылка*:

- § 1.1.3. Эволюция традиционных и интеллектуальных компьютерных систем

Cypher-2022эл

:= стандартное библиографическое описание*:

[Chapter 3. Cypher [Electronic resource] // Neo4j, English, Mode of access: <http://neo4j.com/docs/stable/cypher-query-lang.html>. — Date of access: 01.12.2022]

⇒ аннотация*:

[Страница официального сайта Графовой СУБД Neo4j, посвященная языку запросов Cypher]

⇐ библиографическая ссылка*:

- Глава 6.3. Программная платформа ostis-систем

Dijkstra E.W.CoopeSP-2002bk

:= стандартное библиографическое описание*:

[E. W. Dijkstra, “Cooperating Sequential Processes,” в *The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls*. Berlin, Heidelberg: Springer-Verlag, 2002, с. 65—138, ISBN: 0387954015]

⇒ аннотация*:

[В работе рассмотрены базовые алгоритмы синхронизации параллельных процессов, положившие начало критическим секциям и более сложным механизмам синхронизации.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.4. Принципы синхронизации деятельности sc-агентов

Dorri A..MultiASaS-2018art

:= стандартное библиографическое описание*:

[A. Dorri, S. S. Kanhere и R. Jurdak, “Multi-agent systems: A survey,” *Ieee Access*, т. 6, с. 28 573—28 593, 2018]

⇒ аннотация*:

[Proposed a high-level comprehensive discussion regarding diverse aspects of MAS which helps newcomers to grasp basic concepts of MAS, study existing applications in multiple disciplines, the challenges in developing MAS, and the methods to study MAS performance]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

Dutta S.KnowIPaAAI-1993bk

:= стандартное библиографическое описание*:

[S. Dutta, *Knowledge processing and applied artificial intelligence*, англ. Oxford : Butterworth-Heinemann, 1993]

⇒ аннотация*:

[В данной книге представлены обсуждения бизнес потенциала обработки знаний и рассматриваются аспекты прикладной технологии искусственного интеллекта.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Eve aWBAP-2023el

:= стандартное библиографическое описание*:

[Eve – a web-based agent platform [Electronic resource], English, Mode of access: <http://eve.almende.com/index.html>. — Date of access: 18.01.2023]

⇒ аннотация*:

[Официальный сайт проекта Eve.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Everts R..ImpleIMASUJACK-2004art

:= стандартное библиографическое описание*:

[R. Everts и др., “Implementing Industrial Multi-agent Systems using JACK,” англ., в *Programming multi-agent systems : first International Conference, ICOPMAS 2004, Berlin, Germany, October 2004*, M. M. Dastani, J. Dix и A. E. Fallah-Seghrouchni, ред., Berlin, 2004, с. 18—48]

⇒ аннотация*:

[Статья содержит обсуждение концепций агентного программирования, связанных с платформой JACK.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Excelente-Toledo C.B..DynamSoCM-2004art

:= стандартное библиографическое описание*:

[C. B. Excelente-Toledo и N. R. Jennings, “The Dynamic Selection of Coordination Mechanisms,” англ., *Autonomous Agents and Multi-Agent Systems*, т. 9, № 1/2, с. 55—85, 2004]

⇒ аннотация*:

[В статье рассматривается структура принятия решений, которая позволяет автономным агентам динамически выбирать механизм, который они используют для координации их взаимосвязанных действий.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Ferber J..Agent tOaOVoMAS-2003art

:= *стандартное библиографическое описание**:

[J. Ferber, O. Gutknecht и F. Michel, “From agents to organizations: an organizational view of multi-agent systems,” в *International workshop on agent-oriented software engineering*, Springer, 2003, с. 214—230]

⇒ *аннотация**:

[*annotation*]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

Finin T.KQMLaaACL-1994art

:= *стандартное библиографическое описание**:

[T. Finin и др., “KQML as an agent communication language,” англ., в *CIKM’94 : proc. of the third Intern. Conf. on Inform. a. Knowl. Assoc. for Computing*, New York, 1994, с. 456—463]

⇒ *аннотация**:

[В статье описывается разработка и эксперименты с языком запросов и манипулирования знаниями (KQML), языком и протоколом для обмена информацией и знаниями]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

FIPAACL MSS-2023el

:= *стандартное библиографическое описание**:

[“FIPA ACL message structure specification [Electronic resource],” English, The Foundation for Intelligent Physical Agent, Mode of access: <http://www.fipa.org/specs/fipa00061/SC00061G.html>. — Date of access: 18.01.2023]

⇒ *аннотация**:

[Стандарт языка взаимодействия агентов.]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Ford B.CompoD-2019art

:= *стандартное библиографическое описание**:

[B. Ford, R. Schiano-Phan и J. A. Vallejo, “Component Design,” англ., в *The Architecture of Natural Cooling*, Routledge, 2019, с. 160—174. DOI: [10.4324/9781315210551-7](https://doi.org/10.4324/9781315210551-7). url: <https://doi.org/10.4324/9781315210551-7>]

⇒ *аннотация**:

[***]

⇐ *библиографическая ссылка**:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Fritzson P.ModelLO-2014art

:= *стандартное библиографическое описание**:

[P. Fritzson, “Modelica Library Overview,” англ., в *Principles of Object Oriented Modeling and Simulation with Modelica 3.3*, John Wiley & Sons, Inc., дек. 2014, с. 909—975. DOI: [10.1002/9781118989166.ch16](https://doi.org/10.1002/9781118989166.ch16). url: <https://doi.org/10.1002/9781118989166.ch16>]

⇒ *аннотация**:

[***]

⇐ *библиографическая ссылка**:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Fry R.L.EngineeringCS-2002art

:= *стандартное библиографическое описание**:

[R. L. Fry, “The engineering of cybernetic systems,” в *AIP Conference Proceedings*, American Institute of Physics, т. 617, 2002, с. 497—528]

⇒ *аннотация**:

[*annotation*]

⇐ *библиографическая ссылка**:

- § 1.1.1. Понятие интеллектуальной кибернетической системы

GAMAP-2023el

- := стандартное библиографическое описание*:
 [GAMA Platform [Electronic resource], English, Mode of access: <http://gama-platform.org/>. — Date of access: 18.01.2023]
- ⇒ аннотация*:
 [Официальный сайт проекта GAMA.]
- ⇐ библиографическая ссылка*:
 • § 1.1.2. Понятие интеллектуальной многоагентной системы
 • Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Gao R..PerfoMfISaEP-2002art

- := стандартное библиографическое описание*:
 [R. Gao и L. Tsoukalas, “Performance metrics for intelligent systems: An engineering perspective,” NIST SPECIAL PUBLICATION c. 5—10, 2002]
- ⇒ аннотация*:
 [annotation]
- ⇐ библиографическая ссылка*:
 • § 1.1.3. Эволюция традиционных и интеллектуальных компьютерных систем

Geramian A..FuzzyISAfFAiAI-2017art

- := стандартное библиографическое описание*:
 [A. Geramian и др., “Fuzzy inference system application for failure analyzing in automobile industry,” англ., International Journal of Quality and Reliability Management, c. 1493—1507, 2017. url: <https://doi.org/10.1108/IJQRM-03-2016-0026>]
- ⇒ аннотация*:
 [Nowadays, quality is one of the most important key success factors in the automobile industry. Improving the quality is based on optimizing the most important quality characteristics and usually launched by highly applied techniques such as failure mode and effect analysis (FMEA). According to the literature, however, traditional FMEA suffers from some limitations. Reviewing the literature, on one hand, shows that the fuzzy rule-base system, under the artificial intelligence category, is the most frequently applied method for solving the FMEA problems. On the other hand, the automobile industry, which highly takes advantages of traditional FMEA, has been deprived of benefits of fuzzy rule-based FMEA (fuzzy FMEA). Thus, the purpose of this paper is to apply fuzzy FMEA for quality improvement in the automobile industry.]
- ⇐ библиографическая ссылка*:
 • Глава 3.5. Логические, производственные и функциональные модели решения задач в ostis-системах

GOAL-2023el

- := стандартное библиографическое описание*:
 [“GOAL [Electronic resource],” English, Atlassian, Mode of access: <https://goalapl.atlassian.net/wiki/spaces/GOAL/overview>. — Date of access: 18.01.2023]
- ⇒ аннотация*:
 [Язык для программирования агентов.]
- ⇐ библиографическая ссылка*:
 • § 1.1.2. Понятие интеллектуальной многоагентной системы
 • Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Neo4j-2023el

- := стандартное библиографическое описание*:
 [“Graph Database Platform | Graph Database Management System | Neo4j.” англ. (сент. 2023), url: <https://neo4j.com/>]
- ⇒ аннотация*:
 [annotation]
- ⇐ библиографическая ссылка*:
 • § 1.1.2. Понятие интеллектуальной многоагентной системы
 • § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Godfrey M.D..tCompu avNPi-1993art

- := стандартное библиографическое описание*:

[M. D. Godfrey и D. F. Hendry, “The computer as von Neumann planned it,” *IEEE Annals of the History of Computing*, т. 15, № 1, с. 11—21, 1993. DOI: [10.1109/85.194088](https://doi.org/10.1109/85.194088). url: <https://doi.org/10.1109/85.194088>]

⇒ **аннотация***:

[Статья, в которой рассматривается архитектура фон Неймана.]

⇐ **библиографическая ссылка***:

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

Gungov A.tAmpliLiDtAoAiCR-2018art

:= **стандартное библиографическое описание***:

[A. Gungov, “The ampliative leap in diagnostics: The advantages of abductive inference in clinical reasoning,” англ., *History of Medicine*, т. 5, с. 233—242, янв. 2018. DOI: [10.3897/hmj.5.4.35635](https://doi.org/10.3897/hmj.5.4.35635)]

⇒ **аннотация***:

[Examining diagnostics in logical terms, attention is usually paid to the interaction between deductive and inductive reasoning. This article discusses Ch.S. Peirce’s theory of abductive inference in the clinical diagnosis. The process of diagnostics is seen as a logical transition from the effect (patient’s symptoms and signs) to the cause (the current health disorder), which is the direction common to abductive reasoning. For Peirce, abduction is performed through the transposition of the conclusion and the major premise in the categorical syllogism or, in his later writings, of the result and the rule. An emphasis is put on the ampliative leap from the premise (individual clinical signs and symptoms) to the conclusion (particular diagnosis) abduction features; the universal rule (the nosological unit) mediates between the individual clinical picture and the particular patient’s diagnosis. The abductive inference draws on Kantian view on reflective judgment and G.B. Vico’s ideas about imaginative universals. Reflective judgment aims at identifying a concept for some sensible data, whereas imaginative universals are not rational concepts but contain general characteristics like the regular concepts; formation of an imaginative universal resembles giving a diagnosis where an imagination drive inference is performed based on the combination of general elements of the relevant nosological unit and individual clinical signs and symptoms. Attention is also paid to the principles of coherence and teleology in performing an abductive inference in diagnostics as well as to the dual criterion of its truthfulness based both on coherence and correspondence. Examples from various medical fields are offered to illustrate the validity of the above logical claims in clinical practice.]

⇐ **библиографическая ссылка***:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Hadzic M..OntolBMAS-2009bk

:= **стандартное библиографическое описание***:

[M. Hadzic и др., *Ontology-based multi-agent systems*. Springer, 2009, с. 283]

⇒ **аннотация***:

[annotation]

⇐ **библиографическая ссылка***:

- § 1.1.2. Понятие интеллектуальной многоагентной системы

Hamilton Inter aKEftG-2006art

:= **стандартное библиографическое описание***:

[Hamilton и др., “Interoperability - A Key Element for the Grid and DER of the Future,” в *2005/2006 IEEE/PES Transmission and Distribution Conference and Exposition*, 2006, с. 927—931]

⇒ **аннотация***:

[annotation]

⇐ **библиографическая ссылка***:

- § 1.1.1. Понятие интеллектуальной кибернетической системы

- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Harting R.L..UsingMATrMO-2008art

:= **стандартное библиографическое описание***:

[R. L. Hartung и A. Hakansson, “Using Meta-agents to Reason with Multiple Ontologies,” англ., в *Lecture Notes in Computer Science*, 2008, с. 261—270]

⇒ **аннотация***:

[В статье рассмотрен механизм метаагентов для управления агентами более низкого уровня.]

⇐ **библиографическая ссылка***:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Hoare C.A.R.CommuSP-1983art

:= **стандартное библиографическое описание***:

[C. A. R. Hoare, "Communicating Sequential Processes," Commun. ACM, т. 26, № 1, с. 100—106, янв. 1983, ISSN: 0001-0782. DOI: [10.1145/357980.358021](https://doi.org/10.1145/357980.358021). url: <https://doi.org/10.1145/357980.358021>]

⇒ **аннотация*:**

[annotation]

⇐ **библиографическая ссылка*:**

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.4. Принципы синхронизации деятельности sc-агентов

Hopcroft J.E..Intro tATLaC-2000art

:= **стандартное библиографическое описание*:**

[J. E. Hopcroft, R. Motwani и J. D. Ullman, Introduction to automata theory, languages, and computation, en, 2-е изд. Upper Saddle River, NJ: Pearson, нояб. 2000]

⇒ **аннотация*:**

[Влиятельный учебник Джона Хопкрофта и Джеффри Ульмана по формальным языкам и теории вычислений, первое издание которого вышло в 1968 году. Раджив Мотвани участвовал в более поздних выпусках, начиная с 2000 г. В книге, в частности, описывается абстрактная машина Тьюринга.]

⇐ **библиографическая ссылка*:**

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

Iliadis A.T Tower oBPMDMSwBFO-2019art

:= **стандартное библиографическое описание*:**

[A. Iliadis, "The Tower of Babel Problem: Making Data Make Sense with Basic Formal Ontology," февр. 2019]

⇐ **библиографическая ссылка*:**

- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Iyengar A.CompoDfRD-2021art

:= **стандартное библиографическое описание*:**

[A. Iyengar, "Component Design for Relational Databases," англ., в Data Management, Auerbach Publications, 2021, с. 143—156. DOI: [10.1201/9780429114878-15](https://doi.org/10.1201/9780429114878-15). url: <https://doi.org/10.1201/9780429114878-15>]

⇒ **аннотация*:**

[В этой работе исследуются преимущества компонентного проектирования для управления данными и предлагаются стратегии, которые могут быть использованы при использовании реляционных централизованных баз данных с объектно-ориентированными компонентными архитектурами приложений. Данная работа отвечает на такие вопросы как "Как объектные технологии и бизнес-компоненты используются при управлении данных? Как модели данных связаны с объектными или компонентными моделями?" и другие.]

⇐ **библиографическая ссылка*:**

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Jackson P.IntroES-1998kn

:= **стандартное библиографическое описание*:**

[P. Jackson, Introduction to expert systems, англ., 3rd. Boston: Addison-Wesley, 1998]

⇒ **аннотация*:**

[Третье издание книги Питера Джексона «Введение в экспертные системы» обновляет технологическую базу исследования экспертных систем и включает эти результаты в контекст самых разных областей применения. В более ранних главах используется более практический подход к основным темам, чем в предыдущих изданиях, а в более поздних главах представлены новые тематические области, такие как рассуждения на основе precedентов, коннекционистские системы и гибридные системы. Результаты в смежных областях, таких как машинное обучение и рассуждения с неопределенностью, также подвергаются тщательной обработке.]

⇐ **библиографическая ссылка*:**

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Jagannathan V.BlackAaA-1989bk

:= **стандартное библиографическое описание*:**

[V. Jagannathan, K. Dodhiawala и L. Baum, Blackboard architectures and applications, англ. New York: Academic Press, 1989]

⇒ **аннотация*:**

[В книге подробно рассматривается принцип "доски объявлений" для взаимодействия агентов.]

⇐ **библиографическая ссылка*:**

- § 1.1.2. Понятие интеллектуальной многоагентной системы

- *Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

JAVAADF-2023el

- := *стандартное библиографическое описание**:
 [JAVA Agent DEvelopment Framework [Electronic resource], англ., Режим доступа: <http://jade.tilab.com/>.
 — Дата доступа: 18.01.2023]
- ⇒ *аннотация**:
 [Официальный сайт проекта JADE.]
- ⇐ *библиографическая ссылка**:
 • § 1.1.2. Понятие интеллектуальной многоагентной системы
 • Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Kerr C..aConceMfTI-2006art

- := *стандартное библиографическое описание**:
 [C. I. Kerr и др., “A conceptual model for technology intelligence,” International Journal of Technology Intelligence and Planning, т. 2, № 1, с. 73—93, 2006]
- ⇒ *аннотация**:
 [annotation]
- ⇐ *библиографическая ссылка**:
 • § 1.1.1. Понятие интеллектуальной кибернетической системы

KnowledgeIFS-2023el

- := *стандартное библиографическое описание**:
 [“Knowledge Interchange Format Specification [Electronic resource],” англ., Stanford Logic Group, Mode of access: <http://logic.stanford.edu/kif/specification.html>. — Date of access: 18.01.2023]
- ⇒ *аннотация**:
 [Официальный сайт, описывающий спецификацию языка KIF.]
- ⇐ *библиографическая ссылка**:
 • § 1.1.2. Понятие интеллектуальной многоагентной системы
 • Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Laird J.E..Claim aCiEHIS-2009art

- := *стандартное библиографическое описание**:
 [J. E. Laird и др., “Claims and challenges in evaluating human-level intelligent systems,” в 2nd Conference on Artificiel General Intelligence, Atlantis Press, 2009, с. 80—85]
- ⇒ *аннотация**:
 [annotation]
- ⇐ *библиографическая ссылка**:
 • § 1.1.3. Эволюция традиционных и интеллектуальных компьютерных систем

Lanzenberger M..MakinOTKliSW-2008art

- := *стандартное библиографическое описание**:
 [Lanzenberger, Monika and Sampson, Jennifer and Kargl, Horst and Wimmer, Manuel and Conroy, Colm and O’Sullivan, Declan and Lewis, David and Brennan, Rob and Ramos-Gargantilla, José Ángel and Gómez-Pérez, Asunción and Fürst, Frédéric and Trichet, Francky and Euzenat, Jérôme and Polleres, Axel and Scharffe, François and Kotis, Konstantinos, “Making Ontologies Talk: Knowledge Interoperability in the Semantic Web,” IEEE Intelligent Systems, т. 23, № 6, с. 72—85, 2008]
- ⇒ *аннотация**:
 [annotation]
- ⇐ *библиографическая ссылка**:
 • Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Lopes L.S..SemanItIoT-2022art

- := *стандартное библиографическое описание**:
 [P. Lopes de Lopes de Souza, W. Lopes de Lopes de Souza и R. R. Ciferri, “Semantic Interoperability in the Internet of Things: A Systematic Literature Review,” в ITNG 2022 19th International Conference on Information Technology-New Generation, S. Latifi, ред., Cham: Springer International Publishing, 2022, с. 333—340]
- ⇒ *аннотация**:
 [annotation]

- ⇐ библиографическая ссылка*:
- § 1.1.1. Понятие интеллектуальной кибернетической системы
 - Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Lupea M.aTheorPfCaRDL-2002art

- := стандартное библиографическое описание*:
 [M. Lupea, “A THEOREM PROVER FOR CONSTRAINED AND RATIONAL DEFAULT LOGICS,” англ., янв. 2002]
- ⇒ аннотация*:
 [Default logics represent an important class of the nonmonotonic formalisms. Using simple by powerful inference rules, called defaults, these logic systems model reasoning patterns of the form "in the absence of information to the contrary of. . . and thus formalize the default reasoning, a special type of nonmonotonic reasoning. In this paper we propose an automated system, called DARR, with two components: a propositional theorem prover and a theorem prover for constrained and rational propositional default logics. A modified version of semantic tableaux method is used to implement the propositional prover. Also, this theorem proving method is adapted for computing extensions because one of its purpose is to produce models, and extensions are models of the world described by default theories.]
- ⇐ библиографическая ссылка*:
- ?? ??
 - Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

Melekhova O..aDecenSfCD-2018art

- := стандартное библиографическое описание*:
 [O. Melekhova и др., “A decentralised solution for coordinating decisions in large-scale autonomic systems,” в MATEC Web of Conferences, EDP Sciences, т. 161, 2018, с. 03024]
- ⇒ аннотация*:
 [annotation]
- ⇐ библиографическая ссылка*:
- § 1.1.1. Понятие интеллектуальной кибернетической системы

Memduhoglu M.PossiCoSSMatIMPSDP-2018art

- := стандартное библиографическое описание*:
 [A. Memduhoglu и M. Basaraner, “POSSIBLE CONTRIBUTIONS OF SPATIAL SEMANTIC METHODS AND TECHNOLOGIES TO MULTI-REPRESENTATION SPATIAL DATABASE PARADIGM,” International Journal of Engineering and Technology, vol. 7, no. 14, pp. 1–10, 2018. DOI: [10.26833/ijeg.413473](https://doi.org/10.26833/ijeg.413473). url: <https://doi.org/10.26833/ijeg.413473>]
- ⇒ аннотация*:
 [Today, the amount and variety of spatial data have increased dramatically. In addition, the web has made it easier to disseminate and share this kind of data. Therefore, spatial data integration and interoperability have gained more importance. Spatial data are collected from different sources and often heterogeneous in terms of the levels of detail and the points of view. To be able to meet the demands of different spatial applications, multi-source and heterogeneous spatial datasets need to be integrated as well as the consistency of these datasets needs to be maintained. In this context, multirepresentation spatial database (MRSDB) paradigm has been suggested by researchers. However, the heterogeneity constitutes a significant barrier in this respect and hence the implementations have so far been remained within a rather narrow scope. In this article, it is mainly discussed about the possible contributions of basic methods and technologies of spatial semantics such as ontologies, semantic web and linked data to the data integration for creating a MRSBD. Some examples are also given to illustrate the concept.]
- ⇐ библиографическая ссылка*:
- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Moldovan D.I.SNAPaVLSIAfAIP-1985bk

- := стандартное библиографическое описание*:
 [D. I. Moldovan и Y.-W. Tung, “SNAP: A VLSI architecture for artificial intelligence processing,” Journal of Parallel and Distributed Systems, vol. 2, no. 2, pp. 109–131, 1985, ISSN: 0743-7315. DOI: [https://doi.org/10.1016/0743-7315\(85\)90031-0](https://doi.org/10.1016/0743-7315(85)90031-0). url: <https://www.sciencedirect.com/science/article/pii/0743731585900310>]
- ⇒ аннотация*:
 [annotation]
- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - Глава 3.2. Агентно-ориентированные модели гибридных решателей задач ostis-систем

Nagendra Prasad M.V.LearnSSCiCMAS-1999art

:= *стандартное библиографическое описание**:

[M. V. Nagendra Prasad и V. R. Lesser, “Learning Situation-Specific Coordination in Cooperative Multi-agent Systems,” англ., Autonomous Agents a. Multi-Agent Systems, т. 2, № 2, с. 173—207, 1999]

⇒ *аннотация**:

[В статье рассмотрен еще один вариант координации агентов.]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Neiva F.W.TowarPItSC-2016art

:= *стандартное библиографическое описание**:

[F. W. Neiva и др., “Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature,” Information and Software Technology, т. 72, с. 137—150, 2016, ISSN: 0950-5849. DOI: <https://doi.org/10.1016/j.infsof.2015.12.013>]

⇐ *библиографическая ссылка**:

- § 1.1.1. Понятие интеллектуальной кибернетической системы
- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Nilsson N.J.HumanLAIBS-2005art

:= *стандартное библиографическое описание**:

[N. J. Nilsson, “Human-level artificial intelligence? Be serious!” AI magazine, т. 26, № 4, с. 68—68, 2005]

⇐ *библиографическая ссылка**:

- § 1.1.1. Понятие интеллектуальной кибернетической системы

Norton J.D..aDemonstrationofIoCoII-2019art

:= *стандартное библиографическое описание**:

[J. D. Norton, “A Demonstration of the Incompleteness of Calculi of Inductive Inference,” The British Journal for the Philosophy of Science, т. 70, № 4, с. 1119—1144, 2019. DOI: [10.1093/bjps/axx004](https://doi.org/10.1093/bjps/axx004). url: <https://doi.org/10.1093/bjps/axx004>]

⇒ *аннотация**:

[Статья, в которой рассматривается индуктивный вывод.]

⇐ *библиографическая ссылка**:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Omicini A..Coord fIAD-1999art

:= *стандартное библиографическое описание**:

[A. Omicini и F. Zambonelli, “Coordination for Internet Application Development,” англ., Autonomous Agents a. Multi-Agent Systems, т. 2, № 2, с. 251—269, июнь 1999]

⇒ *аннотация**:

[В статье предложена идея, заключающаяся в том, чтобы вводить в сеть агентов коммуникационные центры, регулирующие общение агентов.]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Ouksel A.M..SemantiiGIS-1999art

:= *стандартное библиографическое описание**:

[A. M. Ouksel и A. Sheth, “Semantic Interoperability in Global Information Systems,” SIGMOD Rec., т. 28, № 1, с. 5—12, март 1999]

⇒ *аннотация**:

[annotation]

⇐ *библиографическая ссылка**:

- § 1.1.1. Понятие интеллектуальной кибернетической системы
- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

OWLWOLDO-2023el

:= *стандартное библиографическое описание**:

[“OWL 2 Web Ontology Language document overview [Electronic resource],” англ., World Wide Web Consortium (W3C), Mode of access: <http://www.w3.org/TR/owl2-overview>. — Date of access: 23.01.2023]

⇒ *аннотация**:

[Официальный сайт, описывающий язык OWL 2]

- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Pau L.F.KnowlBP-1990art

- := стандартное библиографическое описание*:
[“OWL Implementations [Electronic resource],” англ., World Wide Web Consortium (W3C), Mode of access: <https://www.w3.org/2001/sw/wiki/OWL/Implementations/>. — Date of access: 23.01.2023]
- ⇒ аннотация*:
[Официальный сайт, описывающий применение языка OWL]
- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Pavel M..BehavIaCMiSoPHMaC-2015art

- := стандартное библиографическое описание*:
[M. Pavel и др., “Behavioral Informatics and Computational Modeling in Support of Proactive Health Management and Care,” *IEEE Transactions on Biomedical Engineering*, т. 62, № 12, с. 2763—2775, дек. 2015. DOI: [10.1109/tbme.2015.2484286](https://doi.org/10.1109/tbme.2015.2484286). url: <https://doi.org/10.1109/tbme.2015.2484286>]
- ⇒ аннотация*:
[Статья посвящена вопросам решения задач в системах, основанных на знаниях.]
- ⇐ библиографическая ссылка*:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.7. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

Prakash S.P.MicroPA-2022art

- := стандартное библиографическое описание*:
[S. P. Pradhan, “Working with Microsoft Power Apps,” англ., в *Power Platform and Dynamics 365 CE for Absolute Beginners*, Apress, 2022, с. 79—131. DOI: [10.1007/978-1-4842-8600-5_3](https://doi.org/10.1007/978-1-4842-8600-5_3). url: https://doi.org/10.1007/978-1-4842-8600-5_3]
- ⇒ аннотация*:
[***]
- ⇐ библиографическая ссылка*:
- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Pohl J.Inter a tNfISaHP-2004art

- := стандартное библиографическое описание*:
[J. Pohl, “Interoperability and the Need for Intelligent Software: A Historical Perspective,” сент. 2004]
- ⇐ библиографическая ссылка*:
- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Reiter R.LogicDR-1980cm

- := стандартное библиографическое описание*:
[R. Reiter, “A Logic for Default Reasoning,” англ., *Artificial Intelligence*, т. 13, № 13, с. 81—132, 1980]
- ⇒ аннотация*:
[Ключевая статья, посвященная логике умолчаний.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Rumbell T. Emoti iAACAoMaF-2012art

:= стандартное библиографическое описание*:

[T. Rumbell и др., “Emotions in autonomous agents: comparative analysis of mechanisms and functions,” англ., Autonomous Agents a. Multi-Agent Systems, т. 25, № 1, с. 1—45, июнь 2012]

⇒ аннотация*:

[Подход к регулированию деятельности агентов на основе эмоций]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Safawi A.R. tDecisPoAI-2015art

:= стандартное библиографическое описание*:

[S. Abdul Rahman и др., “The Decision Processes of Abductive Inference,” англ., Advanced Science Letters, т. 21, с. 1754—1757, июнь 2015. DOI: [10.1166/asl.2015.6193](https://doi.org/10.1166/asl.2015.6193)]

⇒ аннотация*:

[A vast number of problem solving frameworks have been established in the realms of management, business, mathematics, healthcare, aerospace, law and etcetera towards facilitating effective decision making in normative form in particular. It was learnt that those problem solving frameworks are originated mainly from practical exercises of practitioners and/or improvement/revision of an earlier frameworks. Against this bottom-up approach in the establishment of problem solving frameworks, this paper continues promoting the establishment of a problem solving framework following top-down approach by synthesizing the deductive inference i.e., the second inference process of Pragmatism’s scientific method. The synthesis has resulted in the identification of the decision sub-processes of Pragmatism’s deductive inference namely analysis and demonstration. This paper posits that these two decision processes are aligned with both the key and strategies of the decision processes of the renowned problem solving frameworks in practice in management, business, mathematics, healthcare and law and hence, the decision processes of deductive inference in particular and Pragmatism’s philosophical method in general is an alternative for decision making processes and problem solving frameworks following the top-down approach.]

⇐ библиографическая ссылка*:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Sethy S.S. MediaIS-2021art

:= стандартное библиографическое описание*:

[S. S. Sethy, “Mediate Inference (Syllogism),” англ., в июнь 2021, с. 95—124, ISBN: 978-981-16-2688-3. DOI: [10.1007/978-981-16-2689-0_8](https://doi.org/10.1007/978-981-16-2689-0_8). url: <https://www.researchgate.net/publication/352359427>]

⇒ аннотация*:

[In this article will be discussed mediate inference (i.e. syllogism) in detail with suitable examples. Authors will analyse the moods and figures of syllogism, describe the rules for the syllogism, and apply the rules of syllogism to each mood by linking them to figures of syllogism to find out the valid moods and valid arguments of the syllogism. In the end, they will list out the valid moods of the syllogism corresponding to the four figures of syllogism.]

⇐ библиографическая ссылка*:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Sims M..AutoODfMAS-2008art

:= стандартное библиографическое описание*:

[M. Sims, D. Corkill и V. Lesser, “Automated organization design for multi-agent systems,” англ., Autonomous Agents a. Multi-Agent Systems, т. 16, № 2, с. 151—185, июнь 2008]

⇒ аннотация*:

[В статье рассматривается KB-ORG: полностью автоматизированный конструктор организаций, основанный на знаниях, для многоагентных систем.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Son L.H..PictuISaNFISoPFS-2017art

:= стандартное библиографическое описание*:

[L. Son, P. Van Viet и P. Van Hai, "Picture inference system: a new fuzzy inference system on picture fuzzy set," англ., Applied Intelligence, с. 652—669, 2017. url: <https://doi.org/10.1007/s10489-016-0856-1>]

⇒ *аннотация**:

[In this paper, authors propose a novel fuzzy inference system on picture fuzzy set called picture inference system (PIS) to enhance inference performance of the traditional fuzzy inference system. In PIS, the positive, neutral and negative degrees of the picture fuzzy set are computed using the membership graph that is the combination of three Gaussian functions with a common center and different widths expressing a visual view of degrees. Then, the positive and negative defuzzification values, synthesized from three degrees of the picture fuzzy set, are used to generate crisp outputs. Learning in PIS including training centers, widths, scales and defuzzification parameters is also discussed. The system is adapted for all architectures such as the Mamdani, the Sugeno and the Tsukamoto fuzzy inferences. Experimental results on benchmark UCI Machine Learning Repository datasets and an example in control theory - the Lorenz system are examined to verify the advantages of PIS.]

⇐ *библиографическая ссылка**:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

SPARQLO-2023el

:= *стандартное библиографическое описание**:

[“SPARQL 1.1 Overview [Electronic resource],” англ., World Wide Web Consortium, Mode of access: <https://www.w3.org/TR/sparql11-overview>. — Date of access: 18.01.2023]

⇒ *аннотация**:

[Официальный сайт, описывающий язык SPARQL 1.1]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Uehara K..FuzzyIPaP-2017art

:= *стандартное библиографическое описание**:

[K. Uehara и K. Hirota, “Fuzzy Inference: Its Past and Prospects,” англ., Journal of Advanced Computational Intelligence and Intelligent Informatics, т. 21, с. 13—19, янв. 2017. DOI: [10.20965/jaciii.2017.p0013](https://doi.org/10.20965/jaciii.2017.p0013)]

⇒ *аннотация**:

[Fuzzy inference in the past and its future prospects are described to further promote research in the field: First, the basic methods of fuzzy inference are introduced. Then, the progress of fuzzy inference is reviewed, showing its remarkable achievements, especially in industries. A consideration of fuzzy inference is presented from operational viewpoints. It provides a key to creating fuzzy-inference methods in the future. The growing research area of fuzzy inference is also introduced in order to discuss a current direction, reflecting the consideration mentioned above. Moreover, some future prospects on fuzzy inference are presented, which are expected to stimulate research.]

⇐ *библиографическая ссылка**:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

USBAccel-2022el

:= *стандартное библиографическое описание**:

[“USB Accelerator | Coral.” англ. (дек. 2022), url: <https://coral.ai/products/accelerator/>]

⇒ *аннотация**:

[Официальный сайт USB-ускорителя Coral, обеспечивающего аппаратное ускорение процессов машинного обучения и построенного на базе сопроцессора Google Edge TPU.]

⇐ *библиографическая ссылка**:

- Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем

Vasconcelos W.W..NormaCRiMAS-2009art

:= *стандартное библиографическое описание**:

[W. W. Vasconcelos, M. J. Kollingbaum и T. J. Norman, “Normative conflict resolution in multi-agent systems,” англ., Autonomous Agents and Multi-Agent Systems, т. 19, № 2, с. 124—152, июнь 2009]

⇒ *аннотация**:

[В статье предлагается подход к регулированию поведения агентов на основе неких утвержденных "законов".]

⇐ *библиографическая ссылка**:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Waters J..GlobalUS-2009art

:= *стандартное библиографическое описание**:

[Jeff Waters and Brenda J. Powers and Marion G. Ceruti, “Global Interoperability Using Semantics, Standards, Science and Technology (GIS3T),” Computer Standards & Interfaces, т. 31, № 6, с. 1158—1166, 2009]

⇐ библиографическая ссылка*:

- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Weydert E.DefauLaPaTP-2022art

:= стандартное библиографическое описание*:

[E. Weydert, “Defaults, Logic and Probability – A theoretical perspective,” англ., KI – Künstliche Intelligenz, v.4/01, 44–49 (2001), дек. 2022]

⇒ аннотация*:

[The complexity of the real world and the restricted availability of knowledge call for powerful logical frameworks to deal with uncertainty, reaching from qualitative default formalisms to quantitative probability logics. In particular, reasoning under uncertainty requires defeasible inference notions. Author takes a general look at these issues, with a special emphasis on quasi-probabilistic approaches to default reasoning.]

⇐ библиографическая ссылка*:

- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах

Weyns D.Envir aaFCAiMAS-2007art

:= стандартное библиографическое описание*:

[D. Weyns, A. Omicini и J. Odell, “Environment as a first class abstraction in multiagent systems,” англ., Autonomous Agents a. Multi-Agent Systems, т. 14, № 1, с. 5—30, февр. 2007]

⇒ аннотация*:

[Обзор сред взаимодействия в МАС.]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Wilkes M.V.PrepaopfEDCWSRttEatUoaLoS-1951bk

:= стандартное библиографическое описание*:

[M. Wilkes, The Preparation of Programs for an Electronic Digital Computer: With Special Reference to the EDSAC and the Use of the Computer in Mathematics (Addison-Wesley mathematics series), англ. Addison-Wesley Press, 1951. url: <https://books.google.com/books?id=PzVVAAMAAJ>]

⇒ аннотация*:

[Ее часто считают первой книгой по компьютерному программированию. Она была написана для компьютера EDSAC (автоматический калькулятор электронного хранения с задержкой), который начал работать в 1949 году как первый в мире регулярно работающий компьютер с хранимой программой. Идея библиотеки подпрограмм была разработана для EDSAC и описана в этой книге. Морис Уилкс руководил разработкой EDSAC.]

⇐ библиографическая ссылка*:

- Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

Wooldridge M.aIntro tMS-2009bk

:= стандартное библиографическое описание*:

[M. Wooldridge, An introduction to multiagent systems, англ., 2nd. Chichester: J. Wiley, 2009]

⇒ аннотация*:

[Основные цели книги:

- Познакомить учащихся с понятием агента и многоагентной системы, а также с основными приложениями, для которых они подходят.
- Познакомить с основными проблемами, связанными с проектированием интеллектуальных агентов.
- Познакомить с основными проблемами, связанными с проектированием многоагентных сообществ.
- Представить ряд типичных приложений для агентной технологии.

]

⇐ библиографическая ссылка*:

- § 1.1.2. Понятие интеллектуальной многоагентной системы
- Пункт 3.2.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

WorldWWC-2023el

- := *стандартное библиографическое описание**:
[“World Wide Web Consortium [Electronic resource].” англ. Mode of access: <http://www.w3.org>. — Date of access: 18.01.2023. ()]
- ⇒ *аннотация**:
[Официальный сайт консорциума W3C.]
- ⇐ *библиографическая ссылка**:
- § 1.1.2. Понятие интеллектуальной многоагентной системы
 - § 3.2.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

Yaghoobirafi K..ApprofSIiADIS-2022art

- := *стандартное библиографическое описание**:
[K. Yaghoobirafi и A. Farahani, “An approach for semantic interoperability in autonomic distributed intelligent systems,” *Journal of Software: Evolution and Process*, т. 34, февр. 2022. DOI: [10.1002/smri.2436](https://doi.org/10.1002/smri.2436)]
- ⇒ *аннотация**:
[annotation]
- ⇐ *библиографическая ссылка**:
- Глава 1.2. Интеллектуальные компьютерные системы нового поколения

Yuxuan Z..MissiEAKGIItDGLaT-2022art

- := *стандартное библиографическое описание**:
[Z. Yuxuan и др., “Missing-edge aware knowledge graph inductive inference through dual graph learning and traversing,” англ., в окт. 2022. DOI: [10.1016/j.eswa.2022.118969](https://doi.org/10.1016/j.eswa.2022.118969). url: <https://www.researchgate.net/publication/364404289>]
- ⇒ *аннотация**:
[Knowledge graph (KG) is a kind of structured human knowledge of modeling the relations between real-world entities. This paper studies the KG inductive inference problem, i.e., predicting the relations for out-of-KG entities. However, due to the incomplete nature of the KGs, the connections of some relations are missing. This makes existing differentiable rule learning methods unable to represent some possible rule candidates, which will further affect the inductive inference result. To solve this challenge, author’s research hypothesis is that the semantics of relation’s argument can be well used to reflect the possible connections between relations. There is proposed a KG inductive inference model, RuleNet, which consists of two parts. Firstly, a query-dependent dual graph construction method is proposed, which is able to learn the relation connections using the information of the relation’s argument. Secondly, a dual graph traversing method is proposed, which is able to traverse all possible rule candidates even if some rules cannot be formed due to the missing edges. Performance of the proposed methods is evaluated using the FB15K237 (10%–20%), WN18RR (10%–20%) and YAGO3-10 (10%–20%) benchmarks. Experimental results show that RuleNet achieves a superior performance compared with many strong baselines. Ablation studies have verified the effectiveness of the proposed network components. Qualitative analysis shows that RuleNet can learn meaningful dual graph and logic rules.]
- ⇐ *библиографическая ссылка**:
- Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах