

В.В. Голенков, О.Е. Елисеева, В.П. Иващенко, В.М. Казан

Н.А. Гулякина, Н.В. Беззубенок, Т.Л. Лемешева, Р.Е. Сердюков

И.Б. Фоминых

**ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА
ЗНАНИЙ В
ГРАФОДИНАМИЧЕСКИХ
АССОЦИАТИВНЫХ
МАШИНАХ**

Под редакцией В.В. Голенкова

Минск

2001

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

**ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА
ЗНАНИЙ В
ГРАФОДИНАМИЧЕСКИХ
АССОЦИАТИВНЫХ
МАШИНАХ**

Под редакцией В.В. Голенкова

МИНСК

2001

УДК 007.2 (075.8)

Коллектив авторов:

В.В. Голенков, О.Е. Елисеева, В.П. Иващенко, В.М. Казан

Н.А. Гулякина, Н.В. Беззубенок, Т.Л. Лемешева, Р.Е. Сердюков

И.Б. Фоминых

Представление и обработка знаний в графодинамических ассоциативных машинах /
В.В.Голенков, О.Е.Елисеева, В.П.Иващенко и др.: Под ред. В.В.Голенкова. – Мн.: БГУИР, 2001. – 412 с.

ISBN 985-444-312-4

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.

В книгу вошли результаты проектов, финансируемых Белорусским Республиканским Фондом фундаментальных исследований в рамках совместной белорусско-российской программы фундаментальных исследований «БРФИ-РФФИ 2000»: «Разработка принципов и методов создания интегрированных динамических экспертных систем» (№ 00-7036), «Анализ и моделирование псевдооптических нейронных сетей» (№ 00-7029).

Книга предназначена как для опытных специалистов, так и для аспирантов и студентов, интересующихся проблемами искусственного интеллекта.

Табл. 17. Ил. 270. Библиогр.: 693 назв.

Рецензенты:

доктор технических наук А.С.Гринберг

доктор технических наук Б.М.Лобанов

доктор технических наук Н.А.Ярмош

Издано при поддержке СП International Business Alliance

Рекомендовано к печати Советом БГУИР (протокол № 2 от 25.10.2001)

ISBN 985-444-312-4

© Коллектив авторов, 2001

Belarusian state university of informatics and radioelectronics

**KNOWLEDGE REPRESENTATION AND PROCESSING
IN GRAPH-DYNAMIC ASSOCIATIVE COMPUTERS**

Edited by V.V. Golenkov

Minsk

2001

Knowledge Representation and Processing in Graph-Dynamic Associative Computers. V.V. Golenkov, O.E. Eliseeva, V.P. Ivashenko and others. Edited by V.V. Golenkov. - Minsk, BSUIR, 2001. – 412 c.

ISBN 985-444-312-4

Graph-dynamic associative models for representation and processing of knowledge in artificial intelligence systems are represented. The base of this model is, firstly, a description of different types of knowledge represented by homogeneous semantic networks having a base set-theoretic interpretation and, secondly, interpretation of knowledge processing as a graph-dynamic process, i.e. a process of semantic networks configuration modification.

The results of the projects financed by the Belarusian Republic Fund of Fundamental Researches realized in a framework of a joint program of fundamental researches "BRFFR – RFFR – 2000": "Development of principles and methods for construction of integrated dynamic expert systems" (No. 00-7036), "Analysis and simulation of pseudo-optical neuron networks" (No. 00 – 7029) are included.

The book is intended for skillful specialists as well as for postgraduates and students taking an interest in artificial intelligence.

Reviewers:

Dr.Sc. A.S. Grinberg; Dr.Sc. B.M. Lobanov; Dr.Sc. N.A. Jarmosh.

Issued under sponsorship of JV International Business Alliance.

Содержание

Предисловие	13
Введение	15
1. Графодинамическая парадигма обработки информации	19
1.1. Концепция графодинамических моделей.....	19
1.1.1. Понятие формальной модели обработки информации	20
1.1.2. Классификация формальных моделей обработки информации.....	22
1.1.3. Интеграция и интерпретация формальных моделей обработки информации	23
1.1.4. Семиотические модели обработки информации	24
1.1.5. Графодинамические ассоциативные модели обработки знаний и известные виды моделей представления знаний.....	25
1.2. Графовые языки.....	27
1.2.1. Понятие реляционной структуры как уточнение понятия предметной области и понятия информационной конструкции.....	28
1.2.2. Линейные тексты.....	34
1.2.3. Нелинейные тексты	36
1.2.4. Денотационная семантика текстов	41
1.2.5. Классификация языков	44
1.2.6. Семантические сети и семантические графовые языки	47
1.3. Абстрактные графодинамические ассоциативные машины	51
1.3.1. Абстрактные машины обработки информации и соответствующие им операции, элементарные процессы и микропрограммы.....	51
1.3.2. Классификация абстрактных машин обработки информации	55
1.3.3. Графодинамические параллельные асинхронные абстрактные машины как наиболее перспективный класс абстрактных машин для проектирования сложных интеллектуальных систем.....	58
Выводы к разделу 1	59
2. Теоретико-множественные принципы представления фактографических знаний в памяти графодинамических ассоциативных машин	63
2.1. Базовые понятия, лежащие в основе языка SCB.....	63
2.2. Основные положения языка SCB (Semantic Code Basic).....	73
2.3. Ядро языка SCBg (Semantic Code Basic graphical) – графической модификации языка SCB .	76
2.4. Средства обеспечения наглядности языка SCBg	81
2.5. Ядро языка SCBs (Semantic Code Basic string) – линейной модификации языка SCB	91
2.6. Формирование идентификаторов в языке SCB	94
2.7. Приведение текстов языка SCBs к лаконичному виду	106

2.8. Формальное описание синтаксиса языка SCBs.....	112
Выводы к разделу 2	116
3. Представление основных математических структур на языке SCB	117
3.1. Типология множеств и их представление в языке SCB. Основные множества языка SCB и соответствующие им ключевые узлы.....	117
3.2. Понятие кортежа. Атрибуты элементов кортежа. Представление кортежей в языке SCB. Типология кортежей	127
3.2.1. Понятие кортежа и атрибута	127
3.2.2. Примеры кортежей и их представление в языках SCBg и SCBs	128
3.2.3. Типология кортежей.....	132
3.3. Понятие отношения. Представление отношений в языке SCB. Типология отношений. Классические и неклассические отношения	134
3.3.1. Обобщение традиционной трактовки отношений	134
3.3.2. Типология отношений на основе базовой типологии множеств	137
3.3.3. Типология отношений на основе типологии кортежей, входящих в состав отношения, а также анализа соотношения между кортежами	139
3.3.4. Типология отношений на основе понятия проекции и понятия области определения	142
3.3.5. Типология отношений на основе понятия функциональной зависимости	144
3.3.6. Представление и типология классических отношений.....	145
3.3.7. Отношения предельного вида	146
3.3.8. Типология бинарных отношений и метаотношения над ними	148
3.3.9. Множество соответствий как метаотношение, заданное на множестве бинарных ориентированных отношений	151
3.3.10. Типология тернарных отношений и метаотношения над ними.....	155
3.3.11. Отношения над множествами.....	155
3.3.12. Отношения над кортежами.....	166
3.3.13. Отношения над отношениями	167
3.3.14. Числовые отношения	174
3.4. Представление реляционных структур в языке SCB. Типология реляционных структур. Классические и неклассические реляционные структуры	183
3.4.1. Представление реляционных структур в языке SCB	183
3.4.2. Типология реляционных структур	188
3.4.3. Отношения над реляционными структурами. Реляционные метаструктуры	189
3.4.4. Графовые структуры и отношения над ними	192
Выводы к разделу 3	195

4. Ядро открытого семейства графовых языков представления знаний, построенных на теоретико-множественной основе.....	197
4.1. Основные понятия, лежащие в основе логических языков	197
4.2. Язык SC (Semantic Code), являющийся основой построения различных логических языков и языков представления знаний	199
4.3. Язык SCg (Semantic Code graphical) – графическая модификация языка SC	200
4.4. Язык SCs (Semantic Code string) – линейная модификация языка SC	202
4.5. Ключевые узлы графового языка SC	204
4.6. Понятие sc-подъязыка. Семейство графовых языков, построенных на базе языка SC	209
4.7. Понятие абстрактной sc-машины.....	211
4.8. Понятие формальной sc-модели	215
Выводы к разделу 4	215
5. Представление логических формул и формальных теорий в памяти графодинамических ассоциативных машин.....	217
5.1. Принципы построения графового логического языка SCL (Semantic Code Logic) на теоретико-множественной основе	217
5.2. Запись логических формул с использованием стилизованного естественного языка	218
5.3. Язык SCLs (Semantic Code Logic string) – формальный линейный логический язык классического типа, использующий язык SCs для записи атомарных логических формул	220
5.4. Язык SCLg (Semantic Code Logic graphical) – графический вариант изображения текстов языка SCL.....	222
5.5. Примеры записи логических формул на предложенных логических языках.....	225
5.6. Формальная метатеория и её представление на языке SCL.....	250
Выводы к разделу 5	252
6. Типология знаний и языки представления знаний в графодинамических ассоциативных машинах	253
6.1. Представление знаний, связанных с понятием измерения.....	253
6.2. Описание динамических систем	264
6.3. Описание целей в графодинамических ассоциативных машинах	272
6.4. Гипертекстовые семантические сети	285
6.5. Принципы представления нейросетевых моделей.....	289
6.5.1. Краткое описание полной прямолинейной модели псевдооптической нейронной сети и методов расчета её поведения.....	290
6.5.2. Принципы представления псевдооптических нейросетей в памяти графодинамических машин.....	294
Выводы к разделу 6	296

7.	Навигационно-поисковая графодинамическая ассоциативная машина.....	297
7.1.	Операции навигационно-поисковой графодинамической ассоциативной машины.....	297
7.1.1.	Информационные конструкции, описывающие состояние навигационно-поисковой графодинамической ассоциативной машины.....	297
7.1.2.	Семейство операций поиска теоретико-графовой окрестности указываемого sc-элемента.....	300
7.1.3.	Семейство операций поиска в рамках указываемой формальной теории всех истинных высказываний, релевантных указываемой высказывательной форме (заданному образцу)	301
7.1.4.	Семейство операций поиска семантических окрестностей указываемого sc-элемента.....	307
7.1.5.	Семейство операций поиска семантической связи между (двумя или более) указываемыми sc-элементами.....	309
7.1.6.	Семейство навигационно-поисковых операций в гипертекстовой семантической сети.....	310
7.2.	Пример работы навигационно-поисковой графодинамической ассоциативной машины.....	315
	Выводы к разделу 7	320
8.	Графодинамические ассоциативные машины вывода	321
8.1.	Семейство легко интегрируемых абстрактных графодинамических ассоциативных машин вывода	321
8.1.1.	Информационные конструкции, описывающие состояние абстрактной графодинамической ассоциативной машины вывода.....	323
8.1.2.	Операции абстрактной графодинамической ассоциативной машины вывода	325
8.2.	Решение задач в графодинамических ассоциативных машинах вывода.....	346
	Выводы к разделу 8	370
	Заключение.....	371
	Литература	372

CONTENTS

Annotation.....	13
Introduction.....	15
1. Information Processing. Graph-Dynamic paradigm	19
1.1. Conception of graph-dynamic models	19
1.1.1. Concept of information processing formal model.....	20
1.1.2. Classification of information processing formal models	22
1.1.3. Information processing formal models integration and interpretation.....	23
1.1.4. Semiotic models of information processing	24
1.1.5. Graph-dynamic knowledge processing formal models and admitted model types of knowledge representation.....	25
1.2. Graph languages	27
1.2.1. Concept of a relational structure as the concept specification of the subject and information structure.....	28
1.2.2. Linear texts	34
1.2.3. Nonlinear texts.....	36
1.2.4. Denotative text semantics.....	41
1.2.5. Language classification.....	44
1.2.6. Semantic networks and semantic graphic languages.....	47
1.3. Abstract graph-dynamic associative computers.....	51
1.3.1. Information processing abstract computers and corresponding operations, elementary processes and microprograms	51
1.3.2. Classification of information processing abstract computers	55
1.3.3. Graph-dynamic parallel asynchronous abstract computers as the most perspective class of abstract computers for designing complex intelligent systems	58
Chapter 1 conclusions.....	59
2. Set-Theoretical principles of factual knowledge representation in memory of graph-dynamic associative computers.....	63
2.1. SCB language (Semantic Code Basic). Basic statements.....	63
2.2. SCB language (Semantic Code Basic). Basic concepts	73
2.3. SCBg kernel (Semantic Code Basic graphical) – SCB graphical modification	76
2.4. SCB visual methods features.....	81
2.5. SCBs kernel (Semantic Basic Code string) – SCB linear modification.....	91
2.6. Identifier compilation in SCB.....	94
2.7. SCB texts reduction to laconic form	106
2.8. SCB Syntax. Formal description	112

Summary to Chapter 2	116
3. SCB description of basic mathematical structures.....	117
3.1. Set typology and their representation in SCB. SCB basic sets and their corresponding key nodes	117
3.2. Cortege concept. Attributes of cortege elements. Cortege representation in SCB. Cortege typology	127
3.2.1. Cortege and attribute concept	127
3.2.2. Cortege samples and their description in SCBg and SCBs.....	128
3.2.3. Cortege typology	132
3.3. Concept of relation. Relation representation in SCB. Relation typology. Classical and non classical relations	134
3.3.1. Traditional relation generalization. Resume.....	134
3.3.2. Relation typology based on set base typology	137
3.3.3. Relation typology based on typology of corteges belonging to a relation	139
3.3.4. Relation typology based on projection concept	142
3.3.5. Relation typology based on functional dependence concept.....	144
3.3.6. Description and typology of classical relations.....	145
3.3.7. Boundary relations	146
3.3.8. Binary relations typology and metarelations	148
3.3.9. Set of correspondences as a metarelation assigned on sets of binary oriented relations	151
3.3.10. Ternary relations typology and metarelations.....	155
3.3.11. Relations on sets.....	155
3.3.12. Relations on corteges	166
3.3.13. Relations on relations.....	167
3.3.14. Numeric relations.....	174
3.4. Representation of relational structures in SCB. Relational structures typology. Classical and nonclassical relational structures.....	183
3.4.1. Description of relational structures in SCB.....	183
3.4.2. Typology of relational structures.....	188
3.4.3. Relations on relational structures. Relational metastructures	189
3.4.4. Graphic structures and relations on them	192
Chapter 3. Conclusions	195
4. Kernel of the open set of knowledge representation graph languages created on set-theory base.....	197
4.1. Basic concepts of logic languages	197

4.2.	SC language (Semantic Code) as a base for creating various logical languages and knowledge representation languages.....	199
4.3.	SCg language (Semantic Code graphical) – SC graphical modification	200
4.4.	SCs language (Semantic Code string) – SC linear modification.....	202
4.5.	Key nodes of SC graph language.....	204
4.6.	SC-sub language concept. Set of graph languages created on SC basis	209
4.7.	Concept of abstract sc-computer	211
4.8.	Concept of formal sc-model	215
	Chapter 4 conclusions.....	215
5.	Description of logical formulas and formal theories in memory of graph-dynamic associative computers.....	217
5.1.	Principles of SCL graph logical language (Semantic Code Logic) construction on set-theory basis.....	217
5.2.	Logical formulas recording using conventionalized natural language.....	218
5.3.	SCLs language (Semantic Code Logic string) – a formal linear logical language of a classical type using SCs for recording atomic logical formulas	220
5.4.	SCLg language – graphical variant for reproducing SCL texts	222
5.5.	Samples of recording of logical expressions in conventionalized natural language.....	225
5.6.	Formal meta theory and its description in SCL	250
	Chapter 5 conclusions.....	252
6.	Knowledge typology and knowledge representation languages in graph-dynamic associative computers.....	253
6.1.	Knowledge representation related to a measure concept	253
6.2.	Dynamic systems description.....	264
6.3.	Description of aims in graph-dynamical associative computers.....	272
6.4.	Hypertext semantic networks	285
6.5.	Principles of description of neuronet models	289
6.5.1.	Brief description of a total rectilinear model of pseudo-optical neuronet and of methods of calculation of its behaviour.....	290
6.5.2.	Principles of description of pseudo-optical neuronets in graph-dynamical computers memory.....	294
	Conclusions to Chapter 6.....	296
7.	Navigation and search graph-dynamical associative computer.....	297
7.1.	Navigation and search graph-dynamical associative computer operation	297
7.1.1.	Informational constructions describing a status of navigation and search graph-dynamical associative computer	297

7.1.2.	Search operations set of theoretical graphical area of the specified sc-element.....	300
7.1.3.	Search operations set in a specified formal theory boundaries of all true expressions relevant to the specified expressive form (given specimen)	301
7.1.4.	Search operations set of semantic areas of the specified sc-element.....	307
7.1.5.	Search operations set of the semantic connection between two or more specified sc- elements	309
7.1.6.	Navigation and search operations set in hypertext semantic network	310
7.2.	A sample of a navigation and search graph-dynamical associative computer operation	315
	Conclusions to Chapter 7	320
8.	Graph-dynamical associative logical computers.....	321
8.1.	A set of easily integrated abstract graph-dynamical associative logical computers.....	321
8.1.1.	Information constructions describing a status of an abstract graph-dynamical associa- tive logical computer.....	323
8.1.2.	Abstract graph-dynamical associative logical computer operations.....	325
8.2.	Task solution by abstract graph-dynamical associative logical computers.....	346
	Conclusions to Chapter 8	370
	Conclusion.....	371
	References	372

Предисловие

Данная книга посвящена рассмотрению графодинамических ассоциативных моделей 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.

Особенностью предлагаемых моделей является:

- ассоциативность;
- легкая интегрируемость;
- возможность обработки знаний не только простой, но и сложной структуры;
- единство языка и метаязыка;
- ориентация на параллельную асинхронную обработку знаний.

Данная книга предназначена как для специалистов в области искусственного интеллекта, так и для аспирантов и студентов, интересующихся проблемами искусственного интеллекта. В частности, материал данной книги может быть использован в таких учебных дисциплинах специальности «Искусственный интеллект», как:

- «Математические основы искусственного интеллекта»;
- «Модели представления знаний, базы данных и СУБД»;
- «Нейросетевые модели и нейрокомпьютеры»;
- «Логические основы интеллектуальных систем»;
- «Технология и инstrumentальные средства проектирования интеллектуальных систем».

Материалы данной книги являются результатом выполнения следующих проектов:

- «Параллельная графовая вычислительная система, ориентированная на решение задач искусственного интеллекта». Проект финансировался Министерством Промышленности Республики Беларусь и выполнялся под руководством В.В.Голенкова рядом организаций – Научно-исследовательским институтом электронных вычислительных машин (НИИЭВМ), Институтом технической кибернетики Национальной Академии Наук Беларуси, Белорусским государственным университетом информатики и радиоэлектроники. Со стороны НИИЭВМ большой вклад в выполнение этого проекта внесли В.М.Казан и Л.М.Романовская.
- «Разработать и реализовать семейство легко интегрируемых языков представления знаний и формальных моделей параллельной асинхронной переработки сложноструктурированных знаний, соответствующих разным стратегиям решения трудноформализуемых задач». Проект финансировался Фондом Информатизации РБ.
- «Разработать инструментальные средства построения интеллектуальных систем на основе графового языка логического программирования, ориентированного на распределенную переработку интегрированных баз данных и знаний». Проект финансировался Фондом Информатизации РБ.
- «Разработка принципов и методов создания интегрированных динамических экспертных систем». Проект финансируется Белорусским республиканским фондом фундаментальных исследований в рамках совместной белорусско-российской программы фундаментальных исследований “БРФФИ-РФФИ 2000”, № 00-7036. Руководителем этого проекта с российской стороны является доктор технических наук И.Б.Фоминых (Российский НИИ информационных технологий и автоматизации проектирования).
- «Анализ и моделирование псевдооптических нейронных сетей». Проект финансируется Белорусским республиканским фондом фундаментальных исследований в рамках совместной белорусско-российской программы фундаментальных исследований “БРФФИ-РФФИ 2000”, № 00-7029. Руководителем этого проекта с российской стороны является доктор технических наук О.П.Кузнецов (Институт проблем управления Российской Академии Наук), предоставленные которым материалы были использованы в 6-м разделе данной книги.

Основные исполнители указанных проектов – это лаборатория искусственного интеллекта Института технической кибернетики Национальной Академии Наук Беларусь, лаборатория интеллектуальных систем Белорусского государственного университета информатики и радиоэлектроники (БГУИР), кафедра интеллектуальных информационных технологий БГУИР, осуществляющая подготовку специалистов по специальности «Искусственный интеллект».

Начиная с 1996 года кафедра интеллектуальных информационных технологий БГУИР активно сотрудничает с компанией JV International Business Alliance. Благодаря этому сотрудничеству, кафедра за короткий срок в современных непростых условиях достигла необходимого уровня. Формы сотрудничест-

ва были направлены не только на повышение уровня подготовки молодых специалистов, в том числе путем подключения студентов (начиная со 2-го курса) к серьезным проектам, но и на совместную разработку перспективных конкурентоспособных научноемких программных продуктов. Данная книга является этому подтверждением.

В связи с вышесказанным, авторы книги выражают благодарность компании JV International Business Alliance за сотрудничество в развитии интеллектуальных информационных технологий, финансовую поддержку этих работ и, в частности, поддержку данного издания, а также за содействие в развитии специальности «Искусственный интеллект».

Авторы книги выражают искреннюю признательность всем участникам проектов, результаты которых вошли в книгу, преподавателям и аспирантам кафедры интеллектуальных информационных технологий БГУИР, а также студентам специальности «Искусственный интеллект», принимающим активное участие в выполнении проектов и их внедрении в учебный процесс. Следует особо отметить таких студентов, как А.Рукин, С.Хованский, А.Слиборский, С.Шкред, М.Рафалович, Д.Рожанский.

Особую благодарность авторы книги выражают создателю Российской Ассоциации искусственного интеллекта Д.А.Поспелову, а также всем членам этой Ассоциации за многолетнюю поддержку и плодотворное сотрудничество.

Авторы книги выражают свою признательность С.В.Петрову, работы которого в области графовых грамматик, а также его активный интерес к исследованиям авторов были важным стимулом развития этих исследований.

В книге для более наглядного и понятного восприятия текста принят ряд соглашений.

Текст книги разбивается на нумеруемые разделы, подразделы и пункты. Наименования разделов, подразделов и пунктов выделяются жирным шрифтом разного размера. В тексте встречаются также фрагменты, имеющие свои заголовки, которые выделяются разреженным жирным шрифтом. Такими заголовками выделяются: определения, утверждения, пояснения, доказательства, тексты формальных языков, рисунки, списки ключевых понятий, операции, микропрограммы и др. Указанные фрагменты текста могут иметь номер, который начинается с номера соответствующего раздела, подраздела или пункта.

Второстепенные примечания в тексте выделяются мелким шрифтом.

В начале подразделов и пунктов приводится список ключевых понятий, рассматриваемых в данном подразделе или пункте. В списке ключевых понятий и в той части текста, где это ключевое понятие определяется, оно выделяется жирным шрифтом.

Наиболее важные фразы, на которые рекомендуется обратить особое внимание, выделяются подчеркиванием.

Весь материал книги иллюстрируется подробными примерами и иллюстрациями. Формальные тексты выделяются жирным курсивом и иногда для наглядности заключаются в "рамочки".

Приводимые в тексте библиографические ссылки, помимо порядкового номера в списке литературы, содержат также (в круглых скобках) краткие идентификаторы этих ссылок, в которых указывается фамилия автора, год издания и аббревиатура названия статьи или книги. Например, ссылка ([Поспелов Д.А. 1986kn-СитуаУ](#)) означает, что речь идет о книге, автор которой Поспелов Д.А., издана книга в 1986 году и ее наименование «Ситуационное управление: Теория и практика».

Кроме того, в тексте используются различные шрифты для всевозможных разделителей (таких как точка, запятая, двоеточие, скобки и т.п.). Различие способов отображения разделителей продиктовано тем фактом, что описываемые формальные языки также содержат разделители, которые должны отличаться от тех, которые используются в обычном естественно-языковом тексте. В частности, используется три вида кавычек: "прямые" симметричные кавычки " ... " – для выделения всевозможных иносказаний; асимметричные жирные кавычки " ... " – для явного выделения (если это требуется в данном контексте) фрагментов формального текста описываемых языков; угловые кавычки « ... » – для выделения всевозможных наименований или цитат.

Введение

Расширение областей применения методов и средств искусственного интеллекта приводит к следующему:

- существенно увеличиваются размеры перерабатываемых баз знаний;
- усложняется структура перерабатываемых знаний. Это, в частности, вызвано переходом к предметным областям, имеющим сложную иерархическую, многоуровневую структуру, а также активным использованием метаинформации (метазнаний), что приводит к расслоению базы знаний на уровни в соответствии с тем, какие фрагменты знаний и по отношению к каким являются метазнаниями. В базе знаний современной интеллектуальной системы должно поддерживаться описание связей между связями, связей между целыми структурами, связей между различными фрагментами перерабатываемых баз знаний. Сложно-структурированный характер имеют интенсиональные знания, т.е. знания о свойствах и законах описываемых предметных областей, знания о семантике естественных языков, знания о нестационарных (динамических) системах, которые формально определяются как метасистемы, заданные на стационарных системах, определяющих различные состояния (ситуации) нестационарной системы, знания о конфликтующих системах с различным числом уровней рефлексии;
- усложняется семантика и расширяется типология задач, решаемых в интеллектуальных системах, что, в свою очередь, приводит к усложнению стратегий, используемых для их решения, к существенному усложнению операций (механизмов, правил) переработки знаний и к существенному увеличению их количества. В современных интеллектуальных системах используются не только классические, но и всевозможные не-классические логики (нечеткие логики, ситуационные логики, логики доверия, логики аргументации, логики открытия, логики разрешения противоречий, различные прикладные логики, псевдофизические логики, логики, поддерживающие немонотонный вывод в открытых квазиаксиоматических теориях, логики, моделирующие различного вида правдоподобные рассуждения - рассуждения "здравого смысла", рассуждения по аналогии, рассуждения по ассоциации, рассуждения на основе убеждений);
- существенно повышаются требования к открытости (гибкости, наращиваемости, модифицируемости) интеллектуальных систем. Открытость системы характеризуется возможностью широкого развития системы путем ее локальных изменений, высокой технологичностью (низкими накладными расходами) эволюции системы. Открытый характер предотвращает моральное старение сложных интеллектуальных систем, в ходе эксплуатации которых продолжается активный процесс их создания. В интеллектуальных системах нового поколения в ходе их эксплуатации легко наращиваться и модифицироваться должны не только сами базы знаний, но и языки представления знаний, а также системы операций переработки знаний (правил вывода). Открытый, гибкий, модифицируемый, легко перестраиваемый, адаптируемый характер интеллектуальных систем нового поколения означает то, что их формальной основой являются уже не формальные модели (формальные системы), в которых фиксируются языки, совокупность аксиом и система правил вывода, а семиотические модели, в которых не фиксируются ни языки, ни аксиомы, ни правила вывода;
- появляется необходимость в том, чтобы из нескольких интеллектуальных систем можно было легко построить интегрированную систему. Здесь имеются в виду и интеграция языков представления знаний, и интеграция самих баз знаний, и интеграция систем операций переработки знаний. В современные интеллектуальные системы должны легко интегрироваться (при необходимости) самые различные модели переработки знаний: и классические модели дедуктивного вывода, и модели нечеткого вывода, и модели индуктивного вывода (обнаружения закономерностей, формирования понятий, формирования гипотез), и всевозможные модели распознавания (классификации, идентификации, обнаружения), и модели обучения, и модели принятия решений (планирования поведения, ситуационного управления), и многие другие модели. Возможность такой интеграции различных моделей, в частности, позволяет легко синтезировать конкретную интегрированную модель переработки информации, требуемую для конкретной прикладной интеллектуальной системы, из имеющегося арсенала стратегий и механизмов решения задач;
- существенно повышается потребность в гибридных интеллектуальных системах, которые представляют собой результат интеграции интеллектуальных систем с другими типами систем переработки информации (например, с различными пакетами прикладных программ, с различными информационно-поисковыми системами, поддерживаемыми различными СУБД);
- появляется необходимость в интеграции дискретных и непрерывных моделей переработки информации (к последним, в частности, могут быть отнесены нейросетевые модели) и к созданию моделей переработки информации, сочетающих в себе как дискретные, так и непрерывные свойства (к таким моделям, в частности, могут быть отнесены формальные модели нечеткого логического вывода);
- появляется острая необходимость в создании эффективных средств защиты от несанкционированных изменений, вносимых в интеллектуальную систему в ходе ее эксплуатации (это оборотная сторона высокого уровня гибкости), а также от несанкционированного применения интеллектуальной системы, в частности, от несанкционированного доступа к хранимым знаниям. К указанным средствам защиты относятся, в частности, средства обнаружения противоречий как в базах знаний, так и в системах операций переработки знаний, средства тестирования (отладки) баз знаний и систем операций;

- появляется необходимость в поддержке взаимодействия интеллектуальной системы (в ходе решения задачи) не только с одним пользователем, но и с целым коллективом пользователей. В рамках такого коллективного (распределенного) решения общей задачи интеллектуальная система становится равноправным членом этого коллектива, обеспечивающим координацию совместного решения соответствующей задачи. Указанные интеллектуальные системы строятся на основе специального класса моделей решения задач - кооперативных моделей [1; 64] (*Абдрахманов Б.К. 1990г-ст-ИсслЕВРА ; Macintosh D.J.. 1987art-DistrDEFDES*).

Перечисленные тенденции развития интеллектуальных систем, в свою очередь, приводят к недопустимому увеличению времени реакции создаваемых прикладных интеллектуальных систем, а также к недопустимому увеличению сроков разработки и отладки прикладных интеллектуальных систем.

В настоящее время разработчиков и потребителей прикладных интеллектуальных систем не устраивает ни производительность этих систем, ни степень их гибкости, которая в конечном счете определяется трудоемкостью внесения изменений в ходе отладки интеллектуальных систем или в ходе их совершенствования. Очевидно, что всем этим сдерживается использование в прикладных интеллектуальных системах мощных современных моделей представления и переработки знаний, создаваемых в теоретических исследованиях по искусственноному интеллекту. Признанным в настоящее время является то, что разрешение указанного противоречия лежит в направлении создания компьютеров, имеющих нетрадиционную архитектуру и ориентированных на реализацию интеллектуальных систем. Тем не менее результаты исследований по созданию таких компьютеров представляются в настоящее время достаточно скромными. Причиной этого является не только сложность проблемы, но и то, что работы по созданию средств аппаратной поддержки интеллектуальных систем либо направлены на аппаратную поддержку частных аспектов и не претендуют на повышение эффективности всей технологии проектирования интеллектуальных систем, либо ориентируются на весьма простые на сегодняшний день модели представления и переработки знаний, в результате чего не поддерживаются ни сложные иерархические структуры знаний, ни сложные правила вывода, ни семиотические (самоизменяемые, динамические) модели переработки знаний, ни интеграция различных моделей переработки знаний.

Итак, расширение областей применения методов и средств искусственного интеллекта приводит к необходимости создания принципиально новых инструментальных средств, поддерживающих переработку сложноструктурированных знаний, а также легко расширяемые и модифицируемые стратегии решения неформализованных задач. В основе таких инструментальных средств должны лежать специальные компьютеры, ориентированные на параллельную переработку сложноструктурированных знаний (параллелизм здесь необходим для обеспечения требуемой производительности сложных интеллектуальных систем), а также согласованные с этими компьютерами языки представления знаний и формальные модели параллельной переработки знаний.

Целесообразность комплексного подхода к созданию параллельных компьютеров переработки сложноструктурированных знаний подтверждается опытом разработки различных нетрадиционных компьютеров. Этот опыт показывает, что практическая и, следовательно, коммерческая ценность каждой такой разработки обусловливается не столько эффективностью самого компьютера, сколько эффективностью технологии его использования. В особенности это касается компьютеров, требующих уточнения нетрадиционных информационных технологий.

Для интеллектуальных систем нового поколения необходимость перехода от создания отдельных моделей переработки информации к созданию легко расширяемого семейства согласованных друг с другом моделей обусловлена, во-первых, тем, что в интеллектуальных системах должны использоваться не только собственно модели переработки знаний (модели дедуктивного вывода, модели индуктивного вывода, модели правдоподобных рассуждений и т.д.), но и нейросетевые модели, а также модели, ориентированные на использование хранимых программ, имеющих самую разнообразную денотационную и операционную семантику, и, во-вторых, тем, что практическая реализация всевозможных моделей переработки знаний предполагает их интерпретацию в конечном счете на основе процедурных моделей. Необходимость переноса акцента с разработки отдельных моделей на разработку целого семейства согласованных друг с другом моделей имеет особую актуальность, если речь идет о создании семейства параллельных асинхронных моделей. Это связано с тем, что проблема интерпретации параллельных моделей переработки знаний на основе соответствующих процедурных моделей является нетривиальной, так как подходящие для этой цели процедурные параллельные модели отсутствуют.

В данной книге, а также в работе [411] (*ПрогрВАМ-2001кн*) описываются результаты исследований, которые посвящены созданию легко расширяемого семейства согласованных друг с другом моделей параллельной переработки информации, в состав которого включаются:

- различные модели параллельной переработки знаний, соответствующие различным логикам, различным стратегиям решения задач;
- различные модели, обеспечивающие реализацию хранимых в памяти программ и неформальных процедур, устроенных самым различным способом;
- базовые модели, обеспечивающие интерпретацию всех вышеперечисленных моделей, ориентированные на реализацию в специально предназначенных для этого параллельных компьютерах и определяющие архитектуру этих компьютеров.

Указанные формальные модели, соответствующие различным уровням параллельной интеллектуальной системы, нецелесообразно разрабатывать независимо друг от друга, так как хорошая согласованность этих моделей, основанная на простых принципах их интеграции и интерпретации, является важнейшим фактором эффективности интеллектуальной системы в целом.

Предлагаемый подход к решению поставленных задач основан на следующих положениях:

- ориентация на графодинамическую парадигму переработки информации, в основе которой лежит понятие графодинамической (нелинейной структурно перестраиваемой) памяти, где поддерживается развитая форма ассоциативного доступа, а переработка информации сводится не только к изменению состояния элементов памяти, но и к изменению конфигурации связей между ними;
- создание в рамках графодинамической парадигмы целого семейства графовых языков (графовых языков представления знаний различного вида, графовых языков программирования), т.е. языков, конструкциями (текстами) которых являются графовые конструкции;
- использование принципа языка-ядра (базового языка) при создании указанного выше семейства графовых языков;
- ориентация на открытые языки;
- стремление максимальным образом приблизиться в создаваемых графовых языках к семантическому представлению информации, основанному на теоретико-множественной парадигме, что приводит к использованию специальных семантических сетей в качестве конструкций этих языков;
- ориентация на однородные семантические сети. Семантические сети, относящиеся к классу однородных, обеспечивают более унифицированное представление различного вида информации и тем самым существенно упрощают интеграцию различной информации в рамках одной базы знаний, а также существенно упрощают реализацию различных графодинамических моделей;
- стремление к созданию простых и выразительных метаязыковых средств в используемых однородных семантических сетях, что должно обеспечить представление сложноструктурированной информации любого вида;
- ориентация на асинхронный (децентрализованный) принцип управления процессом переработки информации, который по сравнению с синхронным принципом обеспечивает более высокую гибкость моделей переработки информации.

Основным результатом работы является комплекс инstrumentальных средств проектирования прикладных интеллектуальных систем различного назначения, включающий следующие компоненты:

- фактографический язык SCB (Semantic Code Basic);
- язык SC (Semantic Code), являющийся расширением языка SCB и основой для построения различных логических языков и языков программирования;
- язык SCL (Semantic Code Logic), являющийся одним из возможных (но не единственным) логических языков, построенных на основе языка SC;
- язык описания целей (заданий) в графодинамических ассоциативных машинах;
- язык SCP (Semantic Code Programming), являющийся языком программирования, ориентированным на переработку знаний, представленных в виде текстов языка SC и его подъязыков;
- среда разработки прикладных интеллектуальных систем, включающая средства ввода и редактирования баз знаний на указанных языках, средства разработки и отладки соответствующих программ.

Практическая ценность полученных результатов заключается в том, что предложенный инструментальный комплекс проектирования параллельных интеллектуальных систем позволяет, во-первых, существенно расширить области применения методов и средств искусственного интеллекта и, во-вторых, значительно повысить возможности проектируемых интеллектуальных систем. Это достигается благодаря:

- эффективной поддержке сложноструктурированных знаний;
- поддержке гибких (открытых, легко расширяемых, модифицируемых) и мощных стратегий и механизмов решения трудноформализуемых задач;

- поддержке параллельной переработки знаний.

Перспективными направлениями применения предложенного в данной работе комплекса средств проектирования интеллектуальных систем являются все те приложения, где приходится оперировать сложноструктурными знаниями, где требуется использование сложных моделей решения задач, где необходим высокий уровень гибкости систем, где имеют место серьезные ограничения во времени. Примерами таких приложений являются:

- подсистемы естественно-языкового текстового и речевого интерфейса, для создания которых необходимы лингвистические базы знаний, имеющие весьма сложную иерархическую структуру, переработка которых должна осуществляться в реальном масштабе времени;
- интеллектуальные обучающие и тренажерные системы [236] (*ИнтелОСиВУО-2001кн*);
- интеллектуальные систем автоматизации деятельности организаций, интеллектуальные виртуальные организации [236] (*ИнтелОСиВУО-2001кн*), интеллектуальные системы управления корпоративными знаниями;
- интеллектуальные геоинформационные системы;
- интеллектуальные системы поддержки принятия решений в условиях дефицита времени;
- интеллектуальные системы управления сложными объектами;
- интеллектуальные системы поддержки проектирования сложных объектов;
- интеллектуальные системы защиты информационных ресурсов открытых корпоративных систем.

1. Графодинамическая парадигма обработки информации

Основной задачей данного раздела является определение графодинамической парадигмы обработки информации как наиболее перспективной основы для создания интеллектуальных систем нового поколения, в которых должны поддерживаться сложноструктурированные базы знаний большого объема, сложные стратегии и механизмы решения трудноформализуемых задач, высокий уровень гибкости используемых стратегий и механизмов решения задач, высокий уровень интегрируемости различных моделей переработки знаний, асинхронизм и высокий уровень параллелизма. Кроме того, здесь же осуществляется уточнение основных понятий, связанных с графодинамической парадигмой и уточнение предлагаемого подхода к созданию комплекса средств, ориентированных на поддержку интеллектуальных систем нового поколения.

Данный раздел может быть использован в качестве учебного пособия по дисциплине «Инструментальные средства и технологии проектирования интеллектуальных систем» для студентов специальности «Искусственный интеллект».

1.1. Концепция графодинамических моделей

Ключевые понятия: графодинамические параллельные асинхронные модели; формальная модель обработки информации; интеграция; интерпретация.

Одной из характерных особенностей современного состояния информатики является наличие огромного многообразия способов, парадигм, принципов, методов организации обработки информации. Основным источником такого многообразия на сегодняшний день являются работы в области искусственного интеллекта. При этом следует подчеркнуть, что многообразие способов организации обработки информации находит отражение не только во множестве языков программирования, во множестве программных инструментальных средств проектирования интеллектуальных систем, но и в аппаратных средствах, в архитектурах компьютеров новых поколений.

Для того чтобы разобраться во всем многообразии способов организации обработки информации, необходимо, во-первых, иметь возможность абстрагироваться от несущественных деталей реализации и, во-вторых, иметь возможность так трактовать сравниваемые способы обработки информации, чтобы были исключены их несущественные отличия, несущественные особенности.

Таким образом, важнейшим подходом к исследованию различных способов организации обработки информации в современных системах является исследование соответствующих формальных (абстрактных) моделей.

Основными особенностями рассматриваемых в данной книге моделей являются:

- графодинамический характер, т.е. представление перерабатываемой информации в виде нелинейных конструкций и структурная перестраиваемость системы связей между элементами памяти (абстрактной или аппаратно реализованной) в ходе переработки хранимых в этой памяти информационных конструкций;
- ассоциативность, т.е. использование ассоциативного метода доступа к требуемым фрагментам перерабатываемой информации;
- параллелизм;
- асинхронность;
- возможность обработки информации с помощью операций высокого уровня.

Модели обработки информации указанного класса будем называть **графодинамическими параллельными асинхронными моделями**. Нетрудно заметить, что перечисленные выше свойства этих моделей тесно связаны друг с другом и являются полной противоположностью свойствам классических фон-Неймановских моделей обработки информации, лежащих в основе семейства традиционных компьютеров. К числу свойств классических моделей относятся:

- 1) линейный характер перерабатываемых информационных конструкций, линейный характер связи между элементами памяти и структурно фиксированный характер изменения состояния памяти;
- 2) адресный метод доступа к требуемым фрагментам перерабатываемой информации, в основе которого лежит фиксированное, т.е. неизменяемое в процессе обработки информации, взаимно однозначное соответствие между множеством элементов памяти и множеством их имен (идентификаторов), – такие имена (в качестве которых обычно используются натуральные числа) называются адресами элементов памяти;

- 3) последовательный характер переработки;
- 4) синхронный, т.е. основанный на централизованном управлении, характер организации обработки информации;
- 5) низкий уровень операций обработки информации.

Подробнее о перечисленных свойствах классических моделей обработки информации см. в работе [129] ([Глушков В.М..1974пр-РекурMiBT](#)).

Термин "графодинамика" заимствован нами из работы [10] ([Айзerman M.A..1988ст-ДинамПкАС](#)).

Известными примерами графодинамических моделей обработки информации, в частности, являются алгоритмы Колмогорова [259] ([Колмогоров А.Н..1958ст-кОпредA](#)), вегетативная машина Борщева [64] ([Борщев В.Б.1989ст-ВегетM](#)), всевозможные графовые грамматики [379; 376; 377; 378] ([Петров С.В..1978ст-ГрафоГиA](#); [Петров С.В..1977ст-ГрафоГиЗГ](#); [Петров С.В..1977ст-НормаФГГ](#); [Петров С.В..1977ст-СтандФГГ](#)).

1.1.1. Понятие формальной модели обработки информации

Ключевые понятия: формальная модель обработки информации, первичные элементы, синтаксические правила, начальная информационная конструкция, операции формальной модели, формальная модель, язык, абстрактная машина.

Приведем классическое определение формальной модели.

Определение 1.1.1.1. [390; 392; 401] ([Поспелов Д.А.1981кн-ЛогикЛМ](#) ; [Поспелов Д.А.1986кн-СитуаУ](#) ; [Поспелов Д.А.ред.1990спр-ИскусИ-К2](#)). Формальная модель обработки информации F , называемая также формальной системой, исчислением, определяет процесс переработки информационной конструкции и задается четверкой: $F = (T, P, S, W)$, где

- T – множество **первичных элементов** (терминальных элементов, базовых элементов, элементарных конструкций) формальной модели;
- P – множество **синтаксических правил** формальной модели, которые определяют множество синтаксически правильных (правильно построенных) информационных конструкций (конструктивных объектов), перерабатываемых в рамках данной формальной модели;
- S – **начальная (исходная) информационная конструкция** формальной модели, т.е. начальное состояние перерабатываемой информационной конструкции, которое иногда называют совокупностью аксиом;
- W – множество **операций формальной модели**, т.е. правил построения новых информационных конструкций из уже построенных, правил преобразования (модификации) текущего состояния перерабатываемой информационной конструкции.

Операции формальной модели иногда называют правилами вывода, которые не следует отождествлять с правилами логического вывода, поскольку формальные модели могут быть не только логическими.

Нетрудно заметить, что понятие формальной модели включает в себя три аспекта:

- представление (кодирование) информации в виде некоторых информационных конструкций, устройство этих информационных конструкций, их соотношение с описываемой предметной областью, т.е. устройство языка, используемого формальной моделью, его синтаксис и семантика;
- построение начальной информационной конструкции формальной модели, представляющей собой исходное описание некоторой конкретной предметной области. Для формальной модели переработки знаний, отражающей логико-семантический уровень интеллектуальной системы, начальная информационная конструкция называется исходным состоянием базы знаний;
- организация переработки информационных конструкций. Таким образом, наряду с приведенным выше определением формальной модели можно дать следующее эквивалентное определение.

Определение 1.1.1.2. Формальная модель F задается тройкой $F = (L, S, C)$, где

- L** – язык формальной модели F с присущими ему синтаксисом и семантикой;
- S** – начальная информационная конструкция формальной модели F , которая должна принадлежать языку L ;
- C** – абстрактная машина обработки информации, определяющая операции (правила преобразования) конструкций языка L . Множество операций абстрактной машины C в точности соответствует множеству операций формальной модели, которое в определении 1.1.1 обозначено символом W .

Формальная модель рассматривает процесс обработки информации как процесс преобразования информационной конструкции, хранимой в памяти абстрактной машины. Следовательно, текущее состояние такого процесса полностью определяется текущим состоянием перерабатываемой информационной конструкции, т.е. текущим состоянием памяти абстрактной машины.

Приведенное определение формальной модели обработки информации условно разбивает модель обработки информации на модель представления информации и модель преобразования информации (модель манипулирования информационными конструкциями).

Язык L определяется множеством информационных конструкций, которое называется множеством синтаксически правильных (правильно построенных) конструкций этого языка. Описание синтаксиса языка формальной модели должно быть конструктивным определением множества синтаксически правильных его конструкций, которое соответственно задается а) множеством первичных элементов (базовых элементов, элементарных, атомарных конструкций) языка и б) множеством синтаксических правил (см. определение 1.1.1). То, как соотносится произвольная синтаксически правильная конструкция этого языка с фрагментом предметной области, описываемой этой конструкцией, будем называть денотационной семантикой языка, а соотношение конкретной конструкции языка с описываемым этой конструкцией фрагментом предметной области будем называть денотационной семантикой указанной конструкции.

Абстрактная машина C задается а) абстрактной памятью (абстрактной запоминающей средой), в которой хранятся перерабатываемые информационные конструкции, и б) множеством операций. Текущее состояние абстрактной памяти представляет собой текущее состояние перерабатываемой информационной конструкции. В этом смысле абстрактная память есть нестационарная (динамическая, изменяющаяся во времени) информационная конструкция. Структура памяти абстрактной машины, ее "статические" свойства определяются синтаксисом языка L . Принципы изменения состояния памяти абстрактной машины, т.е. "динамические" свойства хранимых в памяти информационных конструкций, характер преобразования информационных конструкций определяются операциями абстрактной машины. На одной и той же абстрактной машине могут быть реализованы разные формальные модели, отличающиеся друг от друга разными начальными информационными конструкциями, которые задают разное исходное состояние памяти абстрактной машины. Таким образом, каждому сочетанию абстрактной машины и языка соответствует целое семейство формальных моделей, использующих указанный язык и реализуемых на указанной абстрактной машине. Могут существовать формальные модели, отличающиеся разными начальными информационными конструкциями, разными языками, но имеющие одинаковые операции. Такие формальные модели также могут быть реализованы на одной и той же абстрактной машине. Могут существовать формальные модели, отличающиеся разными начальными информационными конструкциями, разным набором операций, но имеющие одинаковые языки.

Язык, которому однозначно ставится в соответствие набор операций, т.е. определенная абстрактная машина, будем называть языком с фиксированной операционной семантикой. Операционная семантика такого языка задается соответствующей абстрактной машиной. Все остальные языки будем называть языками с нефиксированной операционной семантикой. Языками с фиксированной (четко заданной) операционной семантикой являются все языки программирования. В отличие от этого языки представления знаний могут иметь нефиксированную операционную семантику. Это означает, что одному и тому же языку представления знаний могут быть поставлены в соответствие разные методы решения задач в рамках этого языка.

Рассматриваемая нами трактовка формальной модели дает возможность четко выделить три этапа разработки конкретных формальных моделей:

- разработка языка (языка программирования или языка представления знаний);
- разработка абстрактной машины (машины реализации хранимых программ или машины переработки знаний);

- разработка начальной информационной конструкции начального состояния памяти абстрактной машины (конкретной программы вместе с ее конкретными исходными данными или исходного состояния некоторой базы знаний).

Таким образом, рассматриваемая трактовка формальной модели дает возможность явно связать формальную модель с главными компонентами инstrumentальных средств, обеспечивающих ее реализацию, – с соответствующим языком и с соответствующей абстрактной машиной.

Кроме того, используемое понятие абстрактной машины дает возможность с общих позиций рассмотреть принципы организации обработки информации в формальных моделях самого различного вида, а также дает возможность исследовать новые архитектуры компьютеров следующих поколений, в частности, компьютеров, ориентированных на реализацию интеллектуальных систем.

1.1.2. Классификация формальных моделей обработки информации

Ключевые понятия: формальная модель реализации хранимых программ, формальная модель переработки знаний, формальная модель переработки данных, формальные модели реализации хранимых процедурных программ, формальные модели реализации хранимых непроцедурных программ, формальная модель переработки символьных конструкций, формальная модель переработки графовых конструкций, графодинамические формальные модели.

Первым признаком классификации формальных моделей обработки информации является тип денотационной семантики начальных информационных конструкций.

Если начальная информационная конструкция формальной модели включает в себя программы, описывающие методы решения некоторых классов задач, и исходные данные для ряда конкретных задач из указанных классов, а операциями формальной модели являются операции реализации таких программ, то соответствующую формальную модель будем называть **формальной моделью реализации хранимых программ**.

Если начальная информационная конструкция формальной модели включает в себя фактографические (экстенсиональные) знания, относящиеся к некоторой предметной области, и знания о свойствах и законах этой предметной области (интенсиональные знания) [242] ([Кандрашина Е.Ю.. 1989 книга ПредсЗоВиП](#)), а операциями формальной модели являются операции переработки знаний, то соответствующую формальную модель будем называть **формальной моделью переработки знаний**. Такие операции могут поддерживать самые различные стратегии решения задач (стратегии рассуждений) – классические и неклассические, дедуктивные и индуктивные, четкие и нечеткие, строгие и правдоподобные. Частным видом формальных моделей переработки знаний являются формальные модели логического вывода.

Если в начальной информационной конструкции нельзя выделить ни фактографические и интенсиональные знания, ни программы, то соответствующую формальную модель будем называть **формальной моделью переработки данных**. Система операций такой формальной модели есть не что иное, как программа, отражающая некоторый метод решения некоторого класса задач, а начальная информационная конструкция такой модели есть не что иное, как исходные данные для конкретной задачи из указанного класса. Классическим примером формальных моделей переработки данных являются различные неуниверсальные машины Тьюринга с разными системами команд и с разными исходными состояниями памяти.

Формальные модели реализации хранимых программ в зависимости от типа программ делятся на **формальные модели реализации хранимых процедурных программ** и **формальные модели реализации хранимых непроцедурных программ**. Подчеркнем, что каждая формальная модель реализации хранимых непроцедурных программ, строго говоря, включает в себя некоторую формальную модель логического вывода, начальная информационная конструкция которой формируется в результате связывания непроцедурной программы с исходными данными конкретной задачи. Отметим при этом, что для реализации непроцедурных программ обычно используется достаточно простой набор операций логического вывода. Таким образом, непроцедурная программа с семантической точки зрения представляет собой совокупность знаний, являющихся общими для некоторого класса предметных областей и достаточными для решения однотипных (в определенном смысле слова) задач, задаваемых в рамках указанных предметных областей.

Вторым признаком классификации формальных моделей является структурный тип перерабатываемых информационных конструкций.

Если перерабатываемые информационные конструкции являются цепочками (строками) символов, т.е. символьными конструкциями, то соответствующую формальную модель будем называть **формальной моделью переработки символьных конструкций**.

Если перерабатываемые информационные конструкции являются графовыми конструкциями, то соответствующую формальную модель будем называть **формальной моделью переработки графовых конструкций**.

Третьим признаком классификации формальных моделей является характер возможных преобразований перерабатываемых информационных конструкций – возможность изменения связей между элементами этих конструкций. По этому признаку формальные модели делятся на формальные модели переработки структурно фиксированных информационных конструкций и формальные модели переработки структурно перестраиваемых информационных конструкций.

Основным предметом исследования данной работы являются формальные модели переработки графовых структурно перестраиваемых информационных конструкций, которые будем называть **графо-динамическими формальными моделями**.

По типу используемых формальными моделями абстрактных машин они делятся на последовательные и параллельные модели, синхронные и асинхронные и т.д.

1.1.3. Интеграция и интерпретация формальных моделей обработки информации

Ключевые понятия: интеграция, интерпретация.

Важнейшими соотношениями между формальными моделями являются соотношения **интеграции** и **интерпретации**.

Интеграция двух формальных моделей заключается в построении некоторой третьей модели, гарантирующей решение всех задач, решаемых в интегрируемых моделях, и дополнительно обеспечивающей решение некоторых новых задач, для которых требуется совместное использование интегрируемых моделей.

Интеграция формальных моделей в общем случае предполагает:

- перекодировку начальных информационных конструкций интегрируемых моделей;
- соединение перекодированных начальных информационных конструкций;
- преобразование каждой операции интегрируемой модели в эквивалентную операцию интегрированной формальной модели.

Формальные модели, которым соответствует достаточно простая технология интеграции, будем называть **легко интегрируемыми**.

Интерпретация произвольной формальной модели сводится:

- к выбору некоторого языка программирования;
- к описанию на этом языке программы реализации каждой операции интерпретируемой формальной модели в виде демонической (периодически инициируемой) программы;
- к разработке системы операций, обеспечивающих реализацию хранимых программ, написанных на указанном языке программирования;
- к формированию начальной информационной конструкции, включающей в себя начальную информационную конструкцию интерпретируемой формальной модели (возможно, каким-либо способом перекодированную) и тексты демонических программ, описывающих операции интерпретируемой формальной модели.

1.1.4. Семиотические модели обработки информации

Ключевые понятия: семиотическая модель, метаоперация, метаязык, семиотическая абстрактная машина.

В ходе расширения сферы применения интеллектуальных систем стало ясно, что формальные модели отражают не все аспекты обработки информации в интеллектуальных системах. Так, например, при обучении, при самообучении, при адаптации к изменяющейся внешней среде интеллектуальная система должна иметь возможность пересматривать начальную информационную конструкцию, модифицировать язык описания внешней среды и даже модифицировать правила решения задач. В основе интеллектуальных систем нового поколения должны лежать такие модели обработки информации, в которых не фиксируется ни начальная информационная конструкция, ни язык, ни абстрактная машина. Такие модели называются **семиотическими моделями** или семиотическими системами [392] (*Поспелов Д.А. 1986кн-СитуаУ*).

Семиотические модели носят открытый, гибкий, легко перестраиваемый, модифицируемый, самоизменяемый, адаптируемый, эволюционируемый, немонотонный, динамический характер.

В рамках семиотической модели процесс обработки информации рассматривается не только как процесс преобразования информационной конструкции по правилам некоторой формальной модели, но и как процесс перехода от одних формальных моделей к другим.

Текущее состояние процесса обработки информации, описываемого семиотической моделью, в общем случае определяется некоторым множеством взаимодействующих друг с другом формальных моделей, а точнее, текущим состоянием всех формальных моделей, входящих в это множество. Текущее состояние каждой формальной модели определяется текущим состоянием перерабатываемой информационной конструкции (т.е. текущим состоянием памяти соответствующей абстрактной машины), текущим состоянием используемого языка и текущим состоянием используемой системы операций. Таким образом, семиотическую модель можно трактовать как метамодель, описывающую некоторое семейство формальных моделей и, в частности, задающую правила перехода от одних формальных моделей к другим формальным моделям из этого семейства. С другой стороны, семиотическую модель можно трактовать как систему динамических (изменяющихся) формальных моделей.

Определение 1.1.4.1. Семиотическая модель *M* может быть задана парой *M = (FS, WF)*,

где

FS – одна или несколько формальных моделей, определяющих исходное состояние семиотической модели. В исходном состоянии семиотической модели каждая формальная модель, входящая в семейство *FS*, может быть либо активной (инициированной, непосредственно реализуемой), либо пассивной (неинициированной, нереализуемой). При этом 1) активных формальных моделей, реализуемых параллельно (одновременно), может быть сколько угодно, 2) эти формальные модели могут иметь общие фрагменты перерабатываемых информационных конструкций;

WF – **метаоперации** (метаправила) модификации семейства формальных моделей, определяющие текущее состояние семиотической модели. Эти метаоперации обеспечивают: 1) перевод пассивных формальных моделей в активное состояние; 2) перевод активных формальных моделей в пассивное состояние; 3) ликвидацию формальных моделей; 4) синтез новых или модификацию имеющихся формальных моделей.

Синтез новой формальной модели или модификация имеющейся формальной модели в общем случае предполагает синтез (или модификацию) языка, синтез (или модификацию) начальной информационной конструкции, синтез (или модификацию) набора операций.

Каждая метаоперация, входящая во множество *WF*, может задавать любые сочетания различных видов модификации формальных моделей (активизацию, прерывание, ликвидацию формальных моделей, модификацию языка, модификацию начальной информационной конструкции и модификацию операций). Поэтому разбить множество метаправил на подмножества в соответствии с видами модификации формальных моделей представляется возможным не для всех семиотических моделей.

Одним из вариантов реализации семиотической модели $M = (FS, WF)$ является построение формальной метамодели $F = (L, S, C)$, обеспечивающей интерпретацию семиотической модели M . Построение формальной метамодели F включает в себя:

- создание метаязыка L , обеспечивающего описание всевозможных формальных моделей (описание синтаксиса и семантики различных языков, описание семантики различных операций, представление начальных информационных конструкций формальных моделей, представление текущего состояния формальных моделей, т.е. текущего состояния перерабатываемых информационных конструкций), обеспечивающего описание соотношений между формальными моделями, определяющими одно состояние или разные состояния реализуемой семиотической модели, и обеспечивающего описание семантики метаопераций реализуемой семиотической модели, т.е. метаопераций, входящих во множество WF ;
- построение начальной информационной конструкции S , включающей в себя полное описание всех формальных моделей, входящих во множество FS , и полное описание всех метаопераций, входящих во множество WF ;
- создание универсальной абстрактной машины C , обеспечивающей интерпретацию всевозможных формальных моделей по их описаниям на метаязыке L , а также интерпретацию всевозможных метаопераций по их описаниям на метаязыке L . Такие абстрактные машины, использующие метаязык рассмотренного класса, обеспечивающие параллельную интерпретацию нескольких формальных моделей и легко переходящие с интерпретации одних формальных моделей на интерпретацию других формальных моделей с новыми языками, начальными информационными конструкциями и операциями, будем называть **семиотическими абстрактными машинами**. Разработка таких машин является одним из важнейших современных направлений развития работ в области искусственного интеллекта.

1.1.5. Графодинамические ассоциативные модели обработки знаний и известные виды моделей представления знаний

Ключевые понятия: графодинамические модели переработки знаний, сетевые модели, графовая конструкция.

Рассмотрим соотношение **графодинамических моделей переработки знаний** с известными классами моделей представления знаний, к числу которых относятся сетевые модели представления знаний, логические, фреймовые, продукционные модели.

Очевидно, что сетевые модели представления знаний, в которых знания представляются в виде семантических сетей, устроенных тем или иным способом, имеют к графодинамическим моделям переработки знаний самое непосредственное отношение, ибо семантическая сеть есть частный вид графовой конструкции.

Недостатки, обычно приписываемые **сетевым моделям** [120] (*Георгиев В.О. 1993 ст-МоделПЗ*), относятся не столько к самим моделям, сколько к сложностям их реализации на традиционных (фон-Неймановских) компьютерах. Если же вести речь о специальных графодинамических компьютерах, ориентированных на реализацию графодинамических моделей, то преимущество моделей такого класса становится очевидным, ибо их выразительные возможности намного превышают возможности других моделей. Тем более, что графодинамические модели представления и переработки знаний не являются альтернативой фреймовым, логическим и продукционным моделям, поскольку вполне могут существовать графодинамические фреймовые модели (в рамках которых фреймы представляются в виде графовых конструкций), графодинамические логические модели (в рамках которых логические высказывания представляются в виде графовых конструкций, а логический вывод рассматривается как манипуляция такими конструкциями), графодинамические продукционные модели (в рамках которых перерабатываемая продукциями информация и сами продукции представляются в виде графовых конструкций). Более того, в рамках графодинамической парадигмы легко представить себе комплексные модели, сочетающие в себе и фреймовые, и логические, и продукционные аспекты, т.е. модели, в которых интегрируются достоинства перечисленных типов моделей и нейтрализуются их недостатки. О целесообразности создания таких комплексных моделей говорится в целом ряде работ [401; 120] (*Поспелов Д.А. реd. 1990 спр-ИскусИ-К2 ; Георгиев В.О. 1993 ст-МоделПЗ*).

В основе фреймовых моделей представления знаний лежит системно-структурная трактовка описываемой предметной области, при которой предметная область рассматривается как совокупность взаимосвязанных подсистем, каждая из которых описывается в виде соответствующего фрейма. Связи

между фреймами могут быть самыми различными, в частности, они могут иметь достаточно сложную иерархическую структуру. **Графовые конструкции** являются удобным средством представления фреймовых структур, поэтому фреймовые модели часто считают частным случаем сетевых. Соответствующие языки представления знаний условно можно назвать графовыми фреймовыми языками представления знаний.

Графодинамические модели обработки информации хорошо согласуются также и с логическими моделями, так как в этих моделях ничто не запрещает в качестве перерабатываемых информационных конструкций использовать графовые конструкции. В частности, ничто не запрещает строить так называемые графовые логические языки, лежащие в основе графодинамических логических моделей представления и переработки знаний. В указанных логических языках графового типа логические высказывания (логические формулы) представляются не в виде символьных конструкций, как в известных логических языках, а в виде графовых конструкций. Наиболее сложным при создании графовых логических языков является выбор удобного способа представления логических связок и кванторов.

Графодинамическая парадигма является единственной возможной основой для формального рассмотрения и реализации параллельной переработки знаний. Это обусловлено тем, что параллельная переработка знания всегда есть одновременная переработка различных и заранее непредсказуемых фрагментов этого знания и одновременная реализация сразу нескольких стратегий решения одной и той же задачи. Любой фрагмент перерабатываемого знания не может быть выделен из этого знания таким образом, чтобы у этого фрагмента отсутствовали связи с остальной частью знания. Такая связь, в частности, может осуществляться через имена (идентификаторы), присваиваемые различным объектам представляемого знания. Указанное свойство знания можно назвать его связностью или целостностью. Из этого свойства следует, что при переработке каждого фрагмента знания в общем случае может не предсказуемо измениться не только внутренняя структура этого фрагмента, но и его внешние связи (в том числе и метасвязи) с остальной частью перерабатываемого знания, т.е. при переработке каждого фрагмента знания в общем случае требуется учет не только его внутренней структуры, но и его контекста (ближайшего окружения). При последовательной переработке какого-либо знания нет необходимости говорить о различных фрагментах этого знания – в каждый момент времени в качестве перерабатываемого фрагмента можно в этом случае рассматривать все перерабатываемое знание. При параллельной переработке знания картина меняется, ибо приходится говорить об одновременно перерабатываемых фрагментах этого знания. При этом процессы переработки указанных фрагментов могут взаимодействовать друг с другом в силу связности знания – при переработке одного фрагмента могут измениться также его внешние связи, которые непосредственно входят в состав другого одновременно обрабатываемого фрагмента.

Таким образом, процесс параллельной переработки знания есть взаимодействие (своебразная "интерференция") одновременно выполняемых локальных процессов, каждый из которых осуществляет переработку некоторого фрагмента знания. Причем взаимодействие этих локальных процессов осуществляется на основе анализа внешних связей каждого фрагмента. Следовательно, для организации параллельной переработки знания необходимы 1) явное представление всевозможных связей между потенциально обрабатываемыми фрагментами знания и 2) поддержка возможности изменения этих связей. Очевидно, что графодинамическая парадигма переработки знаний этим требованиям удовлетворяет.

Графодинамическая парадигма является также единственной возможной основой для развитых форм ассоциативного доступа к хранимой в памяти информации [269] ([Кохонен Т. 1980 кни-АссоцП](#)). Как известно, без развитых форм ассоциативного доступа к требуемым фрагментам перерабатываемой сложноструктурированной информации, имеющим произвольный размер, произвольную конфигурацию и произвольную "привязку" к остальной части хранимой информации, невозможна реализация наиболее перспективных моделей параллельной переработки сложноструктурированной информации – асинхронных моделей [208] ([Ершов А.П. ред. 1992 кни-АлгорМОиA](#)).

Резюме к подразделу 1.1

Завершая данный подраздел, отметим следующее.

1. Преимущества графодинамических моделей обработки информации по сравнению с другими видами моделей обработки информации (символьными моделями) заключаются в следующем:

- графодинамические модели легко поддерживают развитые формы ассоциативного доступа;

- графодинамические модели легко поддерживают параллельную реализацию операций;
 - графодинамические модели легко поддерживают асинхронную реализацию операций;
 - графодинамические модели проще интегрируются, так как интеграция начальных информационных конструкций интегрируемых графодинамических моделей может быть сведена к простому склеиванию некоторых элементов интегрируемых графовых конструкций при полном сохранении этих конструкций;
 - графодинамические модели легко интерпретируются друг другом, потому что благодаря ассоциативному доступу легко обеспечить независимость программ, описывающих методы реализации операций интерпретируемой модели, от перерабатываемых ими данных, т.е. от начальной информационной конструкции интерпретируемой модели;
 - в графодинамических моделях легко поддерживается работа со сложноструктурированной информацией любой степени сложности.
2. Преимущество графовых средств представления информации по сравнению с символьными средствами и преимущество графодинамических моделей обработки информации отмечается в целом ряде работ. "Всякий раз, когда с задачей удается связать граф, обсуждение резко упрощается и большие фрагменты словесного описания заменяются манипуляциями с картинками" [317] ([Манин Ю.И. 1979 книга-ДоказИН](#)).
3. В основе графодинамических моделей обработки информации лежит понятие графового языка, подробно рассматриваемое в подразделе [1.2](#), и понятие графодинамической абстрактной машины, подробно рассматриваемое в подразделе [1.3](#).
4. Наиболее перспективным классом графодинамических моделей для реализации интеллектуальных систем нового поколения являются графодинамические параллельные асинхронные модели. Параллелизм для графодинамических моделей вполне естественен, а возможность эффективной организации параллельной обработки является одним из главных достоинств графодинамических моделей. Асинхронность организации выполнения операций является естественным свойством моделей переработки знаний (не только графодинамических). Кроме того, без асинхронной организации невозможно обеспечить гибкость модели. Но только в рамках графодинамических моделей представляются широкие возможности для реализации асинхронной переработки знаний, благодаря поддержке развитых форм ассоциативного доступа.
5. Предметом исследования в данной работе являются не только сами модели обработки информации, но и соотношения между ними, а точнее, такие соотношения, которые лежат в основе проектирования (синтеза) новых моделей (в частности, путем интеграции нескольких моделей), а также в основе их практической реализации (в частности, путем интерпретации, т.е. путем сведения реализуемой модели к эквивалентной модели более простого вида).
6. В данной работе исследуются не только принципы (технологии) построения графодинамических моделей параллельной асинхронной переработки знаний, но и принципы практической реализации этих моделей. Суть такой реализации сводится к построению иерархии интерпретирующих графодинамических моделей параллельной асинхронной обработки информации, в рамках которой обеспечивается сведение графодинамических моделей параллельной асинхронной переработки знаний к графодинамическим моделям более простого вида с четкой технологией перехода от уровня к уровню.

1.2. Графовые языки

Ключевые понятия: графовый язык, графовая структура, реляционная структура.

В данном подразделе рассматриваются основные понятия, связанные с нетрадиционным классом языков – классом **графовых языков**, текстами которых являются **графовые структуры**.

Графовые языки составляют основу графодинамических формальных моделей обработки информации, исследуемых в данной работе и неформально рассмотренных в подразделе [1.1](#). Формальное рассмотрение графовых языков дает возможность уточнить понятие графодинамической формальной модели, являющейся центральным в данной работе.

1.2.1. Понятие реляционной структуры как уточнение понятия предметной области и понятия информационной конструкции

Ключевые понятия: реляционная структура, множество, кортеж, атрибут, универсум, отношение подчинения, множество реляционной структуры, кортеж реляционной структуры, атрибут реляционной структуры, первичный элемент реляционной структуры, отношение реляционной структуры, область определения отношения, проекция отношения, классическое отношение, классическая реляционная структура, гомоморфизм, изоморфизм, нестационарная реляционная структура.

Понятие **реляционной структуры** нами используется как способ формального уточнения предметных областей различного вида, т.е. как способ задания их структуры. Нас будут интересовать сложноструктурированные предметные области и соответствующие им сложноструктурированные реляционные структуры.

На предметные области не накладывается никаких ограничений. В частности, предметной областью может быть информационный объект, являющийся описанием какой-то другой предметной области. Информационному объекту можно поставить в соответствие две реляционные структуры. Первая из них определяет внутреннюю структуру (внутреннее строение, синтаксис) информационного объекта. Такую реляционную структуру будем называть информационной конструкцией. Вторая структура задает семантику информационного объекта, т.е. его соотношение с описываемой им предметной областью. Вторая из указанных реляционных структур является метаконструкцией, поскольку с формальной точки зрения является описанием определенного соотношения между двумя реляционными структурами – между информационной конструкцией, определяющей структуру соответствующего информационного объекта, и реляционной структурой, определяющей структуру предметной области, описываемой указанным информационным объектом. Каждый язык (в том числе и каждый графовый язык) формально задается некоторым множеством информационных конструкций, имеющих общие синтаксические и семантические свойства. Введение понятия реляционной структуры позволяет с единых позиций рассматривать различные языки, сравнивать их друг с другом и классифицировать.

Для уточнения понятий сложноструктурированной предметной области и сложноструктурированной информационной структуры вводится понятие реляционной структуры. Это понятие является обобщением классического понятия алгебраической модели [316] ([Мальцев А.И. 1970гн-АлгебС](#)), а также таких понятий, как клубная система [65] ([Борщев В.Б. 1983ст-СхемыНКС](#)), гиперсеть [385] ([Попков В.К. 1986ст-ГиперИиХС](#)), П-граф [213] ([Ефимова С.М. 1983ст-о ОднойФМ](#)).

Реляционные структуры позволяют работать со структурами существенно более сложного вида, чем алгебраические модели, – иерархическими, многоуровневыми структурами.

В основе понятия реляционной структуры лежат понятия **множества, кортежа и атрибута**. Необходимость глубокого семантического анализа реляционных структур заставляет:

- 1) четко противопоставлять понятие знака объекта и понятие самого объекта, обозначаемого этим знаком. Под объектами могут пониматься какие-то отдельные предметы описываемой предметной области, множества, кортежи, целые реляционные структуры;
- 2) ввести специальное бинарное асимметричное отношение принадлежности, связывающее знаки множеств с элементами множеств, обозначаемых этими знаками.

Множество задается знаком (обозначением, именем) множества, перечнем (набором) его элементов, семейством пар отношения принадлежности, связывающих знак множества с каждым элементом этого множества. Элементами множества могут быть объекты любой природы. В зависимости от количества элементов множества бывают пустыми, конечными (1-элементными, 2-элементными и т.д.), бесконечными.

Отношение принадлежности не является однозначным. Одному множеству может принадлежать несколько элементов. Элемент, принадлежащий одному множеству, может также принадлежать и другому множеству. В указанное семейство пар отношения принадлежности одна и та же пара может входить несколько раз, т.е. элементы множества могут входить в его состав многократно. Поэтому отношение принадлежности не может быть отнесено к числу классических бинарных отношений.

Следует четко отличать понятие самого множества (как математической конструкции) от понятия знака множества. Знак одного множества может быть элементом другого множества. Кроме того, знак множества может быть одним из элементов того множества, которое он сам обозначает.

Кортеж задается знаком кортежа, перечнем его элементов (компонентов), семейством пар отношения принадлежности, связывающих знак кортежа с каждым элементом этого кортежа, семейством множеств, элементами которых являются указанные выше пары отношения принадлежности. Такие множества будем называть атрибутами кортежей.

Атрибут $a_{_}$ кортежа t задается знаком атрибута $a_{_}$, перечнем его элементов, каждый из которых является парой отношения принадлежности, связывающего знак кортежа t с его элементами, имеющими атрибут $a_{_}$, семейством пар отношения принадлежности, связывающих знак атрибута $a_{_}$ с каждым его элементом.

Элементами кортежей могут быть объекты любой природы. В общем случае кортежу принадлежит несколько элементов. Элемент, принадлежащий одному кортежу, может принадлежать другому кортежу. Элементы кортежа могут входить в его состав многократно. Знак одного кортежа может быть элементом другого кортежа. Более того, знак кортежа может быть одним из элементов того кортежа, который он сам обозначает.

Каждый атрибут определяет роль соответствующих элементов кортежа в рамках всего кортежа. Каждая пара отношения принадлежности, связывающая знак кортежа с каким-либо его элементом, может либо принадлежать одному атрибуту, либо принадлежать сразу нескольким атрибутам, либо не принадлежать ни одному атрибуту. Элемент кортежа может одновременно являться атрибутом этого же кортежа. Атрибут кортежа может одновременно являться элементом этого же кортежа.

Таким образом, кортеж – это множество, каждому элементу которого ставится в соответствие атрибут этого элемента(в частности, номер), под которым он принадлежит данному кортежу. Атрибут элемента кортежа есть свойство относительное, ибо при интеграции кортежей в единую реляционную структуру элементы одного кортежа могут быть элементами других кортежей и иметь при этом другие атрибуты. Следует отметить, что в предлагаемой трактовке кортежа множество его атрибутов является произвольным множеством, в то время как для классического определения кортежа атрибутами являются номера элементов, которые условно будем обозначать символами $1_{_}$, $2_{_}$, $3_{_}$ и т.д.

Кортеж, имеющий элементы c_1 , c_2 , ..., c_n , атрибутами которых в рамках данного кортежа соответственно являются $a_{1_}$, $a_{2_}$, ..., $a_{n_}$, в символьной записи будем представлять следующим образом: ($a_{1_} : c_1$, $a_{2_} : c_2$, ..., $a_{n_} : c_n$). При этом кортежи классического вида, атрибутами которых являются номера их компонентов, т.е. кортежи ($1_{_} : c_1$, $2_{_} : c_2$, ..., $n_{_} : c_n$), будем представлять также и общепринятым образом: (c_1 , c_2 , ..., c_n).

Утверждение 1.2.1.1. Любой кортеж можно представить в виде классического кортежа, т.е. кортежа, атрибутами которого являются номера его элементов.

Для этого нужно построить взаимно однозначное соответствие между множеством атрибутов неклассического кортежа и отрезком натурального ряда чисел от 1 до n , т.е. пронумеровать атрибуты.

Кортежи друг от друга могут отличаться:

- 1) набором атрибутов (классические кортежи, неклассические кортежи с числовыми атрибутами, неклассические кортежи с нечисловыми атрибутами);
- 2) количеством элементов (унарные, бинарные, тернарные кортежи и т.д.);
- 3) наличием или отсутствием элементов, имеющих одинаковые атрибуты;
- 4) наличием или отсутствием многократного вхождения каких-либо элементов под разными или одинаковыми атрибутами;
- 5) наличием или отсутствием элементов кортежа, являющихся одновременно атрибутами этого же кортежа;
- 6) тем, является ли знак кортежа элементом обозначаемого им кортежа.

Определение 1.2.1.1. Универсум H , построенный на множестве P с атрибутами A , определяется рекурсивно на основе следующих утверждений:

- 1) $P \subset H$. Элементы множества P будем называть первичными (терминальными) элементами универсума H . Само множество P будем называть базовым множеством универсума H ;
- 2) $A \subset H; A \cap P = \emptyset$. Множество A формально трактуется как множество знаков атрибутов;
- 3) если hi есть подмножество универсума H ($hi \subset H$), то знак этого подмножества также является элементом универсума H ;
- 4) если hi есть произвольный элемент универсума H ($hi \in H$), а ai_+ есть произвольный элемент множества A ($ai_+ \in A$), то знак одноэлементного (унарного) кортежа ($ai_+ : hi$) также является элементом универсума H ;
- 5) если hi есть произвольный элемент универсума H , ai_+ есть произвольный элемент множества A , hj есть элемент универсума, являющийся знаком кортежа, то элементом универсума H также является знак кортежа, полученного в результате присоединения к кортежу hj нового элемента hi с атрибутом ai_+ ;
- 6) никаких других элементов множество H не содержит.

Таким образом, в состав универсума H входят следующие элементы:

- первичные (терминальные) элементы;
- знаки атрибутов, которые следует отличать от самих атрибутов;
- вторичные (производные) элементы, являющиеся знаками множеств, состоящих только из первичных элементов универсума, – такие множества будем называть простыми;
- вторичные элементы, являющиеся знаками кортежей, состоящих только из первичных элементов универсума, – такие кортежи будем называть простыми;
- вторичные элементы, являющиеся знаками множеств, в состав которых входит хотя бы один вторичный элемент (знак множества или знак кортежа), – такие множества будем называть метамножествами;
- вторичные элементы, являющиеся знаками кортежей, в состав которых входит хотя бы один вторичный элемент, – такие кортежи будем называть метакортежами.

Завершая рассмотрение понятия универсума, отметим то, что это понятие можно считать обобщением известного понятия шкалы множеств [77] (*Бурбаки Н. 1965кн-ТеориM*).

Определение 1.2.1.2. Пусть hi , hj – произвольные элементы универсума. Будем говорить, что hj принадлежит hi в том, и только в том случае, если либо hj есть элемент множества, знаком которого является hi , либо hj есть элемент кортежа, знаком которого является hi .

Из приведенного определения следует, что каждый элемент универсума, которому принадлежит хотя бы один элемент этого универсума, является вторичным элементом, т.е. является либо знаком множества, либо знаком кортежа.

Транзитивное замыкание отношения принадлежности назовем отношением **подчинения**, которое, соответственно, определяется рекурсивно на основании следующих утверждений:

- 1) если hj принадлежит вторичному элементу hi , то hj подчинен элементу hi ;
- 2) если hj подчинен элементу hi , а hk подчинен элементу hj , то hk подчинен элементу hi .

Теперь перейдем к определению понятия реляционной структуры.

Определение 1.2.1.3. Реляционная структура G задается кортежем:

(P, A, K, R, D) ,

где

- P – множество первичных элементов реляционной структуры G (которые в рамках этой структуры отмечаются атрибутом $primaryEl_$);
- A – множество знаков атрибутов реляционной структуры G (которые в рамках этой структуры отмечаются атрибутом $attr_$);
- K – множество знаков связок реляционной структуры G (которые в рамках этой структуры отмечаются атрибутом $conn_$);
- R – множество знаков отношений реляционной структуры G (которые в рамках этой структуры отмечаются атрибутом $rel_$). Знаки связок и знаки отношений реляционной структуры будем называть вторичными элементами этой структуры;
- D – множество элементов неопределенного типа реляционной структуры G (которые в рамках этой структуры не имеют атрибутов).

При этом должны выполняться следующие условия:

- множества P, A, K, R, D попарно не пересекаются;
- вторичные элементы реляционной структуры G представляют собой знаки множеств или кортежей и являются вторичными элементами универсума H , построенного на множестве P с атрибутами A , т.е. $(K \cup R) \subset H$. Это условие назовем свойством иерархичности реляционных структур, которое заключается в том, что элементами вторичных элементов реляционной структуры являются элементы (в том числе и вторичные элементы) этой же реляционной структуры;
- каждый элемент реляционной структуры G , не являющийся знаком ее отношения, должен быть подчинен знаку хотя бы одного отношения этой реляционной структуры. Понятие отношения подчинения, заданного на элементах реляционной структуры, аналогично понятию отношения подчинения, заданному на элементах универсума (см. определение 1.2.1.1). Это свойство назовем свойством целостности реляционных структур.

Множество первичных элементов реляционной структуры также будем называть базовым множеством этой реляционной структуры.

Множество K знаков связок реляционной структуры разбивается на два подмножества:

$K = S \cup T ; S \cap T = \emptyset$,

где

- S – множество знаков неупорядоченных связок реляционной структуры G , которым в рамках этой конструкции приписывается атрибут $connSet_$;
- T – множество знаков упорядоченных связок (кортежей) реляционной структуры G , которым в рамках этой структуры приписывается атрибут $connTuple_$.

Вторичные элементы, знаки которых принадлежат множеству $S \cup R$, будем называть множествами реляционной структуры G .

Множество реляционной структуры – это множество, которое целиком, как математическая структура, входит в состав (является фрагментом) этой реляционной структуры. Это означает, что в состав указанной реляционной структуры входят знак указанного множества, все элементы этого множества, все пары отношения принадлежности, связывающие знак множества с его элементами.

Кортеж реляционной структуры – это кортеж, который целиком, как математическая структура, входит в состав (является фрагментом) этой реляционной структуры вместе со всеми его элементами, со знаками всех используемых им атрибутов и со всеми соответствующими ему парами отношения принадлежности.

Атрибут реляционной структуры – это атрибут, используемый по крайней мере одним кортежем, входящим в состав указанной реляционной структуры.

Первичный элемент реляционной структуры – это такой элемент, который не является ни знаком множества, целиком входящего в состав указанной реляционной структуры, ни знаком кортежа, целиком входящего в состав этой структуры, ни знаком атрибута такого кортежа. Знаки множеств, знаки

кортежей и знаки атрибутов могут быть первичными элементами реляционной структуры, но в этом случае соответствующие множества и кортежи как математические структуры не считаются фрагментами указанной реляционной структуры. Даже если в число первичных элементов реляционной структуры входит знак некоторого множества, а также все элементы этого множества, то считается, что пары отношения принадлежности, связывающие знаки атрибутов с его элементами, в состав указанной реляционной структуры не входят.

Отношение реляционной структуры – это специально выделенное (чаще всего бесконечное) множество реляционной структуры.

Следует подчеркнуть, что пары отношения принадлежности, связывающие знаки атрибутов реляционной структуры и вторичные элементы реляционной структуры с элементами обозначаемых ими множеств или кортежей, входят в состав реляционной структуры, хотя формально ее элементами не являются.

Определение 1.2.1.4. Пусть r – одно из отношений некоторой реляционной структуры и пусть элементами этого отношения являются вторичные элементы указанной реляционной структуры. Тогда множество m будем называть **областью определения отношения** r в том и только в том случае, если выполняются следующие условия:

- 1) каждый элемент кортежа, принадлежащего отношению r , является элементом множества m ;
- 2) каждый элемент множества, принадлежащего отношению r , является элементом множества m ;
- 3) каждый элемент множества m является либо элементом некоторого кортежа, принадлежащего отношению r , либо элементом некоторого множества, принадлежащего отношению r .

В область определения отношения могут входить не только первичные элементы и элементы неопределенного типа, но и вторичные элементы реляционной структуры, если среди кортежей или множеств, принадлежащих отношению r , имеются метакортежи или метамножества.

Определение 1.2.1.5. Множество I будем называть **проекцией отношения** r по атрибуту a в том и только в том случае, если:

- 1) каждый элемент с атрибутом a кортежа, принадлежащего отношению r , является элементом множества I ;
- 2) каждый элемент множества I является элементом с атрибутом a кортежа, принадлежащего отношению r .

Отношения реляционной структуры бывают классическими (простыми) и неклассическими.

Классическое отношение r представляет собой подмножество декартова произведения $P \times P \times \dots \times P$, где P – множество первичных элементов, т.е. $r \subseteq P \times P \times \dots \times P$.

Если количество сомножителей равно 1, т.е. $r \subseteq P$, то классическое отношение называется унарным. Если количество сомножителей равно 2, т.е. $r \subseteq P \times P$, то классическое отношение называется бинарным. Если количество сомножителей равно 3, т.е. $r \subseteq P \times P \times P$, то классическое отношение называется тернарным и т.д.

Следовательно, классическое отношение – это либо некоторое множество первичных элементов ($r \subseteq P$), либо некоторое множество простых (т.е. состоящих из первичных элементов) кортежей, состоящих при этом из одинакового количества элементов, которым ставятся в соответствие одинаковые наборы атрибутов (натуральные числа от I до n , где n – количество элементов кортежа).

Неклассические отношения реляционной структуры отличаются от классических следующим:

- элементами неклассического отношения могут быть элементы реляционной структуры разного типа (в частности, неклассическому отношению могут одновременно принадлежать как первичные элементы, так и знаки кортежей с разным количеством элементов, знаки множеств с разным количеством элементов, в том числе знаки отношений, а также знаки атрибутов);
- элементами неклассического отношения могут быть не только знаки простых кортежей, элементами которых являются первичные элементы реляционной структуры, но и знаки метакортежей, среди

элементов которых имеется хотя бы один вторичный элемент реляционной структуры. Кроме того, элементами неклассических отношений могут быть знаки как простых множеств, так и метамножеств;

- элементами неклассического отношения могут быть не только знаки кортежей, в которых каждому элементу соответствует свой атрибут, но и знаки кортежей, в которых некоторые элементы имеют одинаковые атрибуты, не имеют атрибутов или имеют несколько атрибутов;
- элементами неклассического отношения могут быть не только знаки классических кортежей, но и знаки кортежей, для которых набор атрибутов представляет собой произвольный набор натуральных чисел, а также знаки кортежей, в которых в качестве атрибутов используются не только натуральные числа.

Отношения, в область определения которых могут входить знаки отношений, а элементами могут быть знаки метакортежей и метамножеств, будем называть метаотношениями.

Реляционные структуры могут быть самыми разными в зависимости от того, какие отношения входят в их состав и как эти отношения связаны друг с другом. Важными характеристиками здесь являются арность, симметричность, однозначность и область определения отношений (так, например, одно отношение реляционной структуры может полностью или частично входить в область определения другого отношения).

Реляционную структуру, все отношения которой являются классическими, будем называть **классической реляционной структурой**. Такую структуру обычно называют алгебраической моделью или реляционной системой [316] ([Мальцев А.И. 1970кн-Алгебр](#)).

Переход от классических реляционных структур к неклассическим (сложноструктурированным) реляционным структурам – это переход к иерархическим, многоуровневым структурам, в которых имеют место связи не только между первичными элементами, но и между связями, а также между целыми конструкциями. Это обстоятельство является очень важным при представлении знаний в интеллектуальных системах, ибо в них часто приходится иметь дело со сложноструктурированными, иерархическими предметными областями.

Особо отметим то, что знак кортежа, представляющего некоторую реляционную структуру, может входить в число вторичных элементов другой реляционной структуры, которую будем называть реляционной метаконструкцией. Более того, в число отношений реляционной метаконструкции может входить отношение "быть знаком реляционной структуры" (т.е. быть знаком кортежа, представляющего реляционную структуру). Таким образом, наша трактовка реляционной структуры позволяет, не выходя за рамки реляционной структуры, описывать сами реляционные структуры и связи между ними.

Примерами реляционных метаконструкций являются структуры, описывающие всевозможные морфизмы между реляционными структурами. Каждый такой морфизм – это соответствие между множествами элементов двух реляционных структур, удовлетворяющее тем или иным свойствам. Важнейшими примерами морфизмов являются гомоморфизмы и изоморфизмы.

Определение 1.2.1.6. Пусть G_1 – реляционная структура, представляющая собой кортеж, множество элементов которого в соответствии с их атрибутами разбивается на семейство подмножеств $(P_1, A_1, K_1, R_1, D_1)$, а G_2 – реляционная структура, представляющая собой кортеж, множество элементов которого разбивается на семейство подмножеств $(P_2, A_2, K_2, R_2, D_2)$ (см. определение 1.2.1.3). Соответствие между множеством элементов реляционной структуры G_1 и множеством элементов реляционной структуры G_2 будем называть **гомоморфизмом** в том и только в том случае, если оно удовлетворяет следующим условиям:

- 1) если $x \in P_1$, то $x^* \in P_2 \cup D_2$;
- 2) если $a \in A_1$, то $a^* \in A_2$;
- 3) если $k \in K_1$, то $k^* \in K_2$;
- 4) если $r \in R_1$, то $r^* \in R_2$;
- 5) если $d \in D_1$, то $d^* \in P_2 \cup A_2 \cup K_2 \cup R_2 \cup D_2$;
- 6) если $k \in r \in R_1$, то $k^* \in r^* \in R_2$;
- 7) если $x \in k \in K_1$, то $x^* \in k^* \in K_2$;
- 8) если $k = (\dots, a : x, \dots)$; $k \in K_1$, то $k^* = (\dots, a^* : x^*, \dots); k^* \in K_2$.

Здесь x^* , a^* , k^* , r^* , d^* есть образы элементов x , a , k , r , d в рамках рассматриваемого соответствия.

Определение 1.2.1.7. Реляционная структура $G1$ гомоморфна реляционной структуре $G2$ в том и только в том случае, если существует гомоморфизм между ними.

Определение 1.2.1.8. Соответствие между множеством элементов реляционной структуры $G1$ и множеством элементов реляционной структуры $G2$ будем называть **изоморфизмом** в том и только в том случае, если оно является гомоморфизмом между $G1$ и $G2$, а также гомоморфизмом между $G2$ и $G1$.

Определение 1.2.1.9. Реляционные структуры $G1$ и $G2$ изоморфны в том и только в том случае, если существует изоморфизм между ними.

Наряду с понятием реляционной структуры введем понятие **нестационарной реляционной структуры**, т.е. реляционной структуры, изменяющейся во времени. В этом смысле обычные реляционные структуры можно назвать стационарными. Для нестационарных реляционных структур вводится понятие ситуативного отношения, т.е. отношения, которое в разных состояниях нестационарной реляционной структуры состоит из разных элементов (разных кортежей, разных множеств или разных первичных элементов).

Важнейшим приемом описания нестационарной реляционной структуры является ее представление в виде обычной (стационарной) реляционной метаконструкции, описывающей соотношение во времени между различными состояниями нестационарной реляционной структуры, каждое из которых описывается конкретной стационарной реляционной структурой. Состояние нестационарной реляционной структуры будем также называть ситуациями этой структуры. Таким образом, описание нестационарных реляционных структур также приводит к сложноструктурированным иерархическим реляционным структурам.

В стационарной реляционной метаконструкции, описывающей нестационарную реляционную структуру, применяется целый ряд метаотношений, заданных на множестве оболочек стационарных реляционных структур, описывающих различные состояния нестационарной структуры. К числу этих метаотношений относятся: метаотношение "Следовать сразу за"; метаотношение "Следовать за через некоторый промежуток времени"; метаотношение "Причина - следствие"; метаотношение "Следовать за с перекрытием во времени"; метаотношение "Происходить одновременно"; метаотношение, связывающее начальную и конечную ситуацию некоторого процесса, метаотношение, связывающее ситуацию с подситуацией, которая является частным случаем (этапом, стадией) для первой ситуации.

Важно подчеркнуть, что при создании сложных интеллектуальных систем обычно приходится иметь дело именно с неклассическими сложноструктурированными иерархическими реляционными структурами самого разного вида, и в частности с реляционными метаконструкциями.

1.2.2. Линейные тексты

Ключевые понятия: линейный текст, символьная конструкция.

Перейдем к рассмотрению информационных конструкций, которые представляют собой реляционные структуры, описывающие структуру информационных объектов, т.е. объектов, которые сами являются описаниями каких-либо предметных областей. Никаких ограничений на структуру информационных объектов не накладывается и, соответственно этому, не накладывается никаких ограничений и на структуру информационных конструкций. Единственное, что в информационных конструкциях невозможно, – это то, чтобы их первичными элементами были предметы описываемой предметной области. Первичными элементами информационных конструкций могут быть не сами предметы описываемой предметной области, а их знаки. Первичными элементами информационных конструкций могут быть также символы, из которых строятся знаки предметов описываемой предметной области.

В данном пункте рассматривается один из видов информационных конструкций – **линейные тексты** (символьные информационные конструкции), называемые также символьными конструкциями, линей-

ными информационными конструкциями, цепочечными информационными конструкциями, цепочками символов, строками символов, которые являются традиционным видом информационных конструкций.

Определение 1.2.2.1. Пусть Gs – информационная конструкция, задаваемая кортежем, множество элементов которого в соответствии с их атрибутами разбивается на семейство подмножеств (Ps, As, Ks, Rs, Ds) (см. определение 1.2.1.3). Информационную конструкцию Gs будем называть **символьной конструкцией** в том и только в том случае, если выполняются следующие условия:

- 1) $As = \{1_-, 2_-\}$;
- 2) если $k \in Ks$, то $k = (1_- : pi, 2_- : pj) ; pi, pj \in Ps$;
- 3) $Rs = Rp \cup \{Ks\}$, где Rp есть семейство унарных отношений, заданных на множестве Ps ;
- 4) не существует $pi, pj, pe \in Ps$ таких, что $(1_- : pi, 2_- : pj) \in Ks$,
 $(1_- : pi, 2_- : pe) \in Ks$;
- 5) не существует $pi, pj, pe \in Ps$ таких, что $(1_- : pi, 2_- : pj) \in Ks$,
 $(1_- : pe, 2_- : pj) \in Ks$;
- 6) $Ds = \emptyset$.

Первичные элементы символической конструкции (т.е. элементы множества Ps) будем называть **символами**. Каждый бинарный асимметричный кортеж, принадлежащий множеству Ks , есть связь непосредственного соседства символов в строке. При этом атрибут “ 1_- ” указывает на предшествующий символ, а атрибут “ 2_- ” указывает на последующий символ. Унарные отношения, входящие в состав Rp , будем называть **типами символов**, а все семейство унарных отношений Rp будем называть **алфавитом символов**.

На множестве символьных конструкций задается целый ряд отношений – отношение равенства, отношение включения, отношение конкатенации.

Символьные конструкции равны, если 1) они изоморфны, 2) у них совпадают алфавиты символов, 3) типы символов в рамках указанного изоморфизма соответствуют сами себе. Очевидно, что множество всевозможных символьных конструкций может быть разбито на классы равных символьных конструкций.

Символьная конструкция, определяемая семейством множеств (Ps, As, Ks, Rs, Ds) , связная, если симметризация и транзитивное замыкание отношения $\{Ks\}$ приводят к декартову произведению $Ps \times Ps$.

Символьная конструкция, определяемая семейством множеств (Ps, As, Ks, Rs, Ds) , является результатом конкатенации связных символьных конструкций, определяемых семейством множеств $(Psi, Asi, Ksi, Rsi, Dsi)$ и семейством множеств $(Psj, Asj, Ksj, Rsj, Dsj)$, если

- 1) $Ps = Psi \cup Psj$;
- 2) $Ks = Ksi \cup Ksj \cup \{k\}$;
- 3) $Rs = Rsi \cup Rsj$;
- 4) $k = (1_- : i, 2_- : j)$, где
 i – самый правый символ первой символьной конструкции;
 j – самый левый символ второй символьной конструкции.

Утверждение 1.2.2.1. Любую реляционную структуру можно представить в виде эквивалентного линейного текста.

Существует большое число способов (языков) представления (кодирования) произвольных реляционных структур в виде символьных конструкций. Определим один из таких языков, который условно назовем Lrs . Этот язык является прообразом формального языка SCBs, который подробно рассмотрен ниже в разделе 2.

Определение 1.2.2.2. Будем утверждать, что реляционная структура, задаваемая семейством множеств (P, A, K, R, D) , и символьная конструкция, принадлежащая языку Lrs и

определенная семейством множеств (P_s, A_s, K_s, R_s, D_s), эквивалентны тогда, и только тогда, когда существует взаимно однозначное соответствие между множеством $P \cup A \cup K \cup R \cup D$ (множеством всех элементов исходной реляционной структуры) и некоторым множеством S , каждый элемент которого представляет собой множество равных символьных конструкций, входящих в состав конструкции (P_s, A_s, K_s, R_s, D_s) и являющихся идентификаторами (именами) соответствующих элементов исходной (представляемой) реляционной структуры. При этом должны выполняться следующие условия:

- 1) если $y \in r, r \in R$, то в символьной конструкции присутствует строка вида $r^* \rightarrow y^*$, где $r^*, y^* \in S$, r^* и y^* – идентификаторы элементов y и r ;
- 2) если в символьной конструкции присутствует строка вида $r \rightarrow y^*$, где $r^*, y^* \in S$, то существуют элементы реляционной структуры $y \in r, r \in R$, для которых r^* и y^* являются идентификаторами;
- 3) если связка $k, k \in K$ является неупорядоченной и состоит из элементов y_1, y_2, \dots, y_n , т.е. $y_1, y_2, \dots, y_n \in k$, то в символьной конструкции присутствует строка вида $k^* \rightarrow y_1^*, y_2^*, \dots, y_n^*$, где $k^*, y_1^*, y_2^*, \dots, y_n^*$ – идентификаторы элементов k, y_1, y_2, \dots, y_n ;
- 4) если в символьной конструкции присутствует строка вида $k^* \rightarrow y_1^*, y_2^*, \dots, y_n^*$, где $k^*, y_1^*, y_2^*, \dots, y_n^* \in S$, то в реляционной структуре существует неупорядоченная связка $k, k \in K$, состоящая из элементов $y_1, y_2, \dots, y_n, y_1, y_2, \dots, y_n \in k$, для которых $y_1^*, y_2^*, \dots, y_n^*$ являются идентификаторами, k^* является идентификатором элемента k ;
- 5) если связка $k, k \in K$ является кортежем, состоящим из элементов y_1, y_2, \dots, y_n , которые, соответственно, имеют атрибуты a_1, a_2, \dots, a_n , то в символьной конструкции присутствует строка вида $k^* \rightarrow a_1^* : y_1^*, a_2^* : y_2^*, \dots, a_n^* : y_n^*$, где $k^*, a_1^*, y_1^*, a_2^*, y_2^*, \dots, a_n^*, y_n^*$ – идентификаторы элементов $k, a_1, y_1, a_2, y_2, \dots, a_n, y_n$;
- 6) если в символьной конструкции присутствует строка вида $k^* \rightarrow a_1^* : y_1^*, a_2^* : y_2^*, \dots, a_n^* : y_n^*$, где $a_1^*, y_1^*, a_2^*, y_2^*, \dots, a_n^*, y_n^* \in S$, то в реляционной структуре существует связка $k, k \in K$, являющаяся кортежем, состоящим из элементов y_1, y_2, \dots, y_n , которые, соответственно, имеют атрибуты a_1, a_2, \dots, a_n , причем $k^*, a_1^*, y_1^*, a_2^*, y_2^*, \dots, a_n^*, y_n^*$ являются идентификаторами элементов $k, a_1, y_1, a_2, y_2, \dots, a_n, y_n$.

Заметим, что в символьной конструкции, которая эквивалентна некоторой реляционной структуре, кроме идентификаторов элементов этой реляционной структуры присутствуют специальные вспомогательные строки символов, выполняющие разделительные и ограничительные функции. К таким специальным символьным конструкциям относятся: “ \rightarrow ”, “ $=$ ”, “ $($ ”, “ $)$ ”, “ $($ ”, “ $)$ ”, запятая “ $,$ ”, точка с запятой “ $;$ ”.

Рассмотренные символьные конструкции противопоставляются графовым структурам, к числу которых будем относить все информационные конструкции, не являющиеся символьными. При определении графовых структур не уточняется, что из себя представляют первичные элементы этих структур, как конкретно выглядят связи графовых структур. Уточнение всего этого приводит к самым различным вариантам представления и изображения графовой структуры. Так, например, можно говорить о матричных представлениях (матрицы смежности, матрицы инцидентности), о списковых, о графических (топологических) представлениях.

1.2.3. Нелинейные тексты

Ключевые понятия: бинарная информационная конструкция, отношение принадлежности, однородная информационная конструкция.

Если рассматривать информационные конструкции, являющиеся реляционными структурами общего вида, как способ представления информации в памяти систем обработки информации, то следует от-

метить недостаточный конструктивизм такого представления, ибо трудно построить соответствие между элементами реляционной структуры общего вида и элементами абстрактной или реальной памяти, в которой эта структура может храниться и преобразовываться. Для того чтобы иметь такое соответствие, нужно ограничить (а в идеале – зафиксировать) набор атрибутов и отношений информационной конструкции, не нарушая при этом семантической мощности. Переходим к частным видам информационных конструкций, удовлетворяющим данному требованию.

Определение 1.2.3.1. Информационную конструкцию Gg , описывающую реляционную структуру G и представляющую собой кортеж, множество элементов которого в соответствии с их атрибутами разбивается на семейство подмножеств (Pg, Ag, Kg, Rg, Dg), будем называть **бинарной информационной конструкцией**, если выполняются следующие условия:

- 1) элементы множества Pg взаимно однозначно соответствуют элементам описываемой реляционной структуры G . Каждый элемент множества Pg , т.е. каждый первичный элемент структуры Gg , считается семантически эквивалентным соответствующему элементу описываемой реляционной структуры G ;
- 2) множество Ag фиксировано: $Ag = \{1_, 2_-\}$;
- 3) связи реляционной структуры Gg представляют собой бинарные кортежи, имеющие вид $(1_ : k, 2_ : g)$, где $k \in Pg$, $g \in (Pg \cup Kg \cup Dg)$, при этом один и тот же кортеж во множество Kg может входить несколько раз (такие кортежи называются кратными). От множества Kg можно перейти ко множеству Kg^* , $Kg^* \subseteq Pg \times (Pg \cup Kg^* \cup Dg)$, оставив для каждой группы кратных кортежей по одному представителю;
- 4) $Rg = Rt \cup Ri$, где Rt – множество знаков унарных отношений (для каждого $r \in Rt$ имеет место $r \subseteq (Pg \cup Dg)$), Ri – множество знаков бинарных отношений (для каждого $r \in Ri$ имеет место $r \subseteq Kg$). Унарные отношения будем также называть метками первичных и неопределенных элементов бинарной информационной конструкции.

Если элемент описываемой реляционной структуры есть ее первичный элемент, представляющий собой некий предмет соответствующей предметной области, то семантическим эквивалентом этого элемента в бинарной информационной конструкции является знак указанного предмета. Если элемент описываемой реляционной структуры есть знак множества или кортежа (атрибуты и отношения считаются частными видами множеств), то семантическим эквивалентом этого элемента в бинарной информационной конструкции является знак семантически эквивалентного множества или кортежа, т.е. множества или кортежа, состоящего из тех, и только тех элементов бинарной информационной конструкции, которые семантически эквивалентны элементам указанного выше множества или кортежа, входящего в состав описываемой реляционной структуры.

Утверждение 1.2.3.1. Каждую реляционную структуру можно представить в виде эквивалентной бинарной информационной конструкции.

Доказательство этого утверждения сводится к построению алгоритма, обеспечивающего для произвольной реляционной структуры построение эквивалентной бинарной информационной конструкции, а также к построению обратного алгоритма, обеспечивающего для произвольной бинарной информационной конструкции построение эквивалентной реляционной структуры общего вида.

Дадим определение эквивалентности реляционных структур общего вида и бинарных информационных конструкций, на основании которого указанные алгоритмы легко могут быть получены.

Определение 1.2.3.2. Пусть G – реляционная структура, представляющая собой кортеж, определяемый семейством множеств (P, A, K, R, D), а Gg – бинарная информационная конструкция, представляющая собой кортеж, определяемый семейством множеств (Pg, Ag, Kg, Rg, Dg), удовлетворяющих следующим условиям:

- 1) $Pg = Pv \cup Pk \cup Pd$, $Pv \cap Pk = \emptyset$, $Pv \cap Pd = \emptyset$, $Pk \cap Pd = \emptyset$;
- 2) $Ag = \{1_, 2_-\}$ (по определению бинарной конструкции);
- 3) если $y \in Kg$, то $y = (1_ : k, 2_ : g)$, $k \in Pk$, $g \in Pg$;
- 4) $Rg = Rt \cup Ri$, $Rt \cap Ri = \emptyset$,
 $Rt = \{Pv, Pk\} \cup Rtr$,
 $Ri = \{p\} \cup Ria$,
где p – отношение принадлежности элемента множеству;

- 5) если $y \in r$, $r \in Rt$, то $y \in Pg$;
- 6) если $y \in r$, $r \in Ri$, то $y \in Kg$;
- 7) $Dg = \emptyset$.

Будем говорить, что реляционная структура G и бинарная информационная конструкция Gg **эквивалентны** тогда и только тогда, когда существуют взаимно однозначные соответствия между P и Pv ($Pv \subseteq Pg$), между A и Ria ($Ria \subseteq Rg$), между K и Pk ($Pk \subseteq Pg$), между R и Rtr ($Rtr \subseteq Rg$), между D и Pd ($Pd \subseteq Pg$) и при этом выполняются следующие условия:

- 1) если $x \in r$, $r \in R$, то $x^* \in r^*$, $r^* \in Rtr$ и наоборот. Здесь x^* , r^* – образы объектов x , r в рамках указанных соответствий;
- 2) если в рамках реляционной структуры имеет место $x \in k$, $k \in K$, где k – вторичный элемент реляционной структуры, являющийся множеством, то $k^* \in Pk$, $x^* \in Pg$, $(1_1 : k^*, 2_1 : x^*) \in p$, $(1_1 : k^*, 2_1 : x^*) \in Kg$ и наоборот;
- 3) если в рамках реляционной структуры имеет место $k \in K$, $a \in A$, $k = (\dots, a : x, \dots)$, то $k^* \in Pk$, $a^* \in Ria$, $x^* \in Pg$, $(1_1 : k^*, 2_1 : x^*) \in a^*$, $(1_1 : k^*, 2_1 : x^*) \in Kg$ и наоборот.

Здесь x^* , a^* , k^* – образы объектов x , a , k в рамках указанных соответствий.

Элементы множества Pg будем называть узлами бинарной информационной конструкции, а элементы множества Kg – дугами бинарной информационной конструкции. Элементы множества Pv – знаки первичных элементов исходной реляционной структуры, элементы множества Pd – знаки элементов неопределенного типа исходной реляционной структуры, элементы множества Pk – знаки связок (упорядоченных или неупорядоченных) исходной реляционной структуры. Подчеркнем, что здесь имеется в виду не связка бинарной конструкции, а связка эквивалентной реляционной структуры общего вида, которая сама в бинарную конструкцию не входит, а входит знак (имя) этой связки.

Как было отмечено, отношение p трактуется как **отношение принадлежности** элемента множеству. Каждое из остальных отношений, входящих в семейство Ri , есть отношение принадлежности элемента кортежу под определенным атрибутом (каждому такому отношению соответствует свой атрибут принадлежности). Итак, переход от реляционной структуры общего вида к эквивалентной бинарной конструкции суть:

- 1) замена первичных элементов и элементов неопределенного типа на их знаки;
- 2) сведение произвольного набора атрибутов к двум атрибутам;
- 3) сведение вторичных элементов произвольного вида к бинарным кортежам;
- 4) сведение произвольного набора отношений к унарным и бинарным отношениям.

В процессе такого перехода атрибуты исходной реляционной структуры "превращаются" в соответствующие отношения принадлежности, вторичные элементы – в первичные, отношения исходной реляционной структуры – в унарные отношения, элементы неопределенного типа – в первичные.

Нетрудно заметить, что рассмотренный переход от реляционных структур общего вида к эквивалентным бинарным конструкциям есть своего рода декомпозиция реляционных структур, в результате чего неявно заданное в реляционной структуре общего вида отношение принадлежности, скрытое внутри вторичных элементов реляционной структуры (знаков множеств и знаков кортежей), приобретает в рамках бинарной конструкции явное очертание. Более того, отношение принадлежности со всеми его модификациями может интерпретироваться как топологическое присоединение (того или иного вида) одного элемента бинарной конструкции к другому. Это обстоятельство является очень важным, так как позволяет говорить не только об алгебраических и теоретико-множественных свойствах бинарных конструкций, но и об их топологических свойствах. Кроме того, простая топологическая интерпретация бинарных конструкций, в частности простая возможность их графического изображения, позволяет говорить о переработке бинарных конструкций более конструктивно, чем о переработке реляционных структур общего вида.

Так, например, первичным элементам бинарной информационной конструкции можно поставить в соответствие элементы памяти, в которой эта конструкция хранится и перерабатывается. Тогда набор унарных отношений, которым принадлежит первичный элемент (т.е. набор его меток), есть не что иное, как текущее состояние соответствующего ему элемента памяти. А набор дуг (пар) принадлежности, ко-

торыми первичный элемент связан с другими элементами, есть не что иное, как система связей, которыми соответствующий элемент памяти связан с другими элементами памяти в текущий момент времени. При этом в процессе переработки бинарной информационной конструкции в самом общем случае может происходить как изменение состояния элементов памяти, так и изменение связей между ними.

Переход от реляционных структур общего вида к эквивалентным бинарным информационным конструкциям можно считать одним из способов упрощения (канонизации) реляционных структур. Дальнейший анализ бинарных информационных конструкций на предмет их упрощения приводит к понятию однородных информационных конструкций – реляционных структур, не имеющих меток и имеющих единственное отношение (бинарное асимметричное отношение принадлежности).

Определение 1.2.3.3. Информационную конструкцию Gg , описывающую реляционную структуру G и представляющую собой кортеж, множество элементов которого в соответствии с их атрибутами разбивается на семейство подмножеств (Pg, Ag, Kg, Rg, Dg), будем называть **однородной информационной конструкцией**, если выполняются следующие условия:

- 1) элементы множества Pg взаимно однозначно соответствуют элементам описываемой реляционной структуры G . Каждый элемент множества Pg , т.е. каждый первичный элемент конструкции Gg , считается семантически эквивалентным соответствующему элементу описываемой реляционной структуры G ;
- 2) множество Ag фиксировано: $Ag = \{1_-, 2_-\}$;
- 3) связи реляционной структуры Gg представляют собой либо простые бинарные кортежи принадлежности, имеющие вид $(1_- : k, 2_- : q)$, где $k \in Pk$, $q \in Pg$, ($Pg = Pv \cup Pk \cup Pd$), либо бинарные метакортежи принадлежности, имеющие вид $(1_- : k, 2_- : c)$, где $k \in Pk$, $c \in Kg$. При этом кортеж c может быть как простым кортежем, так и метакортежем. Связи однородной конструкции будем называть дугами этой конструкции (соответственно простыми дугами и метадугами);
- 4) $Rg = \{Kg\}$. Семейство отношений реляционной структуры Gg включает в себя единственное отношение – отношение принадлежности, представляющее собой множество знаков всех бинарных асимметричных кортежей принадлежности.

Утверждение 1.2.3.2. Каждую бинарную информационную конструкцию можно представить в виде эквивалентной однородной информационной конструкции.

Иными словами, любую бинарную информационную конструкцию можно свести к реляционной структуре с единственным отношением – бинарным асимметричным отношением принадлежности.

Это утверждение легко доказывается на основании следующего определения эквивалентности бинарной информационной конструкции и однородной информационной конструкции.

Определение 1.2.3.4. Пусть Gg – бинарная информационная конструкция, представляющая собой кортеж, определяемый семейством множеств (Pg, Ag, Kg, Rg, Dg), удовлетворяющих условиям:

- 1) $Rg = Rt \cup Ri$, $Rt \cap Ri = \emptyset$, $Ri = \{P\} \cup Ria$.
Здесь P – отношение принадлежности элемента множеству, обозначаемому некоторым узлом в рамках исходной бинарной информационной конструкции;
- 2) $Dg = \emptyset$,
а Gq – однородная информационная конструкция, представляющая собой кортеж, определяемый семейством множеств (Pq, Aq, Kq, Rq, Dq), удовлетворяющих условиям:
 - 1) $Pq = Pqv \cup Pqk$, $Pqv \cap Pqk = \emptyset$,
 $Pqk = Pqrt \cup Pqri$, $Pqrt \cap Pqri = \emptyset$.
Здесь Pqv – множество предметных узлов однородной конструкции, Pqk – множество знаков множеств и знаков кортежей;
 - 2) $Kq = Kqg \cup Kqrt \cup Kqri$, $Kqg \cap Kqrt = \emptyset$,
 $Kqg \cap Kqri = \emptyset$, $Kqrt \cap Kqri = \emptyset$;
 - 3) $Dq = \emptyset$.

Будем говорить, что бинарная информационная конструкция Gg и однородная информационная конструкция Gq эквивалентны тогда и только тогда, когда существуют взаимно однозначные соот-

вествия между Pg и Pqv ($Pqv \subseteq Pg$), между Kg и Kqg ($Kqg \subseteq Kg$), между Rt и $Pqrt$ ($Rt \subseteq Rg$, $Pqrt \subseteq Pg$), между Ria и $Pqri$ ($Ria \subseteq Rg$, $Pqri \subseteq Pg$) и при этом выполняются следующие условия:

- 1) если $k \rightarrow 1_1 : x, 2_1 : y; k \in Kg$, то $k^* \rightarrow 1_2 : x^*, 2_2 : y^*; k^* \in Kqg$ и наоборот. Здесь k^* , x^* , y^* – образы объектов k , x , y в рамках указанных выше соответствий;
- 2) если $k \in Kg, k \in r, r \in Ria$, то $k^* \in Kqg, r^* \in Pqri$,
 $(1_1 : r^*, 2_1 : k^*) \in Kqri$ и наоборот. Здесь k^* , r^* – образы объектов k , r в рамках указанных соответствий;
- 3) если $t \in Pg, t \in r, r \in Rt$, то $t^* \in Pqv, r^* \in Pg$,
 $(1_1 : r^*, 2_1 : t^*) \in Pg$ и наоборот. Здесь t^* , r^* – образы объектов t , r .

Итак, в результате перехода от бинарной информационной конструкции произвольного вида к однородной информационной конструкции каждое унарное отношение исходной бинарной информационной конструкции "превращается" в первичный элемент, являющийся знаком соответствующего множества первичных элементов, а каждое бинарное отношение r исходной бинарной информационной конструкции, относящееся к классу Ria , "превращается" в первичный элемент, являющийся знаком соответствующего множества пар (дуг) принадлежности, каждая из которых связывает знак некоторого кортежа исходной реляционной структуры общего вида с некоторым элементом указанного кортежа, имеющим в рамках этого кортежа атрибут, соответствующий отношению r .

Следует подчеркнуть то, что все вторичные элементы (дуги) однородной информационной конструкции имеют одинаковую и четкую теоретико-множественную интерпретацию. Каждая такая дуга связывает знак некоторого множества (из которого дуга выходит) с одним из элементов этого множества.

Если от однородной информационной конструкции осуществить точно такой же переход, какой мы осуществляли от произвольной реляционной структуры к эквивалентной бинарной информационной конструкции, то получится информационная конструкция, которую назовем предельной. В предельной информационной конструкции дуги (пары) принадлежности (как простые дуги, так и метадуги) будут сведены к кортежам инцидентности, каждый из которых связывает знак дуги принадлежности либо со знаком некоторого множества, либо с первичным элементом, принадлежащим этому множеству. Соответственно этому отношение принадлежности однородной информационной конструкции (не являющееся простым отношением) сводится к двум (простым) бинарным отношениям инцидентности. Одно из этих отношений связывает знаки дуг принадлежности со знаками множеств (т.е. с элементами, из которых эти дуги выходят). А каждая пара инцидентности, принадлежащая другому из этих отношений, связывает знак дуги принадлежности с первичным элементом предельной информационной конструкции, в который эта дуга входит, т.е. с элементом множества, обозначаемого первичным элементом, из которого указанная дуга выходит.

В основе дальнейшего рассмотрения графодинамических моделей обработки информации используется однородные информационные конструкции в силу того, что они имеют очень простую базовую семантическую интерпретацию, носящую теоретико-множественный характер и непосредственно не зависящую от описываемой предметной области.

О перспективности рассмотренных в данном пункте бинарных и однородных информационных конструкций свидетельствует большой интерес к бинарным представлениям (бинарным моделям) баз данных [80; 539; 401] (Вагин В.Н.1989гн–ДедукИО ; Цаленко М.Ш.1989гн–МоделСвБД ; Поспелов Д.А.ред.1990спр–ИскусИ–К2).

Рассматривая соотношение произвольной описываемой реляционной структурой с некоторой эквивалентной символьной информационной конструкцией (см. определение 1.2.2.2), а также с эквивалентной бинарной информационной конструкцией (см. определение 1.2.3.2) и с эквивалентной однородной информационной конструкцией (см. определение 1.2.3.4), мы фактически определяем денотационную семантику произвольных информационных конструкций, относящихся 1) к одному из возможных символьных фактографических языков, 2) к графовому фактографическому языку бинарных информационных конструкций и 3) к графовому фактографическому языку однородных информационных конструкций.

Особенность перечисленных языков заключается в том, что, во-первых, все они являются фактографическими языками и, во-вторых, все они обеспечивают представление фактографической (экстен-

сиональной) информации о реляционных структурах любого вида. Графовый фактографический язык бинарных информационных конструкций и графовый фактографический язык однородных информационных конструкций относятся к классу графовых семантических языков, которые будут рассмотрены в пункте 1.2.6.

1.2.4. Денотационная семантика текстов

Ключевые понятия: денотационная семантика информационной конструкции, знак, Базовая семантическая информационная конструкция, семантическая сеть.

Каждый информационный объект является моделью (описанием) некоторой предметной области и, следовательно, имеет определенные связи с описываемой предметной областью. Внутренняя структура информационного объекта определяется соответствующей информационной конструкцией. Внутренняя структура предметной области определяется соответствующей реляционной структурой. Совокупность связей информационной конструкции с реляционной структурой, определяющей структуру описываемой предметной области, будем называть **денотационной семантикой информационной конструкции**.

Ключевым понятием денотационной семантики информационных конструкций является понятие **знака**. Знак – это минимальный семантически значимый фрагмент информационной конструкции. Знак может быть представлен либо элементом информационной конструкции, либо фрагментом информационной конструкции, состоящим из нескольких элементов. Знак в рамках информационной конструкции представляет (заменяет, изображает) нечто из описываемой предметной области. Этим нечто может быть либо конкретный предмет (объект) описываемой предметной области, либо конкретная связь, имеющая место в описываемой предметной области (это может быть связь между объектами, другими связями, понятиями, фрагментами предметной области), либо конкретный фрагмент описываемой предметной области, либо конкретное понятие этой предметной области. То, что обозначается знаком, называется его денотатом или денотационной семантикой этого знака. Множество знаков, входящих в состав информационной конструкции, однозначно соответствует множеству своих денотатов, т.е. каждому знаку не может соответствовать несколько денотатов. При этом указанное соответствие между множеством знаков, входящих в информационную конструкцию, и множеством их денотатов не обязательно быть взаимно однозначным. Таким образом, разные знаки могут иметь один и тот же денотат, т.е. обозначать одно и то же. Такие знаки будем называть синонимичными.

Рассмотрение денотационной семантики информационных конструкций различного вида основано на понятии базовой семантической информационной конструкции.

Определение 1.2.4.1. Базовая семантическая информационная конструкция – это такая информационная конструкция, которая изоморфна некоторому фрагменту реляционной структуры, определяющей структуру описываемой предметной области. Все элементы базовой семантической информационной конструкции являются знаками, семантически эквивалентными соответствующим элементам указанного представляемого (кодируемого) фрагмента реляционной структуры.

Информационная конструкция называется семантической информационной конструкцией или **семантической сетью**, если морфизм между этой конструкцией и семантически эквивалентной ей базовой семантической информационной конструкцией является взаимно однозначным соответствием между множествами элементов указанных конструкций:

- 1) каждый первичный элемент базовой семантической конструкции является знаком, обозначающим некоторый конкретный предмет описываемой предметной области и взаимно однозначно соответствующим этому предмету;
- 2) каждый знак связки базовой семантической конструкции обозначает некоторую конкретную связь, имеющую место в описываемой предметной области, и взаимно однозначно соответствует этой связи;
- 3) каждый знак атрибута базовой семантической конструкции взаимно однозначно соответствует некоторому относительному понятию (множеству однотипных ролей, выполняемых в рамках определенных связей соответствующими компонентами этих связей);
- 4) каждый знак отношения базовой семантической конструкции взаимно однозначно обозначает некоторое понятие описываемой предметной области (некоторое множество знаков аналогичных в том или ином смысле предметов или связей).

Конкретная связь, имеющая место в описываемой предметной области, не обязательно должна быть простой, т.е. не обязательно должна быть связью между предметами описываемой предметной облас-

ти. В число связей описываемой предметной области входят также некоторые фрагменты этой предметной области, состоящие из некоторого количества предметов, некоторого количества связей. К числу связей относятся также и ролевые структуры (фреймы).

Построение базовой семантической информационной конструкции, соответствующей описываемой предметной области, является важнейшим этапом формального описания предметной области, а также важнейшим этапом формирования базы знаний.

В качестве примера рассмотрим базовую семантическую информационную конструкцию, соответствующую геометрии Евклида. Первичными элементами этой конструкции являются знаки конкретных геометрических точек, конкретных прямых, конкретных плоскостей. Вторичными элементами этой конструкции являются знаки конкретных отрезков, треугольников, окружностей и т.д. Каждая такая геометрическая фигура трактуется как множество точек, удовлетворяющее определенным требованиям. Вторичными элементами рассматриваемой информационной конструкции являются также знаки связей инцидентности, конгруэнтности; связей сравнения по длине, по площади, по объему; связей, каждая из которых связывает тройку точек, одна из которых лежит между двумя другими, и т.д. Отношениям рассматриваемой информационной конструкции соответствуют такие абсолютные понятия, как "быть геометрической точкой", "быть прямой", "быть отрезком", "быть треугольником", "быть инцидентными геометрическими фигурами", "быть конгруэнтными геометрическими фигурами" и т.д. Атрибутами рассматриваемой информационной конструкции являются такие относительные понятия, как "быть более длинной линией", "быть менее длинной линией", "быть точкой, лежащей между" и т.д.

Таким образом, построить базовую семантическую информационную конструкцию описываемой предметной области – это, прежде всего, сформировать понятийный аппарат указанной предметной области, что требует глубокого анализа этой предметной области.

В основе нашего рассмотрения денотационной семантики информационных конструкций лежит следующее свойство базовой семантической информационной конструкции.

Утверждение 1.2.4.1. Любую фактографическую информацию (любые экстенсиональные знания) о любой предметной области можно представить в виде фрагмента базовой семантической информационной конструкции.

Если от базовой семантической информационной конструкции перейти к эквивалентным информационным конструкциям частного вида, то получатся различные виды семантических конструкций – семантическая бинарная конструкция, семантическая однородная конструкция. Введенные семантические конструкции различного вида можно считать различными вариантами уточнения понятия семантической сети.

Если рассматривать множество информационных конструкций, эквивалентных некоторой базовой семантической информационной конструкции, то последнюю можно считать формальным уточнением денотационной семантики указанных конструкций.

Важнейшим свойством семантических конструкций различного вида является их ассоциативность, т.е. наличие достаточно простой процедуры, позволяющей для любых объектов или связей описываемой предметной области выделить их окрестность по отношению принадлежности. Другими словами, ассоциативность семантических конструкций – это наличие простых процедур, с помощью которых легко находятся ответы на следующие вопросы:

- какими связями связан данный предмет с другими предметами;
- в каких связях заданный предмет выполняет заданную роль;
- какие предметы и под какими атрибутами участвуют в заданной связи;
- каким классам принадлежит заданный предмет или заданная связь;
- какими связями связано между собой некоторое число заданных предметов и (или) связей.

Такая ассоциативность семантических конструкций становится возможной благодаря тому, что каждый знак (знак какого-либо предмета, какой-либо связи, какого-либо понятия) в каждую семантическую конструкцию входит однократно.

Итак, семантическая конструкция есть такая информационная конструкция (общего или частного вида), в которой:

- 1) в качестве знаков используются элементы этой конструкции, а не более сложные ее фрагменты, состоящие из нескольких элементов;
- 2) отсутствует синонимия знаков, т.е. разные знаки не могут иметь совпадающие денотаты.

Очевидно, далеко не каждая информационная конструкция является семантической конструкцией.

Классическим примером несемантических информационных конструкций являются символьные конструкции (см. пункт 1.2.2.). Во-первых, знаки в символьной конструкции представлены идентификаторами – фрагментами символьной конструкции, состоящими в общем случае из нескольких символов. Это обусловлено тем, что алфавит символов в символьных конструкциях конечен и обычно состоит из небольшого числа унарных отношений, заданных на первичных элементах символьных конструкций, тогда как количество предметов, связей, понятий, обозначаемых знаками информационных конструкций, является, в общем случае, неограниченным. Во-вторых, в символьных конструкциях при описании нетривиальных предметных областей принципиально невозможно избежать синонимии знаков, т.е. многократного вхождения в символьную конструкцию знаков, обозначающих одно и то же.

Обычно синонимичные знаки в символьных конструкциях представляются равными (совпадающими, одинаковыми) строками символов. Неравные строки символов также могут быть синонимичными знаками. В символьных конструкциях встречаются омонимичные идентификаторы, т.е. идентификаторы, которые являются равными (совпадающими) строками символов, но имеют разную денотационную семантику.

В основе дальнейшего рассмотрения графодинамических моделей обработки информации будем использовать однородные информационные конструкции. Благодаря их однородности существенно сокращается номенклатура механизмов обработки этих конструкций на низших уровнях.

Первым семантическим свойством однородных информационных конструкций является то, что они имеют простую базовую семантическую интерпретацию, носящую теоретико-множественный характер и непосредственно не зависящую от описываемой предметной области. Каждый первичный элемент однородной информационной конструкции является либо знаком некоторого предмета описываемой предметной области (первичные элементы будем называть знаками предметов или предметными узлами), либо знаком некоторого множества, состоящего из первичных и (или) вторичных элементов однородной информационной конструкции. Таким образом, все первичные элементы (узлы) однородной информационной конструкции являются знаками двух типов: знаками предметов или знаками множеств. Причем элементами указанных множеств могут быть только первичные и вторичные элементы однородной информационной конструкции. Каждый вторичный элемент (дуга) однородной информационной конструкции является позитивным высказыванием о принадлежности некоторого первичного или вторичного элемента однородной информационной конструкции некоторому множеству, обозначаемому узлом, из которого указанная дуга выходит (т.е. тем узлом, который является первым компонентом этой дуги).

Таким образом, множество дуг однородной информационной конструкции определяет классифицирующее отношение, аналогичное АКО-отношению (a kind of...) и ISA-отношению (is a...), которые рассмотрены в работе [242] ([Кандрашина Е.Ю., 1989гн-ПредсЗоВиП](#)). Эти классифицирующие отношения не следует путать с классифицирующим отношением, которое связывает знаки множеств не с элементами этих множеств, а со знаками их подмножеств (подклассов). Такое классифицирующее отношение будем называть родовидовым отношением. Классифицирующие отношения являются основой для наследования свойств, что играет важную роль в решении большого числа задач.

Вторым семантическим свойством однородных информационных конструкций является то, что они всегда представляют собой семантические информационные конструкции, причем даже тогда, когда они являются информационными метаконструкциями, описывающими структуру несемантических информационных конструкций. Так, например, переходя от символьной информационной конструкции к ее эквивалентному представлению в виде однородной информационной конструкции, эта символьная информационная конструкция "превращается" в семантическую сеть, которая описывает синтаксис указанной символьной конструкции и которую можно легко нарастить семантической сетью, аналогичным образом устроенной и описывающей семантику этой символьной конструкции.

Итак, однородные информационные конструкции всегда являются семантическими информационными конструкциями. При этом по семантическим признакам выделяются два класса однородных информационных конструкций:

- 1) однородные информационные конструкции, описывающие структуру (синтаксис) информационных конструкций, не являющихся семантическими. Денотатами (описываемыми предметными областя-

- ми) таких однородных метаконструкций являются реляционные структуры, определяющие внутреннюю структуру несемантических информационных конструкций;
- 2) однородные информационные конструкции, описывающие структуру семантических информационных конструкций, которые, в свою очередь, описывают произвольные предметные области. Денотатами таких однородных информационных конструкций можно считать как указанную семантическую информационную конструкцию, так и предметную область, описанную этой семантической информационной конструкцией.

Исследуя семантические свойства информационных конструкций, можно говорить о фактографических и логических высказываниях (фактографических и логических информационных конструкциях). Фактографические высказывания непосредственно представляют структуру описываемой предметной области, т.е. представляют конкретные факты, имеющие место в описываемой предметной области. В отличие от этого логические высказывания представляют информацию о свойствах и законах описываемой предметной области. Благодаря этому появляется возможность компактного описания бесконечных предметных областей с помощью информационных конструкций, состоящих из конечного числа элементов.

Отличительными особенностями логических высказываний по сравнению с фактографическими являются:

- 1) появление логических переменных наряду со знаками конкретных предметов и связей предметной области (указанные знаки иногда называют логическими константами). Семантически каждая логическая переменная является знаком произвольного элемента из некоторого множества, которое может содержать как логические константы, так и логические переменные. Указанное множество элементов называют областью возможных значений соответствующей переменной;
- 2) появление простых (атомарных) высказываний и сложных (конъюнктивных, дизъюнктивных, импликативных) высказываний;
- 3) появление позитивных, негативных и нечетких высказываний;
- 4) появление кванторных высказываний, т.е. высказываний, свободные переменные которых связываются кванторами (кванторами существования, кванторами всеобщности и т.д.);
- 5) появление формальных теорий для стационарных и нестационарных предметных областей. Формальная теория – это конъюнктивное высказывание, изначально считающееся истинным. Соответственно этому позитивность, негативность и нечеткость компонентов (логических множителей) формальных теорий определяет истинные, ложные и неопределенные высказывания в рамках данной формальной теории.

Основным принципом представления информации о законах какой-либо предметной области (информационных знаний о предметной области) является формальное рассмотрение законов не самой описываемой предметной области, а универсального фактографического высказывания, которое включает в себя всю фактографическую информацию описываемой предметной области. Таким образом, логические высказывания по своей сути являются информационными метаконструкциями, т.е. конструкциями, описывающими другие информационные конструкции.

Для рассмотренных выше информационных конструкций фактографического типа выделим следующие классы:

- 1) базовые семантические информационные конструкции;
- 2) семантические конструкции различного вида, не являющиеся базовыми (в первую очередь, однородные информационные конструкции);
- 3) символные конструкции.

Совершенно аналогичные классы можно выделить и для информационных конструкций логического типа. Для этого в перечисленные фактографические высказывания необходимо дополнительно ввести средства представления логических переменных, логических связок, кванторов, высказываний, формальных теорий.

1.2.5. Классификация языков

Ключевые понятия: символный язык, графовый язык, семантический язык, сложно-структурированная предметная область, фактографический язык, логический язык, нестационарная предметная область, семантическая мощность языка.

С формальной точки зрения каждый конкретный язык представляет собой некоторое множество (являющееся бесконечным для практически интересных языков) информационных конструкций, которые имеют общие синтаксические свойства (т.е. общие принципы своего внутреннего устройства), а также общие денотационно-семантические свойства (т.е. общие принципы своего соотношения с описываемыми предметными областями). Общие синтаксические свойства информационных конструкций, принадлежащих языку, будем называть синтаксисом этого языка, а общие денотационно-семантические свойства информационных конструкций, принадлежащих языку, – денотационной семантикой этого языка. Синтаксис языка чаще всего задается как конструктивное определение множества так называемых синтаксически правильных (правильно построенных) конструкций соответствующего языка. Множество синтаксически правильных конструкций языка – это условное расширение множества конструкций, принадлежащих данному языку. Конструктивное определение множества синтаксически правильных конструкций языка задается следующими множествами:

- 1) множеством элементарных (атомарных, примитивных) конструкций;
- 2) множеством синтаксических правил, указывающих, как из имеющихся синтаксически правильных конструкций можно построить также синтаксически правильную конструкцию.

Денотационная семантика языка чаще всего формально задается как метод (в частности, алгоритм) перевода произвольной конструкции описываемого языка (языка-объекта) на некий другой язык, денотационная семантика которого считается известной и является достаточно простой. Языками с наиболее простой денотационной семантикой являются языки, конструкции которых представляют собой семантические конструкции.

С прагматической точки зрения каждому языку можно поставить в соответствие присущий только этому языку метод представления информации, являющейся описанием некоторого класса областей, т.е. метод построения информационных конструкций. Каждый такой метод ограничивает (уточняет) вид (внутреннее строение) информационных конструкций и соотношение информационных конструкций с описываемыми предметными областями.

Классификация языков осуществляется в соответствии с тем, какими синтаксическими и семантическими особенностями обладают информационные конструкции, принадлежащие этим языкам.

Первый признак классификации языков – линейность языковых конструкций. По этому признаку языки делятся на символные (линейные) и графовые (нелинейные, сетевые). **Символьные языки** – это языки, которым принадлежат только символные конструкции. Все остальные языки будем называть графовыми.

Во множестве **графовых языков**, в свою очередь, можно выделить следующие подклассы: языки информационных конструкций общего вида, языки бинарных информационных конструкций, языки однородных информационных конструкций, языки предельных информационных конструкций. Из сказанного ранее (см. пункт 1.2.3.) следует, что от каждого графового языка, принадлежащего одному из перечисленных классов, достаточно легко перейти к эквивалентному графовому языку, который принадлежит любому другому из перечисленных классов графовых языков.

Важнейшим свойством символьных языков является то, что для каждого такого языка при конечном фиксированном для всех конструкций языка алфавите символов (разным языкам могут соответствовать разные алфавиты) обеспечивается возможность формирования неограниченного количества знаков, соответствующих неограниченному количеству денотатов.

Предложенный нами язык однородных информационных конструкций также обеспечивает открытый характер при одинаковом для всех языковых конструкций наборе отношений, задающих эти конструкции.

По второму признаку классификации языки делятся на семантические и несемантические. **Семантические языки** – это языки, которым принадлежат только семантические конструкции. Все остальные языки считаются несемантическими.

Символьная информационная конструкция может быть семантической сетью, если описываемая предметная область имеет линейную структуру и если связи между символами в символьной информационной конструкции ставятся в соответствие связям между предметами в описываемой предметной области. Но далеко не каждая фактографическая конструкция может быть одновременно и семантической информационной конструкцией, и символьной информационной конструкцией. В частности, не существует символьного семантического языка, обеспечивающего эквивалентное представление произвольных информационных конструкций общего вида.

Третий признак классификации языков – степень сложности структур описываемых предметных областей. По этому признаку выделяют языки, ориентированные на описание сложноструктурированных предметных областей. В **сложноструктурированных предметных областях** отсутствуют ограничения на вид связей. В этих предметных областях связи могут быть между предметами, связями, фрагментами предметной области, относительными понятиями, абсолютными понятиями.

Четвертый признак классификации языков – наличие средств описания свойств предметных областей. По этому признаку языки делятся на фактографические и логические. **Фактографический язык** – это язык, обеспечивающий представление только фактографической информации (экстенсиональных знаний), т.е. информации о фактах, имеющих место в описываемой предметной области [242] ([Кандрашина Е.Ю..1989кн-ПредсЗоВиП](#)). Следовательно, фактографическому языку принадлежат только фактографические высказывания. Логический язык – это язык, обеспечивающий представление не только фактографической информации, имеющей отношение к некоторой предметной области, но и информации о свойствах и законах этой предметной области. Такую информацию называют интенсиональными знаниями [242] ([Кандрашина Е.Ю..1989кн-ПредсЗоВиП](#)). **Логический язык** – это язык, которому принадлежат как фактографические, так и логические высказывания (как фактографические, так и логические информационные конструкции).

Особенностями логических языков, отличающими их от языков фактографических, являются наличие логических переменных, наличие сложных (конъюнктивных, дизъюнктивных, импликативных) высказываний, негативных и нечетких высказываний, кванторных высказываний, наличие формальных теорий. Логические языки могут быть самыми различными как по синтаксическим, так и по семантическим особенностям. Классическими примерами логических языков являются языки предикатов 1-го порядка (языки 1-й ступени) и языки предикатов 2-го порядка (языки 2-й ступени) [316] ([Мальцев А.И.1970кн-АлгебС](#)).

Пятый признак классификации языков – наличие фактора времени. По этому признаку языки делятся на языки описания стационарных предметных областей и языки описания нестационарных предметных областей, т.е. предметных областей, состояние которых меняется во времени. **Нестационарная предметная область** формально трактуется как множество упорядоченных во времени стационарных предметных областей, называемых состояниями (ситуациями). Каждому состоянию нестационарной предметной области ставится в соответствие информационная конструкция, описывающая это состояние. В целом описание нестационарной предметной области есть метаописание системы информационных конструкций, каждая из которых описывает одно из состояний нестационарной предметной области. Следовательно, язык описания нестационарных предметных областей должен включать в себя язык описания стационарных предметных областей. Описание нестационарной предметной области можно также трактовать как метаописание некоторой нестационарной информационной конструкции. При этом за основу такого метаописания можно брать не только систему состояний нестационарной информационной конструкции, но и систему взаимодействующих переходных процессов, осуществляющих переход от одного состояния к другому. Важнейшим классом языков описания нестационарных предметных областей являются языки процедурного программирования, описывающие преобразование информационных конструкций.

Шестой признак классификации языков – наличие метаязыковых средств, достаточных для полного самоописания языка. По этому признаку языки делятся на стратифицированные, у которых указанные средства отсутствуют, и нестратифицированные, которые являются собственными метаязыками и которые, следовательно, не имеют четкой грани между языком-объектом и метаязыком, а также между информационной конструкцией-объектом и метаконструкцией.

Седьмой признак классификации языков – семантическая мощность. **Семантическая мощность языка** – это все то, что может быть описано с помощью конструкций этого языка. Языки с неограниченной семантической мощностью, т.е. языки, в которых имеется потенциальная возможность представления любой информации, будем называть универсальными. Согласно рассматриваемому признаку классификации, языки делятся на универсальные и специализированные. В свою очередь, специализированные языки могут сравниваться по своей семантической мощности, т.е. по многообразию описываемых предметных областей, а также по многообразию свойств, описываемых в этих областях. Универсальные языки можно рассматривать как результат интеграции всевозможных специализированных языков, поэтому при создании универсального языка принципиально важной является разработка методов интеграции специализированных языков в рамках создаваемого универсального языка. В основе такой интеграции, в частности, может лежать выделение языка-ядра, общего для всех интегрируемых языков. Универсальные языки по определению должны быть языками, ориентированными на описание сложноструктурированных предметных областей, логическими языками, языками описания нестационарных предметных областей, нестратифицированными языками.

Подчеркнем, что языки представления знаний для интеллектуальных систем нового поколения должны быть универсальными, а также открытыми языками, т.е. языками, позволяющими достаточно легко интегрировать самые различные специализированные языки.

Важнейшим классом специализированных языков являются языки программирования, включающие в себя языковые средства представления самих программ и языковые средства представления информационных конструкций (данных), которые перерабатываются в процессе реализации этих программ. Каждая программа есть описание некоторого метода решения произвольной задачи из определенного класса задач. Существуют два принципиально разных класса методов решения задач – процедурные и непроцедурные методы. Процедурный метод решения произвольной задачи из определенного класса задач – это явная декомпозиция всего процесса решения задачи на иерархическую систему более простых и, в конечном счете, элементарных процессов. Таким образом, описание процедурного метода решения некоторого класса задач (такое описание будем называть процедурной программой) есть не что иное, как описание некоторого класса нестационарных информационных конструкций. Следовательно, языки процедурного программирования следует отнести к классу языков описания нестационарных предметных областей (см. пятый признак классификации языков). В отличие от этого непроцедурный метод решения произвольной задачи из определенного класса задач явно не рассматривает сам процесс решения задачи, а представляет собой информационную конструкцию, которая, будучи соединена с исходными данными задачи из указанного класса, предоставляет решателю (абстрактной машине) дополнительную информацию, достаточную для решения произвольной задачи соответствующего класса. При этом способ использования решателем этой дополнительной информации может быть самый различный.

Важным классом специализированных языков для интеллектуальных систем являются языки спецификаций программ. Язык спецификаций программ – это метаязык, обеспечивающий описание программ, и в частности описание денотационной семантики программ, т.е. формальное определение класса задач, решаемых с помощью каждой программы. Наличие описаний спецификаций программ, имеющихся в памяти интеллектуальной системы, дает возможность интеллектуальной системе найти для появившейся у нее задачи программу, обеспечивающую решение этой задачи, если, конечно, такая программа имеется.

1.2.6. Семантические сети и семантические графовые языки

Ключевые понятия: графовый язык, графовый семантический язык представления знаний.

Семантические языки, т.е. языки, которым принадлежат только семантические информационные конструкции, являются основным объектом исследований в данной работе. Все семантические языки с нетривиальной семантической мощностью являются **графовыми языками**.

Можно говорить о целом семействе семантических графовых языков, называемых также языками семантических сетей. В частности, в соответствии с типологией семантических информационных конструкций, рассмотренных выше, можно выделить следующие семантические языки: язык базовых семантических информационных конструкций, язык бинарных информационных конструкций, язык однородных информационных конструкций.

Примерами графовых семантических языков (как специализированных, так и языков, претендующих на универсальность) также являются:

- предложенный В.С.Лозовским сетевой язык представления фреймов [401] ([Поспелов Д.А.ред.1990спр-ИскусИ-К2](#));
- язык растущих пирамидальных сетей [125] ([Гладун В.П.1987кн-ПланиР](#));
- способ организации семантических сетей, используемый В.Н.Вагиным для поддержки параллельного логического вывода [80; 401] ([Вагин В.Н.1989кн-ДедукИО](#); [Поспелов Д.А.ред.1990спр-ИскусИ-К2](#));
- способы организации семантических сетей, рассмотренные в работе [445] ([Скрэгг П.1983ст-СеманСкМП](#));
- используемый Э.Ф.Скороходько сетевой способ описания синтаксиса и семантики текстов естественного языка [444] ([Скороходько Э.Ф. 1983кн-СеманСиАот](#));
- используемый Р.Шенком сетевой способ описания семантики текстов естественного языка [554] ([Шенк Р.1980кн-ОбрабКИ](#));

- графовый язык представления информации в памяти вегетативной машины, предложенной В.Б.Борщевым [66] (*Борщев В.Б.1990ст-ПаралАВ*);
- графовый язык представления информации в памяти абстрактной сетевой машины, предложенной А.М.Степановым [452] (*Степанов А.М.1981пр-ФреймИПСВ*);
- графовый язык представления алгоритмов А.Н.Колмогорова и данных, перерабатываемых этими алгоритмами [259] (*Колмогоров А.Н..1958ст-кОпредA*);
- графовые языки представления различным образом устроенных производственных программ (систем продукции), ориентированных на переработку семантических сетей [276] (*Кузнецов В.Е.1989кн-ПредсВЭВМНП*);
- графовые логические языки, являющиеся различными способами сетевого представления логических высказываний, т.е. различными способами введения логических связок и кванторов в семантическую сеть [631; 80] (*Hendrix G.G.1979art-EncodKiP ; Вагин В.Н.1989кн-ДедукИО*).

К достоинствам семантических графовых языков представления знаний, по сравнению с другими способами представления знаний, относятся:

- компактность, обусловленная тем, что в отличие от символьного текста в семантической сети знак каждого объекта или понятия описываемой предметной области присутствует только в одном экземпляре и состоит из одного элемента;
- ассоциативность, заключающаяся в существовании простых процедур поиска элементов семантической сети, связанных заданным образом с заданными элементами;
- наличие простой возможности введения метаинформации в семантическую сеть путем простого наращивания исходной семантической сети метасетью без какого-либо изменения исходной семантической сети;
- возможность рассмотрения описываемых предметных областей одновременно на неограниченном числе уровней детализации;
- приспособленность к поддержке структур любого вида, и в частности к поддержке сложноструктурированных знаний;
- приспособленность к интеграции самых различных специализированных языков и самых различных моделей представления знаний;
- приспособленность к представлению различного рода лингвистических знаний (о синтаксисе, о семантике, о прагматике естественных языков), что делает эффективным использование семантических графовых языков для создания естественноязыковых интерфейсных подсистем в интеллектуальных системах;
- приспособленность к параллельной асинхронной переработке знаний, что делает эффективным использование семантических графовых языков для создания интеллектуальных систем, поддерживающих сложноструктурированные знания и сложные логические операции.

Подробнее о достоинствах семантических графовых языков см. в работах [127; 125; 444; 80] (*Гладун В.П.1977кн-ЭврисПвСС ; Гладун В.П.1987кн-ПланиР ; Скороходько Э.Ф.1983кн-СеманСиАОТ ; Вагин В.Н.1989кн-ДедукИО*).

В данной работе мы будем рассматривать графовый семантический язык представления знаний, ориентированный на описание сложноструктурированных предметных областей, обеспечивающий представление логических высказываний самого различного вида и обладающий высокой степенью открытости. Кроме того, мы будем рассматривать различные графовые языки параллельного асинхронного программирования, легко интегрируемые в состав указанного выше графового семантического языка представления знаний. Напомним при этом, что программы, представляющие собой описания различных методов решения различных классов задач, составляют важнейшую часть баз знаний интеллектуальных систем.

Предлагаемый в данной работе **графовый семантический язык представления знаний**, обладающий указанными выше свойствами, назван языком SCL (Semantic Code Logic) – см. раздел 5. В качестве базового языка представления знаний (языка-ядра), на основе которого осуществляется интеграция всевозможных специализированных языков в состав языка SCL, предлагается язык SC (Semantic Code) – см. раздел 4.

Резюме к подразделу 1.2

В подразделе 1.2 графовые языки рассматриваются в сравнении с символьными языками и в контексте общей типологии языков. Кроме того, формально рассмотрены синтаксические и семантические свойства информационных конструкций, принадлежащих графовым языкам.

В целях формального уточнения понятия предметной области и понятия информационного объекта введено понятие реляционной структуры как обобщение классического понятия алгебраической модели. Рассмотрены наиболее интересные виды информационных конструкций: бинарные, однородные, символьные информационные конструкции.

В целях формального уточнения денотационной семантики информационных конструкций введено понятие базовой семантической информационной конструкции. Денотационная семантика информационной конструкции определяется как ее соотношение с эквивалентной базовой семантической информационной конструкцией. Информационные конструкции произвольной структуры, имеющие наиболее простой вид соотношения с эквивалентными им базовыми семантическими информационными конструкциями, определены как семантические конструкции.

Графовые языки в настоящее время не являются практически используемыми языками, несмотря на их преимущества по сравнению с символьными языками. В данной работе мы пытаемся это устраниć. При этом нас будут интересовать не просто графовые языки, а графовые семантические языки, ибо языки именно этого класса дают достаточно "прозрачную" возможность реализации моделей параллельной асинхронной переработки знаний путем их сведения к более простым и в конечном счете к непосредственно реализуемым моделям.

Для того чтобы оценить практическую значимость идеи создания и реализации графовых языков, сделаем небольшой экскурс в становление теории графов как области математики. Особенность теории графов заключается в том, что исследуемые ею математические объекты были хорошо известны и до нее. Это конечные алгебраические модели частного вида. Но на эти алгебраические модели теория графов взглянула, условно говоря, "топологическим" взглядом. И сразу оказался обнаженным для этих структур целый ряд аспектов, имеющих, как скоро выяснилось, большое практическое значение. Конечно, графы как картинки, иллюстрирующие ту или иную задачу, использовались задолго до появления теории графов. Практика показала плодотворность теоретико-графового взгляда на большое количество задач, и не только при постановке задач, но и при рассмотрении процесса их решения. Но все это в настоящее время сдерживается определенной сложностью погружения теоретико-графового взгляда на решаемую задачу в современные технологии программирования, которые в конечном счете скрывают, камуфлируют то, что теоретико-графовый взгляд пытается обнажить. С появлением графовых языков представления знаний и графовых языков программирования это противоречие снимается.

Наиболее острая потребность в графовых языках ощущается в CAD-CAM-задачах, в задачах структурного распознавания, в задачах, связанных с обработкой иерархической сложноструктурированной информации.

В заключение сделаем несколько общих примечаний о графовых структурах.

Примечание 1. Графовый способ представления дискретной информации обладает универсализмом в том смысле, что любое представление дискретной информации можно рассматривать как графовую структуру. Следовательно, есть все основания считать понятие графовой структуры и понятие текста (в общем виде) тождественными понятиями. Действительно, когда мы говорим о графе, мы всегда подразумеваем некую предметную область, которую этот график описывает, будь то схема улиц города, сеть железных дорог, принципиальная электрическая схема, структура молекулы какого-либо органического соединения или система состояний конечного автомата. А текст всегда можно рассматривать как график, вершины и связи которого являются элементами текста (разумеется, элементы текста физически могут быть представлены самым различным образом). Так, например, текст, записанный на ленте машины Тьюринга, есть не что иное, как орграф с помеченными вершинами. Вершины этого орграфа есть ячейки ленты, в которые записывается тот или иной символ, а его дуги физически реализуются смежностью (соседством) ячеек ленты и связывают каждую ячейку ленты с непосредственно следующей за ней ячейкой.

Графовая трактовка текста встречается в целом ряде работ по формальным системам. Текстовая трактовка графовой структуры также встречается в ряде работ. Так, например, А.А.Зыков рассматривает графовую структуру как "средство описания тех или иных взаимосвязей между математическими объектами" [228] ([Зыков А.А. 1969гн-ТеориКГ](#)).

Итак, графовая структура можно рассматривать как объект или процесс, являющийся дискретной моделью некоторого другого объекта или процесса. Под дискретностью модели здесь понимается не фи-

зическое свойство объектов, являющихся дискретными моделями, а тот факт, что для восприятия информации, содержащейся в дискретной модели, достаточно изучить ее строение только до определенного уровня. Так, например, изменение написания какого-либо символа в рукописном тексте, не переходящее грани возможного его написания, не изменяет содержащейся в тексте информации, т.е. не является существенным для текста.

Примечание 2. Одну и ту же информацию можно представлять различными видами графов. Отсюда следуют два вывода:

- необходимо разрабатывать стандарты;
- среди этих стандартов нужно выявлять наиболее удобные способы графового представления информации.

Примечание 3. До сих пор в основном мы пользовались такими языками, которые являются способами представления дискретной информации в виде линейных (цепочечных) графовых структур. Сюда относятся все естественные языки, подавляющее большинство искусственных языков. Не исключением является и способ представления информации в памяти современных ЭВМ.

Примечание 4. Выход за рамки линейности текстов [317] ([Манин Ю.И. 1979кн-ДоказИН](#)) при разработке языков представления дискретной информации представляет значительные удобства.

Нелинейная графовая структура является естественной и удобной формой описания любой сложной системы (стационарной, нестационарной, абстрактной), в которой удается выделить некоторое множество элементов системы и некоторый набор отношений, заданных на этом множестве элементов. Эффективность нелинейного способа описания систем может быть проиллюстрирована следующими примерами.

Формализация описания систем, исследуемых органической химией, привела к графовой форме описания молекулы органических соединений.

Описание электронной системы оказалось удобнее всего строить в виде принципиальной электрической схемы, которая представляет собой не что иное, как граф.

Примечание 5. Среди способов (языков) нелинейного представления дискретной информации наиболее удобным для переработки является представление в виде семантических сетей. Об этом свидетельствует интенсивное развитие теории семантических сетей, которая сформировалась в рамках исследований по системам искусственного интеллекта, системам автоматического перевода с одного естественного языка на другой и которая рассматривается сейчас как формальный аппарат исследования сложных процессов обработки информации (распознавания образов, планирования целенаправленных действий, обнаружения закономерностей и т.д.).

Примечание 6. В рамках способа представления информации при помощи семантических сетей, в свою очередь, также возможен полиморфизм. Это требует разработки стандартных способов представления семантических сетей, т.е. специальных языков семантических сетей. Одним из таких языков является рассматриваемый ниже язык SC (Semantic Code).

Рассмотрение интеллектуальных систем на семантическом уровне является наиболее перспективным подходом к формальному уточнению понятия интеллектуальной системы. Такой подход развивается в работах И.П.Кузнецова [287; 288] ([Кузнецов И.П. 1978кн-МеханОСИ](#) ; [Кузнецов И.П. 1986кн-СеманП](#)), В.П.Гладуна [125; 127] ([Гладун В.П. 1987кн-ПланиР](#) ; [Гладун В.П. 1977кн-ЭврисПвСС](#)), Р.Шенка [554] ([Шенк Р.1980кн-ОбрабКИ](#)), Э.В.Попова [387] ([Попов Э.В. 1982кн-ОбщепСЭВМиЕЯ](#)) и многих других авторов.

Рассмотрение семантического уровня информационно-логической организации интеллектуальных систем позволяет в большой степени абстрагироваться от несущественных деталей технической реализации этих систем и, таким образом, позволяет сконцентрировать внимание на принципиальных особенностях их организации.

Рассмотрение интеллектуальных систем на семантическом уровне прежде всего требует выяснения того, что следует называть семантическим языком представления знаний. Проблема представления (организации) знаний в памяти интеллектуальной системы носит ключевой характер, так как от способа представления знаний, а точнее, от степени близости этого представления к семантическому самым существенным образом зависят простота и удобство их переработки [341; 386; 515; 127] ([Минский М.1978кн-СтрукДПЗ](#) ; [Попов Э.В. 1976кн-АлгорОИР](#) ; [Хант Э.1978кн-ИскусИ](#) ; [Гладун В.П. 1977кн-ЭврисПвСС](#)). Суть проблемы представления знаний заключается не в том, чтобы суметь представить зна-

ния (представить их можно, например, и на естественном языке), а в том, чтобы представить эти знания в памяти так, чтобы ими можно было достаточно удобно пользоваться. Следовательно, проблему представления и организации знаний нельзя решать в отрыве от логики системы и от организации ее памяти. Так, например, представление знаний существенным образом зависит от способа организации доступа к памяти. Хорошо организованные знания должны быть записаны в память так, чтобы достаточно просто можно было найти требуемую в текущий момент информацию (требуемый фрагмент знаний).

Наиболее перспективной в этом плане является ассоциативная память. При вводе информации в ассоциативную структурно перестраиваемую память не возникает проблемы распределения информации по памяти, так как при обращении к нужному фрагменту информации, хранимой в ассоциативной памяти, не требуется знать, в каком месте памяти находится этот фрагмент. Особенность ассоциативной организации знаний интеллектуальной системы заключается в том, что здесь требуется реализация ассоциативного доступа к таким фрагментам хранимых знаний, которые имеют в общем случае произвольный размер, вид и структуру. Реализация такого доступа к фрагментам знаний оказывается возможной, если знания представить в виде семантической сети. Поэтому семантическую сеть иногда называют ассоциативной сетью. Семантическая сеть (которую иногда также называют семантической моделью предметной области, смысловым графом, концептуальным графом, сетью концептуальной зависимости) представляет собой специальным образом организованную графовую структуру, вершины которой однозначно соответствуют понятиям представляющей предметной области, а связи между вершинами однозначно соответствуют связям между этими понятиями. Преимущество семантических сетей по сравнению с другими способами представления знаний обусловлено их компактностью, однозначностью и ассоциативностью [559] ([Шуберт Л. 1979 ст-УсилевМСС](#)).

О преимуществах семантических сетей свидетельствует удобство использования и широкое распространение теоретико-графовых методов решения различного рода задач [274; 425; 228] ([Кристофицес Н. 1978 кн-ТеориГ](#) ; [Рейнгольльд Э.. 1980 кн-КомбиАТИП](#) ; [Зыков А.А. 1969 кн-ТеориКГ](#)), а также активная разработка языковых [320; 446] ([Маркевичус Р. 1977 ст-ЯзыкПдОГ](#) ; [Скриган Н.И.. 1979 пр-СредсОИГС](#)), программных [44] ([Белоглазов В.Н.. 1974 ст-СистеAPЗ](#)) и аппаратных [193] ([Додонов А.Г.. 1979 ас-УстроДИГ](#)) средств решения теоретико-графовых задач.

Связь теоретико-графовых методов с семантическими сетями обусловлена тем, что теоретико-графовая трактовка задачи всегда неявно подразумевает формализацию перерабатываемых данных не просто в виде некоторой графовой структуры, а в виде такой графовой структуры, которая обладает вполне определенными семантическими свойствами, а именно – является семантической сетью, хотя явно о семантических сетях в теории графов речь не идет. Этим обстоятельством обусловлены естественность, наглядность и удобство теоретико-графовой трактовки задачи, в чем легко убедиться, сравнив содержательную запись какого-либо теоретико-графового алгоритма с записью программы, обеспечивающей его реализацию на современных ЭВМ.

Кроме традиционных приложений в органической химии, в электротехнике теоретико-графовые методы в настоящее время широко используются в социологии, экономике, теории вероятностей, генетике, математической лингвистике, проектировании дискретных устройств и т.д. Об эффективности теоретико-графовых методов говорят также результаты, полученные при разработке структурного подхода к распознаванию образов, который позволяет сократить сложность процедуры распознавания по сравнению с другими подходами и обеспечивает распознавание в тех случаях, когда другие подходы оказываются неприемлемыми [512] ([Фу К.С. 1977 кн-СтрукМвРО](#)). Как отмечается в работе [43] ([Белов В.В.. 1976 кн-ТеориГ](#)), "возможность приложения теории графов к столь различным областям заложена, в сущности, уже в самом понятии графа, сочетающего в себе теоретико-множественные, комбинаторные и топологические аспекты".

1.3. Абстрактные графодинамические ассоциативные машины

Ключевые понятия: абстрактная машина обработки информации; графодинамическая параллельная асинхронная абстрактная ассоциативная машина.

Цель данного подраздела – рассмотреть графодинамические параллельные асинхронные абстрактные машины с общих позиций, как подкласс абстрактных машин обработки информации, и показать, что этот подкласс абстрактных машин является наиболее перспективным для проектирования интеллектуальных систем нового поколения, а следовательно, и для проектирования компьютеров нового поколения, ориентированных на поддержку интеллектуальных систем.

1.3.1. Абстрактные машины обработки информации и соответствующие им операции, элементарные процессы и микропрограммы

Ключевые понятия: абстрактная машина обработки информации, память, операция.

Как было отмечено выше, каждому языку можно поставить в соответствие его синтаксис, отражающий свойства внутреннего строения конструкций этого языка, и денотационную семантику, отражающую общие свойства соотношения между конструкциями языка и описываемой ими предметной областью. Некоторым языкам можно поставить в соответствие также и методы манипулирования их конструкциями, направленные на решение различных задач. Такие методы манипулирования языковыми конструкциями (модели обработки информации, модели вычислений, модели решения задач) могут быть различными. Так, например, разным языкам программирования и разным языкам представления знаний могут соответствовать абсолютно разные принципы организации решения задач, абсолютно разные "системы мышления" [563] ([ЯзыкПАда-1981кн](#)).

Правила манипулирования языковыми конструкциями, соответствующие некоторому языку и направленные на решение различных задач в рамках этого языка, называются его операционной семантикой [358] ([Непейвода Н.Н.1983ст-СеманАЯ](#)). Операционная семантика различных языков может существенно отличаться друг от друга. Более того, некоторым наиболее мощным языкам (например, языкам представления знаний) может быть поставлено в соответствие несколько операционных семантик, т.е. некоторые языки допускают достаточно большую свободу в выборе методов (приемов, стратегий) решения задач.

Для того чтобы уточнить (формализовать) понятие способа организации процесса решения задач, для того чтобы разобраться во всем многообразии таких способов, вводится понятие абстрактной машины обработки информации (абстрактной информационной машины). Важность этого понятия отмечается в ряде работ, в частности в работе [482] ([Тыугу Э.Х.1984кн-Концеп](#)). Заметим также, что самому термину "абстрактная машина обработки информации" в других работах могут соответствовать такие термины, как абстрактный решатель задач, абстрактная вычислительная машина, абстрактный вычислитель, гипотетическая информационная машина, виртуальная информационная машина, абстрактный интерпретатор некоторого языка, абстрактная машина манипулирования конструктивными объектами, абстрактный компьютер. Введение понятия абстрактной машины преследует цель с некоторых единых позиций охватить все многообразие моделей вычисления, способов организации процесса решения задач, задаваемых самыми различными языками программирования, языками представления знаний, самыми различными архитектурами вычислительных систем. На основе понятия абстрактной машины различные способы организации вычислительного процесса можно трактовать как различные виды абстрактных машин, что дает формальную основу для сравнения различных способов организации обработки информации.

Итак, задать тот или иной способ организации обработки информации – это задать ту или иную абстрактную машину обработки информации. Не случайно поэтому различные абстрактные машины использовались для различных уточнений понятия алгоритма (машины Тьюринга, машины Поста).

В основе предлагаемой в данной работе трактовки абстрактной машины обработки информации лежит стремление привести к общему виду абстрактные машины самого различного вида. В частности, это выражается в том, что рассматриваемые нами абстрактные машины не содержат никаких устройств управления (которые являются уникальными для каждого вида абстрактных машин) и осуществляют управление взаимодействием операций только через память. Нетрудно показать, что к такому виду можно привести любую абстрактную машину путем формирования специальных управляющих структур в памяти.

Определение 1.3.1.1. Абстрактная машина обработки информации C задается парой $C = (CS, W)$,

где

CS – некоторым образом организованная **память** (запоминающая среда), в которой хранятся перерабатываемые информационные конструкции;

W – множество **операций** (правил вывода), выполняемых над указанной памятью.

Язык, которому принадлежат информационные конструкции, хранимые и преобразуемые в памяти абстрактной машины обработки информации, и есть тот самый язык, операционная семантика которого

определяется этой абстрактной машиной. Указанный язык вместе с соответствующей абстрактной машиной и информационной конструкцией, принадлежащей этому языку и определяющей начальное состояние памяти абстрактной машины, задают конкретную формальную модель обработки информации (см. пункт 1.1.1).

Память абстрактной машины обработки информации с формальной точки зрения есть некоторым образом устроенная нестационарная информационная конструкция, трактуемая как процесс преобразования хранимой в памяти стационарной информационной конструкции, определяющей текущее состояние реализуемой формальной модели. Формальное рассмотрение указанной нестационарной информационной конструкции осуществляется путем декомпозиции этой конструкции на систему определенным образом взаимодействующих между собой непрерываемых элементарных процессов. В разных абстрактных машинах обработки информации эта декомпозиция осуществляется разным образом и в известной мере носит условный (договорной) характер.

Множество всевозможных элементарных процессов абстрактной машины обработки информации разбивается на классы и каждому такому классу ставится в соответствие активный исполнитель, осуществляющий реализацию каждого элементарного процесса из соответствующего класса, если в текущем состоянии перерабатываемой информационной конструкции возникнут необходимые и достаточные условия выполнения этого элементарного процесса. Указанный активный исполнитель будем называть операцией или процессором абстрактной машины обработки информации. Абстрактная машина обработки информации в общем случае может иметь неограниченное число операций. Таким образом, элементарный процесс обработки информации в абстрактной машине – это процесс реализации одной из операций этой абстрактной машины.

В абстрактной машине обработки информации каждой ее операции ставится в соответствие программа, описывающая то, как осуществляется выполнение произвольного элементарного процесса из соответствующего для этой операции класса элементарных процессов. Указанные программы будем называть микропрограммами операций. Микропрограмма каждой операции абстрактной машины должна описывать три этапа выполнения элементарного процесса [80] (*Вагин В.Н. 1989 книга-ДедукИО*):

- 1) проверку условия выполнения элементарного процесса, т.е. поиск соответствующего фрагмента перерабатываемой информационной конструкции, над которым указанный элементарный процесс должен быть выполнен;
- 2) собственно выполнение элементарного процесса (преобразование перерабатываемой информационной конструкции);
- 3) оформление результата, включающее в себя формирование информационных конструкций, описывающих тип полученного результата, удаление вспомогательных информационных конструкций, создаваемых только для выполнения данного элементарного процесса.

В абстрактных машинах обработки информации, поддерживающих параллельное и асинхронное выполнение элементарных процессов, первый этап их выполнения может оказаться существенно сложнее второго.

В некоторых элементарных процессах второй и даже третий этапы их выполнения могут отсутствовать. Такие элементарные процессы будем называть нерезультивативными. Нерезультивативный элементарный процесс – это безуспешная попытка найти в текущем состоянии памяти некоторую информационную конструкцию, являющуюся достаточной для того, чтобы этот элементарный процесс закончился результативно. Нерезультивативные элементарные процессы никак не влияют на процесс переработки информационных конструкций, хранимых в памяти абстрактной машины. Тем не менее нерезультивативные элементарные процессы абстрактной машины необходимо исследовать, так как одним из направлений повышения эффективности абстрактной машины является снижение частоты появления таких процессов, чего можно добиться 1) путем подбора системы приоритетов на множестве операций, 2) путем совершенствования структуры перерабатываемых информационных конструкций, 3) путем совершенствования микропрограмм операций.

Микропрограмма каждой операции абстрактной машины представляет собой описание семантики соответствующей операции этой машины. Систему операций абстрактной машины можно считать первым уровнем декомпозиции абстрактной машины, а систему соответствующих микропрограмм – вторым уровнем ее декомпозиции. Микропрограммы не должны непосредственно храниться в памяти этой абстрактной машины. Хотя очевидно, что память абстрактной машины, интерпретирующей некоторую другую абстрактную машину, должна хранить микропрограммы всех операций интерпретируемой абстрактной машины.

Операции абстрактной машины взаимодействуют друг с другом только через память абстрактной машины и в этом смысле являются автономными (самостоятельными). Операции абстрактной машины реализуются асинхронно, и если в текущем состоянии памяти абстрактной машины для нескольких операций одновременно существуют условия их реализации, то эти операции могут быть реализованы параллельно. Каждая операция абстрактной машины реагирует на соответствующую этой операции ситуацию, возникающую в текущем состоянии памяти абстрактной машины. Таким образом, синхронизация выполнения операций абстрактной машины, и в частности организация их последовательного выполнения, осуществляется через память этой машины с помощью специальных управляющих структур данных, описывающих текущее состояние процесса обработки информации. Следовательно, управление последовательностью выполнения элементарных процессов абстрактной машины обработки информации осуществляется децентрализованным образом и непосредственно входит в компетенцию каждой операции.

Из сказанного следует, что микропрограммы всех операций абстрактной машины носят активный, демонический характер, т.е. инициируются самопроизвольно в произвольные моменты времени, независимо друг от друга и без каких-либо внешних причин. Другими словами, для того чтобы инициировать микропрограмму любой операции абстрактной машины, никакого явного обращения к ней из другой микропрограммы, никакого явного ее вызова не требуется.

Язык, операционная семантика которого определяется абстрактной машиной обработки информации, будем называть внутренним языком этой абстрактной машины. Наряду с внутренним языком абстрактная машина обработки информации может иметь несколько внешних языков, с помощью которых осуществляются ввод информации в память и вывод информации из памяти абстрактной машины. Такой обмен информацией с пользователями машины обработки информации поддерживается соответствующими операциями ввода и вывода информационных конструкций. Эти операции осуществляют трансляцию вводимых информационных конструкций с некоторого внешнего языка абстрактной машины обработки информации на ее внутренний язык, а также трансляцию выводимых информационных конструкций с внутреннего языка на тот или иной внешний язык. Очевидно, что такие операции-трансляторы могут быть достаточно сложными, если внешние языки абстрактной машины обработки информации сильно отличаются от ее внутреннего языка.

Противопоставление внешнего и внутреннего языка является для абстрактных машин относительным, ибо для любого способа представления информационных конструкций, который является внешним для одной абстрактной машины, можно построить другую абстрактную машину, в которой указанный способ представления будет внутренним.

Кроме операций ввода и вывода информационных конструкций, предназначенных для взаимодействия с пользователем, абстрактная машина обработки информации может взаимодействовать с внешней средой также с помощью различных рецепторных и эффекторных операций. Рецепторные операции осуществляют формирование информационных конструкций в памяти абстрактной машины путем описания (на внутреннем языке) текущего состояния соответствующих рецепторных подсистем. Эффекторные операции осуществляют воздействие на внешнюю среду путем изменения состояния соответствующих эффекторных подсистем.

Завершая рассмотрение понятия абстрактной машины обработки информации, определяющей операционную семантику некоторого языка путем его непосредственной интерпретации, подчеркнем, что это понятие имеет также большую практическую ценность, поскольку строгое описание абстрактной машины обработки информации (перечисление всех ее операций и соответствующих им микропрограмм) содержит всю информацию, необходимую для реализации указанного языка любым выбранным способом (программным, аппаратным, аппаратно-программным).

Понятие абстрактной машины обработки информации является также важной методологической опорой при проектировании вычислительных машин с нетрадиционной (не фон-Неймановской) архитектурой, особенно параллельных машин. "Если мы занимаемся исследованием предельных теоретических машин, а не практическим инженерным анализом существующих устройств, то необходимо абстрагироваться от многих реальных деталей и особенностей систем. Большой частью это абстрагирование заходит так далеко, что остается лишь скелетное представление о структуре последовательности событий внутри машины, т.е некоторого рода "символическая" или "информационная" структура машины. При таком уровне абстракции мы игнорируем геометрию расположения частей. Мы игнорируем вопросы, касающиеся энергии. Мы даже разбиваем время на последовательность отдельных, не связанных между собой моментов и вообще игнорируем пространство как таковое! Может ли вообще столь абстрактная теория быть теорией чего-либо? Как ни странно, может. Выделяя только те вопросы, которые касаются логических выводов из определенного вида причинно-следственных отношений, мы можем сконцентрировать наше внимание на немногих действительно фундаментальных про-

блемах. Разобравшись в этих вопросах, мы можем вернуться в мир практики, где столь ясное понимание сути дела было бы невозможным из-за множества несущественных деталей" [342] ([Минский М.1971кн-ВычисИА](#)).

Использование понятия абстрактной машины для уточнения различных моделей вычислений осуществляется в работе [24] ([Ахо А..1979кн-ПострИАВА](#)).

Понятие абстрактной машины как средства формального рассмотрения операционной семантики языков программирования, причем не обязательно процедурного программирования, используется в работах [5; 520; 408] ([Агафонов В.Н.1990кн-СпецИП](#) ; [Хендерсон П.1983кн-ФункЦП](#) ; [Пратт Т.1979кн-ЯзыкиИП](#)).

О трактовке машин Тьюринга как о частном виде параллельных асинхронных абстрактных машин см. в работе [5] ([Агафонов В.Н.1990кн-СпецИП](#)).

1.3.2. Классификация абстрактных машин обработки информации

Ключевые понятия: абстрактные машины с символьной памятью; абстрактные машины с графовой памятью; абстрактные машины со структурно фиксированной памятью; абстрактные машины со структурно перестраиваемой памятью; абстрактные машины переработки знаний; абстрактные машины реализации хранимых программ; последовательные абстрактные машины обработки информации; параллельные абстрактные машины обработки информации; синхронные абстрактные машины обработки информации; асинхронные абстрактные машины обработки информации; абстрактные машины обработки информации, в памяти которых описание элементарных информационных процессов не осуществляется; абстрактные машины обработки информации, управляемые потоком команд; абстрактные машины обработки информации, управляемые потоком данных.

Как было отмечено в пункте [1.1.2](#), тип абстрактной машины и тип внутреннего языка этой машины полностью определяют тип соответствующей формальной модели. Типология языков рассмотрена в пункте [1.2.5](#). Классификацию абстрактных машин будем проводить по следующим признакам.

1. Структурный тип информационных конструкций, хранимых в памяти абстрактной машины. По этому признаку можно выделить:

- **абстрактные машины с символьной (линейной) памятью**, т.е. абстрактные машины, в памяти которых могут храниться только символьные конструкции;
- **абстрактные машины с графовой (нелинейной) памятью**, т.е. абстрактные машины, в памяти которых непосредственным образом могут храниться графовые конструкции.

2. Характер изменения состояния памяти, возможность изменения связей между элементами памяти. По этому признаку можно выделить:

- **абстрактные машины со структурно фиксированной памятью**, обработка информации в которых сводится только к изменению состояния элементов памяти;
- **абстрактные машины со структурно перестраиваемой (модифицируемой) памятью**, обработка информации в которых сводится к изменению состояния элементов памяти и к изменению связей между ними.

Абстрактные машины со структурно фиксированной и структурно перестраиваемой памятью могут перерабатывать как символьные, так и графовые информационные конструкции. Абстрактные машины с графовой структурно перестраиваемой памятью будем называть графодинамическими. Такие машины являются основным объектом рассмотрения в данной работе.

В структурно фиксированной памяти жестко задана структура элементов памяти, система связей между ними. Поэтому в структурно фиксированной памяти возникает проблема размещения хранимой информационной конструкции в памяти, т.е. проблема наложения информационной конструкции на структуру памяти. Кроме того, при переработке информации в структурно фиксированной памяти часто возникает проблема переразмещения (перераспределения) информационных конструкций, хранимых в памяти.

Структурно фиксированная память является неудобной даже для переработки символьных конструкций. Так, например, в символьной структурно фиксированной памяти весьма громоздко реализуются такие элементарные и часто используемые преобразования, как вставка строки символов или удаление подстроки. Реализация этих преобразований в символьной структурно фиксированной памяти требует достаточно утомительных действий по перезаписи (сдвигу) хранимых в памяти строк символов. Очевидно, что действий по перезаписи хранимой в памяти информации можно вообще избежать, если разрешить менять не только состояние элементов памяти, но и связи между ними. Благодаря этому становится возможной вставка (запись в память) новых информационных конструкций, а также удаление (стирание в памяти) фрагментов перерабатываемой информационной конструкции, оказавшихся в какой-то момент лишними, без какой-либо перезаписи остальной части хранимой в памяти информационной конструкции.

3. Уровень организации доступа к нужным фрагментам перерабатываемой информационной конструкции, который определяется количеством усилий, затрачиваемых программистом (при составлении программы), и количеством усилий, затрачиваемых операциями абстрактной машины на выделение (локализацию, поиск) требуемого фрагмента перерабатываемой информационной конструкции. Вид доступа к требуемым фрагментам определяется 1) ограничениями на размеры и структуру этих фрагментов и 2) значимыми признаками искомого фрагмента, на основании которых осуществляется поиск, т.е. тем, что должна знать абстрактная машина о требуемом фрагменте хранимой информационной конструкции для того, чтобы выделить этот фрагмент в рамках всей хранимой информационной конструкции. В настоящее время наиболее известными методами доступа являются **последовательный доступ, адресный (прямой) доступ, доступ по ключу и обобщенный ассоциативный доступ**.

Первые три из перечисленных методов доступа предполагают расчленение хранимой информационной конструкции на некоторые области (зоны, блоки) фиксированной или переменной длины, к которым и осуществляется непосредственный доступ. При последовательном методе доступа на множестве указанных областей задается бинарное отношение непосредственного соседства, являющееся отношением строгого порядка. Описание области, к которой осуществляется непосредственный доступ, заключается в указании того, где она находится по отношению к области, обозреваемой в текущий момент (справа от нее или слева). Таким образом, последовательный метод доступа в каждый момент времени обеспечивает возможность доступа только к двум смежным областям хранимой информационной конструкции. Адресный метод доступа устраняет этот недостаток путем условного задания взаимно однозначного отображения множества областей хранимой информационной конструкции во множество имен, называемых адресами этих областей. Описание области, к которой осуществляется непосредственный доступ, здесь заключается в указании ее адреса. Адресный метод доступа, как и последовательный, требует того, чтобы программист точно знал, где в памяти находится каждая область хранимой информационной конструкции. Это вызывает проблему распределения памяти, которая заключается в том, что сам программист должен "раскладывать по полочкам" перерабатываемую им информацию. Это, очевидно, требует значительных накладных расходов.

Поскольку адрес требуемой области перерабатываемой информационной конструкции не всегда легко определить, требуемую область удобнее задавать не ее адресом, а некоторой известной ее подструктурой (например, известными значениями некоторых разрядов). В этом случае программисту не требуется знать, в каком месте памяти находится необходимая ему область [508] (**Фостер К. 1981 книга АссоцИПП**). Такой метод доступа называется доступом по ключу (по ключевому набору заранее известных признаков).

Общим недостатком первых трех методов доступа является наличие определенных ограничений на структуру и "размеры" областей (единиц доступа) хранимой информационной конструкции. Снятие каких бы то ни было ограничений на размеры и структуру областей информационных конструкций, требующее отсутствия разбиения этих объектов на области, предпринято при организации обобщенного ассоциативного метода доступа (доступа по значению) [187] (**Дейт К. 1980 книга ВведенСБД**). Здесь описание требуемой области есть указание ее структуры (с точностью до изоморфизма) или некоторой ее подструктуры, дополненной указанием некоторых свойств. С логической точки зрения такое описание требуемого фрагмента информационной конструкции представляет собой не что иное, как его определение. Следовательно, такой язык описания (задания) фрагментов можно с полным основанием считать декларативным языком запросов. Подчеркнем, что обобщенный ассоциативный метод доступа оказывается единственным возможным для языка программирования, специально ориентированного на переработку баз данных, поскольку программист в этом случае не может заранее знать, где и как хранятся требуемые ему данные (это оказывается возможным, только когда сам программист организует для себя все необходимые для него данные). Таким образом, обобщенный ассоциативный метод доступа обеспечивает произвольную переместимость данных в памяти и независимость данных

от программ, их использующих. Следует также заметить, что переработку базы данных с использованием обобщенного ассоциативного метода доступа логически очень просто организовать как вычисление, управляемое потоком данных.

Нетрудно заметить, что все виды доступа можно формально проинтерпретировать как частный случай обобщенного ассоциативного доступа. Очевидно также, что снятие ограничений на вид и размер требуемых фрагментов информационной конструкции позволяет существенно увеличить множество реализуемых абстрактной машиной операций.

Нетрудно заметить, что в полном, наиболее развитом варианте обобщенный ассоциативный метод доступа реализуем только на основе использования графовых языков и граffодинамической памяти.

Подчеркнем, что все исследуемые в данной работе абстрактные машины используют обобщенный ассоциативный метод доступа к фрагментам хранимой информационной конструкции.

4. Денотационная семантика перерабатываемых информационных конструкций и смысл операций абстрактной машины. По этому признаку можно выделить **абстрактные машины переработки знаний и абстрактные машины реализации хранимых программ.**

5. Возможность или невозможность одновременного выполнения нескольких результативных элементарных процессов. По этому признаку можно выделить:

- **последовательные абстрактные машины обработки информации** (абстрактные машины обработки информации, осуществляющие последовательное выполнение элементарных процессов);
- **параллельные абстрактные машины обработки информации.**

Подчеркнем, что абстрактные машины обработки информации в общем случае считаются параллельными, что следует из определения абстрактных машин. Следовательно, последовательные абстрактные машины считаются абстрактными машинами частного вида. Согласно определению абстрактной машины, ее параллельность или последовательность определяется не самими операциями, а особенностями хранимой в памяти информационной конструкции, ибо каждая операция абстрактной машины считается абсолютно автономной и может реализовываться, как только в памяти возникнет ситуация, удовлетворяющая условию ее применения. При этом реализуется операция абсолютно независимо от того, реализуется в это время какая-то другая операция или нет. Следовательно, для последовательной абстрактной машины характерно то, что в ее памяти никогда не возникает ситуация, которая бы одновременно удовлетворяла условиям применения сразу нескольких операций.

Необходимо отметить, что в параллельных машинах обработки информации понятие текущего состояния ее памяти становится неопределенным, так как может не существовать ни одного момента, в который бы не выполнялось ни одного элементарного процесса. То есть память параллельной машины обработки информации всегда может находиться в переходном состоянии, когда до завершения каждого элементарного процесса начинается и не заканчивается какой-либо другой элементарный процесс или несколько элементарных процессов.

6. Характер взаимодействия элементарных процессов (насколько произвольным может быть выбор момента начала выполнения элементарных процессов). По этому признаку можно выделить:

- **синхронные абстрактные машины обработки информации;**
- **асинхронные абстрактные машины обработки информации.**

Из определения абстрактных машин обработки информации следует, что в общем случае такие машины являются асинхронными. Синхронность в абстрактной машине обработки информации достигается с помощью специальных управляющих (синхронизирующих) информационных конструкций, которые явно фиксируют факты завершения элементарных процессов того или иного вида и соответствующим образом анализируются при проверке условий выполнения элементарных процессов.

Суть асинхронного взаимодействия элементарных процессов абстрактной машины обработки информации заключается в том, что в каждый момент времени могут существовать условия инициирования нескольких элементарных процессов. Но последовательность выполнения этих процессов произвольна. Таким образом, в асинхронной машине обработки информации наличие условия выполнения того или иного элементарного процесса означает не предписание (команду), а только разрешение на выполнение этого процесса.

7. Характер описания в памяти и инициирования элементарных информационных процессов. По этому признаку можно выделить:

- **абстрактные машины обработки информации, в памяти которых описание элементарных информационных процессов не осуществляется ни в каком виде (этот класс абстрактных машин обработки информации полностью совпадает с абстрактными машинами переработки данных);**
- **абстрактные машины обработки информации, управляемые потоком команд;**
- **абстрактные машины обработки информации, управляемые потоком данных;**
- **абстрактные машины обработки информации, управляемые потоком целей.**

Команда, представляющая собой инициированный оператор хранимой процедурной программы, содержит полную информацию, необходимую для выполнения соответствующего элементарного информационного процесса.

В абстрактной машине обработки информации, управляемой потоком данных, хранимая (уже непрограммная, а точнее, функциональная) программа также представляет собой совокупность операторов. Но условием реализации каждого из этих операторов является не его явное инициирование, а вычислительность всех его операндов.

В абстрактных машинах обработки информации, управляемых потоком целей, операции и элементарные процессы имеют дело уже не с операторами, каждый из которых задает преобразование заранее известного типа, а с описаниями целей, способ достижения которых определяется не только самими целями, но и их контекстами.

8. Степень открытости (гибкости, модифицируемости) абстрактных машин.

Для открытых абстрактных машин характерно то, что добавление в этих машинах новых операций или модификация имеющихся не требует модификации остальных операций либо требует их локальной модификации.

Высокая степень открытости абстрактных машин является важнейшим требованием, предъявляемым к современным системам обработки информации, особенно к современным системам переработки знаний.

Принятая трактовка абстрактных машин создает все необходимые предпосылки для создания абстрактных машин с высокой степенью открытости, так как эта трактовка основана 1) на отсутствии какого-либо непосредственного взаимодействия между операциями (отсутствие вызова одной операции из другой), 2) на использовании самопроизвольного инициирования операций, 3) на использовании только одного способа фиксации текущего состояния машины – через текущее состояние памяти, 4) на организации взаимодействия и синхронизации элементарных процессов только через память абстрактной машины.

Тем не менее следует подчеркнуть, что наиболее высокого уровня открытости можно добиться только от абстрактных машин, имеющих графовую структурно перестраиваемую память. Высокий уровень открытости таких абстрактных машин обеспечивается высоким уровнем их ассоциативности. Именно поэтому данная работа, направленная на создание максимально открытых средств переработки знаний, базируется на исследовании графодинамических абстрактных машин.

1.3.3. Графодинамические параллельные асинхронные абстрактные машины как наиболее перспективный класс абстрактных машин для проектирования сложных интеллектуальных систем

Преимущество использования графодинамических параллельных асинхронных абстрактных машин в качестве инструмента для создания интеллектуальных систем нового поколения обусловлено следующими обстоятельствами:

- 1) в графодинамических машинах принципиально проще реализуется ассоциативный метод доступа к перерабатываемой информации;
- 2) в графодинамических машинах существенно проще поддерживается открытый характер как самих машин, так и реализуемых на них формальных моделей. В частности, на графодинамических машинах существенно проще реализуются семиотические модели;

- 3) графодинамические параллельные асинхронные машины являются удобной основой для интеграции самых различных моделей обработки информации, ибо на базе графодинамических параллельных асинхронных абстрактных машин легко реализуются не только графодинамические, но и символные формальные модели, не только параллельные, но и последовательные модели, не только асинхронные, но и синхронные модели, не только различные модели логического вывода, но и всевозможные модели реализации хранимых процедурных программ.

Остальные достоинства графодинамических параллельных асинхронных машин обусловлены достоинствами графовых информационных конструкций и графовых языков (см. подраздел 1.2). Подчеркнем при этом, что нас будут интересовать в первую очередь такие графодинамические параллельные асинхронные машины, внутренними языками которых являются семантические языки.

Резюме к подразделу 1.3

Соображения, приведенные в подразделе 1.3, позволяют сделать вывод о том, что графодинамические параллельные асинхронные абстрактные машины являются наиболее перспективной основой для реализации интеллектуальных систем нового поколения. При этом особое внимание следует уделить трем видам графодинамических параллельных асинхронных абстрактных машин, определяющим три уровня параллельных интеллектуальных систем:

- 1) графодинамическим параллельным асинхронным абстрактным машинам логического вывода. Перерабатываемая информация (перерабатываемые знания) в таких машинах представляется в виде семантических конструкций (семантических сетей) на некотором графовом семантическом языке представления знаний. А операциями в этих машинах являются операции логического вывода, поддерживающие самые различные стратегии решения задач в базах знаний;
- 2) графодинамическим параллельным асинхронным абстрактным машинам реализации хранимых процедурных программ, специально ориентированным на интерпретацию графодинамических параллельных асинхронных абстрактных машин логического вывода;
- 3) графодинамическим распределенным (!) асинхронным абстрактным машинам реализации хранимых процедурных программ, определяющим крупнозернистую архитектуру графодинамического параллельного асинхронного компьютера, ориентированного на поддержку всевозможных графодинамических параллельных асинхронных моделей обработки информации. Операции абстрактных машин данного уровня, в отличие от абстрактных машин второго уровня, имеют ограниченную область действия, т.е. имеют доступ не ко всей памяти, а только к какой-то ее области. Следовательно, вся память абстрактной машины разбивается на области, каждой из которых ставится в соответствие набор работающих над ней операций. При этом наборы операций, работающих над различными областями памяти, абсолютно аналогичны и отличаются только областью действия операторов. Таким образом, графодинамическая распределенная асинхронная абстрактная машина реализации хранимых процедурных программ представляет собой коллектив графодинамических абстрактных машин (которые будем называть модулями распределенной машины), взаимодействующих между собой путем обмена сообщениями. Особенностями распределенных абстрактных информационных машин являются а) наличие в каждом модуле входных и выходных буферов для приема и передачи сообщений, б) наличие специальных операций пересылки сообщений из выходного буфера модуля-передатчика во входной буфер модуля-приемника (модуля-адресата), в) наличие специальных операций обработки принятых сообщений.

В настоящее время исследования по графовым языкам, графодинамическим машинам и графодинамическим формальным моделям в основном носят частный теоретический характер и пока не привели к широкому появлению практически пригодных и конкурентоспособных графовых языков программирования, графовых языков представления знаний, графодинамических логик и графодинамических компьютеров. К числу работ, связанных с графодинамическими моделями обработки информации, можно отнести работы по графовым грамматикам [376; 11; 379] ([Петров С.В..1977ст-ГрафоГиЗГ ; Айзerman M.A..1977ст-ДинамПкАС](#) ; [Петров С.В..1978ст-ГрафоГиA](#)), работы по исследованию логического вывода в семантических сетях [80] ([Вагин В.Н.1989кн-ДедукиО](#)), работы по исследованию алгоритмов А.Н.Колмогорова [259; 174; 175] ([Колмогоров А.Н..1958ст-кОпредA ; Григорьев Д.Ю.1976ст-АлгорКСМ ; Григорьев Д.Ю.1977ст-ТеореВдМТ](#)), работы А.Шенхаге по исследованию абстрактных машин с модифицируемой памятью [674] ([Schonhage A.1980art-StoraMM](#)), работы В.Б.Борщева по вегетативной машине [65] ([Борщев В.Б.1983ст-СхемыНКС](#)), работы А.М.Степанова по абстрактной сетевой машине [452; 453] ([Степанов А.М.1981пр-ФреймИПСВ ; Степанов А.М.1981пр-ЭкспеСП](#)).

Выводы к разделу 1

1. Поскольку решение задач в интеллектуальных системах требует использования самых различных методов (процедурных программ, непроцедурных программ, баз знаний), важной проблемой является не только разработка каждого из этих методов, но и разработка способов их эффективной интеграции, т.е. эффективного совместного использования в рамках одной интеллектуальной системы. Поэтому актуальным является создание принципов, в рамках которых такая интеграция была бы достаточно легко реализуемой.
2. На современном этапе развития языков программирования и языков представления знаний особую актуальность приобретает совершенствование таких их свойств, как открытость (модифицируемость), ассоциативность доступа к перерабатываемой информации, поддержка параллельной асинхронной реализации операций.
3. Совершенствование методов описания операционной семантики самых различных языков и, соответственно, методов интерпретации этих языков настоятельно требует введения общего понятия абстрактной машины, в рамках которого можно было бы достаточно легко специфицировать всевозможные методы организации обработки информации.
4. Вводится понятие реляционной структуры, являющееся способом формального уточнения сложно-структурированных предметных областей, сложноструктурированных информационных объектов и представляющее собой обобщение таких понятий, как алгебраическая модель, гиперсеть, клубная система. По семантическому принципу реляционные структуры разбиваются на два типа: 1) реляционные структуры, определяющие структуру различных предметных областей, и 2) реляционные структуры, определяющие структуру (синтаксис) различных информационных объектов. Реляционные структуры 2-го типа называются информационными конструкциями.
5. Вводится понятие графовой информационной конструкции. Рассматриваются графовые информационные конструкции частного вида (бинарные информационные конструкции и однородные информационные конструкции), являющиеся различными способами кодирования произвольных реляционных структур.
6. Уточняется понятие семантической сети (семантической графовой информационной конструкции).
7. На современном этапе развития языков программирования и языков представления знаний перспективным является класс графовых языков, обеспечивающих представление информации в виде тем или иным способом устроенных семантических сетей. Графовые языки поддерживают открытость, ассоциативность, параллельность, асинхронизм. Графодинамическая парадигма обработки информации является наиболее перспективной основой для создания интеллектуальных систем нового поколения, в которых должны поддерживаться сложноструктурированные базы знаний большого объема, сложные стратегии и механизмы решения трудноформализуемых задач, гибкость (модифицируемость) используемых стратегий и механизмов решения задач, интегрируемость различных моделей переработки знаний, асинхронизм и высокий уровень параллелизма.
8. Рассмотрены основные понятия, связанные с графодинамической парадигмой обработки информации, – понятие графодинамической формальной модели, понятие графовой конструкции, понятие семантической графовой конструкции, понятие графового языка, понятие графового семантического языка, понятие графодинамической абстрактной машины, понятие графодинамической параллельной асинхронной абстрактной машины.
9. Графодинамические модели обработки информации, графовые языки программирования, графовые языки представления знаний, графодинамические абстрактные машины – все это является естественным результатом эволюции традиционных моделей обработки информации, направленной на поддержку сложноструктурированности, гибкости, модифицируемости, асинхронности, ассоциативности, параллельности.
10. Основными направлениями работ по созданию комплекса средств, ориентированных на поддержку интеллектуальных систем нового поколения, являются: создание базового графового языка, обеспечивающего легкую интегрируемость различных графодинамических формальных моделей, построенных с использованием этого языка; создание графового логического языка расширяемого типа и соответствующей ему графодинамической абстрактной машины логического вывода; создание графовых языков

программирования и соответствующих им графодинамических абстрактных машин, обеспечивающих легкую интегрируемость с графодинамическими моделями логического вывода; создание графового языка программирования и соответствующей ему графодинамической абстрактной машины для эффективной интерпретации всевозможных графодинамических параллельных моделей; создание графодинамического параллельного асинхронного ассоциативного компьютера, ориентированного на эффективную интерпретацию всевозможных графодинамических параллельных моделей.

11. В основе формального рассмотрения интеллектуальных систем на самом верхнем уровне лежит специальный класс моделей обработки информации, называемых моделями логического вывода. Основными особенностями таких моделей являются:

- 1) рассмотрение обработки информации на семантическом уровне;
- 2) сложность операций, семантически трактуемых как операции логического вывода, как содержательно осмысливаемые механизмы решения задач, поддерживающие ту или иную стратегию решения;
- 3) объективно присущие этим моделям параллельность и асинхронность реализации операций.

Добавим к этому то, что модели логического вывода, соответствующие практически полезным интеллектуальным системам, оперируют перерабатываемыми информационными конструкциями (базами знаний) большого размера и сложной структуры. Последнее обусловлено не только тем, что практически интересные модели логического вывода часто бывают модифицируемыми, но и тем, что даже в классических (немодифицируемых) моделях логического вывода практически интересные модели обычно имеют дело со знаниями, которые описывают сложноструктурированные предметные области.

Известные модели представления и переработки знаний а) имеют свои достоинства и недостатки, б) не противоречат друг другу, т.е. не являются альтернативными, и в) дополняют друг друга, поскольку каждая из этих моделей акцентирует свое внимание не на всех, а только на некоторых аспектах представления знаний.

Следовательно, одним из направлений повышения эффективности известных моделей представления знаний является их интеграция, т.е. создание комплексной модели представления знаний, в рамках которой гармонично бы сочетались и логическая модель представления знаний, и сетевая (графовая) модель представления знаний, и фреймовая, и продукционная модель, а также различные модели представления процедурных знаний, модели представления и реализации программ. О целесообразности интеграции различных моделей представления знаний говорится, в частности, в работе [401] ([Поспелов Д.А.ред. 1990 спр-ИскусИ-K2](#)).

Построение комплексной (интегрированной) модели представления знаний целесообразно начинать с создания базовой (унифицированной, стандартизованной) сетевой модели, которая бы являлась ядром, основой всего комплекса моделей представления знаний. Целесообразность такого подхода обусловлена тем, что основным препятствием для интеграции различных моделей представления знаний является отсутствие базового языка.

В случае создания удачного базового графового языка, т.е. базовой сетевой (графовой) модели представления знаний, интеграция различных моделей представления знаний может быть осуществлена путем погружения в эту базовую модель.

Важнейшими дополнительными требованиями, предъявляемыми к комплексной модели представления знаний, являются:

- гибкость, т.е. легкая возможность модификации модели;
- параллельность, асинхронизм и адекватность аппаратной поддержке в компьютерах нового поколения, ориентированных на параллельную и асинхронную переработку знаний.

Из сказанного следует, во-первых, то, что модели представления и переработки знаний удобнее всего строить как графодинамические, и, во-вторых, то, что реализовывать модели переработки знаний естественнее не на традиционных (фон-Неймановских) вычислительных машинах, а на графодинамических параллельных асинхронных машинах, которые с полным основанием можно считать вычислительными машинами нового поколения, специально ориентированными на использование в интеллектуальных системах.

12. Рассмотрены семиотические модели обработки информации, являющиеся для интеллектуальных систем важнейшим классом моделей, на основе которых, в частности, осуществляется реализация всевозможных псевдофизических логик. Поскольку основным путем реализации семиотических моделей обработки информации является их сведение к формальным моделям, необходимы языки с развитыми метаязыковыми средствами, с помощью которых можно было бы описывать правила модификации языка и правила модификации системы операций. Быть основой для таких метаязыков – это одно из основных требований, предъявляемых к базовому графовому языку SC (Semantic Code), рассматриваемому в разделе 4. Формальные модели обработки информации, осуществляющие интерпретацию семиотических моделей, являются типичным примером моделей переработки сложноструктурированных знаний, которые носят иерархический характер со сложным переплетением информации и метаинформации.

13. Наиболее перспективным классом формальных моделей обработки информации являются графодинамические параллельные асинхронные модели, так как они:

- наиболее близки к семантическому уровню рассмотрения и обработки информации;
- достаточно просто интерпретируют друг друга;
- легко интегрируются.

Соответственно этому перспективным классом языков и абстрактных машин являются графовые языки и графодинамические параллельные асинхронные абстрактные машины.

К числу графодинамических формальных моделей можно отнести также и нейросетевые модели, обработка информации в которых сводится к изменению состояния формальных нейронов и (при обучении нейронной сети) к изменению веса связей между ними.

Понятие графодинамической модели обработки информации, понятие графового языка и понятие графодинамической параллельной асинхронной абстрактной машины являются центральными понятиями всей данной работы.

1. Теоретико-множественные принципы представления фактографических знаний в памяти графодинамических ассоциативных машин

Данный раздел может быть использован в качестве учебного пособия по дисциплине «Математические основы искусственного интеллекта» для студентов специальности «Искусственный интеллект».

1.1. Базовые понятия, лежащие в основе языка SCB

Ключевые понятия

Ключевыми понятиями данного подраздела являются понятия, перечисленные в табл. 2.1.1, табл. 2.1.2 и табл. 2.1.3, а также следующие понятия: **элемент множества** (второй компонент пары принадлежности); **петля принадлежности**; **нормализованная пара принадлежности**; **классическая пара принадлежности**; **мощность множества**; **петля**.

Основными из указанных ключевых понятий являются такие понятия, как **множество**, **знак множества** и **изображение знака множества**. Далее по значимости идут такие понятия, как **пара принадлежности** и **предмет**. Пара принадлежности трактуется как **ориентированное** (упорядоченное) множество специального вида, имеющее мощность, равную двум. Понятие предмета также сводится к понятию множества – каждый предмет заменяется множеством, состоящим только из этого предмета (такое множество будем называть **предметным**). Следующими по значимости являются такие понятия, как **система множеств** и **нормализованное множество**. Система множеств так же, как и пара принадлежности, трактуется как множество частного вида и является способом формального уточнения трактовки всевозможных (в том числе и математических) структур. Нормализованные множества представляют собой множества, состоящие из знаков множеств, и являются для нас важнейшим видом множеств, к которым мы будем приводить все остальные (ненормализованные) множества за исключением предметных множеств.

Перечисленные понятия являются основой для фактографического языка SCB (Semantic Code Basic), семантика и синтаксис которого будут рассмотрены ниже. Особенностью языка SCB является то, что в его основу положен теоретико-множественный принцип представления фактографических знаний.

В табл. 2.1.1 приведена базовая для языка SCB типология **множеств** по трем следующим признакам:

- * нормализованность множеств (в таблице такие типы множеств помечены символом “ * ”),
- мощность и ориентированность (упорядоченность) множеств (в таблице такие типы множеств помечены символом “ ■ ”),
- семантика множеств (в таблице такие типы множеств помечены символом “ • ”).

Более подробная типология множеств рассматривается в разделе 3.

В табл. 2.1.2 приведена типология **знаков множеств**, в частности соответствующая рассмотренной выше типологии множеств. Аналогично в табл. 2.3.1 приведена соответствующая типология **изображений знаков множеств**.

Таблица 2.1.1. Типология множеств

Множество
* нормализованное множество
* ненормализованное множество
* почти нормализованное множество
▪ одномощное множество
▪ предметное множество (предмет)
▪ пара
▪ простая ориентированная пара
▪ пара принадлежности
▪ пара непринадлежности
▪ пара нечёткой принадлежности

Окончание табл. 2.1.1

<ul style="list-style-type: none"> ■ неориентированная пара <ul style="list-style-type: none"> ■ пара синонимии ■ пара несинонимии ■ пара нечёткой синонимии ■ тройка
<ul style="list-style-type: none"> ● пара принадлежности
<ul style="list-style-type: none"> ● узловое множество
<ul style="list-style-type: none"> ● предметное множество (предмет) <ul style="list-style-type: none"> ● узловое непредметное множество <ul style="list-style-type: none"> ● простая ориентированная узловая пара <ul style="list-style-type: none"> ● пара непринадлежности ● пара нечёткой принадлежности ● неориентированная пара <ul style="list-style-type: none"> ● пара синонимии ● пара несинонимии ● пара нечёткой синонимии ● семейство дуг принадлежности ● семейство узловых множеств <ul style="list-style-type: none"> ● семейство предметных множеств ● семейство узловых непредметных множеств ● система множеств

Таблица 2.1.2. Типология знаков множеств

знак множества
<ul style="list-style-type: none"> * знак нормализованного множества * знак ненормализованного множества <ul style="list-style-type: none"> * знак почти нормализованного множества
<ul style="list-style-type: none"> ■ знак одномощного множества <ul style="list-style-type: none"> ■ знак предметного множества (знак предмета, предметный знак) ■ знак пары <ul style="list-style-type: none"> ■ знак простой ориентированной пары (дуга) <ul style="list-style-type: none"> ■ знак пары принадлежности (дуга принадлежности) ■ знак пары непринадлежности ■ знак пары нечёткой принадлежности ■ знак неориентированной пары (ребро) <ul style="list-style-type: none"> ■ знак пары синонимии ■ знак пары несинонимии ■ знак пары нечёткой синонимии ■ знак тройки
<ul style="list-style-type: none"> ● знак пары принадлежности (дуга принадлежности) ● знак узлового множества <ul style="list-style-type: none"> ● знак предметного множества (знак предмета, предметный знак) ● знак узлового непредметного множества <ul style="list-style-type: none"> ● знак простой ориентированной узловой пары <ul style="list-style-type: none"> ● знак пары непринадлежности ● знак пары нечёткой принадлежности ● знак неориентированной пары <ul style="list-style-type: none"> ● знак пары синонимии ● знак пары несинонимии ● знак пары нечёткой синонимии

Окончание табл. 2.1.2

- знак семейства дуг принадлежности
- знак семейства узловых множеств
 - знак семейства предметных множеств
 - знак семейства узловых непредметных множеств
- знак системы множеств

Таблица 2.1.3. Типология изображений знаков множеств

изображение знака множества
* изображение знака нормализованного множества
* изображение знака ненормализованного множества <ul style="list-style-type: none"> * изображение знака почти нормализованного множества
■ изображение знака одномощного множества <ul style="list-style-type: none"> ■ изображение знака предметного множества (изображение знака предмета)
■ изображение знака пары <ul style="list-style-type: none"> ■ изображение знака простой ориентированной пары <ul style="list-style-type: none"> ■ изображение знака пары принадлежности ■ изображение знака пары непринадлежности ■ изображение знака пары нечёткой принадлежности ■ изображение знака неориентированной пары <ul style="list-style-type: none"> ■ изображение знака пары синонимии ■ изображение знака пары несинонимии ■ изображение знака пары нечёткой синонимии
■ изображение знака тройки
● изображение знака пары принадлежности
● изображение знака узлового множества <ul style="list-style-type: none"> ● изображение знака предметного множества (изображение знака предмета) ● изображение знака узлового непредметного множества <ul style="list-style-type: none"> ● изображение знака простой ориентированной узловой пары <ul style="list-style-type: none"> ● изображение знака пары непринадлежности ● изображение знака пары нечёткой принадлежности ● изображение знака неориентированной пары <ul style="list-style-type: none"> ● изображение знака пары синонимии ● изображение знака пары несинонимии ● изображение знака пары нечёткой синонимии ● изображение знака семейства дуг принадлежности ● изображение знака семейства узловых множеств <ul style="list-style-type: none"> ● изображение знака семейства предметных множеств ● изображение знака семейства узловых непредметных множеств ● изображение знака системы множеств

Пояснение 2.1.1. **Множество** – одно из фундаментальных математических понятий, относящихся к числу неопределяемых. Это понятие можно только пояснить. Приведем несколько таких пояснений.

- Множество – это некоторое количество каких-то объектов, объединяемых в одно целое. Указанные объекты считаются элементами формируемого из них множества. См. [100] (**Вilenkin Н.Я. 1969гн-Расском**).
- «Множество – это многое, мыслимое нами как единое» (Г.Кантор – основатель теории множеств).
- «Представим прозрачную непроницаемую оболочку, нечто вроде плотно закрытого прозрачного мешка. Предположим, что внутри этой оболочки находятся все элементы некоторого множества **S** и что кроме них внутри оболочки никаких других объектов не находится. Эта оболочка с объектами **x**, находящимися внутри неё, и может служить образом множества **S**, составленного из элементов **x**. Сама же эта прозрачная оболочка, охватывающая все элементы (и ничего другого

кроме них), довольно хорошо изображает тот акт объединения элементов x , в результате которого создаётся множество S » (Н.Н.Лузин) [100] ([Виленкин Н.Я.1969кн-РасскОМ](#)).

- Множество – это условная "надстройка" над некоторой группой объектов любой природы, объединяющая эту группу объектов в некоторое мысленное (абстрактное) целое. Принципы (критерии) формирования множеств, т.е. признаки принадлежности тех или иных объектов к формируемому множеству, могут быть самыми различными.

Важно подчеркнуть то, что различные множества, выделяемые (формируемые) в рамках анализируемой (описываемой) предметной области, есть нечто субъективно (мысленно) привносимое. В природе множеств нет. Мы просто искусственно пытаемся зафиксировать факт сходства или факт целостности какой-то группы объектов. При анализе одной и той же области один субъект "увидит" сходство или целостность каких-то объектов по одним признакам и сформирует одни множества, другой – другие. Следует подчеркнуть, что из любого набора объектов можно составить множество, элементами которого будут эти и только эти объекты. То есть множество – это достаточно условная, произвольная математическая структура (конструкция). Эффективность использования этой математической структуры при построении математических моделей различных предметных областей обеспечивается тем, насколько существенными являются общие свойства тех объектов, из которых составляется множество.

Существенно также подчеркнуть, что на вид объектов, являющихся элементами множеств, не накладывается никаких ограничений. Элементами множеств могут быть самые разнообразные физические (материальные) объекты, явления, процессы. "Элементами множеств могут быть буквы, атомы, числа, функции, точки, узлы и т.д. Отсюда с самого начала ясна чрезвычайная широта теории множеств и ее приложимость к очень многим областям знания – математике, механике, физике" [311] ([Лузин Н.Н.1958кн-Собрас](#)). «Сегодня мы знаем, что, логически говоря, возможно вывести почти всю современную математику из единого источника – теории множеств» [77] ([Бурбаки Н.1965кн-ТеориM](#)). Элементами одних множеств также могут быть другие множества. Это позволяет над исходными выделенными объектами той или иной области мысленно надстраивать сложную, иерархическую **систему множеств**, которая как раз и задает структуру этой области.

Множество, элементами которого являются другие множества, называют множеством множеств, или семейством множеств. В качестве синонима термина "множество" кроме термина "семейство" используются также термины: "группа", "набор", "совокупность", "класс", "тип", "род", "вид", "собрание", "ансамбль", "коллекция" и др.

Для того чтобы дать строгое определение понятия **системы множеств**, необходимо ввести понятие **знака множества** и понятие **пары принадлежности**. После введения этих понятий систему множеств можно будет трактовать как множество специального вида.

Пояснение 2.1.2. Знак множества – это некий объект, главным свойством которого является обозначать, быть представителем некоторого конкретного множества. При этом сам облик и "внутреннее" устройство знака множества может быть самым различным, т.е. является предметом индивидуального творчества автора знака, предметом его согласования с авторами других знаков и с "читателями" текстов, в которых используются эти знаки. Другими словами, понятие знака множества абстрагируется от того, как этот знак множества будет изображен (представлен, записан) в том или ином тексте. Следовательно, необходимо четко отличать **знак** какого-либо множества как абстракцию и конкретное **изображение знака**, т.е. конкретное представление (конкретное воплощение, конкретную материализацию, конкретную запись) указанного знака в некотором тексте, конкретное вхождение этого знака в текст. Таким образом, каждому знаку множества можно поставить в соответствие:

- множество, обозначаемое этим знаком (денотат указанного знака);
- множество всевозможных (!) изображений (воплощений) этого знака в различных текстах.

Пояснение 2.1.3. Изображение знака множества – это его материализованное воплощение, входящее в состав некоторого материализованного текста. При этом в одном тексте может встречаться несколько вхождений (несколько изображений) одного и того же знака. Для символьных (линейных) языков это вообще характерная ситуация, поскольку без множественного вхождения одного и того же знака в текст невозможно представить (изобразить) большое количество связей обозначаемого этим знаком объекта с другими объектами. При этом должен существовать простой способ, позволяющий для любых двух изображений знаков установить, являются они (1) изображениями одного и того же знака или (2) изображениями двух разных знаков.

К конкретным изображениям знаков в формальных языках предъявляются следующие очевидные требования:

- знаки, обозначающие разные объекты, должны легко различаться;
- сходство знаков, обозначающих один и тот же объект (т. е. сходство разных вхождений одного и того же знака в один и тот же или разные тексты), должно быть легко устанавливаемым;
- между пользователями соответствующего языка должна быть достигнута договоренность о принципах установления связи между знаком и объектом, который обозначается этим знаком, а для некоторых знаков эта связь должна быть оговорена конкретно.

Очевидно, что для каждого множества можно построить знак, обозначающий это множество. Как уже было отмечено при рассмотрении понятия множества, из любого набора объектов любой природы можно сформировать множество, элементами которого эти объекты являются, т. е. на природу и вид элементов множеств не накладывается никаких ограничений. Следовательно, элементами множеств могут быть как множества, так и знаки множеств.

Множество, каждый элемент которого является знаком множества, будем называть **нормализованным множеством**. Понятие нормализованного множества имеет важное значение, т. к. нормализованное множество достаточно хорошо "подготовлено" к его изображению или описанию в виде текста того или иного языка. Такая "подготовленность" обусловлена тем, что в нормализованном множестве все его элементы являются знаками, которые вместе со знаком самого этого множества могут быть изображены произвольным образом в соответствии с требованиями любого языка. **Ненормализованное множество** – это множество, среди элементов которого имеется по крайней мере один объект, не являющийся знаком множества.

Пара принадлежности трактуется как множество частного вида, состоящее либо из двух элементов, один из которых считается знаком некоторого множества, а второй – одним из элементов указанного множества, либо из двух вхождений одного и того же элемента, который трактуется как знак некоторого множества, включающего в качестве одного из своих элементов знак самого себя. Пару принадлежности второго вида будем называть **петлей принадлежности**.

Итак, **пара принадлежности** – это связь двух (возможно совпадающих) объектов, один из которых (условно назовем его первым компонентом пары) в рамках этой пары играет роль знака некоторого множества, а второй (второй компонент пары) – роль непосредственно одного из элементов указанного выше множества (т. е. множества, обозначаемого первым компонентом пары принадлежности).

Объект, являющийся первым компонентом некоторой пары принадлежности, может быть вторым компонентом в рамках другой пары принадлежности. Аналогично этому объект, являющийся вторым компонентом какой-либо пары принадлежности, в рамках другой пары принадлежности может быть первым компонентом.

Пусть дана пара принадлежности (s, e) ,
где

s – первый компонент пары принадлежности (знак некоторого множества);

e – второй компонент пары принадлежности (непосредственно сам один из элементов множества, обозначаемого знаком s).

Будем при этом говорить, что указанная пара принадлежности:

- соединяет объект s (каковым может быть только знак некоторого множества) с объектом e (каковым может быть все что угодно);
- инцидентна знаку s и объекту e ;
- инцидентна слева знаку s и инцидентна справа объекту e ;
- проведена из знака s в объект e .

Пару принадлежности, вторым компонентом которой является знак множества, будем называть **нормализованной парой принадлежности**, в противном случае – **ненормализованной парой принадлежности**. Очевидно, что нормализованная пара принадлежности есть пара принадлежности, являющаяся нормализованным множеством, т. к. первым компонентом пары принадлежности всегда является знак множества.

Следует чётко отличать семантику введённых нами пар принадлежности, связывающих знак множества с объектом, который непосредственно сам является одним из элементов этого множества, от семантики классических пар принадлежности, связывающих знак множества со знаком объекта, являющегося одним из элементов этого множества. Очевидно, что указанные пары принадлежности

второго (классического) вида не относятся к числу введенных нами пар принадлежности, поскольку вторым компонентом этих пар является не сам элемент множества, обозначаемого первым компонентом, а знак этого элемента. Заметим при этом, что математическая запись вида s e представляет собой запись не пары принадлежности, а запись соответствующей классической пары принадлежности (поэтому такие пары принадлежности и называются классическими). Поскольку элементом множества может быть знак какого-либо объекта, вторым компонентом пары принадлежности (неклассической) может быть знак. Такие пары принадлежности мы называем нормализованными. Нормализованные пары принадлежности очень легко спутать с классическими парами принадлежности, в каждой из которых вторым компонентом всегда является знак некоторого объекта. Таким образом, следует чётко отличать нормализованную пару принадлежности, которая имеет вид

(знак множества — знак, являющийся элементом указанного множества)

и классическую пару принадлежности вида

(знак множества — знак объекта, являющегося элементом указанного множества).

Завершая рассмотрение понятия пары принадлежности, отметим следующее. Поскольку каждая пара принадлежности является множеством (специального вида) и поскольку для каждого множества можно построить (тем или иным способом) знак, обозначающий это множество, то соответственно этому и для каждой пары принадлежности можно построить знак, обозначающий эту пару принадлежности. Таким образом, мы ввели понятие знака пары принадлежности. Знак пары принадлежности, как и знак любого другого множества, может быть элементом какого-либо множества, т.е. может быть вторым компонентом какой-либо пары принадлежности.

Узловое множество — это множество, не являющееся парой принадлежности. Узловое множество может быть нормализованным узловым множеством (т. е. состоящим только из знаков множеств) и ненормализованным узловым множеством. Узловое множество, как и любое другое множество, может иметь знак, который обозначает это узловое множество.

Элементом ненормализованного множества может быть либо некоторое множество, либо знак некоторого множества, либо объект, не являющийся ни множеством, ни знаком множества. Объект, который не является ни множеством, ни знаком множества, будем называть **предметом**. Чаще всего предметы — это материальные объекты (физические объекты, процессы, явления). Каждому предмету, как и каждому множеству, можно поставить в соответствие некоторый знак. Такие знаки будем называть **предметными знаками**. Существенно при этом подчеркнуть, что предметный знак мы будем, строго говоря, трактовать не как знак соответствующего предмета, а как знак множества, состоящего из одного и однократно входящего в его состав элемента, каковым является указанный предмет. Такое специфическое множество будем называть **предметным множеством** и соответственно предметный знак будем иногда называть знаком предметного множества.

Из сказанного следует, что каждый рассматриваемый нами предмет мы фактически заменяем соответствующим ему предметным множеством и тем самым сводим понятие предмета к понятию множества. Очевидно, что все предметные множества являются ненормализованными. Очевидно также, что пара принадлежности, связывающая знак предметного множества (предметный знак) с элементом этого множества, также является ненормализованной.

Поскольку предметное множество не является парой принадлежности, оно является частным видом узлового множества. Частным видом узлового множества является также **узловое непредметное множество**, т. е. узловое множество, не являющееся предметным. Узловое непредметное множество может быть как нормализованным, так и ненормализованным, и так же, как и все множества, может иметь знак.

Мощность множества — это суммарное количество вхождений в это множество всех его элементов. Таким образом, каждый элемент множества может входить в это множество однократно (т. е. один раз), двукратно (два раза), трехкратно (три раза) и т. д. Мощность множества определяется общим количеством пар принадлежности, выходящих из знака этого множества. Мощность нормализованного множества определяется общим количеством нормализованных пар принадлежности, выходящих из знака этого множества.

Семейство всевозможных множеств в соответствии со значением их мощности разбивается на следующие классы:

- **пустые множества** (0-мощные множества) — множества, не имеющие элементов;

- **одномощные множества** – множества, имеющие мощность, равную единице (это 1-элементные множества с однократным вхождением этого единственного элемента);
- **пары** (2-мощные множества, би-мощные множества) – множества, имеющие мощность, равную двум (это либо 1-элементные множества с двукратным вхождением этого единственного элемента, либо двухэлементные множества с однократным вхождением каждого элемента), в частности:
 - **петли** – 1-элементные множества с двукратным вхождением этого единственного элемента;
- **тройки** (3-мощные множества);
- и т. д.

Простую ориентированную пару (с неуточняемой семантикой) будем трактовать как ориентированное (упорядоченное) множество, состоящее:

- либо из двух элементов, каждый из которых однократно входит в состав этого множества и при этом один из них считается первым элементом (первым компонентом) указанного множества, а другой считается вторым его элементом;
- либо из одного элемента, но двукратно входящего в состав определяемого множества, при этом одно вхождение рассматриваемого элемента считается первым вхождением, а другое – вторым вхождением.

Простую ориентированную пару второго вида будем называть простой ориентированной петлей. Как и любому другому множеству, ориентированной паре можно поставить в соответствие знак этой пары. Знаки простых ориентированных пар будем называть **дугами**.

Простая ориентированная узловая пара – эта простая ориентированная пара, являющаяся узловым множеством, т.е. пара, которая не является парой принадлежности.

К числу простых ориентированных пар относятся рассмотренные выше пары принадлежности, а также рассматриваемые ниже пары непринадлежности и пары нечёткой принадлежности. **Пара непринадлежности** – это ориентированная пара, второй компонент которой не является элементом множества, обозначенного первым её компонентом. **Пара нечёткой принадлежности** – это ориентированная пара, о которой достоверно не известно, является она парой принадлежности или нет.

Подчеркнем то, что пары непринадлежности и пары нечеткой принадлежности являются не только частным видом простых ориентированных пар, но и частным видом узловых непредметных множеств. Кроме того, частным видом узловых непредметных множеств являются также **неориентированные пары**. Каждая неориентированная (неупорядоченная) пара есть множество, состоящее либо из двух однократно входящих в его состав элементов, либо из одного элемента, двукратно входящего в его состав. Частными видами неориентированных пар являются: **пары синонимии**, связывающие семантически эквивалентные (синонимичные) знаки; **пары несинонимии**, представляющие собой неориентированные пары, не являющиеся парами синонимии; **пары нечёткой синонимии**, представляющие собой неориентированные пары, о которых достоверно не известно, являются они парами синонимии или нет, а также **пары омонимии**, связывающие семантически омонимичные знаки. Знаки неориентированных пар будем называть **ребрами** (например, ребрами синонимии, ребрами несинонимии, ребрами нечёткой синонимии, ребрами омонимии).

Заметим, что введённые нами типы простых ориентированных пар являются частным видом классических кортежей, которые представляют собой ориентированные (упорядоченные) множества произвольной мощности, и которые подробно нами рассматриваются в подразделе 3.1. Каждому из введенных нами типов пар ставится в соответствие некоторое бинарное отношение (см. пункт 3.3.8):

- отношение принадлежности;
- отношение непринадлежности;
- отношение нечёткой принадлежности;
- отношение синонимии;
- отношение несинонимии;
- отношение нечёткой синонимии;
- **отношение омонимии**.

Каждое бинарное отношение трактуется как множество, каждым элементом которого является знак некоторой пары. Подробно бинарные отношения рассматриваются в пункте 3.3.8.

Продолжим рассмотрение типов множеств, перечисленных в табл. 2.1.1.

Семейство пар принадлежности – это множество, каждым элементом которого является дуга принадлежности, т. е. знак некоторой пары принадлежности. Очевидно, что введенное выше отношение принадлежности является одним из семейств пар принадлежностей, так как представляет собой семейство всевозможных пар принадлежностей.

Семейство узловых множеств – это множество, каждым элементом которого является знак некоторого множества.

Семейство предметных множеств – это множество, каждым элементом которого является знак некоторого предметного множества (т. е. знак некоторого предмета).

Семейство узловых непредметных множеств – это множество, каждым элементом которого является знак некоторого узлового непредметного множества.

Вернемся к введенному выше понятию системы множеств. Приведём теоретико-множественную трактовку этого понятия. **Система множеств** – это множество, состоящее из некоторого количества узловых множеств или их знаков, а также пар принадлежности (или знаков этих пар), которые связывают между собой указанные выше узловые множества и/ или их знаки. Каждая пара принадлежности, являющаяся элементом системы множеств, или пары принадлежности, знак которой является элементом системы множеств, непосредственно связывает следующие элементы системы множеств. Первым компонентом таких пар является знак некоторого узлового множества. Вторым компонентом таких пар может быть множество любого вида или его знак, т. е. это может быть:

- пара принадлежности или знак пары принадлежности;
- предмет, предметное множество или знак предметного множества;
- узловое непредметное множество или его знак.

Теперь поставим перед собой задачу нормализации системы множеств, т. е. задачу приведения системы множеств к некоторому каноническому виду, от которого достаточно легко можно перейти к тексту, изображающему (представляющему, записывающему) эту систему множеств. Рассмотрим произвольную систему множеств. Для каждой пары принадлежности, входящей в состав этой системы множеств, построим знак этой пары. Для каждого узлового множества, входящего в состав этой системы множеств, построим знак этого множества, а также построим пары принадлежности, связывающие этот знак со всеми элементами обозначаемого им множества. При этом, если какое-либо множество *si*, входящее в рассматриваемую систему множеств, имеет в качестве одного из своих элементов некоторое другое множество *sj*, каковым, в частности, может быть и пара принадлежности, то множество *si* преобразуем в *si**, отличающееся от исходного тем, что в нем элементом является не само множество *sj*, а знак этого множества. Далее для каждого предмета, входящего в состав рассматриваемой системы множеств, построим знак соответствующего предметного множества. При этом, если какое-либо множество *sm*, входящее в рассматриваемую систему множеств, имеет в качестве одного из своих элементов некоторый предмет *sk*, то множество *sm* преобразуем во множество *sm**, отличающееся от исходного тем, что в нем элементом является не сам предмет *sk*, а знак предметного множества, соответствующего этому предмету, т. е. множества, единственным элементом которого является указанный предмет.

Если в результате рассмотренных выше преобразований исходной системы множеств:

- все множества, входящие в исходную систему множеств, будут иметь соответствующие им знаки;
- все построенные знаки множеств будут связаны парами принадлежности со всеми элементами соответствующих им множеств;
- все пары принадлежности вида
(множество sj является элементом множества si)
будут заменены парами принадлежности вида
(знак множества sj является элементом множества si*);
- все предметы, входящие в исходную систему множеств, будут иметь соответствующие предметные знаки;
- все пары принадлежности вида
(предмет sk является элементом множества sm), за исключением пар, в которых множество *sm* является предметным множеством (т. е. унарным множеством, которое состоит только из соответствующего предмета), будут заменены парами вида
(знак предметного множества, соответствующего предмету sk,
является элементом множества sm*),

то полученную систему множеств будем называть **нормализованной системой множеств**, или почти нормализованным множеством.

Множество s является **почти нормализованным множеством** в том и только в том случае, если:

- множество s является ненормализованным;
- не существует ни одного элемента множества s , который является множеством (т. е. элементами множества s могут быть только знаки множеств и предметы);
- каждый элемент множества s , являющийся знаком ненормализованного множества, представляет собой знак пары принадлежности, соединяющей знак предметного множества с соответствующим предметом (т. е. других ненормализованных множеств в составе множества s нет);
- для каждого элемента множества s , являющегося предметом, существует пара принадлежности, соединяющая указанный предмет со знаком соответствующего предметного множества, причем знак указанной пары принадлежности, а также знак указанного предметного множества, являются элементами множества s .

Очевидно, что каждую систему множеств можно привести к нормализованному виду. Переход от ненормализованной к нормализованной системе множеств дает возможность строго трактовать систему множеств как множество специального вида, а именно как множество, состоящее из следующих элементов:

- некоторого количества знаков узловых непредметных нормализованных множеств;
- некоторого количества предметных знаков;
- некоторого количества знаков нормализованных пар принадлежности;
- некоторого количества предметов;
- некоторого количества ненормализованных пар принадлежности, связывающих предметные знаки с соответствующими им предметами.

Подчеркнем, что все множества, знаки которых входят в состав нормализованной системы множеств, кроме предметных множеств и пар принадлежности, связывающих предметные знаки с предметами, являются нормализованными множествами.

Поскольку в состав нормализованной системы множеств могут входить предметы, нормализованная система множеств в общем случае не является нормализованным множеством. Если из числа элементов нормализованной системы множеств исключить предметы, а также знаки пар принадлежности, вторыми компонентами которых являются предметы, то мы получим нормализованную систему множеств, которая также является и нормализованным множеством. Такую систему множеств будем называть **полностью нормализованной системой множеств**. Таким образом, полностью нормализованная система множеств – это такая система множеств, все элементы которой являются знаками множеств (либо предметных множеств, либо узловых непредметных нормализованных множеств, либо нормализованных пар принадлежности).

Полностью нормализованную систему множеств мы будем рассматривать как семантическую структуру текста, который является изображением (представлением, записью) структуры той или иной описываемой области (описываемого мира).

Очевидно, что полностью нормализованную систему множеств можно изобразить (представить, записать) в виде текста некоторого языка. Создание такого языка предполагает:

- выработку условных правил изображения (построения) знаков множеств, не являющихся парами принадлежности;
- выработку условных правил изображения знаков пар принадлежности (явным или неявным образом);
- выработку правил изображения (записи) самих пар принадлежности. Для каждой пары принадлежности, входящей в изображаемую нормализованную систему множеств, должны быть указаны:
 - знак этой пары принадлежности;
 - знак некоторого узлового множества;
 - объект, являющийся непосредственно одним из элементов указанного узлового множества.

Очевидно, выработка правил изображения знаков множеств и, в частности, знаков пар принадлежности не составляет большого труда, т. к. на формирование любых знаков никаких принципиальных ограничений не накладывается – эти ограничения определяются исключительно условностями того или иного языка. А вот при изображении пары принадлежности может возникнуть

проблема, если вторым компонентом пары принадлежности (элементом множества) является не знак какого-либо множества, а предмет. Дело в том, что знак вполне может быть фрагментом текста, тогда как предмет для любого текста всегда есть нечто внешнее и соответственно не может непосредственным образом входить в состав текста.

Следовательно, для нормализованной системы множеств мы можем изобразить (записать) все пары принадлежности, кроме ненормализованных, т. е. кроме тех, которые связывают предметные знаки с предметами, соответствующими этим знакам. Но это совершенно нормальное явление, когда связь между знаками предметов и обозначаемыми предметами из "внешнего" описываемого мира осуществляется дополнительными "внезыковыми" средствами.

Итак, нормализованная система множеств в отличие от полностью нормализованной системы множеств в полном объеме не может быть изображена в виде текста из-за ненормализованных пар принадлежности. Таким образом, изобразить (представить, записать) в виде некоторого текста можно только полностью нормализованную систему множеств.

При этом существенно подчеркнуть то, что в силу внезыкового характера связи между предметными знаками и соответствующими им предметами переход от нормализованной системы множеств к полностью нормализованной системе множеств путем исключения предметов и знаков инцидентных им пар принадлежности из состава нормализованной системы множеств не приводит к потере информации с точки зрения текста, изображающего систему множеств.

Тот факт, что в число элементов множества, представляющего собой полностью нормализованную систему множеств, мы включаем не сами пары принадлежности, а их знаки, дает нам возможность достаточно легко расширять полностью нормализованную систему множеств, добавляя в неё знак самой этой системы множеств, знак различных фрагментов этой системы множеств и соответственно этому знаки пар принадлежности, связывающих добавленные знаки с их элементами.

Упражнения к подразделу 2.1.

Упражнение 2.1.1. Пусть дано некоторое множество, знак этого множества, некоторое изображение знака указанного множества, множество всевозможных изображений знака указанного множества. Что значит задать множество и как задание множества может выглядеть в виде текста формального языка? Предложите несколько вариантов.

Упражнение 2.1.2. Какие ограничения накладываются на элементы множеств? Из каких объектов можно построить множество?

Упражнение 2.1.3. Какие ограничения накладываются на изображения знаков множеств? Любой ли объект можно использовать в качестве изображения знака какого-либо множества?

Упражнение 2.1.4. Какими преимуществами обладают нормализованные множества?

Упражнение 2.1.5. Можно ли изобразить ненормализованную пару принадлежности в виде текста формального языка?

Упражнение 2.1.6. Все ли пары синонимии являются нормализованными множествами?

Упражнение 2.1.7. Чем отличаются пары принадлежности, используемые в языке SCB, от классической трактовки пар принадлежности?

Упражнение 2.1.8. Является ли пара принадлежности множеством?

1.2. Основные положения языка SCB (Semantic Code Basic)

Ключевые понятия:

SCB; тройка принадлежности; дуга принадлежности; scb-узел; предметный scb-узел; непредметный scb-узел; scb-узел неопределенного типа; scb-текст; scb-элемент; scb-элемент неопределенного типа; инцидентность scb-элементов; смежность scb-элементов; петля принадлежности; кратные дуги принадлежности; встречные дуги принадлежности; содержимое scb-узла; scb-узел с содержимым.

Язык **SCB** (Semantic Code Basic) – это фактографический язык, обеспечивающий представление (изображение и запись) всевозможных математических структур путем трактовки каждой такой структуры как полностью нормализованной системы множеств. Каждая полностью нормализованная система множеств однозначно задается множеством троек принадлежности, которые имеют следующий вид: (v, e, g) ,

где

v – знак нормализованного множества, не являющегося парой принадлежности (знак узлового непредметного множества);

e – один из элементов множества **v**, каковым может быть только знак некоторого множества, поскольку множество **v** является нормализованным;

g – знак нормализованной пары принадлежности, проведенной из знака **v** в знак **e**.

Знак нормализованной пары принадлежности в языке SCB будем называть **дугой принадлежности**.

Знак узлового множества, т. е. множества, не являющегося парой принадлежности, в языке SCB будем называть **scb-узлом**. При этом будем отличать **предметные scb-узлы**, которые являются предметными знаками (знаками предметных множеств), и **непредметные scb-узлы**, которые являются знаками нормализованных узловых непредметных множеств, а также scb-узлы неопределенного (неуточняемого, неустановленного, неизвестного типа), принадлежность которых к классу предметных scb-узлов или к классу непредметных scb-узлов в текущий момент не установлена.

Тексты языка SCB будем называть **scb-текстами**, или scb-конструкциями.

Элементарные фрагменты **scb-текста** будем называть **scb-элементами**. К числу scb-элементов относятся **дуги принадлежности** и **scb-узлы** (как предметные узлы, так и непредметные узлы), а также scb-элементы неопределенного типа (неуточняемого, неустановленного, неизвестного типа), принадлежность которых к классу дуг принадлежности или к классу scb-узлов не установлена. Никаких других scb-элементов не существует. Каждый scb-элемент является "синтаксически" элементарным (поскольку его "внутренняя" структура в языке SCB не требует уточнения), а также семантически значимым (поскольку каждый scb-элемент представляет собой знак некоторого множества).

Узловые непредметные множества в языке SCB могут состоять из знаков множеств того или иного типа. Соответственно этому можно говорить о типологии знаков узловых непредметных множеств, а следовательно, и о типологии **непредметных scb-узлов**. Согласно этому среди непредметных scb-узлов можно выделить:

- непредметные scb-узлы, являющиеся знаками различных множеств, состоящих только из знаков пар принадлежности;
- непредметные scb-узлы, являющиеся знаками различных множеств, состоящих только из знаков узловых множеств (т. е. множеств, не являющихся парами принадлежности), и в частности,
 - непредметные scb-узлы, являющиеся знаками различных множеств, состоящих только из знаков предметных множеств (т. е. из предметных знаков);
 - непредметные scb-узлы, являющиеся знаками различных множеств, состоящих только из знаков узловых непредметных множеств (т. е. множеств, не являющихся парами принадлежности и не являющихся предметными множествами);
- непредметные scb-узлы, являющиеся знаками различных систем множеств. Напомним, что каждая система множеств есть множество, в состав которого входят как знаки узловых множеств, так и знаки пар принадлежности.

На основании введённых понятий языка SCB тройка принадлежности (v, e, g) будет трактоваться следующим образом:

v – некоторый непредметный scb-узел;

e – знак некоторого множества, который представляется в виде некоторого scb-элемента и который является одним из элементов множества, обозначаемого узлом **v**. Подчеркнем, что scb-элемент **e** может быть как scb-узлом, так и дугой принадлежности;

g – дуга принадлежности, проведенная из scb-узла **v** в scb-элемент **e**.

При изображении рассматриваемой тройки принадлежности на языке SCB будем также говорить, что:

- дуга принадлежности **g** выходит из узла **v** и входит в scb-элемент **e**;
- дуга **g** инцидентна справа узлу **v** (т. е. является непосредственно правым соседом узла **v**) и инцидентна слева scb-элементу **e** (т. е. является непосредственно левым соседом scb-элемента **e**);

- узел v инцидентен слева дуге принадлежности g ;
- scb-элемент e инцидентен справа дуге принадлежности g ;
- узел v и дуга принадлежности g инцидентны друг другу, т. к. непосредственно соседствуют;
- scb-элемент e и дуга принадлежности g инцидентны друг другу;
- узел v и scb-элемент e смежны друг другу, т. е. соединены дугой принадлежности.

Подчеркнем, что при представлении тройки принадлежности (v, e, g) в языке SCB не изображение scb-элемента e считается элементом множества v , а сам знак, которым является указанный scb-элемент e . Таким образом, следует четко отличать сам знак (как некую абстракцию) от изображения (представления, вхождения) этого знака в том или ином тексте (в частности, в scb-тексте). Так, например, один и тот же знак в рамках одного и того же текста (информационной конструкции) может быть изображен несколько раз, т. е. может иметь несколько вхождений. Это означает, что два разных (похожих или совсем не похожих) изображения могут быть изображениями одного и того же знака. Такие изображения будем называть синонимичными.

Уже на данном этапе рассмотрения языка SCB можно сформулировать некоторые свойства синтаксически и семантически корректных scb-текстов.

Утверждение 2.2.1 (свойство SCB-1). Дуга принадлежности не может выходить из дуги принадлежности, т.к. в языке SCB связь знака пары принадлежности с элементами этой пары задается не другими парами принадлежности, а связью инцидентности. Из данного свойства следует, что дуга принадлежности не может выходить из самой себя.

Утверждение 2.2.2 (свойство SCB-2). Дуга принадлежности не может выходить из предметного scb-узла.

Утверждение 2.2.3 (свойство SCB-3). Если дуга принадлежности выходит из scb-узла неопределенного типа, то этот узел следует трактовать как непредметный, преобразовав тип этого узла соответствующим образом.

Утверждение 2.2.4 (свойство SCB-4). Если дуга принадлежности выходит из scb-элемента неопределенного типа, то этот scb-элемент следует трактовать как непредметный узел, преобразовав соответствующим образом тип этого scb-элемента.

Утверждение 2.2.5 (свойство SCB-5). Дуга принадлежности не может входить в саму себя.

Примечание. Из 5-го и 1-го свойств следует, что дуга принадлежности не может быть инцидентна самой себе. Таким образом, третий компонент тройки принадлежности (g) не может совпадать ни с первым компонентом (т. к. узел не может совпадать с дугой принадлежности), ни со вторым компонентом. То есть дуга принадлежности не может входить сама в себя и не может выходить сама из себя.

Утверждение 2.2.6 (свойство SCB-6). Дуга принадлежности может входить в scb-элемент любого типа (в другую дугу принадлежности, в предметный узел, в непредметный узел, в узел неопределенного типа, в scb-элемент неопределенного типа). При этом удаление или добавление дуги принадлежности, входящей в scb-элемент, не меняет семантики этого scb-элемента.

Утверждение 2.2.7 (свойство SCB-7). Из двух инцидентных scb-элементов один обязательно должен быть дугой принадлежности – либо дугой, выходящей из инцидентного ей scb-элемента, либо дугой, входящей в этот элемент. Следовательно, два разных scb-узла не могут быть инцидентны друг другу.

Примечание. Из свойств SCB-1, SCB-5 и SCB-7 следует, что scb-элемент любого типа не может быть инцидентен сам себе.

Утверждение 2.2.8 (свойство SCB-8). Пусть scb-элемент ei инцидентен слева (является левым соседом) по отношению к scb-элементу ej и пусть здесь ej является дугой принадлежности. Тогда ei является непредметным узлом и соответственно не может быть дугой принадлежности. В этом случае будем говорить, что дуга ej выходит из узла ei .

Утверждение 2.2.9 (свойство SCB-9). Пусть scb-элемент *ei* инцидентен слева scb-элементу *ej* и пусть здесь *ei* является дугой принадлежности. Тогда *ej* может быть scb-элементом любого типа (как узлом, так и дугой принадлежности). В этом случае будем говорить, что дуга *ei* входит в элемент *ej*.

Утверждение 2.2.10 (свойство SCB-10). SCB-элемент, из которого дуга принадлежности выходит, и scb-элемент, в который дуга принадлежности входит, могут совпадать. Такую дугу принадлежности будем называть **петлей принадлежности**.

Примечание. Таким образом, первые два компонента тройки принадлежности могут совпадать. Это будет означать, что множество *v* в качестве одного из своих элементов имеет знак самого себя. Очевидно, что эта ситуация существенно отличается от ситуации, когда множество *v* в качестве одного из своих элементов имеет не свой знак, а само себя. Правда, такая ситуация в нормализованных системах множеств невозможна.

Утверждение 2.2.11 (свойство SCB-11). В scb-конструкциях могут встречаться **кратные дуги принадлежности**, т.е. дуги принадлежности, которые выходят из одного и того же scb-элемента и входят в другой, но совпадающий для этих дуг scb-элемент.

Утверждение 2.2.12 (свойство SCB-12). В scb-конструкциях могут существовать **встречные дуги принадлежности**.

Утверждение 2.2.13 (свойство SCB-13). Для каждой дуги принадлежности существует один и только один узел, являющийся по отношению к этой дуге инцидентным слева, а также один и только один scb-элемент, являющийся по отношению к этой дуге инцидентным справа. То есть каждая дуга принадлежности выходит из одного узла и входит только в один scb-элемент. Это свойство следует из того, что дуга принадлежности является знаком пары принадлежности, т.е. знаком множества, состоящего только из двух элементов – из элемента, инцидентного слева от этой дуги принадлежности, и элемента, инцидентного справа от неё.

Любую дискретную информационную конструкцию можно закодировать (представить, изобразить) в виде scb-текста, в котором предметными узлами являются знаки элементарных (атомарных, неделимых) фрагментов (символов) кодируемой информационной конструкции.

Таким образом, все дискретные информационные **конструкции** (см. раздел 1) и соответствующие им языки можно свести к языку SCB. Но несмотря на такую возможность, далеко не всегда это целесообразно, поскольку привычные способы кодирования информационных конструкций могут оказаться нагляднее и могут иметь развитые средства обработки. Таким образом, одной и той же информационной конструкции может соответствовать несколько узлов, соответствующих разным способам кодирования этой информационной конструкции. Подчеркнем, что указанные узлы не являются синонимичными, т.к. формально они обозначают разные объекты. Таким образом, следует четко отличать синонимию самих узлов и синонимию различных информационных конструкций. В языке SCB синонимичные узлы отождествляются (логически склеиваются).

Если имеется scb-узел, обозначающий некоторую информационную **конструкцию**, не принадлежащую языку SCB, то указанную информационную конструкцию будем называть **содержимым** указанного узла. Таким образом, можно ввести новый тип scb-узлов – узлы, у которых существует содержимое, т.е. узлы, являющиеся знаками "внешних" текстов. В частности, можно говорить об узлах, содержимое которых известно (сформировано). Содержимое могут иметь только предметные scb-узлы. Отсюда следует 14-е свойство scb-текста.

Утверждение 2.2.14 (свойство SCB-14). Если некоторый scb-узел имеет содержимое, то его следует трактовать как предметный scb-узел.

Упражнения к подразделу 2.2.

Упражнение 2.2.1. Могут ли быть инцидентными две разные дуги принадлежности?

Упражнение 2.2.2. Почему текст языка SCB трактуется как множество троек принадлежности и почему его нельзя трактовать как множество пар принадлежности?

1.3. Ядро языка SCBg (Semantic Code Basic graphical) – графической модификации языка SCB

Ключевые понятия: графический язык; SCBg; графический примитив языка SCBg; scbg-изображение scb-элемента неопределённого типа; scbg-изображение дуги принадлежности; scbg-изображение scb-узла неопределённого типа; scbg-изображение предметного scb-узла; scbg-изображение непредметного scb-узла неопределённого типа; scbg-изображение дуги неопределённого типа (ориентированный связующий элемент графового текста – линия со стрелкой); scbg-изображение дуги непринадлежности; scbg-изображение дуги нечёткой принадлежности; scbg-изображение ребра неопределённого типа (неориентированный связующий элемент графового текста – линия без стрелок); scbg-изображение ребра синонимии scbg-элементов; scbg-изображение ребра несинонимии scbg-элементов; scbg-изображение ребра нечёткой синонимии scbg-элементов; scbg-изображение знака семейства дуг принадлежности; scbg-изображение знака семейства scb-узлов неуточняемого типа; scbg-изображение знака семейства предметных scb-узлов; scbg-изображение знака семейства непредметных scb-узлов; scbg-изображение знака системы множеств.

В самом языке SCB способ изображения scb-элементов при представлении (записи) троек принадлежности не уточняется. Всё это уточняется в рамках различных модификаций (вариантов) языка SCB, которые являются подъязыками языка SCB. В частности, в данной книге рассматриваются две модификации языка SCB:

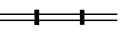
- графическая модификация, которой соответствует графический язык SCBg – Semantic Code Basic graphical. Тексты этого языка будем называть scbg-текстами. Используемые в языке SCBg графические примитивы приведены в табл. 2.3.1;
- линейная модификация, которой соответствует линейный язык SCBs – Semantic Code Basic string. Тексты этого языка будем называть scbs-текстами.

Таблица 2.3.1. Алфавит графических примитивов, используемых для изображения основных типов scb-элементов в графическом языке SCBg (см. табл. 2.1.2)

Наименование	Изображение
изображение знака множества (изображение scb-элемента неуточняемого типа)	•
• изображение знака пары принадлежности (изображение дуги принадлежности)	→
• изображение знака узлового множества (изображение узла неуточняемого типа)	○
• изображение знака предметного множества (изображение предметного узла)	●
• изображение знака узлового непредметного множества (изображение непредметного узла)	◎
• изображение знака простой ориентированной узловой пары (изображение дуги неуточняемого типа)	↔→
• изображение знака пары непринадлежности (изображение дуги непринадлежности)	→↔
• изображение знака пары нечёткой принадлежности (изображение дуги нечеткой принадлежности)	↔→

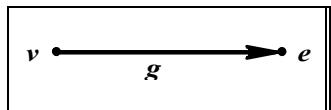
Окончание табл. 2.3.1

Наименование	Изображение
• изображение знака неориентированной пары (изображение ребра неуточняемого типа)	↔↔

• изображение знака пары синонимии (изображение ребра синонимии)	
• изображение знака пары несинонимии (изображение ребра несинонимии)	
• изображение знака пары нечёткой синонимии (изображение ребра нечеткой синонимии)	
• изображение знака семейства дуг принадлежности	
• изображение знака семейства узловых множеств	
• изображение знака семейства предметных множеств	
• изображение знака семейства узловых непредметных множеств	
• изображение знака системы множеств	

В языке SCBg scb-элементы неуточняемого типа изображаются в виде закрашенного маленького кружка, узлы – в виде большого кружка, а дуги принадлежности – отрезком линии со стрелкой на одном из его концов, указывающей направление scb-дуги. В кружок, изображающий узел, могут быть помещены условные обозначения типа узла (вертикальные, горизонтальные, наклонные и прочие черточки в различных комбинациях, а также точки, штриховки и прочее). Однако любой scb-текст может быть представлен на основе только двух изображений scb-элементов: изображения scb-элемента неуточняемого типа и изображения scb-дуги (см. scb-текст 2.3.1).

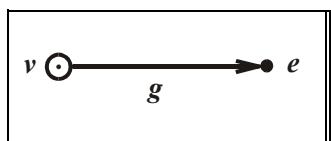
SCBg-текст 2.3.1. Базовая scb-конструкция для изображения тройки принадлежности



Здесь тип scb-элементов с идентификаторами **v** и **g** может быть уточнен с помощью соответствующих графических примитивов (см. scbg-тексты 2.3.2 – 2.3.6)

На scbg-текстах 2.3.2 – 2.3.6 рассмотрены пять вариантов представления (изображения) троек принадлежности в языке SCBg, соответствующих первым пятью типам scb-элементов, указанных в табл. 2.3.1.

SCBg-текст 2.3.2. Вариант 1 для изображения тройки принадлежности

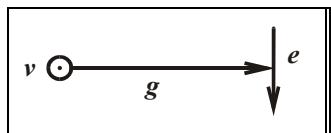


Здесь первый (левый) кружок с точкой изображает непредметный узел неуточняемого типа (знак узлового непредметного множества), являющийся первым компонентом (**v**) изображаемой тройки принадлежности.

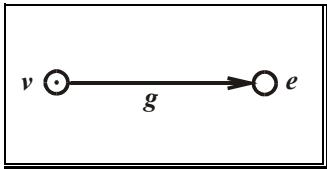
Маленьким кружком изображается знак множества неуточняемого типа, т.е. элемент scb-текста неуточняемого типа (либо дуга принадлежности, либо узел), если, конечно, scb-элемент **e** **таковым является**.

Дуга принадлежности **g** в языке SCBg изображается в виде линии со стрелкой. При этом указанная линия может иметь любую форму, но без пересечений. Пересекаться могут только линии, изображающие различные дуги принадлежности.

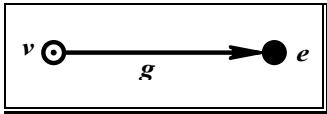
SCBg-текст 2.3.3. Вариант 2 для изображения тройки принадлежности



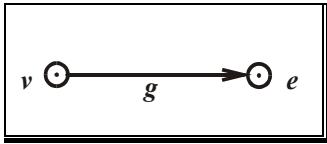
Этот вариант имеет место, если scb-элемент **e** является знаком некоторой пары принадлежности.

SCBg-текст 2.3.4. Вариант 3 для изображения тройки принадлежности

Незаштрихованный кружок изображает узел неуточняемого типа, являющийся знаком узлового множества неуточняемого типа (либо знаком предметного множества, либо знаком узлового непредметного множества), если, конечно, *e* таковым знаком является.

SCBg-текст 2.3.5. Вариант 4 для изображения тройки принадлежности

Заштрихованный кружок изображает узел, являющийся предметным знаком предметного множества, если *e* таковым знаком является.

SCBg-текст 2.3.6. Вариант 5 для изображения тройки принадлежности

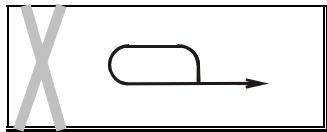
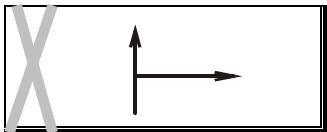
В данном варианте изображения тройки принадлежности второй компонент изображаемой тройки принадлежности (*e*) также является непредметным узлом неуточняемого типа. Поэтому он также изображается в виде кружка с точкой.

На самом деле количество вариантов изображения троек принадлежностей намного больше, т.к. второй компонент тройки принадлежности (элемент *e*) может быть элементом любого типа и соответственно может быть изображён с помощью любого графического примитива, приведённого в табл. 2.3.1.

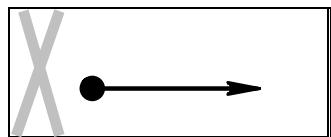
Текст (конструкция) языка SCBg представляет собой совокупность изображений троек принадлежности с физическим склеиванием синонимичных scbg-элементов, обозначающих одно и то же множество, т.е. являющихся изображениями одного и того же scb-элемента.

Проиллюстрируем на языке SCBg перечисленные в подразделе 2.2 общие свойства языка SCB, являющегося надъязыком по отношению к языку SCBg. Здесь для наглядности сохранена нумерация, принятая при перечислении свойств ядра языка SCB (см. утверждения 2.2.1 – 2.2.14).

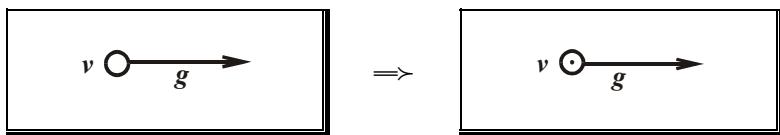
Утверждение 2.3.1 (свойство SCBg-1). Дуга принадлежности не может выходить из дуги принадлежности и, в частности, из самой себя (см. свойство SCB-1 утверждения 2.2.1), т.е. запрещены конструкции следующего вида:



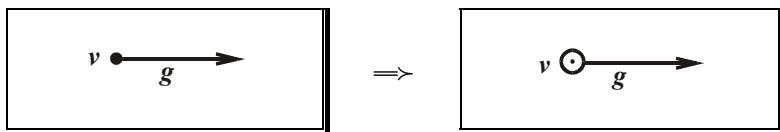
Утверждение 2.3.2 (свойство SCBg-2). Дуга принадлежности не может выходить из предметного узла (см. свойство SCB-2 утверждения 2.2.2):



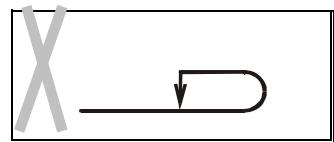
Утверждение 2.3.3 (свойство SCBg-3). Дуга принадлежности, выходящая из узла неопределённого типа, позволяет уточнить тип этого узла (см. свойство SCB-3 утверждения 2.2.3):



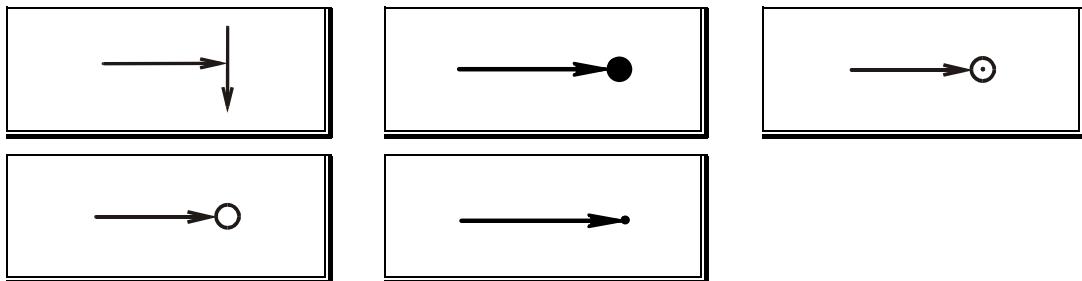
Утверждение 2.3.4 (свойство SCBg-4). Дуга принадлежности, выходящая из scb-элемента неопределённого типа, позволяет уточнить тип этого элемента (см. свойство SCB-4 утверждения 2.2.4):



Утверждение 2.3.5 (свойство SCBg-5). Дуга принадлежности не может входить в саму себя (см. свойство SCB-5 утверждения 2.2.5):

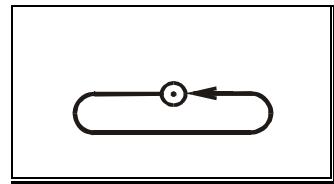


Утверждение 2.3.6 (свойство SCBg-6). Дуга принадлежности может входить в scb-элемент любого типа, т.е. могут существовать, например, конструкции вида (см. свойство SCB-6 утверждения 2.2.6):



Утверждение 2.3.7 (свойство SCBg-10).

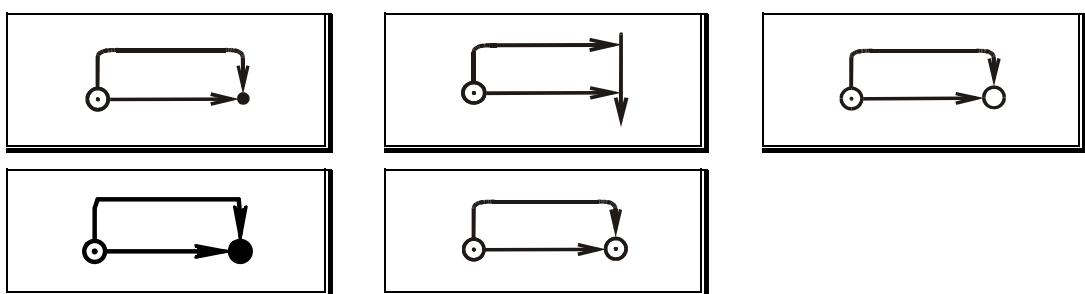
SCB-элемент, из которого дуга принадлежности выходит, и scb-элемент, в который дуга принадлежности входит, могут совпадать (см. свойство SCB-10 утверждения 2.2.10), т. е. может существовать конструкция вида:



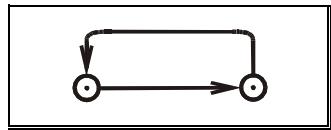
Но в силу свойств SCBg-1 и SCBg-2 не могут существовать конструкции вида:



Утверждение 2.3.8 (свойство SCBg-11). В scb-конструкциях могут встречаться кратные scb-дуги, т.е. могут существовать конструкции вида (см. свойство SCB-11 утверждения 2.2.11):



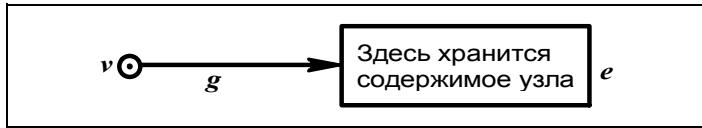
Утверждение 2.3.9 (свойство SCBg-12). В scb-конструкциях могут присутствовать встречные scb-дуги (см. свойство SCB-12 утверждения 2.2.12):



Среди предметных scb-узлов (т.е. scb-узлов, являющихся знаками конкретных предметов) особое место занимают scb-узлы, обозначающие информационные конструкции самого различного вида. Каждую из указанных информационных конструкций будем называть **содержимым** того scb-узла, который эту конструкцию обозначает.

Узел, имеющий известное содержимое, в графическом языке SCBg изображается в виде прямоугольной замкнутой линии (см. scbg-текст 2.3.7), которая ограничивает изображение информационной конструкции, являющейся этим содержимым. Очевидно, что в данном случае содержимое узла должно быть визуально воспроизводимо, т.е. может быть изображено на листе бумаги. Таким содержимым может быть символьный текст какого-либо естественного или искусственного языка. Таким содержимым может быть рисунок, фотография, карта, схема, чертёж, нотная запись какого-либо музыкального произведения. Содержимым может быть также закодированный звуковой сигнал (музыкальное произведение, речевое сообщение), видеофильм, хотя такое содержимое нет возможности представить на бумажном носителе.

SCBg-текст 2.3.7. Изображение scb-узла с содержимым



Завершая рассмотрение ядра языка SCBg, сделаем несколько примечаний.

Примечание 1. Приведенный в подразделе 2.3 алфавит графических примитивов языка SCBg будет расширен для повышения наглядности текстов этого языка (см. следующие подразделы). Кроме того, расширение фактографического языка SCB до логического языка SCL (см. раздел 5) потребует введения принципиально новых текстовых элементов (в частности, переменных), что также расширит алфавит графических примитивов.

Примечание 2. Теоретически минимальный алфавит графических примитивов языка SCBg включает в себя:

- маленький кружок, изображающий знак множества неуточняемого типа;
- линию со стрелкой, изображающую знак пары принадлежности.

Все остальные графические примитивы можно свести к двум указанным (это показано в следующем подразделе).

Упражнения к подразделу 2.3.

Упражнение 2.3.1. Сколько и какие графические примитивы используются в языке SCBg для изображения предметного scb-узла? SCB-элемента неопределенного типа? Дуги принадлежности?

Упражнение 2.3.2. Сколько и какие графические примитивы используются в языке SCBg для изображения узлов различных типов?

1.4. Средства обеспечения наглядности языка SCBg

Ключевые понятия и идентификаторы ключевых scbg-узлов:
 синонимичные scbg-элементы; scbg-изображение непредметного scb-узла в виде замкнутой линии; изображение увеличения контактной зоны scbg-изображения scb-элемента; *пара принадлежности*; *узловое множество*; *предмет*; *узловое непредметное множество*; *простая ориентированная узловая пара*; *пара непринадлежности*; *пара нечёткой принадлежности*; *неориентированная пара*; *пара синонимии*; *пара несинонимии*; *пара нечёткой синонимии*; *семейство пар принадлежности*; *семейство узловых множеств*; *семейство предметных множеств*; *семейство узловых непредметных множеств*; *система множеств*.

Наглядность scbg-текста в первую очередь обеспечивается тем, насколько удачно автор текста разместит на листе бумаги или экране монитора графические изображения узлов и дуг принадлежности. Удачное размещение, в частности, приводит к сокращению числа пересечений линий, изображающих дуги принадлежности, а также к упрощению формы этих линий (в идеальном случае указанные линии должны быть отрезками, соединяющими scb-элементы).

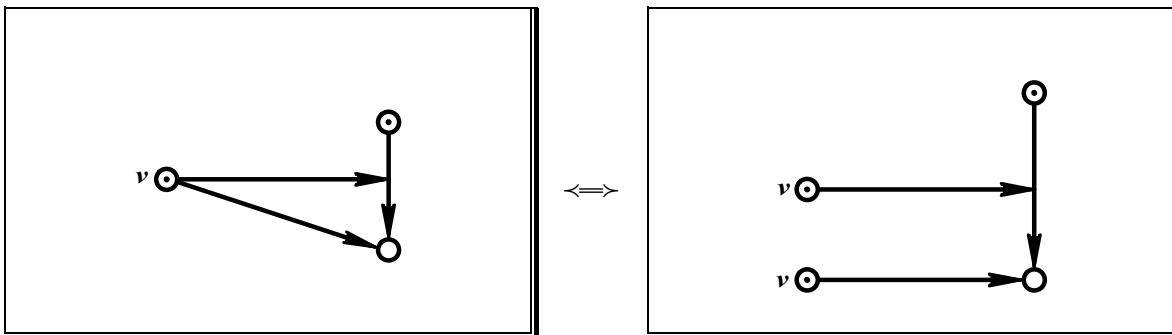
К числу дополнительных мер, обеспечивающих повышение наглядности scbg-текста, можно отнести:

- расширение алфавита графических примитивов, используемых для изображения scb-элементов и указывающих тип изображаемого scb-элемента. В частности, желательно иметь графические примитивы, которые соответствуют различным типам непредметных узлов (см. табл. 2.3.1);
- введение нескольких копий изображения одного и того же scb-элемента, т. е. допущение многократного вхождения (в scbg-текст) изображений одного и того же знака. Это означает, что в scbg-тексте могут встречаться синонимичные scbg-элементы (см. scbg-текст 2.4.1);
- введение неявно изображаемых дуг принадлежности, а именно дуг принадлежности, неявно проводимых из непредметного узла, изображенного замкнутой линией, во все scb-элементы, изображенные внутри этой замкнутой линии (см. scbg-текст 2.4.2);
- увеличение размера "контактной зоны" изображения узла, если этот узел имеет большое количество инцидентных дуг принадлежности (см. scbg-текст 2.4.3);
- введение неявно изображаемых дуг принадлежности, проводимых из узлов, идентификаторы которых содержат двоеточие, в scb-элементы, которым указанные идентификаторы с двоеточием присвоены (см. scbg-текст 2.4.4);
- синонимичные scbg-элементы необходимо явно указать. Это, в частности, осуществляется путем присыпывания синонимичным scbg-элементом одного и того же идентификатора, который считается символным эквивалентом (символьным вариантом изображения) соответствующего scb-элемента. Поскольку два синонимичных scbg-элемента семантически представляют собой изображения одного и того же знака, они могут быть логически склеены. Подчеркнем при этом, что возможность в языке scbg склеивать синонимичные scbg-элементы не только логически, но и "физически" придает этому языку хорошие "ассоциативные" свойства, т.е. возможность поиска нужной информации по связям. Для языка SCBg по умолчанию считается, что два разных scbg-элемента (в том числе scbg-элементы без идентификаторов, см. scbg-текст 2.4.5) считаются изображениями разных scb-элементов.

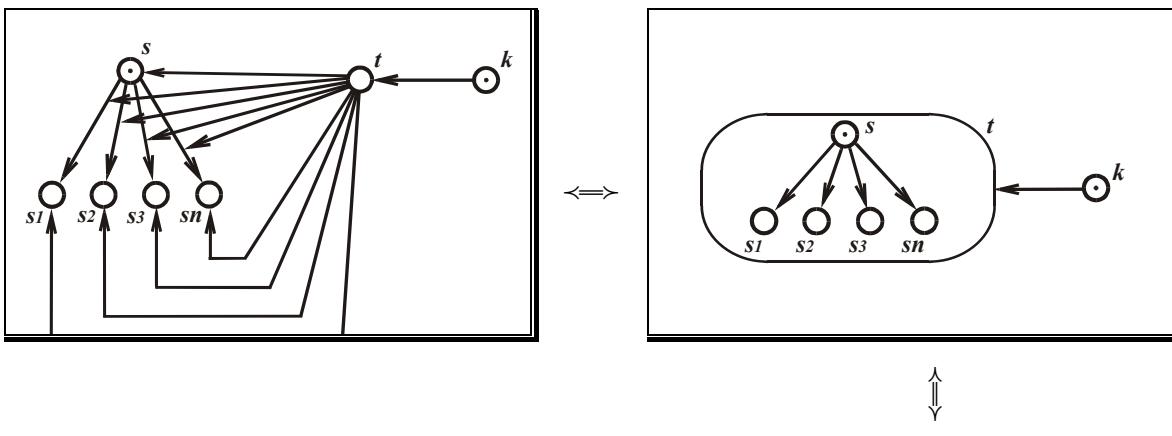
Примечание. Следует отличать:

- синонимичные scbg-элементы, которые имеют обычно разные идентификаторы и которые являются разными знаками, но одного и того же множества (см. scbg-текст 2.4.6);
- синонимичные scbg-элементы, которые имеют одинаковые идентификаторы и которые являются изображениями одного и того же scb-элемента, т.е. являются разными изображениями (разными вхождениями в scbg-текст) одного и того же знака (см. scbg-текст 2.4.1);
- синонимичные вхождения идентификаторов, которые представляют собой одинаковые строки символов и которые являются изображениями одного и того же scb-элемента, т.е. являются разными вхождениями в scbg-текст (см. подраздел 2.5) одного и того же знака.

SCBg-текст 2 . 4 . 1 . Использование многократного вхождения изображений одного и того же знака

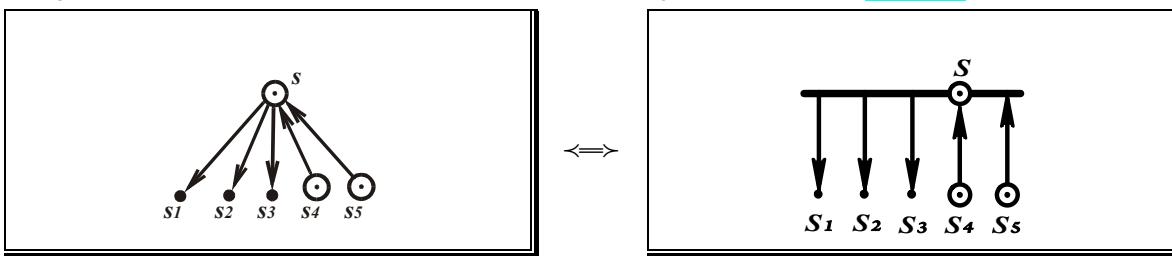


SCBg-текст 2 . 4 . 2 . Использование изображения непредметных scb-узлов в виде замкнутых линий



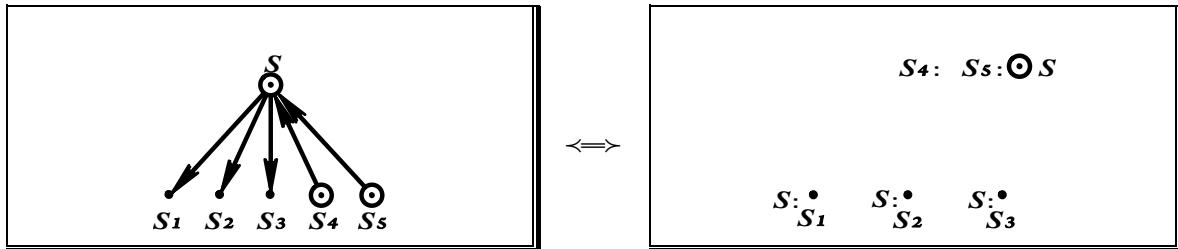
Примечание. Замкнутая линия изображает знак множества всех тех и только тех знаков множеств, которые являются полностью нормализованной системой множеств, изображение которой ограничено замкнутой линией. При этом линия, изображающая дугу принадлежности, и замкнутая линия изображающая непредметный узел, должны полностью входить внутрь области, ограниченной замкнутой линией

SCBg-текст 2 . 4 . 3 . Увеличение "контактной зоны" scb-узла с помощью толстых линий

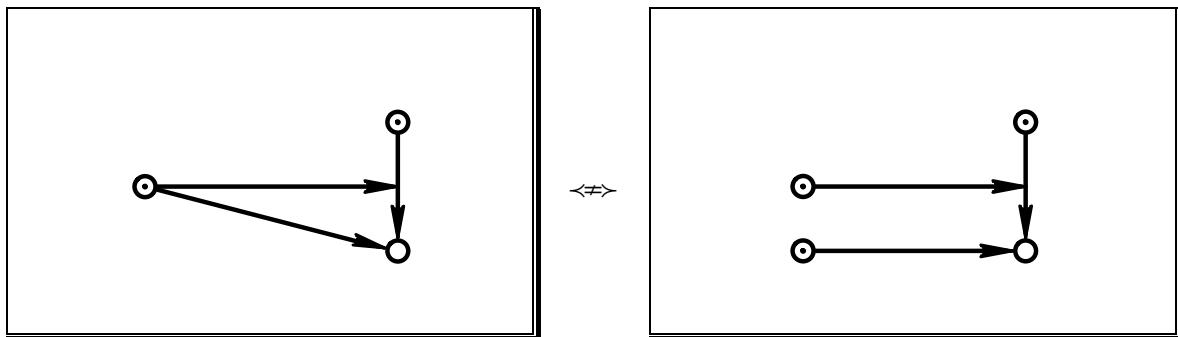


Примечание. Увеличение "контактной зоны" изображения узла (**s**) целесообразно тогда, когда узел (**s**) имеет большое количество инцидентных ему дуг принадлежности.

SCBg-текст 2.4.4. Использование неявно изображаемых scb-дуг, проводимых из scb-узлов, идентификаторы которых содержат двоеточие



SCBg-текст 2.4.5. Изображения элементов без идентификаторов обозначают разные scb-элементы



Итак, кроме перечисленных в табл. 2.4.1 типов изображений, в алфавит графических примитивов языка SCBg входят следующие дополнительные изображения, перечисленные в табл. 2.5.1.

Таблица 2.4.1. Алфавит дополнительных графических примитивов языка SCBg

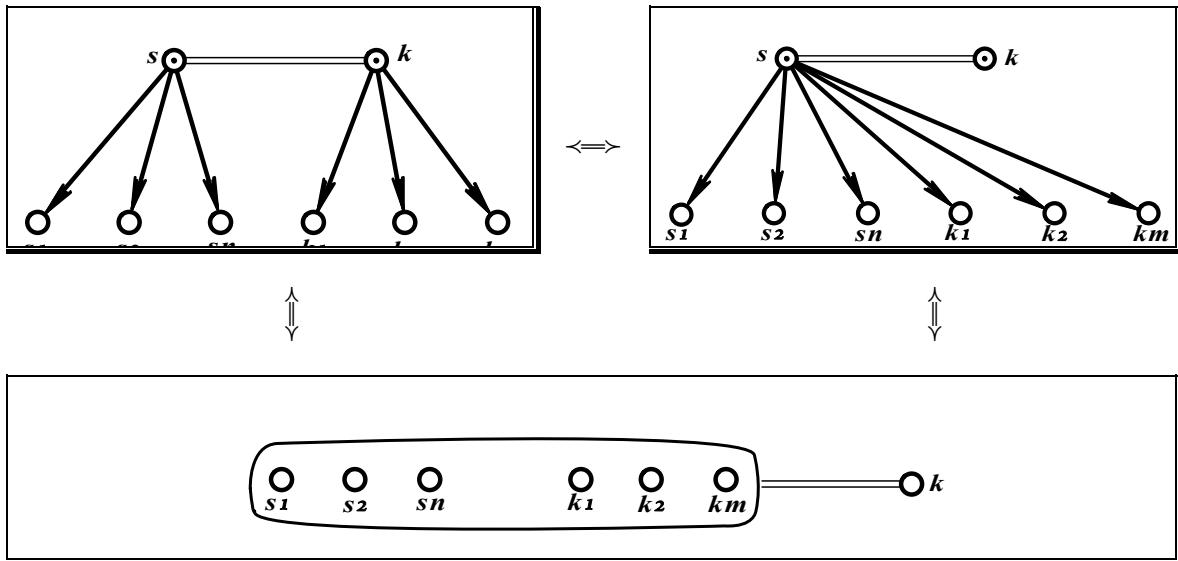
Наименование	Изображение
• замкнутые линии, изображающие непредметные узлы, обозначающие множества scb-элементов, изображенных внутри этих замкнутых линий	
• замкнутые линии, изображающие узлы с явно указываемым содержимым	
• толстые линии (которые будем называть шиной), увеличивающие размер "контактной зоны" узлов, т.е. зоны, с которой соприкасаются изображения входящих и выходящих дуг принадлежности	
• двоеточия	:

Синонимичные scb-элементы могут быть указаны не только путем присыпывания им одинаковых идентификаторов, но и с помощью специальных пар синонимии (см. табл. 2.1.1). Знаки множеств, связываемые каждой такой парой, либо имеют разные идентификаторы (см. scbg-текст 2.4.6), либо не имеют идентификаторов для обоих знаков, либо не имеют идентификатора для одного из них. Подчеркнем при этом, что по умолчанию scb-элементы считаются несинонимичными, т.е. пары несинонимии (пары неравенства знаков) в основном указываются по умолчанию. Это значит, что, если какие-либо два scb-элемента не связаны друг с другом парой синонимии (парой равенства знаков), то эти два scb-элемента считаются связанными парой несинонимии.

Два синонимичных scb-элемента обозначают равные множества (а точнее, одно и то же множество). Хотя обратное не является верным – два равных множества (т.е. множества, состоящие из одинаковых элементов) могут считаться разными объектами и, следовательно, обозначаться разными несинонимичными scb-элементами. Следовательно, пару, связывающую два синонимичных

scb-элемента (два равных знака), следует отличать от пары, которая связывает знаки двух равных множеств. Равные множества, если они вводятся, считаются разными объектами, и поэтому их знаки не могут быть синонимичными. Следует при этом подчеркнуть, что равные множества можно вводить только в случае особой необходимости – для представления кратных или встречных связок, принадлежащих одному отношению.

SCB_g-текст 2 . 4 . 6 . Использование рёбер синонимии



В языке SCB_g приписывание идентификаторов scb-элементам, изображаемым в scbg-конструкциях, не является обязательным. В этом возникает необходимость только тогда, когда изображение одного и того же scb-элемента несколько раз входит в одну или разные scbg-конструкции, например, на разных страницах. В этом случае приписывание разным scb-элементам одинаковых идентификаторов является указанием того, что они являются изображениями одного и того же scb-элемента.

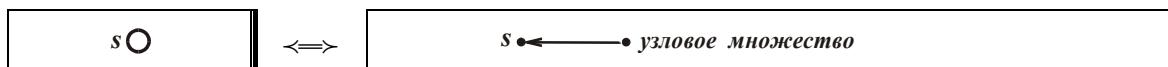
Явное указание типа изображаемого scb-элемента в языке SCB_g осуществляется либо путем использования графического примитива, который соответствует типу изображаемого scb-элемента, либо явного изображения дуги принадлежности, которая связывает изображаемый scb-элемент со специальным узлом, который является знаком всевозможных scb-элементов соответствующего типа. В связи с этим введем ряд специальных узлов, которые соответствуют некоторым графическим примитивам, приведенным в табл. 2.3.1. Особо подчеркнем то, что из перечисленных в табл. 2.3.1 типов scb-элементов базовыми следует считать два первых – знаки множеств неуточняемого типа и знаки пар принадлежности (см. scbg-текст 2.3.1). Для того чтобы все остальные типы scb-элементов свести к указанным двум типам, достаточно каждому небазовому типу scb-элементов поставить в соответствие специальный scb-элемент, являющийся знаком всех scb-элементов соответствующего типа и только их. Такие scb-элементы будем называть **ключевыми scb-элементами**. Каждому ключевому scb-элементу поставим в соответствие уникальный идентификатор, взаимно однозначно соответствующий идентифицируемому ключевому scb-элементу. Общие правила построения идентификаторов scb-элементов рассмотрены в подразделе 2.6, а сейчас просто перечислим идентификаторы нужных ключевых элементов:

- “**пара принадлежности**” (быть знаком пары принадлежности) – это идентификатор scb-элемента, обозначающего множество знаков всевозможных пар принадлежности и только их;
- “**узловое множество**” (быть знаком узлового множества) – это идентификатор scb-элемента, обозначающего множество знаков всевозможных узловых множеств и только их (см. scbg-текст 2.4.7);
- “**предмет**” (быть знаком предметного множества) – это идентификатор scb-элемента, обозначающего множество знаков всевозможных предметных множеств и только их (см. scbg-текст 2.4.8);
- “**узловое непредметное множество**” (быть знаком узлового непредметного множества) – это идентификатор scb-элемента, обозначающего множество знаков всевозможных узловых непредметных множеств и только их (см. scbg-текст 2.4.9);
- “**простая ориентированная узловая пара**” (быть знаком простой ориентированной узловой пары) – это идентификатор scb-элемента, обозначающего множество знаков всевозможных простых ориентированных узловых пар и только их (см. scbg-текст 2.4.10);

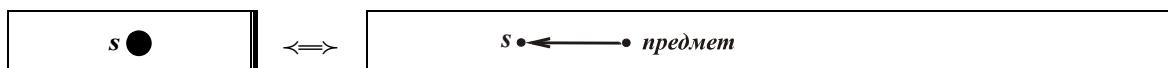
- “**пара непринадлежности**” (быть знаком пары непринадлежности) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных пар непринадлежности и только их (см. scbg-текст 2.4.11);
- “**пара нечёткой принадлежности**” (быть знаком пары нечёткой принадлежности) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных пар нечёткой принадлежности и только их (см. scbg-текст 2.4.12);
- “**неориентированная пара**” (быть знаком неориентированной пары) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных неориентированных пар и только их (см. scbg-текст 2.4.13);
- “**пара синонимии**” (быть знаком пары синонимии) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных пар синонимии и только их (см. scbg-текст 2.4.14);
- “**пара несинонимии**” (быть знаком пары несинонимии) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных пар несинонимии и только их (см. scbg-текст 2.4.15);
- “**пара нечёткой синонимии**” (быть знаком пары нечёткой синонимии) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных пар нечёткой синонимии и только их (см. scbg-текст 2.4.16);
- “**семейство пар принадлежности**” (быть знаком семейства пар принадлежности) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных семейств пар принадлежности и только их (см. scbg-текст 2.4.17);
- “**семейство узловых множеств**” (быть знаком семейства узловых множеств) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных семейств узловых множеств и только их (см. scbg-текст 2.4.18);
- “**семейство предметных множеств**” (быть знаком семейства предметных множеств) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных семейств предметных множеств и только их (см. scbg-текст 2.4.19);
- “**семейство узловых непредметных множеств**” (быть знаком семейства узловых непредметных множеств) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных семейств узловых непредметных множеств и только их (см. scbg-текст 2.4.20);
- “**система множеств**” (быть знаком системы множеств) – это идентификатор scbg-элемента, обозначающего множество знаков всевозможных систем множеств и только их (см. scbg-текст 2.4.21).

На следующих scbg-текстах рассмотрены правила приведения различных типов scbg-элементов к знакам множеств неуточняемого типа и к знакам пар принадлежности.

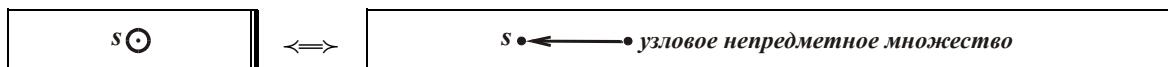
С C B g - т е к с т 2 . 4 . 7 . Изображение знака **узлового множества**



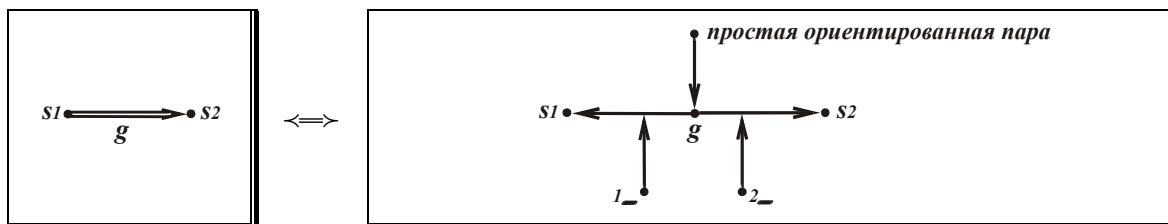
С C B g - т е к с т 2 . 4 . 8 . Изображение знака **предмета**



С C B g - т е к с т 2 . 4 . 9 . Изображение знака **узлового непредметного множества**

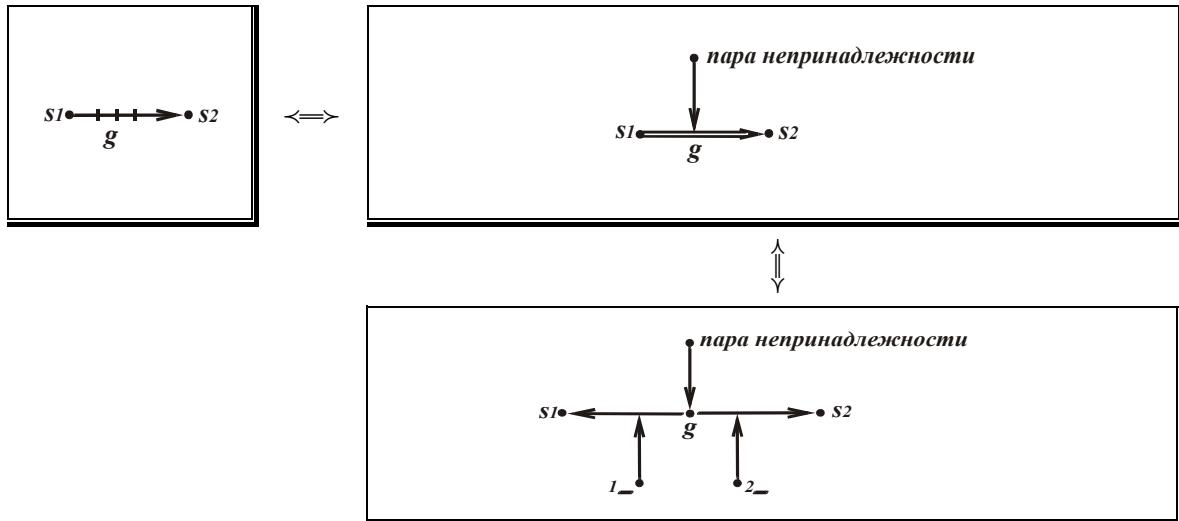


С C B g - т е к с т 2 . 4 . 1 0 . Изображение знака **простой ориентированной пары**

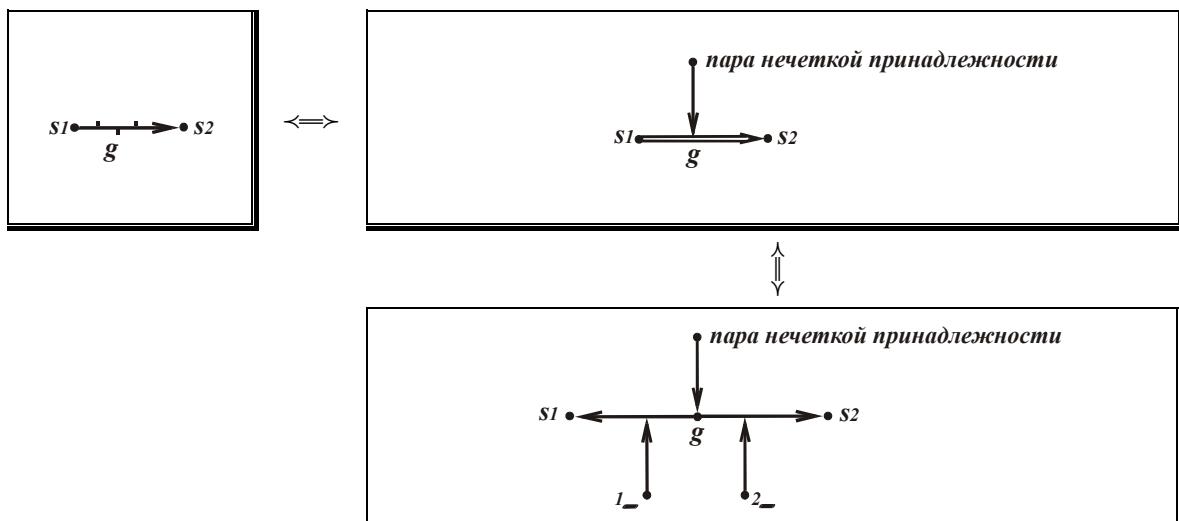


Примечание. Здесь scb-элемент с идентификатором “1” является знаком множества всевозможных знаков пар принадлежности, связывающих знаки ориентированных (упорядоченных) множеств с первыми элементами этих множеств. Аналогично этому scb-элемент с идентификатором “2” задает вторые элементы ориентированных множеств. Подробнее об этом – в разделе 3.

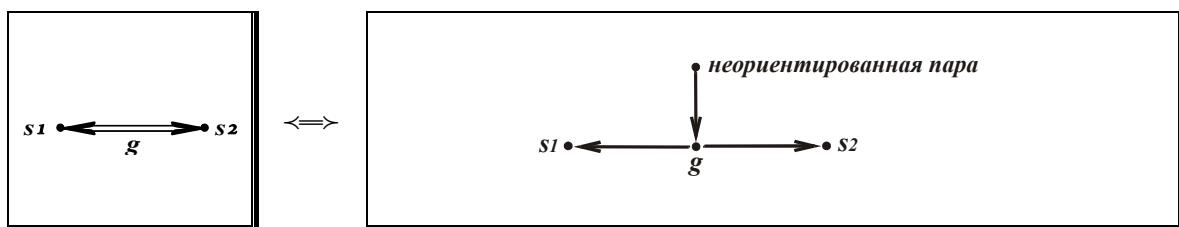
С C B g - т е к с т 2 . 4 . 1 1 . Изображение знака *пары непринадлежности*

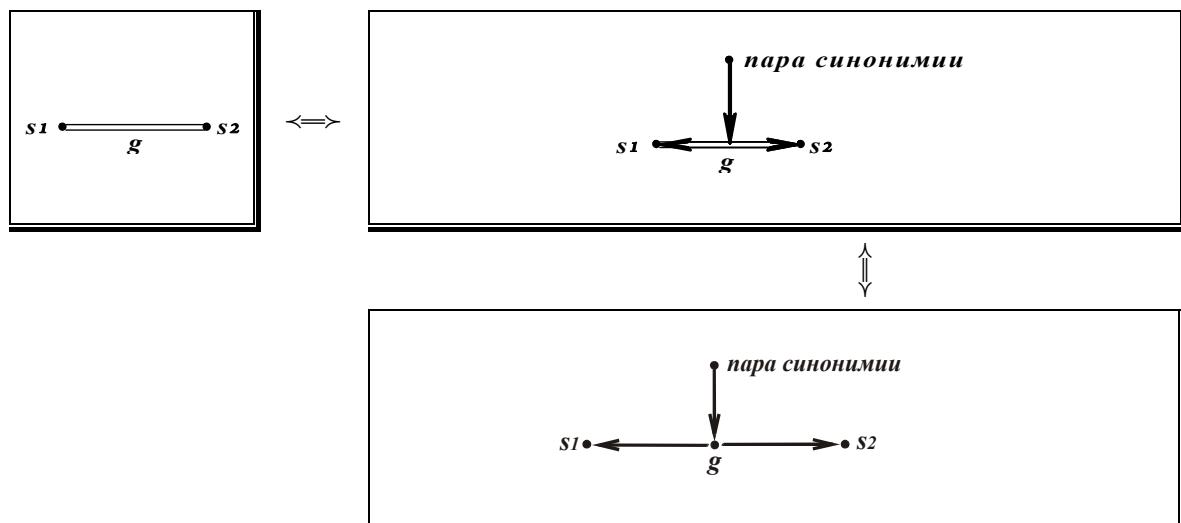
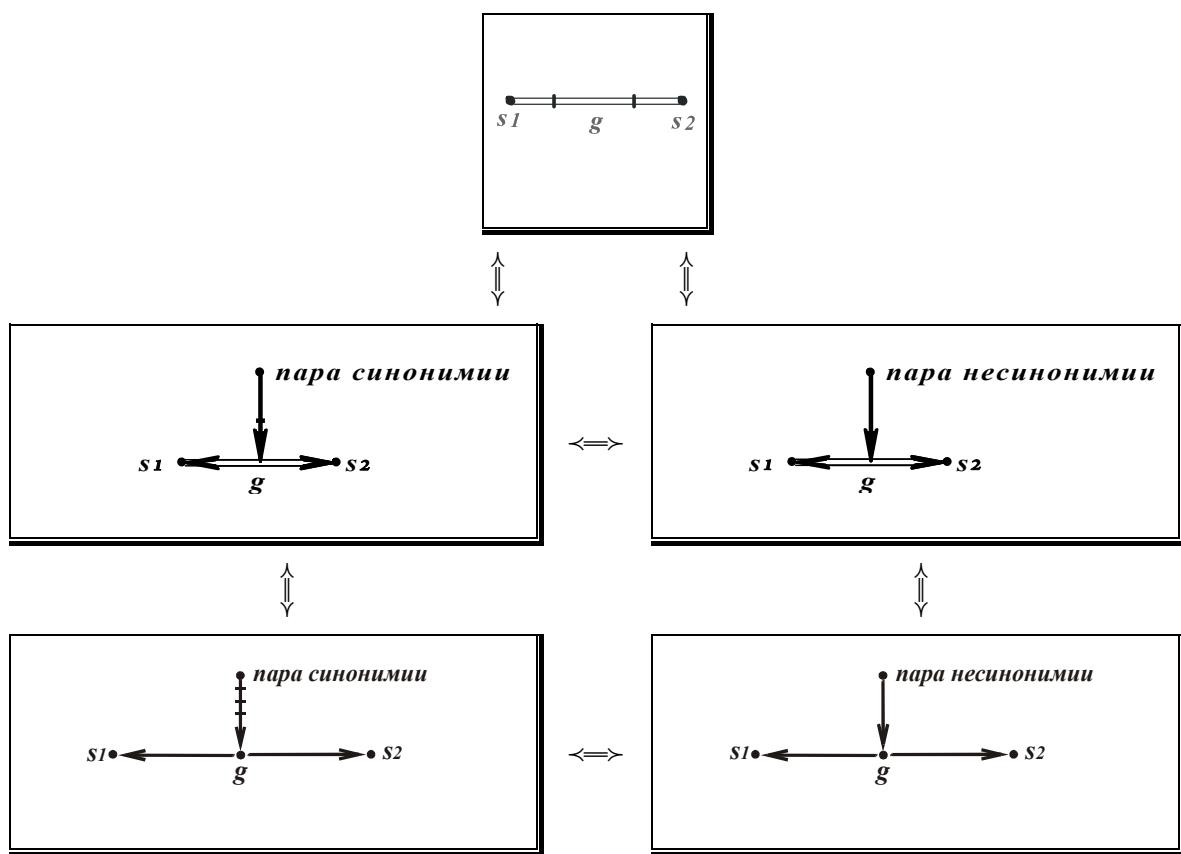


С C B g - т е к с т 2 . 4 . 1 2 . Изображение знака *пары нечёткой принадлежности*

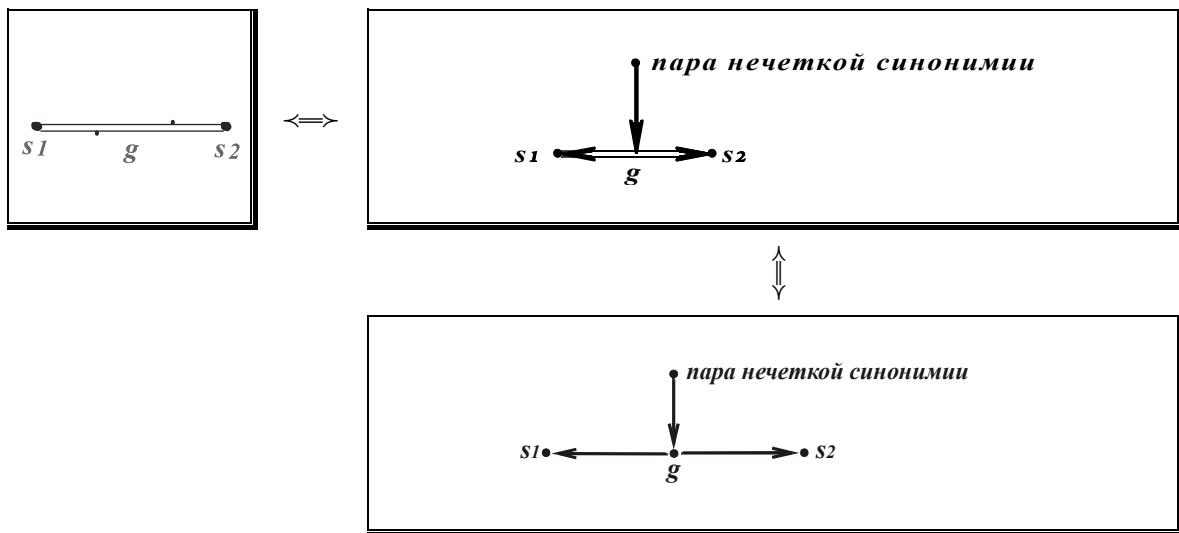


С C B g - т е к с т 2 . 4 . 1 3 . Изображение знака *неориентированной пары*



С В г - т е к с т 2 . 4 . 1 4 . Изображение знака пары синонимии**С В г - т е к с т 2 . 4 . 1 5 .** Изображение знака пары несинонимии

SCB-текст 2.4.16. Изображение знака [пары нечёткой синонимии](#)



SCB-текст 2.4.17. Изображение знака [семейства пар принадлежности](#)



SCB-текст 2.4.18. Изображение знака [семейства узловых множеств](#)



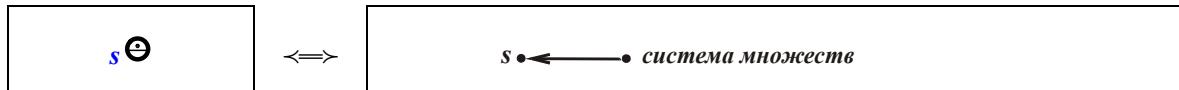
SCB-текст 2.4.19. Изображение знака [семейства предметов](#)



SCB-текст 2.4.20. Изображение знака [семейства узловых непредметных множеств](#)

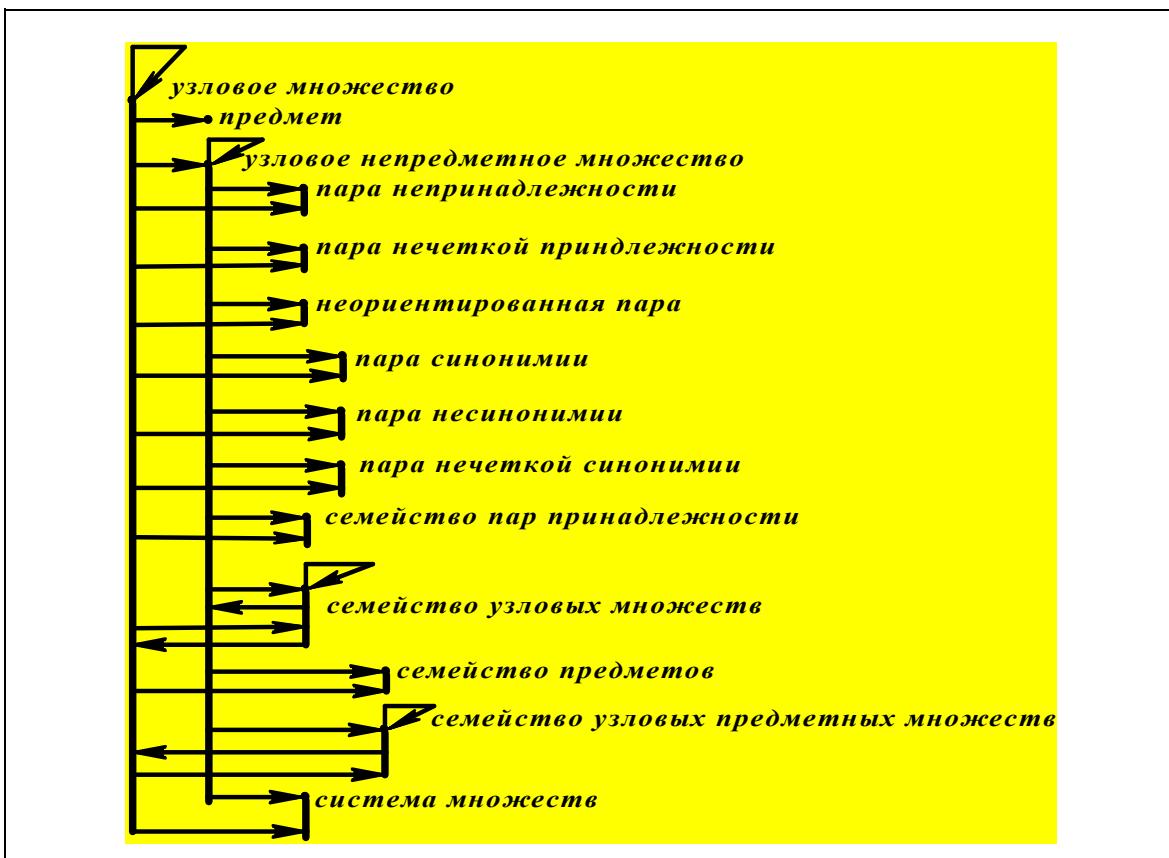


SCB-текст 2.4.21. Изображение знака [системы множеств](#)



Заметим также, что введенные нами ключевые scb-элементы связаны между собой целым рядом пар принадлежности, которые приведены на scbg-тексте [2.4.22](#).

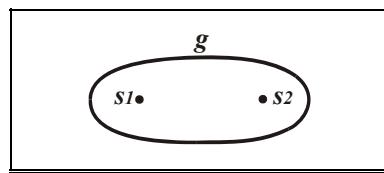
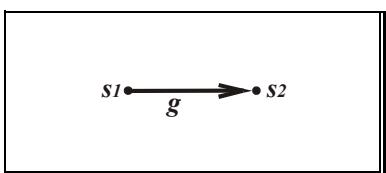
SCB-текст 2.4.22. Перечень пар принадлежности, связывающих между собой введенные выше ключевые scb-элементы



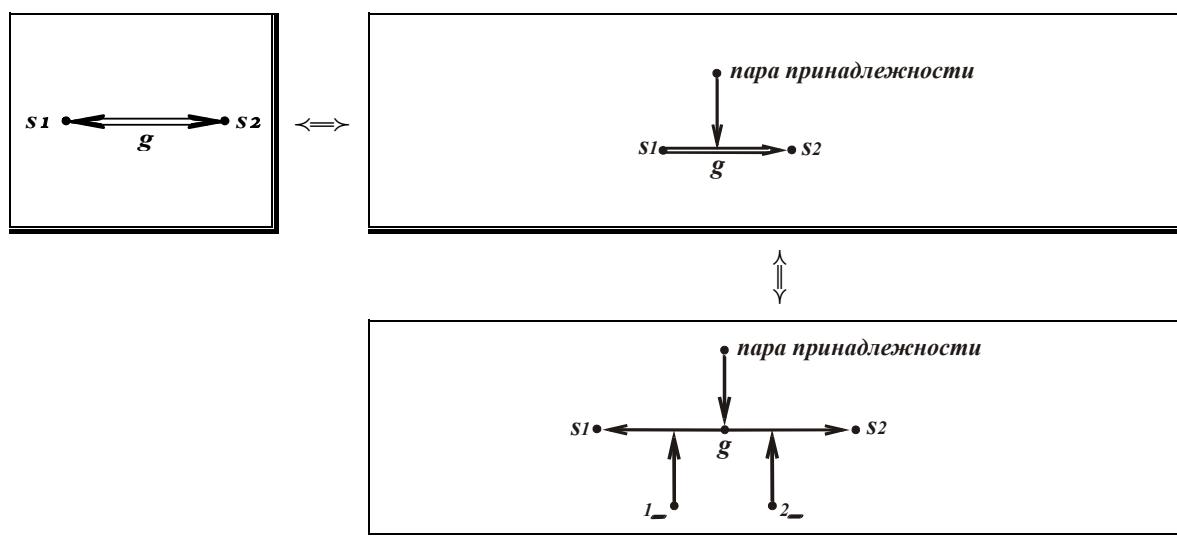
Примечание. Пары принадлежности не следует путать с парами отношения теоретико-множественного включения.

Упражнения к подразделу 2.4.

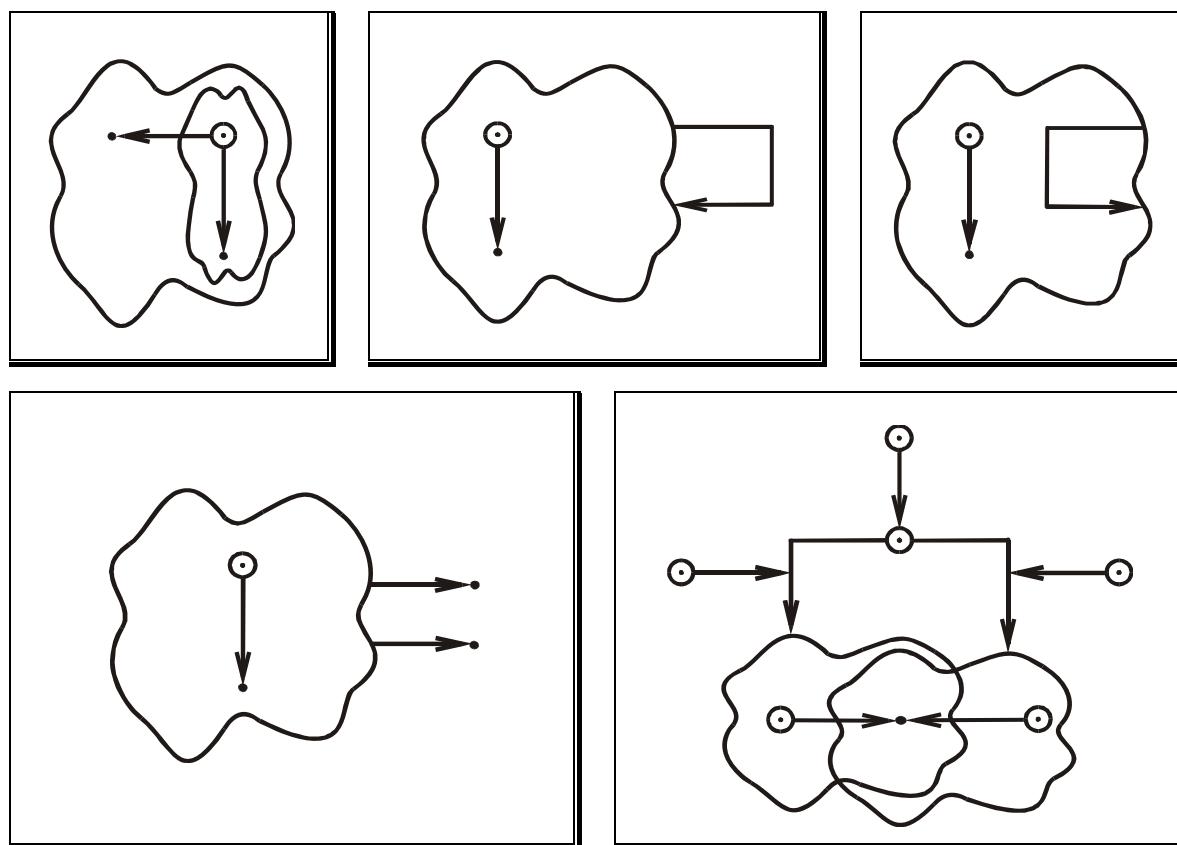
Упражнение 2.4.1. Можно ли считать семантически эквивалентными следующие scbg-конструкции?



Упражнение 2.4.2. Корректны ли следующие преобразования?



Упражнение 2.4.3. Преобразуйте приведённые ниже scbg-тексты, исключив из них изображения непредметных scb-узлов в виде замкнутых линий:



1.5. Ядро языка SCBs (Semantic Code Basic string) – линейной модификации языка SCB

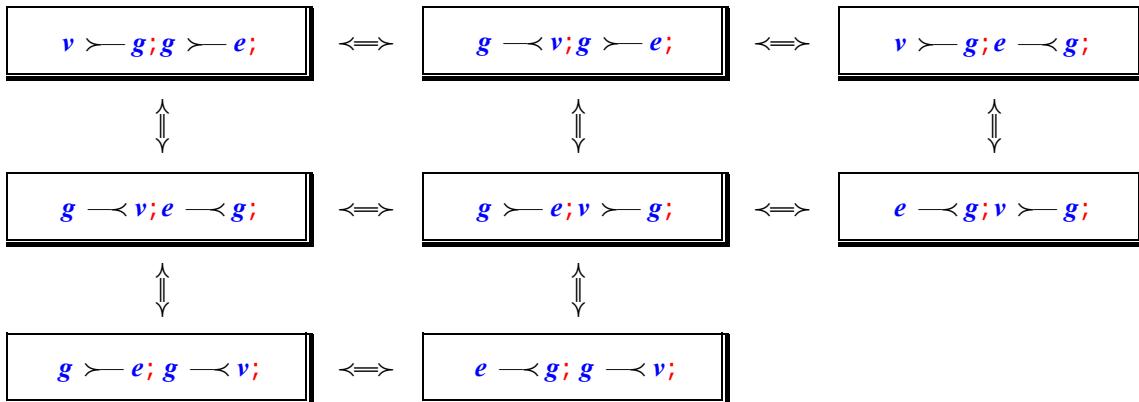
Ключевые понятия: язык SCBs; scbs-текст (**scbs-конструкция**); scbs -предложение; scbs -разделитель; scbs-разделитель инцидентности; scbs-разделитель принадлежности; scbs-разделитель синонимии; scbs-разделитель несинонимии; scb-идентификатор (**идентификатор scb-элемента**).

Символьная модификация языка SCB, а именно **язык SCBs** (Semantic Code Basic string), дает возможность записать тексты языка SCB в линейном (строковом) виде. Для этого в языке SCBs принят ряд соглашений по формированию и использованию идентификаторов scb-элементов и введены специальные разделители, обозначающие связи между scb-элементами, а также способствующие наиболее понятному "прочтению" scbs-текстов.

Итак, текст, записанный на языке SCBs, назовем **scbs-текстом** (или scbs-конструкцией). Элементарные составляющие scbs-текстов будем называть **scbs-предложениями**.

На языке SCBs, так же как на графической модификации – SCBg (см. подраздел 2.4), имеется возможность представлять тройки принадлежности различными способами. Возможные эквивалентные варианты изображения тройки принадлежности даны на scbs-тексте 2.5.1 (см. также scbg-текст 2.3.1).

SCBs-текст 2.5.1. Варианты изображения тройки принадлежности на языке SCBs



Здесь разделители “ \succ ” и “ \prec ” будем называть **scbs-разделителями инцидентности**, а символ “ $;$ ” (точка с запятой) – **разделителем scbs-предложений**. Каждый из приведенных вариантов scbs-текстов состоит из двух scbs-предложений.

Заметим, что стилистически, для повышения наглядности scbs-текстов, в состав разделителей, подобных связкам инцидентности, принято включать пробелы слева и справа от разделителя. При этом допускается произвольное количество пробелов до и после разделителя языка SCBs.

Символами **v**, **g**, **e** обозначены идентификаторы соответствующих scb-элементов, т. е. символные варианты изображения (в виде строк символов) тех знаков множеств, которые эквивалентны (синонимичны) соответствующим scb-элементам. Эквивалентные (синонимичные) знаки – это знаки, обозначающие одно и то же. Для языка SCB и его модификаций – это знаки, обозначающие одно и то же множество, т. к. в указанных языках нет ничего, кроме множеств и их знаков. Кроме того, указанные три идентификатора **v**, **g**, **e** соответствуют scb-элементам разного типа. Идентификатор **v** соответствует непредметному узлу (знаку узлового непредметного множества, см. scbg-текст 2.3.2). Идентификатор **g** соответствует дуге принадлежности (знаку пары принадлежности). Идентификатор **e** соответствует scb-элементу, тип которого может быть любым (см. scbg-тексты 2.3.2 – 2.3.6).

SCBs-конструкцию **ei >-> ej ;** будем называть элементарной scbs-конструкцией 1-го вида.

SCBs-конструкцию

$ej \rightarrow ei ;$

будем называть элементарной scbs-конструкцией 2-го вида.

Элементарные scbs-конструкции могут быть элементарными scbs-предложениями соответственно 1-го и 2-го вида. Справедливо следующее эквивалентное преобразование элементарных scbs-предложений 1-го вида в элементарные scbs-предложения 2-го вида и наоборот:

$ei \succ ej ;$

\Leftrightarrow

$ej \prec ei ;$

Кроме того, эквивалентным преобразованием scbs-текста является любая перестановка предложений. Результатом указанных эквивалентных преобразований scbs-текстов являются перечисленные на scbs-тексте 2.5.1 варианты изображения троек принадлежности.

Элементарные scbs-конструкции описывают инцидентность (соседство) scb-элементов. Соответственно этому scbs-конструкция:

$ei \succ ej ;$

или эквивалентная ей конструкция:

$ej \prec ei ;$

трактуется следующим образом:

- scb-элемент ei инцидентен слева scb-элементу ej ;
- scb-элемент ej инцидентен справа scb-элементу ei .

По аналогии с рассмотрением ядра языка SCBg (см. утверждения 2.2.1 – 2.2.14 в подразделе 2.3 и утверждения 2.3.1 – 2.3.9 в подразделе 2.4) перечислим основные свойства языка SCB в рамках языка SCBs. Здесь для наглядности сохранена нумерация, принятая при перечислении свойств языка SCB.

Утверждение 2.5.1 (свойство SCBs-1). Дуга принадлежности не может выходить из дуги принадлежности и, в частности, не может выходить из самой себя (см. свойство SCB-1 утверждения 2.2.1). То есть если gi и gj являются идентификаторами дуг принадлежности, то в языке SCBs конструкции вида:

$gi \succ gj ;$

и эквивалентная ей scbs-конструкция вида:

$gj \prec gi ;$

означают, что дуга принадлежности gi входит в дугу принадлежности gj . Трактовка: дуга принадлежности gj выходит из дуги принадлежности gi – является некорректной.

Утверждение 2.5.2 (следствие из свойств SCB-1 утверждения 2.2.1, SCB-5 утверждения 2.2.5, SCB-7 утверждения 2.2.7). SCB-элемент любого типа не может быть инцидентен сам себе, т.е. некорректными считаются следующие scbs-конструкции:

 $e \succ e ;$

 $e \prec e ;$

Утверждение 2.5.3 (следствие из свойства SCB-7 утверждения 2.2.7). SCB-узлы не могут быть инцидентны друг другу, т. е. если vi и vj являются идентификаторами scb-узлов, то следующие scbs-конструкции некорректны:

 $vi \succ vj ;$

\Leftrightarrow

 $vj \prec vi ;$

Утверждение 2.5.4 (свойство SCBs-13). Если g есть идентификатор некоторой дуги принадлежности, то существует только один scb-элемент v , для которого справедлива scbs-конструкция вида (см. свойство SCB-13 утверждения 2.2.13):

$v \succ g ;$

или эквивалентная ей конструкция вида:

$g \prec v ;$

а также существует только один scb-элемент e , для которого справедлива scbs-конструкция вида:

$g \succ e;$

или эквивалентная ей конструкция вида:

$e \prec g;$

Как сказано выше, каждому scb-узлу, обозначающему информационную конструкцию, можно поставить в соответствие содержимое этого узла, каковым указанная информационная конструкция и является. Содержимым узла может быть информационная конструкция любого вида (в частности, любой файл).

Если узел является знаком представления некоторого числа в некоторой системе счисления, то содержимым этого узла следует считать представление указанного числа в заданной системе счисления.

Если узел является знаком некоторого текста, то содержимым этого узла следует считать сам указанный текст (см. scbg-текст 2.3.7).

На языке SCBs присваивание узлу соответствующего содержимого осуществляется следующим образом:

$v = "/ t "/;$

где

- v – scb-идентификатор некоторого узла;
- $=$ – scbs-разделитель синонимии (подробнее см. подраздел 2.6)
- t – непосредственно сам символьный текст (цепочка символов), который является содержимым узла v ;
- косая черта с кавычкой используется как левый ограничитель текста, являющегося содержимым;
- кавычка с косой чертой используется как правый ограничитель текста, являющегося содержимым.

Таким образом, в языке SCBs разделитель “ $/$ ” и разделитель “ $"$ ” являются scbs-ограничителями (слева и справа соответственно) для произвольного текста, представляющего собой содержимое некоторого scb-узла.

Далее будем называть scbs-ограничителями те scbs-разделители, которые имеют характер открывающей и закрывающей "скобок". Так, ограничитель “ $/$ ” можно назвать открывающей скобкой содержимого scb-элемента, а ограничитель “ $"$ ” – закрывающей скобкой указанного содержимого.

Комментарий в тексте языка SCBs можно вставлять в любом месте, ограничивая его левым ограничителем, в качестве которого используется косая черта со звёздочкой “ $/*$ ”, и правым ограничителем, в качестве которого используется звёздочка с косой чертой “ $*/$ ” (см. scbs-текст 2.5.2).

SCBs-текст 2.5.2. Использование ограничителей комментария и содержимого узла

$v \succ g; g \succ e;$

/ Так как элемент e является узлом, можем присвоить ему содержимое. Далее следует учитывать, что узел e является предметным, и из него не могут выходить дуги. */*

$e = "/$ Здесь хранится содержимое узла “ $/$; */* См. scbg-текст 2.4.7 */*

В табл. 2.5.1 приведены базовые средства языка SCBs для записи scb-текстов в линейном виде. Этих средств достаточно для того, чтобы записать любой scb-текст. Ниже будут рассмотрены дополнительные средства, облегчающие понимание таких текстов.

Таблица 2.5.1. Основные разделители языка SCBs

Разделитель	Описание
 	scbs-разделители инцидентности
	разделитель scbs-предложений
=	scbs-разделитель синонимии
	левый scbs-ограничитель содержимого scb-узла
	правый scbs-ограничитель содержимого scb-узла
	левый scbs-ограничитель комментария (в scbs-тексте)
	правый scbs-ограничитель комментария (в scbs-тексте)

Упражнения к подразделу 2.5.

Упражнение 2.5.1. Запишите на базовых средствах языка SCBs тексты, предложенные в упражнении 2.4.3.

1.6. Формирование идентификаторов в языке SCB

Ключевые понятия: scb-идентификатор; имя собственное; имя нарицательное; простой scb-идентификатор; сложный scb-идентификатор.

SCB-идентификатор (идентификатор scb-элемента) – это класс одинаковых строк символов, каждая из которых является элементарным семантически значимым фрагментом scb-текста. В указанный класс строк символов входят всевозможные строки символов, совпадающие с некоторой условно заданной строкой. С семантической точки зрения каждый scb-идентификатор есть символьное изображение соответствующего (идентифицируемого) scb-элемента, т. е. символьное изображение знака того множества, которое обозначается указанным scb-элементом. Таким образом, идентификатор scb-элемента есть не что иное, как имя множества, обозначаемого этим scb-элементом.

Идентификаторы scb-элементов формируются по следующим принципам:

- каждому scb-элементу ставится в соответствие не более одного идентификатора (идентифицироваться могут как знаки узловых множеств, так и знаки пар принадлежности);
- у некоторых scb-элементов идентификаторы могут отсутствовать;
- каждому идентификатору соответствует один и только один идентифицируемый им scb-элемент.

Следует четко отличать сам идентификатор как класс одинаковых строк символов от вхождения этого идентификатора в соответствующие фрагменты текста. Один и тот же идентификатор может входить в состав scbs-текста несколько раз.

В языке SCBs запрещена неявная омонимия идентификаторов, т. е. разные вхождения одного и того же идентификатора не могут (например, в зависимости от контекста) быть семантическим эквивалентом разных scb-элементов, если это не оговорено специальным образом. Поэтому при вводе в базу знаний некоторого scbs-текста автор этого текста должен:

- гарантировать отсутствие неявной омонимии в рамках построенного им scbs-текста;
- убедиться в том, что каждый идентификатор, как имеющийся в базе знаний, так и во вводимом scbs-тексте, имеет одинаковую семантическую трактовку соответственно и в имеющейся базе знаний и во вводимом scbs-тексте;
- если же нет гарантии того, что вводимый идентификатор уникален, то необходимо явным образом описать омонимию данного идентификатора.

Явное задание омонимии scb-элементов на языке SCBs производится с помощью **scbs-разделителя несинонимии** “ = ” (см. scbs-текст 2.6.1):

SCBs-текст 2.6.1. Явное задание омонимии scb-элементов

v = **v** ;

Очевидно, что при этом требуется также уточнение того, с какой семантической трактовкой используется каждый из омонимичных идентификаторов в каждом отдельном scbs-тексте. Для разрешения указанного противоречия вводятся синонимичные идентификаторы (см. ниже).

Каждый scbs-текст, автор которого гарантирует отсутствие неявной омонимии (в рамках этого текста), имеет специальный признак начала (каковым является ограничитель “ **Begin** ”) и специальный признак конца (каковым является ограничитель “ **End** ”). Кроме того, в таком тексте (после “ **Begin** ”) указывается ФИО автора, дата создания scbs-текста и точка с запятой (см. scbs-текст 2.6.2).

SCBs-текст 2.6.2. Использование ограничителей начала и конца scbs-текста с гарантированным отсутствием неявной омонимии (здесь используется расширенный метаязык Бэкуса-Наура, который описан в подразделе 2.8)

Begin Иванов И.И. 15.01.2001 ;

{* < scbs-предложение > ; *} ;

End ;

Синонимия идентификаторов в языке SCBs разрешена и аналогично омонимии должна быть явно указана с помощью предложений вида:

ei = **ej** ;

Это предложение означает, что идентификатор **ei** синонимичен (семантически эквивалентен) идентификатору **ej**.

Другими словами, оба эти идентификатора изображают один и тот же scb-элемент. Указанное предложение (со знаком равенства) можно также трактовать как указание на "склеивание" scb-элемента с идентификатором **ei** и scb-элемента с идентификатором **ej**.

Примечание. Следует обратить внимание на то, что связка синонимии “ = ” используется также при формировании текстовых содержимых scb-узлов. Таким образом, запись вида:

v = /* **t** */ ;

означает, что scb-элемент с идентификатором **v** синонимичен scb-элементу /* **t** */ , обозначающему некоторый текст.

Если в scbs-тексте необходимо подчеркнуть отсутствие синонимии похожих идентификаторов (например, **ei** и **ej**), используются предложения с scbs-разделителем “ ≠ ”, имеющие вид:

ei ≠ **ej** ;

В случае же омонимии идентификаторов (см. scbs-текст 2.7.1) может иметь место, например, scbs-конструкция следующего вида:

ei = **v** ≠ **v** = **ej** ;

Здесь идентификаторы **ei** и **ej** следует рассматривать как синонимичные "заменители" омонимичного идентификатора **v**.

Итак, в дополнение к табл. 2.5.1, в табл. 2.6.1 приведём дополнительные средства языка SCBs для записи scb-текстов в линейном виде.

Таблица 2.6.1. Дополнительные средства языка SCBs

Символ	Описание
<i>Begin</i>	признак начала scbs-текста, в котором гарантировано отсутствие неявной омонимии
<i>End;</i>	признак конца scbs-текста, в котором гарантировано отсутствие неявной омонимии
=	scbs-разделитель несинонимии (в частности, омонимии) scb-элементов

Далее рассмотрим более подробно правила формирования идентификаторов scb-элементов.

SCB-идентификаторы можно классифицировать по целому ряду признаков:

- откуда заимствован идентификатор:
 - русскоязычные идентификаторы;
 - англоязычные идентификаторы;
 - условные обозначения;
- наличие сокращений:
 - аббревиатурные идентификаторы (всевозможные сокращения);
 - неаббревиатурные идентификаторы;
- семантика идентификатора:
 - имена собственные;
 - имена нарицательные;
- наличие вхождений других идентификаторов:
 - простые идентификаторы (атомарные);
 - сложные идентификаторы (составные, производные, неатомарные) – это scb-идентификаторы, в состав которых входят scb-идентификаторы, соответствующие другим scb-элементам.

Итак, scb-идентификатор – это строковое изображение знака некоторого множества, т.е. изображение знака в виде строки (последовательности) символов, принадлежащих тому или иному алфавиту. Идентификаторы называют также именами, термами, символьными обозначениями.

Простой scb-идентификатор – это некоторое словосочетание, в состав которого не входят разделители языка SCBs. Заметим, что пробелы и символы подчеркивания не являются разделителями языка SCBs.

Примечание. Некоторые разделители и ограничители языка SCBs состоят из нескольких символов. Некоторые специальные символы (например дефис), используемые в таких разделителях и ограничителях, могут использоваться и в простых идентификаторах.

Суть сложного идентификатора (неатомарного, составного, производного идентификатора) – это описание того, как идентифицированный scb-элемент связан с другими уже идентифицированными scb-элементами. При этом указанные идентифицированные элементы могут иметь как простые, так и сложные идентификаторы. Сложный идентификатор иногда называют выражением (арифметическим выражением, теоретико-множественным выражением), сложным (неатомарным, составным, производным) именем, сложным (неатомарным, составным, производным) термом. Элементарными составляющими сложных идентификаторов являются:

- простые идентификаторы;
- имена функций (эти имена, строго говоря, идентификаторами не считаются);
- обозначения 2-мощных операций;
- разделители, используемые при формировании сложных идентификаторов;
- ограничители, используемые при формировании сложных идентификаторов.

Ниже при рассмотрении проблемы сокращения словаря (набора) используемых простых идентификаторов (см. подраздел 2.7), а также при рассмотрении теоретико-множественных и числовых соотношений (см. раздел 3) будут введены некоторые конкретные правила построения сложных идентификаторов.

Перечислим некоторые "стилистические правила" построения простых идентификаторов scb-элементов. Некоторые из правил будут иметь вид схем, в которых приняты следующие соглашения и обозначения:

- текст, оформленный ***жирным курсивом***, является непосредственной частью формируемого идентификатора;
- в схему могут быть включены комментарии, оформленные с помощью тех же разделителей, которые имеются в языке SCBs. В процессе формирования идентификатора эти комментарии опускаются;
- в текст схемы могут входить слова в скобках специального вида “**< >**”. Это означает, что вместо слов, написанных внутри таких скобок, следует подставить те слова, о которых говорится в формулировке правила (см. также подраздел 2.8).

Правило 2.6.1. Простые идентификаторы должны строиться на основе русскоязычного или англоязычного словаря так, что бы те, кто знает соответствующий язык, могли достаточно легко понять смысл идентификаторов, т. е. могли достаточно легко "прочитать" scb-текст.

Правило 2.6.2. Простые scb-идентификаторы, являющиеся именами нарицательными, пишутся с маленькой буквы, а scb-идентификаторы, являющиеся именами собственными, пишутся с большой буквы. Имя нарицательное – это "апелляция" к свойству всех элементов именуемого (обозначаемого) множества и только их, т.е. к свойству всех тех и только тех объектов, которые являются элементами именуемого множества. Имя собственное – это апелляция к свойству (смыслу) самого именуемого (обозначаемого) множества как единого целого или к смыслу знака этого множества.

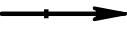
Правило 2.6.3. Если строка **< имя нарицательное >** есть простой scb-идентификатор, построенный как имя нарицательное, не начинающийся словом "**знак**" или словами "**быть знаком**", то этот идентификатор семантически эквивалентен (сионимичен) идентификатору, построенному как имя собственное по следующей схеме:

Знак множества всех тех /* всевозможных */ и только тех знаков, каждый из которых обозначает некоторый /* некоторую, некоторое */ < имя нарицательное > /* в соответствующем падеже */

Приведём в качестве примеров имена нарицательные и имена собственные, которые являются идентификаторами узлов, задающих те типы scb-элементов, каждому из которых соответствует свой графический примитив в языке SCBg (см. табл. 2.3.1).

Таблица 2.6.2. Синонимичные идентификаторы узлов, задающих типы scb-элементов

Графический примитив	SCBs-тексты, отражающие синонимию идентификаторов узлов
	<p>пара принадлежности = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности ;</p>

	<p>узловое множество</p> <ul style="list-style-type: none"> = множество, не являющееся парой принадлежности /* Запятая, являющаяся разделителем членов scbs-предложений, не может входить в идентификатор. В идентификаторах используется запятая специального вида (см. правило 2.6.10) */ = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторое узловое множество = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторое множество, <u>не являющееся парой принадлежности</u> ;
	<p>узловое непредметное множество</p> <ul style="list-style-type: none"> = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторое множество, не являющееся парой принадлежности <u>узловое непредметное множество</u> ;
	<p>пара непринадлежности</p> <ul style="list-style-type: none"> = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару непринадлежности ;
	<p>пара нечёткой принадлежности</p> <ul style="list-style-type: none"> = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару нечеткой принадлежности ;
	<p>неориентированная пара</p> <ul style="list-style-type: none"> = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую неориентированную пару ;
	<p>пара синонимии</p> <ul style="list-style-type: none"> = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару синонимии ;

Продолжение табл. 2.6.2

Графический примитив	SCBs-тексты, отражающие синонимию идентификаторов узлов
	<p><i>пара несинонимии</i></p> <p>= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару несинонимии ;</p>
	<p><i>пара нечёткой синонимии</i></p> <p>= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару нечеткой синонимии ;</p>
	<p><i>семейство пар принадлежности</i></p> <p>= множество <u>знаков</u> пар принадлежности</p> <p>= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторое множество знаков пар принадлежности ;</p>
	<p><i>семейство узловых множеств</i></p> <p>= множество знаков узловых множеств</p> <p>= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторое множество знаков узловых множеств ;</p>
	<p><i>семейство предметов</i></p> <p>= множество знаков предметных множеств</p> <p>= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторое множество знаков предметных множеств ;</p>
	<p><i>семейство узловых непредметных множеств</i></p> <p>= множество знаков узловых непредметных множеств</p> <p>= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторое множество знаков узловых непредметных множеств ;</p>
	<p><i>система множеств</i></p> <p>= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую систему множеств ;</p>

Окончание табл. 2.6.2

Графический примитив	SCBs-тексты, отражающие синонимию идентификаторов узлов
	<p><i>простая ориентированная пара</i> = Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую простую ориентированную пару ;</p>

Правило 2.6.4. Если строка `<имя нарицательное>` есть простой scb-идентификатор, который построен как имя нарицательное и начинается словами “*быть знаком*” или словом “*знак*”, либо включает слово “*знак*” как главное определяемое слово, то этот идентификатор семантически эквивалентен (синонимичен) идентификатору, построенному как имя собственное по следующей “схеме”:

Знак множества всех тех и только тех знаков, каждый из которых является <имя нарицательное> / в соответствующем падеже */*

На scbs-текстах 2.6.3 приведены примеры использования правила 2.6.4.

SCBs-текст 2.6.3. Пример формирования синонимичных идентификаторов по правилу 2.6.4

знак пары принадлежности
= Знак множества всех тех и только тех знаков, каждый из которых является знаком пары принадлежности ;

знак предметного множества
= предметный знак
= Знак множества всех тех и только тех знаков, каждый из которых является предметным знаком ;

Правило 2.6.5. Поскольку scb-элементы являются знаками и поскольку язык SCB оперирует множествами, состоящими только из знаков множеств (за исключением предметных множеств), слово “*знак*” в простых идентификаторах scb-элементов можно опускать (см. scbs-текст 2.6.4).

SCBs-текст 2.6.4. Пример формирования синонимичных идентификаторов по правилу 2.6.5

знак пары принадлежности
= пара принадлежности
= Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности
= Множество всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности;

Правило 2.6.6. Словосочетание “*всех тех и только тех знаков /** или объектов **/*, *каждый из которых обозначает /** или является **/ некоторым /** или некоторой **/*” заменяется в простых идентификаторах на слово “*всевозможных*” (см. scbs-текст 2.6.5).

SCBs-текст 2.6.5. Пример формирования синонимичных идентификаторов по правилу 2.6.6

<p><i>Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности</i></p> <p>= <i>Знак множества <u>всевозможных</u> знаков пар принадлежности ;</i></p>
--

Правило 2.6.7. Словосочетание “*множество всевозможных*” в простых идентификаторах преобразуется во множественное число (см. scbs-текст 2.6.6).

SCBs-текст 2.6.6. Пример формирования синонимичных идентификаторов по правилу 2.6.7

<p><i>множество всевозможных пар принадлежности</i></p> <p>= <i>пары принадлежности ;</i></p>

Правило 2.6.8. Если слово “*некоторых*” стоит после слова “*множество*” или слова “*семейство*”, то его можно исключить. Таким образом, в простых идентификаторах после слова “*множество*” или слова “*семейство*” по умолчанию подразумевается слово “*некоторых*”, а не слово “*всевозможных*” (см. scbs-текст 2.6.7).

SCBs-текст 2.6.7. Пример формирования синонимичных идентификаторов по правилу 2.6.8

<p><i>множество знаков пар принадлежности</i></p> <p>= <i>множество знаков <u>некоторых</u> пар принадлежности ;</i></p> <p style="text-align: center;"><i>/* = множество знаков <u>всевозможных</u> пар принадлежности */</i></p>
--

Правило 2.6.9. Если идентификатор, являющийся именем нарицательным, начинается словом “*быть*”, то его можно исключить (см. scbs-текст 2.6.8).

SCBs-текст 2.6.8. Пример формирования синонимичных идентификаторов по правилу 2.6.9

<p><i>быть парой принадлежности</i></p> <p>= <i>пара принадлежности ;</i></p> <p><i>быть знаком пары принадлежности</i></p> <p>= <i>знак пары принадлежности ;</i></p>

Правило 2.6.10. В состав идентификаторов могут входить запятые, но запятые специального вида “*,*”, которые не следует путать с запятыми вида “*,*”, используемыми в языке SCBs в качестве разделителей (см. подраздел 2.8). Кроме того, в идентификаторах допускается использование таких знаков, как дефис “*-*”, тире “*–*”, знак подчеркивания “*_*”.

Проиллюстрируем перечисленные правила формирования простых scb-идентификаторов в языке SCBs на примере множества идентификаторов, синонимичных идентификатору “*пара принадлежности*” (см. scbs-текст 2.6.9).

SCBs-текст 2.6.9. Пример формирования синонимичных идентификаторов по различным правилам

пара принадлежности

- = *Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности*
/* результат преобразования имени нарицательного в эквивалентное имя собственное */
- = *Знак множества всевозможных знаков пар принадлежности*
/* результат замены словосочетания "всех тех и только тех ..., которые" */
- = *Множество всевозможных знаков пар принадлежности*
/* результат исключения слова "знак" */
- = *Множество всевозможных пар принадлежности*
/* результат исключения слова "знак" */
- = *пары принадлежности*
/* результат преобразования слова "множество" во множественное число */
- = *быть парой принадлежности*
- = *быть знаком пары принадлежности*
- = *знак пары принадлежности*
- = *Знак множества всех тех и только тех знаков, каждый из которых является знаком некоторой пары принадлежности ;*

Следует также отметить, что преобразование имени нарицательного в имя собственное, осуществлённое путём изменения "размера" первой буквы идентификатора, меняет семантику идентификатора (см. scbs-тексты 2.6.10 и 2.6.11).

SCBs-текст 2.6.10. Пример несинонимичного преобразования идентификатора

пара принадлежности

- = *Пара принадлежности ;*

Здесь имя собственное "**Пара принадлежности**" обозначает какую-то одну конкретную пару принадлежности в отличие от имени нарицательного "**пара принадлежности**".

Если необходимо таким путем идентифицировать несколько конкретных пар принадлежности, то для каждой такой пары в идентификатор необходимо вводить дополнительный индивидуальный признак, например, индексы: "**Пара принадлежности.1**", "**Пара принадлежности.2**" и т. д.

SCBs-текст 2.6.11. Пример несинонимичного преобразования идентификатора

Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности

- = *знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности ;*

Здесь второй идентификатор (имя нарицательное) является знаком множества, состоящего из одного элемента, каковым является знак, изображаемый первым идентификатором. Дело в том, что здесь в имени нарицательном явно указывается такое свойство, каким обладает только один объект.

Замена в идентификаторе словосочетаний "всех тех и только тех" на слово "некоторых" меняет семантику идентификатора (см. scbs-текст 2.6.12).

SCBs-текст 2.6.12. Пример несинонимичного преобразования идентификатора

Знак множества всех тех и только тех знаков, каждый из которых обозначает некоторую пару принадлежности / имя нарицательное */*

= *Знак множества некоторых, /* т.е. не всех */ знаков, каждый из которых обозначает некоторую пару принадлежности ;*

Здесь множество, обозначаемое вторым идентификатором, содержит больше одного элемента, в отличие от множества, обозначаемого первым идентификатором.

Следует также помнить о том, что добавление к имени нарицательному слова “**множество**” меняет смысл идентификатора (см. scbs-текст 2.6.13).

SCBs-текст 2.6.13. Пример несинонимичного преобразования идентификатора

*Множество всевозможных пар принадлежности
/* = пара принадлежности */*

= *множество пар принадлежности ;
/* = множество некоторых пар принадлежности */*

Множество всевозможных треугольников

/ = треугольник */*

= *множество треугольников ;
/* = множество некоторых треугольников */*

В табл. 2.6.3 для некоторых ранее введённых идентификаторов приведены синонимичные им англоязычные и аббревиатурные scb-идентификаторы.

Таблица 2.6.3. Англоязычные и аббревиатурные идентификаторы основных понятий языка SCB

Русскоязычный идентификатор	Англоязычный идентификатор	Аббревиатурный идентификатор
<i>пара принадлежности</i>	= <i>arc;</i>	
<i>узловое множество</i>	= <i>node;</i>	
<i>предметное множество</i>	= <i>object</i>	= <i>obj;</i>
<i>узловое непредметное множество</i>	= <i>set;</i>	
<i>множество знаков пар принадлежности</i>		= <i>setArc;</i>
<i>множество знаков узловых множеств</i>		= <i>setNode;</i>
<i>множество знаков предметных множеств</i>		= <i>setObj;</i>

Окончание табл. 2.6.3

Русскоязычный идентификатор	Англоязычный идентификатор	Аббревиатурный идентификатор
<i>множество знаков узловых непредметных множеств</i>	= <i>setSet</i> ;	
<i>система множеств</i>	= <i>structure</i>	= <i>str</i> ;
<i>сложноструктурированная система множеств</i>		= <i>strStr</i> ;

При формировании идентификатора scb-элемента может быть условным образом указан тип соответствующего scb-элемента. Приведем некоторые рекомендуемые правила построения условных обозначений, используемых в качестве простых scb-идентификаторов в табл. 2.6.4. Здесь символ “ \rightarrow ” обозначает scbs-разделитель принадлежности (см. подраздел 2.8).

Таблица 2.6.4. Условные обозначения, используемые в scb-идентификаторах для указания типа scb-элемента

Идентификатор scb-узла, обозначающего множество scb-элементов соответствующего типа	"Шаблон" условного обозначения для scb-элемента указанного типа
<i>неуточняемый scb-элемент</i>	\rightarrow <i>e-< строка ></i> ; /* без пробелов слева и справа от дефиса */
<i>arc</i>	\rightarrow <i>g-< строка ></i> ;
<i>node</i>	\rightarrow <i>v-< строка ></i> ;
<i>obj</i>	\rightarrow <i>o-< строка ></i> ;
<i>set</i>	\rightarrow <i>s-< строка ></i> ;
<i>setArc</i>	\rightarrow <i>sg-< строка ></i> ;
<i>setNode</i>	\rightarrow <i>sv-< строка ></i> ;
<i>setObj</i>	\rightarrow <i>so-< строка ></i> ;
<i>setSet</i>	\rightarrow <i>ss-< строка ></i> ;
<i>str</i>	\rightarrow <i>c-< строка ></i> ;

Окончание табл. 2.6.4

Идентификатор scb-узла, обозначающего множество scb-элементов соответствующего типа	"Шаблон" условного обозначения для scb-элемента указанного типа
<i>strStr</i>	→ <i>cc-< строка ></i> ;

При формировании scb-идентификаторов также допустимо использование натуральных чисел в качестве индексов. Индексы от первой части такого идентификатора и друг от друга отделяются точкой. Например:

*e.1 , e.2 , ..., e.1.1 , e.1.2 , ... ,
g.1 , g.2 , ..., g.1.1 , g.1.2 , ... ,*

Точка в языке SCB используется также в качестве разделителя целой и дробной части числа в составе идентификатора известного числа. В этом случае идентификатор числа совпадает с его десятичным представлением.

В заключение рассмотрим в качестве примера правила формирования идентификатора библиографического источника, которые используются в данной книге. Для каждого библиографического источника вводится идентификатор, представляющий собой его аббревиатуру. Этот идентификатор указывается сразу после номера библиографического источника. В списке литературы библиографические источники упорядочиваются по алфавиту их идентификаторов.

Идентификатор библиографического источника оформляется ***жирным курсивом***, так как он трактуется как идентификатор предметного scb-узла, обозначающего текст, опубликованный в виде указанного библиографического источника.

Идентификаторы библиографических источников оформляются по следующим правилам (здесь используются средства расширенного языка Бэкуса-Наура, которые подробнее рассмотрены в подразделе 2.8):

1. Если идентификатор библиографической ссылки содержит фамилию первого автора или научного редактора, то:

```
< идентификатор библиографического источника > ::=  
< фамилия и инициалы первого автора или научного редактора >  
< вторая точка > /* если указывается первый, но не единственный автор */  
ред. || сост. /* если указанная выше фамилия является фамилией не первого автора, а научного  
редактора или составителя */  
< год издания >  
[* кн || bk || уч || спр || ст || арт || сб || слв || отч || rep || пр || тез || дис || мо || http ||  
mes *]  
/* указание типа библиографического источника:  
  кн – книга (монография); bk – иноязычная книга (монография); уч – учебник, учебное пособие,  
  методическое пособие и т.п.; спр – справочник; ст – статья; арт – иноязычная статья; сб – сборник; слв  
  – словарь (в частности, энциклопедический словарь); отч – отчет; rep – иноязычный отчет; пр –  
  препринт; тез – тезисы; дис – диссертация; мо – материалы по математическому обеспечению ЭВМ;  
  http – ссылка на источник в Интернет; mes – иноязычное сообщение и т.п. */
```

< дефис >

```
< первые пять букв первого слова названия библиографического источника >  
/* если первым словом является предлог, то указывается первая буква этого предлога (малым шрифтом), а  
потом (с большой буквы и без пробела) указываются пять букв второго слова */
```

< первые буквы остальных слов названия >

```
/* большими буквами и без пробелов; малыми буквами пишутся первые буквы предлогов, которые не следуют  
за словами, сокращаемыми до пятибуквенных аббревиатур */
```

2. Если идентификатор библиографической ссылки не содержит фамилий, то в начале идентификатора указывается сокращенное название библиографической ссылки, а именно:

```
< идентификатор библиографического источника > ::=  
< первые пять букв первого слова названия библиографического источника >
```

```

/* если первым словом является предлог, то указывается первая буква этого предлога (малым шрифтом), а
потом (с большой буквы и без пробела) указываются пять букв второго слова */
<первые буквы остальных слов названия>
/* большими буквами и без пробелов; малыми буквами пишутся первые буквы предлогов, которые не следуют
за словами, сокращаемыми до пятибуквенных аббревиатур */
<дефис>
<год издания>
[* kn || bk || уч || spr || ст || art || сб || слв || отч || rep || пр || тез || дис || мо || http ||
mes *]
/* указание типа библиографического источника */

```

Например, идентификатор данной книги, сформированный по первому правилу, примет вид “*Голенков В.В.ред.2001кн-ПредстИОЗвГАМ*”, а по второму правилу – “*ПредстИОЗвГАМ-2001кн*”.

Упражнения к подразделу 2.6.

Упражнение 2.6.1. Верны ли следующие scbs-тексты:

Знак множества = знак множества ;

узел = знак множества ;

1.7. Приведение текстов языка SCBs к лаконичному виду

Ключевые понятия: scbs-разделитель принадлежности; scbs-разделитель инцидентности; сложное scbs-предложение; сложный scbs-идентификатор.

Как уже было отмечено, текст языка SCBs есть последовательность предложений, которая изображает некоторое множество троек принадлежности.

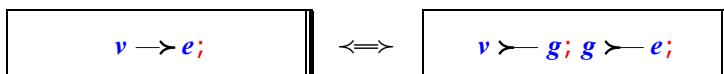
Очевидно, что с помощью рассмотренных выше scbs-конструкций 1-го и 2-го вида можно изобразить (представить) любое множество троек принадлежности, имеющих любую конфигурацию связи между собой (эти конфигурации, в частности, задаются тем, какие элементы изображаемых троек принадлежности совпадают между собой). Тем не менее, для обеспечения более лаконичного изображения множества троек принадлежности, кроме указанных выше типов конструкций, введем еще несколько видов, каждому из которых можно поставить в соответствие некоторое правило лаконизации (правило сокращения). Каждое из таких правил приводит либо к сокращению числа вхождений идентификаторов в состав текста, либо вообще к сокращению общего числа вводимых идентификаторов.

Правило 2.7.1. Изображение неидентифицируемой (неименуемой) scb-дуги.

В языке SCBs совсем не обязательно вводить идентификатор каждой scb-дуги. В частности, в этом нет никакой необходимости, если в scb-дугу не входит другая scb-дуга. В таком случае тройка принадлежности будет выглядеть следующим образом:



Эти два варианта эквивалентны любой из записей, приведенных на scbs-тексте 2.5.1. Например:



Здесь разделители “ \rightarrow ” и “ \leftarrow ” будем называть **scbs-разделителями принадлежности**. Их следует отличать от **scbs-разделителями инцидентности** “ \succ ” и “ \prec ” (см. подраздел 2.5).

SCBs-конструкцию будем называть элементарной scbs-конструкцией 3-го вида.

SCBs-конструкцию

$e \leftarrow v;$

будем называть элементарной scbs-конструкцией 4-го вида.

Справедливо следующее эквивалентное преобразование элементарных scbs-предложений 3-го вида в элементарные scbs-предложения 4-го вида и наоборот:

$v \rightarrow e;$

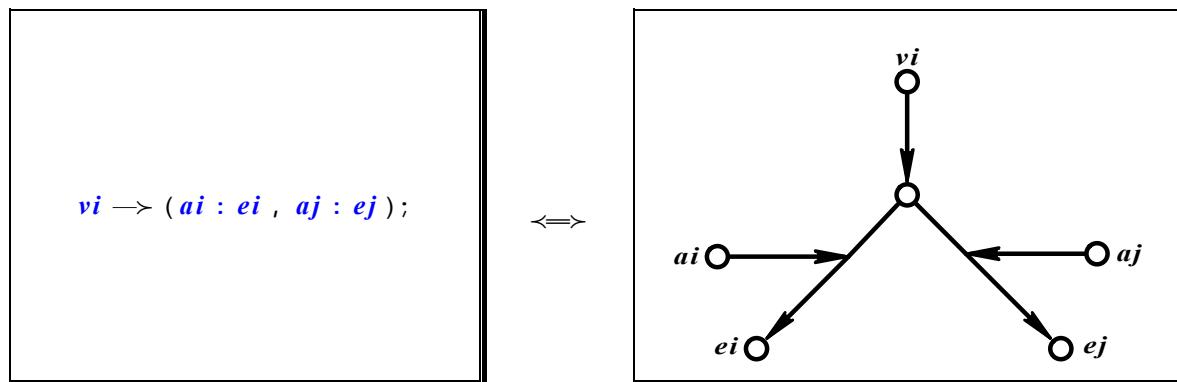
$e \leftarrow v;$

Приведём семантическую трактовку указанных элементарных scbs-конструкций 3-го и 4-го вида:

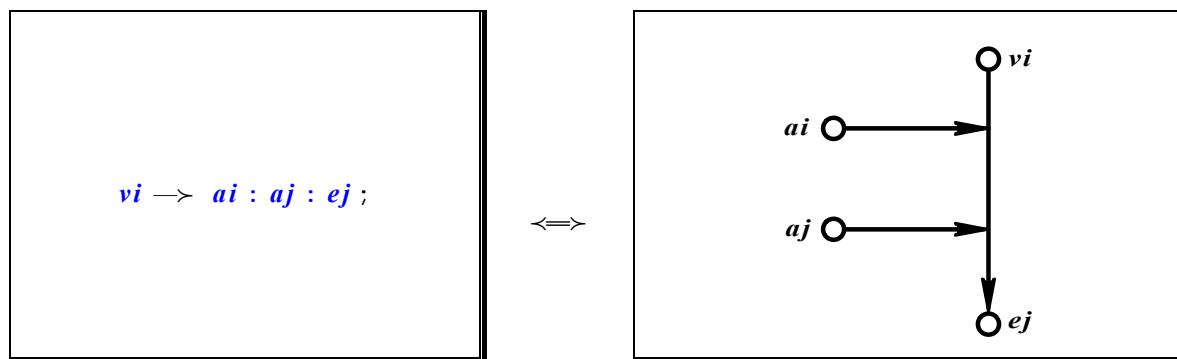
- scb-элемент e принадлежит множеству v , т. е. множеству, обозначаемому scb-элементом v , каковым, согласно свойствам языка SCB, может быть только непредметный scb-узел;
- scb-элементы v и e связаны парой принадлежности, выходящей из scb-элемента v и входящей в scb-элемент e ;
- scb-элементы v и e связаны scb-дугой, выходящей из scb-элемента v и входящей в scb-элемент e .

В случае, если в некоторую дугу входит другая дуга, ее также можно задать неявно, используя знак двоеточия “ : ”. Например, такой способ описания неидентифицируемой scb-дуги используется при задании элементов кортежа, имеющих атрибуты (см. scbs-текст 2.7.1), либо для описания вхождений нескольких scb-дуг в одну и ту же scb-дугу (см. scbs-текст 2.7.2).

SCBs-текст 2.7.1. Использование двоеточия для описания кортежа

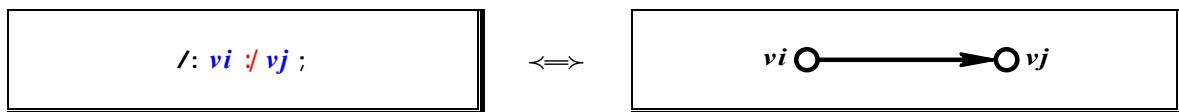


SCBs-текст 2.7.2. Использование двоеточия для описания вхождения нескольких scb-дуг в одну и ту же scb-дугу

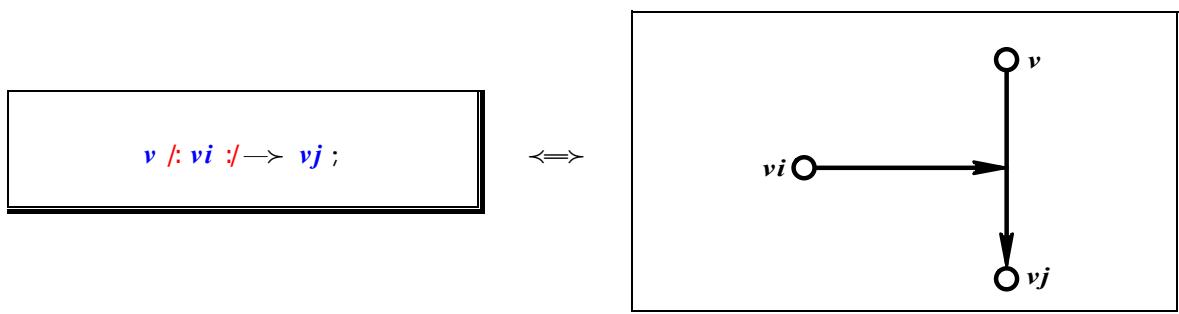


Помимо перечисленных выше ограничителей комментария и содержимого sc-элемента в языке SCBs используются также парные ограничители вида “ /: ” (левый) и “ \: ” (правый). Указанные ограничители используются для описания вхождения некоторой sc-дуги (идентификатор которой задается между указанными ограничителями) в некоторый элемент, который указан сразу после правого ограничителя (см. scbs-текст 2.7.3 и см. scbs-текст 2.7.4).

SCBs-текст 2.7.3. Пример использования ограничителей вида “`/:`” и “`;/`”



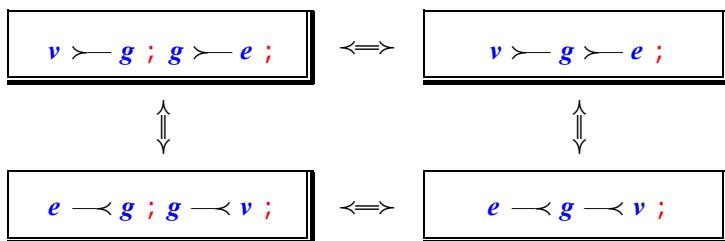
SCBs-текст 2.7.4. Пример использования ограничителей вида “`/:`” и “`;/`”



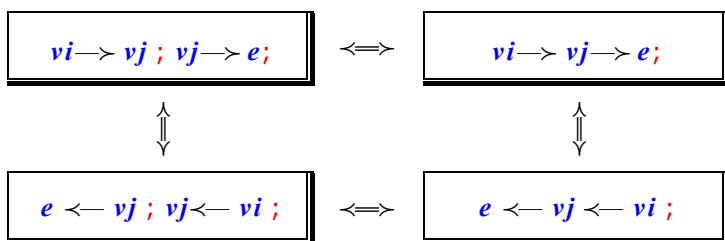
Правило 2.7.2. Сокращение числа вхождений идентификаторов.

Если какое-либо scbs-предложение завершается некоторым разделителем и некоторым идентификатором, с которого начинается какое-либо другое предложение, то эти предложения можно соединить в одно, "склеивая" указанные вхождения заданного идентификатора (см. scbs-тексты 2.7.5 – 2.7.8).

SCBs-текст 2.7.5. Сокращение числа вхождений идентификаторов для разделителя инцидентности



SCBs-текст 2.7.6. Сокращение числа вхождений идентификаторов для разделителя принадлежности



SCBs-текст 2.7.7. Сокращение числа вхождений идентификаторов для разделителя синонимии



SCBs-текст 2.7.8. Сокращение числа вхождений идентификаторов для "разнонаправленных" разделителей принадлежности



Правило 2.7.3. Сокращение числа вхождений идентификаторов для совокупности элементарных scbs-предложений. Если в нескольких элементарных scbs-предложениях одного вида совпадают первые или вторые идентификаторы, то их вхождения можно склеить, перечислив идентификаторы через запятую “ , ”, являющуюся разделителем идентификаторов в scbs-предложении (см. scbs-текст 2.7.9). Полученное scbs-предложение будем называть **сложным scbs-предложением**.

SCBs-текст 2.7.9. Сокращение числа вхождений идентификаторов для совокупности элементарных scbs-предложений, включающих одинаковые идентификаторы

$$\boxed{v \rightarrow e_1; v \rightarrow e_2; \dots; v \rightarrow e_n;} \Leftrightarrow \boxed{v \rightarrow e_1, e_2, \dots, e_n;}$$

Правило 2.7.4. Кроме сокращения числа вхождений используемых в тексте идентификаторов, целесообразно также сокращать и само количество используемых простых идентификаторов. Далеко не для каждого scb-элемента изображаемой scb-конструкции целесообразно вводить уникальный (простой) идентификатор. Простые идентификаторы целесообразно вводить для scb-узлов, которые соответствуют тем или иным понятиям (такие scb-узлы мы будем называть ключевыми), но для остальных scb-элементов целесообразно вводить не простые, а сложные идентификаторы. Каждый сложный идентификатор состоит из нескольких идентификаторов, каковыми могут быть как простые, так и сложные. Сложные идентификаторы строятся по принципу определения обозначаемого множества. Перечислим некоторые типы сложных идентификаторов (примеры использования указанных обозначений см. в разделе 3):

- $\boxed{(v \rightarrow e)}$ – обозначение неидентифицируемой scb-дуги, которая проведена из scb-узла с именем v в scb-элемент с именем e .
- $\boxed{\{ e_1, e_2, \dots, e_n \}}$ – обозначение неидентифицируемого scb-узла, являющегося знаком множества, в состав которого в качестве элементов входят все те, и только те знаки, которые изображаются scb-элементами с именами e_1, e_2, \dots, e_n .
- $\boxed{[\dots]}$ – обозначение неидентифицируемого scb-узла, являющегося знаком системы множеств, которая представлена в виде scbs-текста, ограниченного указанными квадратными скобками.
- $\boxed{(\dots)}$ – обозначение неидентифицируемого scb-узла, являющегося знаком кортежа, перечисление компонентов которого ограничено угловыми скобками.

Правило 2.7.5. Кроме scb-идентификаторов, которые взаимно однозначно соответствуют идентифицируемым scb-элементам, в языке SCBs используются строки символов, которые взаимно однозначно соответствуют некоторым типам пар и являются способом изображения знаков пар того или иного типа. Очевидно, что такое символическое изображение знаков пар является омонимичным, поскольку разные вхождения одинаковых строк символов такого вида в scb-текст являются изображениями знаков в общем случае разных пар. Перечислим используемые в языке SCBs такого рода способы изображения знаков пар в табл. 2.7.1.

Таблица 2.7.1. Изображения знаков пар различного типа в языке SCBs

Наименование	Изображение	
изображение знака простой ориентированной пары	\Rightarrow	\Leftarrow
изображение знака пары принадлежности	\rightarrow	\leftarrow
изображение знака пары непринадлежности	$\rightarrow\rightarrow$	$\leftarrow\leftarrow$
изображение знака пары нечеткой принадлежности	\rightsquigarrow	$\leftarrow\sim$
изображение знака пары включения множества	\subset	\supset

изображение знака пары невключение множества	$\not\subset$	\supset
изображение знака пары синонимии	=	
изображение знака пары несинонимии		\neq
изображение знака пары нечеткой синонимии		\approx
изображение знака пары равенства множеств		\equiv
изображение знака пары неравенства множеств		$\not\equiv$
изображение знака пары эквивалентности множеств по набору элементов		\sqsubseteq
изображение знака пары неэквивалентности множеств по набору элементов		$\not\sqsubseteq$
изображение знака пары пересекающихся множеств	\square	
изображение знака пары непересекающихся множеств		\blacksquare

В линейной (строковой) модификации языка SCB и его подъязыков большой текст (например, база знаний какой-либо предметной области) представляется в виде последовательности словарных статей. Каждая такая словарная статья является представлением некоторого фрагмента семантической окрестности соответствующего scb-элемента. Указанный scb-элемент назовем ключевым scb-элементом словарной статьи. Перечислим правила оформления таких словарных статей (см. scbs-текст 2.7.10):

- словарные статьи упорядочиваются по алфавиту идентификаторов своих ключевых scb-элементов;
- разделителем между словарными статьями является строка из не менее пяти дефисов;
- первым предложением каждой словарной статьи является идентификатор ключевого scb-элемента этой словарной статьи;
- если входящее в словарную статью предложение начинается с идентификатора ключевого scb-элемента этой словарной статьи, то указанный идентификатор можно опустить. Признаком предложений, полученных в результате такого преобразования, является то, что они начинаются с какого-либо разделителя языка SCBs;
- входящие в словарную статью предложения рекомендуется начинать с новой строки;
- идентификатор ключевого scb-элемента словарной статьи, входящий в предложение этой словарной статьи (кроме первого предложения), может быть заменен специальным символом “~”;
- в состав словарной статьи могут входить не только предложения языка SCBs, но и конструкции языка SCBg.

Содержательно словарная статья может включать следующие компоненты:

- синонимы;
- омонимы (если имеются);
- близкие (синтаксически похожие) идентификаторы;
- частные понятия (в том числе разбиение);
- общие понятия;
- примеры;
- пояснения;
- библиографические ссылки;
- и др.

С В с -т екст 2.7.10 . Пример оформления словарной статьи базы знаний

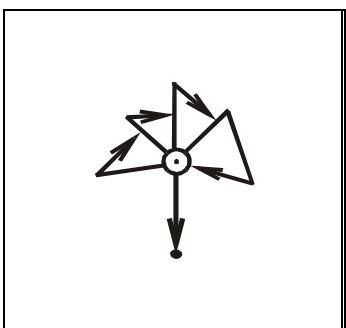
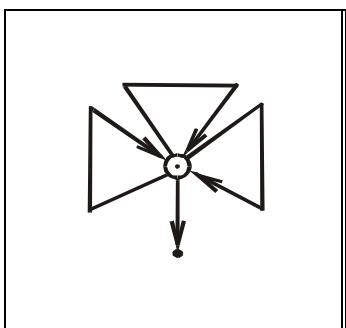
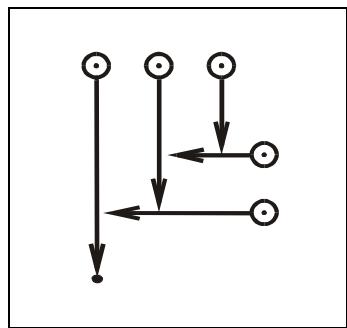
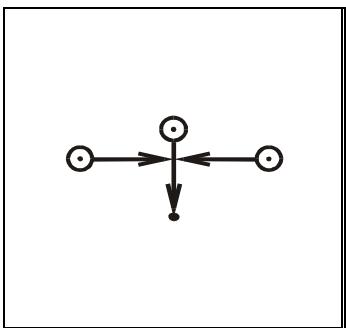
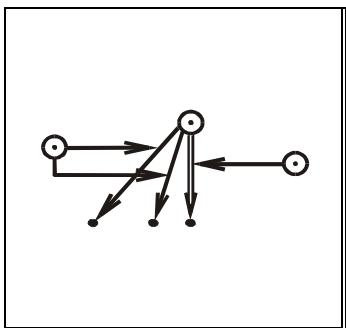
```
/*
attr_ ;                                     /* подробнее см. раздел 3, 4 */
← главный синоним ;
= attribute_ ;                               /* перечисление синонимов */
= attribute sign_ ;
= быть знаком атрибута_ ;
/* оформление комментария */

/: комментарий :/ => /"Ключевой узел attr_ является знаком специального атрибута,
используемого некоторыми метаотношениями. В частности, в метаотношении relConstruct
(relation construct) атрибут attr_ указывает на знаки атрибутов, используемых каждой конкретной
реляционной структурой, и относится к группе ключевых узлов, используемых для задания
реляционной структуры"/;

/* оформление библиографической ссылки */
=> комментарий :ПредстИОЗвГАМ-2001кн-п4.4 ;
```

Упражнения к подразделу 2.7.**Упражнение 2.7.1.**

Запишите на языке SCBs в нескольких вариантах следующие scbg-тексты.

**1.8. Формальное описание синтаксиса языка SCBs**

Ключевые понятия: метаязык описания синтаксиса, расширенный язык Бэкуса-Наура, метаконстанта, метапеременная.

В качестве метаязыка для описания синтаксиса языка SCBs используется расширенный метаязык (расширенная форма) Бэкуса-Наура. Этот метаязык широко используется для описания синтаксиса линейных языков, к которым относится и язык SCBs.

Основными лексемами расширенного метаязыка Бэкуса-Наура являются:

- метаконстанта, изображаемая жирным курсивом;
- метапеременная, которая ограничивается специальными угловыми скобками, и значениями которой являются конструкции описываемого языка, т.е. различные строки символов;
- условное обозначение пустой (нулевой, отсутствующей) конструкции, т.е. пустой строки символов, которое изображается строкой < *null* > .

Выражением расширенного языка Бэкуса-Наура является:

- метаконстанта;
- метапеременная;
- обозначение пустой строки;
- необязательное выражение (оно ограничивается специальными квадратными скобками со звёздочкой);
- конкатенация выражений, изображаемая в виде простой последовательности выражений;
- многократная конкатенация одного и того же выражения, заключенного в специальные фигурные скобки со звёздочкой;
- перечень альтернатив – это выражение изображается в виде последовательности альтернативных выражений, отделенных друг от друга двойной вертикальной чертой. Если перечень альтернатив является компонентом конкатенации выражений, он ограничивается специальными круглыми скобками со звёздочкой.

На множестве рассмотренных выражений определены некоторые правила эквивалентного преобразования выражений. Так, например, конкатенация выражений, каждое из которых является перечнем альтернатив, эквивалентна выражению, которое является общим перечнем (объединением) всех этих альтернатив.

Текст расширенного метаязыка Бэкуса-Наура есть последовательность предложений. Предложение расширенного языка Бэкуса-Наура состоит из следующих элементов:

- некоторой определяемой метапеременной;
- разделителя левой и правой части предложения (этот разделитель изображается строкой “ ::= ” ;
- некоторого выражения, которое уточняет структуру конструкций, являющихся значениями определяемой метапеременной.

Все разделители и ограничители рассмотренного метаязыка приведены в табл. 2.8.1.

Примечания. Все разделители и ограничители используемого нами метаязыка Бэкуса-Наура не могут совпадать с какими-либо разделителями и ограничителями языков SCBs, SCs, SCLs.

Т а б л и ц а 2 . 8 . 1 . Разделители и ограничители метаязыка Бэкуса-Наура

Символ		Назначение
<	>	скобки, ограничивающие лингвистические переменные
[*	*]	скобки, ограничивающие конструкцию, которая может отсутствовать
{*	*	скобки, ограничивающие конструкцию, которая может повторяться сколько угодно раз
(*	*)	скобки, ограничивающие список альтернатив. Выражение, ограниченное этими скобками, обозначает какую-либо одну из перечисленных альтернатив
		разделитель альтернатив
::=		разделитель, задающий операцию "присваивания" левой части правила Бэкуса-Наура, конструкцию, описанную в правой его части
< <i>null</i> >		обозначение пустой строки

Текст любого линейного языка (в том числе и языка SCBs) представляет собой последовательность лексем. Лексема (слово) – это элементарная осмысленная конструкция линейного языка. К числу лексем относятся знаки объектов и понятий описываемой предметной области, а также всевозможные разделители и ограничители символьных конструкций обеспечивающие им структуризацию. Наиболее часто используемые лексемы (например, некоторые разделители и ограничители) могут быть односимвольными конструкциями.

< *scbs-текст* > ::=

```
Begin < пробелы > < фамилия и инициалы автора > < пробелы > < дата > ;
{* < scbs-предложение > ; *}
End;
```

< scbs-предложение > ::=

```
< перечень scb-идентификаторов бинарного отношения >
[* {* < scbs-связка > [* < перечень scb-идентификаторов с атрибутами > *} *]
```

< перечень scb-идентификаторов > ::=

```
[* < метки > *] < scb-идентификатор >
[* {*, [* < метки > *] < scb-идентификатор > *} *]
```

< метки > ::=

```
/: < scb-идентификатор >
[* {*, < scb-идентификатор > *} *] :/
```

< перечень scb-идентификаторов с атрибутами > ::=

```
[* < атрибуты > *] [* < метки > *] < scb-идентификатор >
[* {*, [* < атрибуты > *] [* < метки > *] < scb-идентификатор > *} *]
```

< перечень scb-идентификаторов с атрибутами > ::=

```
[* < атрибуты > *] [* < метки > *] < scb-идентификатор >
[* {*, [* < атрибуты > *] [* < метки > *] < scb-идентификатор > *} *]
```

< атрибуты > ::=

```
{* < scb-идентификатор > : *}
```

< scbs-связка бинарного отношения > ::=

	/* связки инцидентности */ /* связки принадлежности */ /* связки непринадлежности */ /* связки нечеткой принадлежности */ /* ориентированные связки с дополнительно уточняемой семантикой */
--	--

$\Leftarrow\Rightarrow$	\parallel	<i>/* неориентированная связка с дополнительно уточняемой семантикой */</i>
\sqsubset	\sqsubset	<i>/* связки строгого включения множеств */</i>
\sqsubseteq	\sqsupseteq	<i>/* связки нестрогого включения множеств */</i>
∇	∇	<i>/* связки строгого невключение множеств */</i>
$\not\equiv$	$\not\equiv$	<i>/* связки нестрогого невключение множеств */</i>
\equiv	\equiv	<i>/* связка равенства множеств */</i>
\neq	\neq	<i>/* связка неравенства множеств */</i>
\vdash	\vdash	<i>/* связка эквивалентности множеств по набору элементов */</i>
\dashv	\dashv	<i>/* связка неэквивалентности множеств по набору элементов */</i>
\sqcap	\sqcap	<i>/* связка пересекающихся множеств */</i>
\sqcup	\sqcup	<i>/* связка непересекающихся множеств */</i>
$=$	$=$	<i>/* связка синонимии scb-элементов */</i>
\neq	\neq	<i>/* связка несинонимии scb-элементов */</i>
\approx	\approx	<i>/* связка нечеткой синонимии scb-элементов */</i>
$<$	$>$	$\parallel \leq \parallel \geq$ <i>/* связки сравнения чисел */</i>

$\ll \text{scb-идентификатор} \gg ::=$

$\ll \text{простой scb-идентификатор} \gg \parallel$
 $\ll \text{сложный scb-идентификатор} \gg$

$\ll \text{сложный scb-идентификатор} \gg ::=$

($\ll \text{scb-идентификатор} \gg \ll \text{scbs-связка бинарного отношения} \gg$
 $\ll \text{scb-идентификатор} \gg) \parallel$
{ $\ll \text{перечень scb-идентификаторов} \gg \parallel$
 $\text{!} \ll \text{перечень scb-идентификаторов с атрибутами} \gg \parallel$
[{ \ast $\ll \text{scbs-предложение} \gg ; \ast$ }] \parallel
/ " $\ll \text{информационная конструкция произвольного вида} \gg / \parallel$
 $\ll \text{имя унарной функции} \gg (\ll \text{scb-идентификатор} \gg) \parallel$
 $\ll \text{имя бинарной функции} \gg (\ll \text{scb-идентификатор} \gg , \ll \text{scb-идентификатор} \gg) \parallel$
($\ll \text{scb-идентификатор} \gg \ll \text{scbs-связка бинарной функции} \gg$
 $\ll \text{scb-идентификатор} \gg) \parallel$

$\ll \text{имя унарной функции} \gg ::=$

- $\parallel \underline{\text{abs}} \parallel ! \parallel \underline{\text{exp}} \parallel \underline{\text{ln}} \parallel \underline{\text{sin}} \parallel \underline{\text{cos}} \parallel \underline{\text{tg}} \parallel \underline{\text{ctg}}$

$\ll \text{имя бинарной функции} \gg ::=$

log

< scbs-связка бинарной функции > ::=

у || \wedge || \cap || \ || Δ || \times || + || \bullet || - || / || \uparrow || \vee

Выходы к разделу 2

Данный раздел был посвящен рассмотрению основных изобразительных средств графового языка SCB (Semantic Code Basic). В разделе 3 приводятся подробные примеры использования языка SCB для представления основных математических структур. В основе языка SCB лежат:

- базовые понятия теории множеств;
- чёткое противопоставление понятия множества, понятия знака множества и понятия изображения знака множества;
- введение специальных пар принадлежности, каждая из которых связывает знак некоторого множества с одним из элементов этого множества;
- ориентация на нормализованные множества, элементами которых являются знаки множеств;
- трактовка пар принадлежности как множеств и введение знаков пар принадлежности;
- трактовка всех элементов текстов языка SCB как знаков множеств.

Всё это даёт возможность на языке SCB описывать структуры любой степени сложности и легко переходить от описания к метаописанию.

1. Представление основных математических структур на языке SCB

Данный раздел посвящен рассмотрению способов представления основных математических структур на языке SCB (Semantic Code Basic), основные положения которого были приведены в разделе 2.

Данный раздел может быть использован в качестве учебного пособия по дисциплине «Математические основы искусственного интеллекта» для студентов специальности «Искусственный интеллект».

1.1. Типология множеств и их представление в языке SCB. Основные множества языка SCB и соответствующие им ключевые узлы

Ключевые понятия и идентификаторы ключевых scb-узлов:
пара; узловое множество; предмет; узловое непредметное множество; семейство пар принадлежности; семейство узловых множеств; система множеств; канторовское множество (классическое множество); мульти множество (неклассическое множество); рефлексивное множество; нерефлексивное множество; мощность множества; одномощное множество; 2-мощное множество; 3-мощное множество; семейство множеств одинаковой мощности; семейство множеств неодинаковой мощности; семейство одномощных множеств; семейство 2-мощных множеств; семейство 3-мощных множеств; количество элементов множества; одноэлементное множество; 2-элементное множество; конечно-мощное множество; бесконечно-мощное множество; конечно-элементное множество; бесконечно-элементное множество; конечно множество; бесконечное множество; включение множества (быть нестрогим подмножеством); строгое включение множества (быть строгим подмножеством, быть собственным подмножеством); равенство множеств; эквивалентность множеств по совпадению элементов; пересекающиеся множества; объединение множеств; пересечение множеств; разность множеств; симметрическая разность множеств; соединение множеств; разбиение множеств; приведение мульти множества к канторовскому виду; **множество (Универсальное нормализованное множество)**.

Как было отмечено в разделе 2, язык SCB обеспечивает представление (изображение, запись, задание) только нормализованных множеств, т.е. множеств, элементами которых являются знаки множеств. Но это, как тоже было отмечено, не снижает семантической мощности языка SCB, т.к. любое множество легко преобразуемо к нормализованному виду.

Прежде, чем перейти к рассмотрению типологии множеств, приведем несколько примеров использования изобразительных средств различных модификаций языка SCB.

Пусть имеется некоторое множество **s**, состоящее из элементов **s₁**, **s₂**, ..., **s_n**.

Следует отличать само множество **s** от знака этого множества, который является условно формируемым представителем обозначаемого им множества во всевозможных текстах (см. подраздел 2.1).

Следует также отличать само множество **s** от системы множеств, которая является представлением множества **s** и которая является множеством, включающим в себя в качестве элементов:

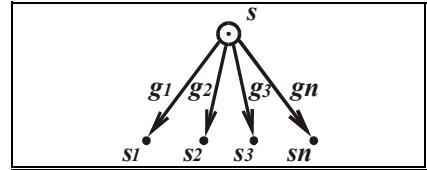
- все элементы множества **s**;
- знак множества **s**;
- знаки всех пар принадлежности, связывающих знак множества **s** с его элементами.

В нижеследующих примерах scbg-текстов 3.1.1, 3.1.2 и scbs-текстов 3.1.1 – 3.1.4 приведено представление (изображение, запись, задание) на языке SCB узлового непредметного множества, т.е. множества, не являющегося парой принадлежности и не являющегося предметным множеством. Здесь:

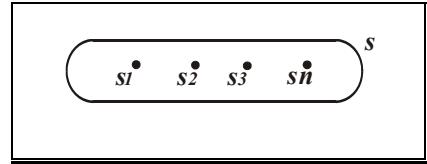
- узел с идентификатором **s** есть знак некоторого нормализованного множества, представлением (изображением) которого является данная scbg-конструкция;

- scb-элементы с идентификаторами s_1, s_2, \dots, s_n есть такие знаки множеств, которые являются элементами множества s . Указанные знаки могут быть предметными узлами (т.е. знаками унарных ненормализованных множеств), дугами принадлежности (т.е. знаками пар принадлежности), непредметными узлами (т.е. знаками нормализованных множеств, не являющимися парами принадлежности).

SCBg-текст 3.1.1. Вариант 1g изображения узлового непредметного множества (с явным изображением дуг, выходящих из узла s)



SCBg-текст 3.1.2. Вариант 2g изображения узлового непредметного множества (с неявным изображением scb-дуг, выходящих из узла s)



SCBs-текст 3.1.1. Вариант 1s изображения узлового непредметного множества (с использованием именуемых scb-дуг, выходящих из узла s). Данный текст эквивалентен scbg-тексту 3.1.1)

```
s >— g1 >— s1 ; s >— g2 >— s2 ; s >— g3 >— s3 ; ... ; s >— gn >— sn ;
```

SCBs-текст 3.1.2. Вариант 2s изображения узлового непредметного множества (с использованием неименуемых scb-дуг, выходящих из узла s). Данный текст эквивалентен scbg-тексту 3.1.2)

```
s —> s1 ; s —> s2 ; s —> s3 ; ... ; s —> sn ;
```

SCBs-текст 3.1.3. Вариант 3s изображения узлового непредметного множества (с использованием неименуемых scb-дуг, выходящих из узла s , и уменьшением количества вхождений идентификаторов. Данный текст эквивалентен scbg-тексту 3.1.2 и scbs-тексту 3.1.2)

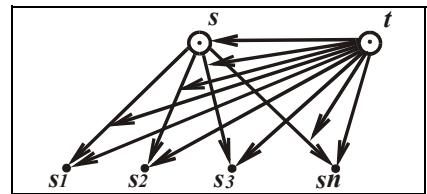
```
s —> s1 , s2 , s3 , ... , sn ;
```

SCBs-текст 3.1.4. Вариант 4s изображения узлового непредметного множества (с использованием неименуемых scb-дуг, выходящих из узла s , и scbs-разделителя синонимии). Данный текст эквивалентен scbg-тексту 3.1.2 и scbs-текстам 3.1.2 – 3.1.3)

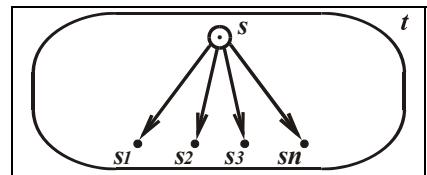
```
s = { s1 , s2 , s3 , ... , sn } ;
```

На scbg-текстах 3.1.3 – 3.1.4 и scbs-текстах 3.1.5 – 3.1.14 приведены примеры изображения на языке SCB множества, которое представляет собой систему множеств, являющуюся представлением множества s . Обозначим эту систему множеств идентификатором t .

SCBg-текст 3.1.3. Вариант 1g изображения системы множеств (с явным изображением дуг, выходящих из узла t)



SCB_g-текст 3.1.4. Вариант 2g изображения системы множеств (с неявным изображением дуг, выходящих из узла t)



SCB_s-текст 3.1.5. Вариант 1s изображения системы множеств

```
/* Явное описание дуг, выходящих из узла s (см. scbs-текст 3.1.1) */
s → g1 → s1; s → g2 → s2; s → g3 → s3; ...; s → gn → sn;

/* Явное описание дуг, выходящих из узла t и входящих во все узлы, обозначающие элементы множества s, и в знак самого множества s */
t → gs1 → s1; t → gs2 → s2; t → gs3 → s3; ...; t → gsn → sn;
t → gs → s;

/* Явное описание дуг, выходящих из узла t и входящих во все дуги, которые выходят из узла s и входят в узлы, обозначающие элементы множества s */
t → gg1 → g1; t → gg2 → g2; t → gg3 → g3; ...; t → ggn → gn;
```

SCB_s-текст 3.1.6. Вариант 2s изображения системы множеств

```
/* Явное описание дуг, выходящих из узла s (см. scbs-текст 3.1.1) */
s → g1 → s1; s → g2 → s2; s → g3 → s3; ...; s → gn → sn;

/* Неявное описание дуг, выходящих из узла t и входящих во все узлы, обозначающие элементы множества s, и в знак самого множества s */
t → s; t → s1; t → s2; t → s3; ...; t → sn;

/* Неявное описание дуг, выходящих из узла t и входящих во все дуги, которые выходят из узла s и входят в узлы, обозначающие элементы множества s */
t → g1; t → g2; t → g3; ...; t → gn;
```

SCB_s-текст 3.1.7. Вариант 3s изображения системы множеств

```
/* Явное описание дуг, выходящих из узла s (см. scbs-текст 3.1.1) */
s → g1 → s1; s → g2 → s2; s → g3 → s3; ...; s → gn → sn;

/* Неявное описание дуг, выходящих из узла t и входящих в знак множества s, во все узлы, обозначающие элементы множества s, и во все дуги, которые выходят из узла s и входят в узлы, обозначающие элементы множества s */
t → s, s1, s2, s3, ..., sn, g1, g2, g3, ..., gn;
```

SCBs-текст 3.1.8. Вариант 4s изображения системы множеств

```

/* Явное описание дуг, выходящих из узла s (см. scbs-текст 3.1.1) */
s >— g1 >— s1 ; s >— g2 >— s2 ; s >— g3 >— s3 ; ... ; s >— gn >— sn ;

/* Неявное (с использованием scbs-разделителя синонимии) описание дуг, выходящих из узла t и входящих в
знак множества s, во все узлы, обозначающие элементы множества s, и во все дуги, которые выходят из узла s и
входят в узлы, обозначающие элементы множества s */
t = { s, s1, s2, s3, sn, g1, g2, g3, ..., gn } ;

```

SCBs-текст 3.1.9. Вариант 5s изображения системы множеств

```

/* Неявное описание всех дуг, выходящих из узла t */
t —> s, s1, s2, s3, ..., sn, (s —> s1), (s —> s2), (s —> s3), ..., (s —> sn) ;

```

SCBs-текст 3.1.10. Вариант 6s изображения системы множеств

```

/* Комбинирование неявного и явного описания дуг */
t = { s, s1, s2, s3, ..., sn, (s —> s1), (s —> s2), (s —> s3), ..., (s —> sn) } ;

```

SCBs-текст 3.1.11. Вариант 7s изображения системы множеств

```

/* Использование обозначения scb-узла, являющегося знаком системы множеств */
t = [ s >— g1 >— s1 ; s >— g2 >— s2 ; s >— g3 >— s3 ; ... ; s >— gn >— sn ] ;

```

SCBs-текст 3.1.12. Вариант 8s изображения системы множеств

```

/* Использование обозначения scb-узла, являющегося знаком системы множеств, в комбинации с неявным
описанием дуг, выходящих из узла s */
t = [ s —> s1 ; s —> s2 ; s —> s3 ; ... ; s —> sn ] ;

```

SCBs-текст 3.1.13. Вариант 9s изображения системы множеств

```

/* Использование обозначения scb-узла, являющегося знаком системы множеств, в комбинации с неявным
описанием дуг, выходящих из узла s, и с сокращением числа вхождений идентификаторов */
t = [ s —> s1, s2, s3, ..., sn ] ;

```

SCBs-текст 3.1.14. Вариант 10s изображения системы множеств

```

/* Комбинированное использование обозначения scb-узла, являющегося знаком системы множеств, и обозна-
чения scb-узла, являющегося знаком узлового непредметного множества */
t = [ s = { s1, s2, s3, ..., sn } ] ;

```

Перейдем к рассмотрению типологии множеств.

По признакам нормализованности множества можно разбить на два следующих класса (см. подраздел 2.1):

- нормализованные множества (множества, все элементы которых являются знаками множеств);
- ненормализованные множества (множества, среди элементов которых имеется хотя бы один, не являющийся знаком какого-либо множества).

Напомним, что в языке SCB явно представить (изобразить) можно только нормализованное множество. При этом вспомним также, что любое ненормализованное множество можно привести к нормализованному виду (см. подраздел 2.1).

По признаку "является или не является множество парой принадлежности" множества разбиваются на два следующих класса:

- пары принадлежности – в языке SCB этот класс множеств обозначается ключевым узлом "**пара принадлежности**";
- узловые множества (т.е. множества, не являющиеся парами принадлежности) – в языке SCB этот класс множеств обозначается ключевым узлом "**узловое множество**".

В свою очередь семейство узловых множеств разбивается на следующие классы:

- предметные множества (унарные ненормализованные множества) – в языке SCB этот класс множеств обозначается ключевым узлом "**предмет**";
- узловые непредметные множества – в языке SCB этот класс множеств обозначается ключевым узлом "**узловое непредметное множество**".

В свою очередь семейство узловых непредметных нормализованных множеств разбивается на:

- множества, состоящие из знаков пар принадлежности, – в языке SCB этот класс множеств обозначается ключевым узлом "**семейство пар принадлежности**";
- множества, состоящие из знаков узловых множеств, – в языке SCB этот класс множеств обозначается ключевым узлом "**семейство узловых множеств**";
- полностью нормализованные системы множеств (т.е. множества, состоящие из знаков пар принадлежности и знаков узловых множеств) – в языке SCB этот класс множеств обозначается ключевым узлом "**система множеств**".

По признаку наличия многократного вхождения элементов множества делятся на:

- множества без кратных элементов (т.е. множества, все элементы которых входят в эти множества однократно, – такие множества называются канторовскими или классическими множествами [464] (*Таран Т.А. 1998гн-ОсновыДМ*) – в языке SCB этот класс множеств обозначается ключевым узлом "**канторовское множество**";
- множества с кратными элементами (т.е. множества, некоторые элементы которых входят в эти множества многократно, – такие множества называются мульти множествами, см. scb-текст 3.1.1) – в языке SCB этот класс множеств обозначается ключевым узлом "**мульти множество**".

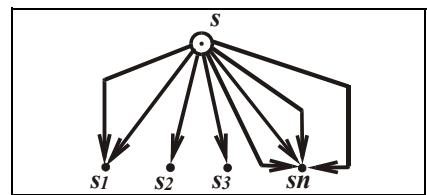
По признаку вхождения в число элементов множества собственного знака множества делятся на:

- рефлексивные множества (множества, которые включают в число своих элементов собственные знаки, см. scb-текст 3.1.2) – в языке SCB этот класс множеств обозначается ключевым узлом "**рефлексивное множество**";
- нерефлексивные множества – в языке SCB этот класс множеств обозначается ключевым узлом "**нерефлексивное множество**".

SCB-текст 3.1.1. Представление на обеих модификациях языка SCB множества с кратными элементами (с неоднократным вхождением некоторых элементов; такие множества будем называть мульти множествами):

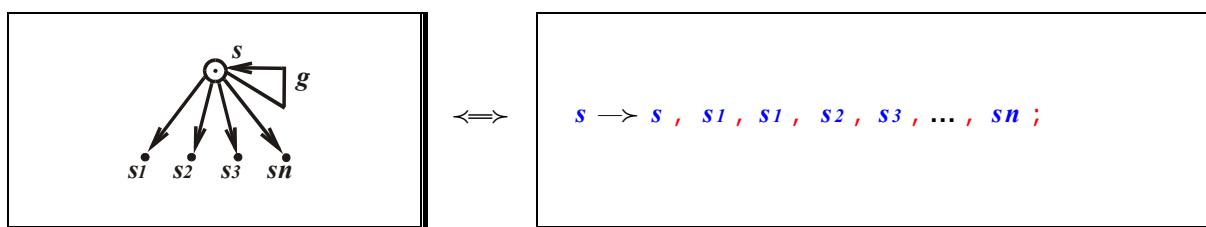
- знак, изображенный здесь scb-элементом s_1 , входит в качестве элемента во множество, знак которого изображен scb-элементом s как минимум двукратно;
- знак s_2 является как минимум однократным элементом множества s ;
- знак s_3 является как минимум однократным элементом множества s .

Знак s_n является как минимум четырехкратным элементом множества s .



$s \rightarrow s_1, s_1, s_2, s_3, s_n, s_n, s_n, s_n;$

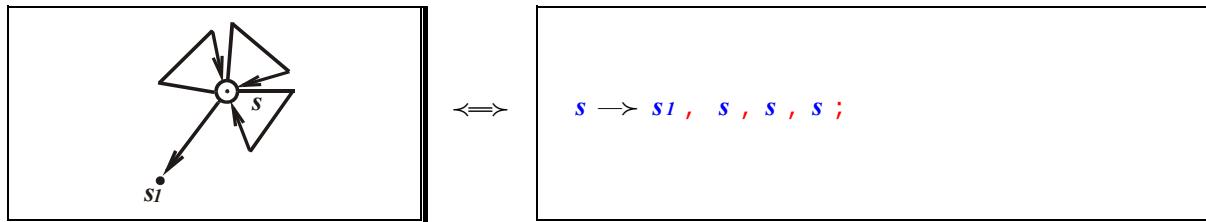
SCB-текст 3.1.2. Представление на обеих модификациях языка SCB рефлексивного множества (здесь петлевая scb-дуга g означает, что одним из элементов множества s является знак самого этого множества)



$s \rightarrow s, s_1, s_1, s_2, s_3, \dots, s_n;$

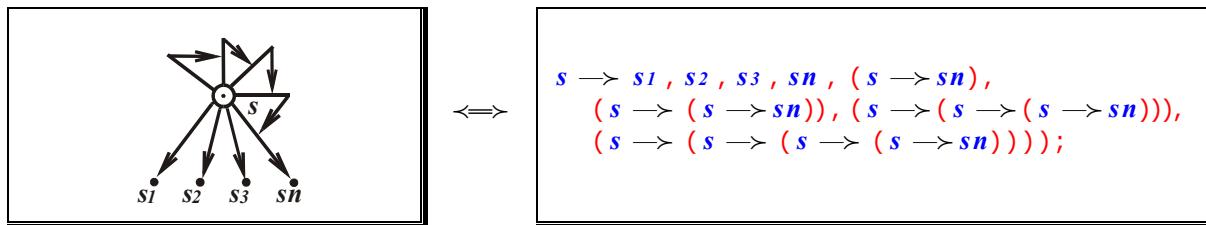
Приведем еще несколько примеров "экзотических" множеств (см. scb-тексты 3.1.3 – 3.1.5).

SCB-текст 3.1.3. Представление на обеих модификациях языка SCB рефлексивного множества с многократным вхождением знака самого себя



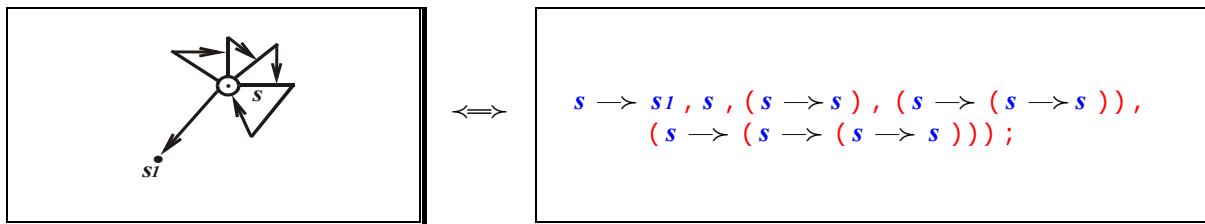
$s \rightarrow s_1, s, s, s;$

SCB-текст 3.1.4. Представление на обеих модификациях языка SCB множества с "рекурсивным" вхождением дуг



$s \rightarrow s_1, s_2, s_3, s_n, (s \rightarrow s_n), (s \rightarrow (s \rightarrow s_n)), (s \rightarrow (s \rightarrow (s \rightarrow s_n))), (s \rightarrow (s \rightarrow (s \rightarrow (s \rightarrow s_n))));$

С С В - т е к с т 3 . 1 . 5. Представление на обеих модификациях языка SCB рефлексивного множества с "рекурсивным" вхождением дуг (вариант 2)



Важнейшей измеряемой (числовой) характеристикой множества является **мощность множества** – общее количество вхождений всех элементов множества. По этому признаку множества делятся на следующие классы (перечислим scb-идентификаторы ключевых узлов, обозначающих эти классы):

- “**одномощное множество**” – класс множеств, которые состоят из одного однократно входящего в его состав элемента;
- “**2-мощное множество**” – класс множеств, мощность которых равна 2;
- “**3-мощное множество**” – класс множеств, мощность которых равна 3;
- И т.д.

Мощность нормализованного множества в языке SCB определяется количеством дуг принадлежности, выходящих из знака этого множества.

Понятие “мощность множества” позволяет выделить еще целый ряд классов множеств (перечислим scb-идентификаторы ключевых узлов, обозначающих эти классы):

- “**семейство множеств одинаковой мощности**” (быть семейством множеств одинаковой мощности);
- “**семейство множеств неодинаковой мощности**” (в состав каждого такого семейства входят по крайней мере два множества с различной мощностью).

В свою очередь множество знаков, обозначающих всевозможные семейства множеств одинаковой мощности, разбивается на следующие подклассы (перечислим соответствующие им scb-идентификаторы):

- “**семейство одномощных множеств**”;
- “**семейство 2-мощных множеств**”;
- “**семейство 3-мощных множеств**”;
- И т.д.

Можно говорить ещё об одной измеряемой характеристике множеств – **количество элементов**. Очевидно, что если множество не имеет кратных элементов, то его мощность и количество его элементов совпадают.

Согласно рассматриваемой характеристике, в семействе всевозможных множеств можно выделить следующие классы множеств:

- “**одноэлементное множество**” (множества, состоящие из одного элемента, – не путать с унарными множествами);
- “**2-элементное множество**”;
- И т.д.

На основании свойства “мощность множества” и свойства “количество элементов множества” можно выделить также следующие классы множеств (перечислим соответствующие scb-идентификаторы):

- “**конечно-мощное множество**” (множества, мощность которых является конечным числом);
- “**бесконечно-мощное множество**” (∞ - мощные множества);
- “**конечно-элементное множество**”;
- “**бесконечно-элементное множество**” (∞ - элементные множества).

Очевидно, что:

- могут существовать конечно-элементные, но ∞ - мощные множества;
- не существует множеств, которые являются ∞ - элементными, но конечно-мощными;
- каждое ∞ - элементное множество является ∞ - мощным.

Множество, являющееся конечно-мощным и соответственно конечно-элементным, будем называть **конечным множеством**, а класс таких множеств обозначим ключевым узлом с scb-идентификатором “**конечное множество**”. Множество, являющееся ∞ - мощным или ∞ - элементным, будем называть **бесконечным множеством**, а класс таких множеств обозначим scb-узлом с scb-идентификатором “**бесконечное множество**”.

Конечные множества, знаки которых входят в состав scb-текста, делятся на два важных класса:

- конечные множества, частично представленные в текущем состоянии scb-текста, – для каждого такого множества в текущем состоянии scb-текста присутствуют не все дуги принадлежности, выходящие из знака этого множества и входящие в его элементы;
- конечные множества, полностью представленные в текущем состоянии scb-текста, – для каждого такого множества в текущем состоянии scb-текста присутствуют все дуги принадлежности, выходящие из знака этого множества и входящие в его элементы.

Примечание. В процессе переработки информации состояния scb-текста может меняться. Изменение состояния scb-текста сводится к появлению новых scb-элементов (в частности, новых дуг принадлежности и узлов), а также к удалению (стиранию) имеющихся scb-элементов. В результате этого некоторые конечные частично представленные множества могут стать полностью представленными (если будут сгенерированы все элементы некоторого множества и все дуги принадлежности, выходящие из знака этого множества), а также некоторые полностью представленные множества могут стать частично представленными (если будут удалены некоторые дуги принадлежности, выходящие из знака полностью представленного множества). Все это должно учитываться при преобразовании scb-текста.

Пусть даны два множества — множество s_1 и множество s_2 . Рассмотрим то, как могут соотноситься между собой эти множества, и то, как эти отношения записываются в языке SCBs:

Таблица 3.1.1. Описание соотношений между множествами в языке SCBs

SCBs-текст	Комментарий
$s_1 \rightarrow s_2 ; /*$ или $s_2 \leftarrow s_1 */$	Знак множества s_2 является одним из элементов множества s_1
$s_1 \leftrightarrow s_2 ; /*$ или $s_2 \leftrightarrow s_1 */$	Знак множества s_2 <u>не</u> является одним из элементов множества s_1
$s_2 \rightarrow s_1 ; /*$ или $s_1 \leftarrow s_2 */$	Знак множества s_1 является одним из элементов множества s_2
$s_2 \leftrightarrow s_1 ; /*$ или $s_1 \leftrightarrow s_2 */$	Знак множества s_1 <u>не</u> является одним из элементов множества s_2
$s_1 \supseteq s_2 ; /*$ или $s_2 \subseteq s_1 */$	Множество s_2 является нестрогим подмножеством множества s_1 , т.е. <u>каждый</u> элемент множества s_2 является также элементом множества s_1 . При этом количество вхождений каждого элемента во множество s_2 <u>не</u> превышает количества вхождений <u>этого же</u> элемента во множество s_1
$s_1 \not\supseteq s_2 ; /*$ или $s_2 \not\subseteq s_1 */$	Множество s_2 <u>не</u> является нестрогим подмножеством множества s_1

Окончание табл. 3.1.1.

SCBs-текст	Комментарий
$s2 \supseteq s1 ; /*$ или $s1 \subseteq s2 ; */$	Множество $s1$ является нестрогим подмножеством множества $s2$
$s2 \not\supseteq s1 ; /*$ или $s1 \not\subseteq s2 ; */$	Множество $s1$ <u>не</u> является нестрогим подмножеством множества $s2$
$s1 \supset s2 ; /*$ или $s2 \subset s1 ; */$	Множество $s2$ является строгим подмножеством множества $s1$, т.е. каждый элемент множества $s2$ является элементом множества $s1$, при этом существует по крайней мере один элемент множества $s1$, не являющийся элементом множества $s2$ либо имеющий во множестве $s2$ меньшее количество вхождений
$s1 \not\supset s2 ; /*$ или $s2 \not\subset s1 ; */$	Множество $s1$ <u>не</u> является строгим подмножеством множества $s2$
$s2 \supset s1 ; /*$ или $s1 \subset s2 ; */$	Множество $s1$ является строгим (собственным) подмножеством множества $s2$
$s2 \not\supset s1 ; /*$ или $s1 \not\subset s2 ; */$	Множество $s1$ <u>не</u> является строгим подмножеством множества $s2$
$s1 = s2 ; /*$ или $s2 = s1 ; */$	Множество $s1$ и множество $s2$ равны, т.е. каждый элемент множества $s1$ является элементом множества $s2$ и наоборот. При этом, если какой-либо элемент в одно из этих множеств входит много раз, то в другое множество он входит столько же раз
$s1 \neq s2 ; /*$ или $s2 \neq s1 ; */$	Множество $s1$ и множество $s2$ <u>не</u> равны
$s1 \leftarrow s2 ; /*$ или $s2 \leftarrow s1 ; */$	Множество $s1$ и множество $s2$ содержат одни и те же элементы, но, возможно, с разной кратностью
$s1 \leftrightarrow s2 ; /*$ или $s2 \leftrightarrow s1 ; */$	Существует по крайней мере один элемент множества $s1$, не являющийся элементом множества $s2$, или наоборот
$s1 \square s2 ; /*$ или $s2 \square s1 ; */$	Множество $s1$ и множество $s2$ являются пересекающимися множествами, т.е. множествами, у которых имеется по крайней мере один общий элемент
$s1 \not\square s2 ; /*$ или $s2 \not\square s1 ; */$	Множество $s1$ и множество $s2$ <u>не</u> являются пересекающимися множествами

Примечание. Следует отличать:

- scb-элементы, которые являются знаками разных, но равных множеств (т.е. фактически являются знаками разных объектов);
- синонимичные scb-элементы, которые являются разными знаками, но одного и того же множества (т.е. разные знаки одного и того же объекта).

Подчеркнем при этом, что введение в формальные тексты равных множеств необходимо проводить весьма аккуратно – делать это тогда, когда без этого действительно трудно обойтись.

Кроме того, из двух заданных множеств можно формировать новые множества с помощью ряда теоретико-множественных операций. Рассмотрим некоторые из этих операций и приведем используемые в языке SCBs способы идентификации (обозначения) множеств, формируемых с помощью указанных теоретико-множественных операций.

Таблица 3.1.2. Сложные идентификаторы scb-элементов, формируемые с помощью теоретико-множественных операций

Идентификатор	Комментарий
$(s_1 \cup s_2)$ /* или $(s_2 \cup s_1)$ */	Множество, являющееся результатом <u>объединения множеств</u> s_1 и s_2 . В это множество входят все те и только те элементы, которые являются либо элементами множества s_2 , либо элементами множества s_1 . При этом, если некий элемент x входит в состав одного из указанных множеств n -кратно, а в состав другого m -кратно (причем $m \geq n$), то в состав множества $(s_1 \cup s_2)$ указанный элемент будет входить m -кратно. Кратность, равная нулю, означает отсутствие вхождений элемента во множество
$(s_1 \cap s_2)$ /* или $(s_2 \cap s_1)$ */	Множество, являющееся результатом <u>пересечения множеств</u> s_1 и s_2 . В это множество входят все те и только те элементы, которые являются как элементами множества s_1 , так и элементами множества s_2 . При этом, если некий элемент x входит в состав одного из указанных множеств n -кратно, а в состав другого m -кратно (причем $m \geq n$), то в состав множества $(s_1 \cap s_2)$ указанный элемент будет входить n -кратно
$(s_1 \setminus s_2)$	Множество, являющееся результатом <u>разности множеств</u> s_1 и s_2 . В это множество входят все те и только те элементы множества s_1 , которые <u>не являются</u> элементами множества s_2
$(s_1 \Delta s_2)$ /* или $(s_2 \Delta s_1)$ */	Множество, являющееся результатом <u>симметрической разности</u> множеств s_1 и s_2 . В это множество входят все те и только те элементы множества $(s_1 \cup s_2)$, которые <u>не являются</u> элементами множества $(s_1 \cap s_2)$. Очевидно, что сложные имена вида $(s_1 \Delta s_2)$ всегда синонимичны сложным именам вида $((s_2 \cup s_1) \setminus (s_1 \cap s_2))$, $((s_2 \setminus s_1) \cup (s_1 \setminus s_2))$
$(s_1 \sqcup s_2)$ /* или $(s_2 \sqcup s_1)$ */	Множество, являющееся результатом <u>соединения множеств</u> s_1 и s_2 . В это множество входят все те и только те элементы, которые являются либо элементами множества s_1 , либо элементами множества s_2 . При этом, если некий элемент x входит в состав одного из указанных множеств n -кратно, а в состав другого m -кратно, то в состав множества $(s_1 \sqcup s_2)$ указанный элемент будет входить $(n + m)$ -кратно

Будем говорить, что множество s_1 и множество s_2 являются разбиением множества s на два подмножества в том и только в том случае, если:

- $s = (s_1 \cup s_2)$ /* здесь знак равенства указывает на синонимию имени */;
- $s_1 \sqsupseteq s_2$;

Будем говорить, что множество s_2 является результатом приведения мульти множества s_1 к канторовскому виду в том и только в том случае, если:

-
- $s_1 \rightarrow s_2;$
 - $s_2 \leftarrow \text{канторовское множество} /* \text{множество без кратных элементов */};$

Универсальное нормализованное множество – это множество, по отношению к которому все остальные нормализованные множества являются подмножествами. В языке SCB универсальное нормализованное множество обозначается scb-идентификатором “ **множество** ”.

Итак, мы дополнительно ввели в язык SCBs:

- ряд новых разделителей, соответствующих различным связям между множествами (\rightarrow , \leftarrow , \supseteq , \subseteq , \neq , \supset , \subset , $\supseteq\!\!$, $\subset\!\!$, $=$, \neq , $-$, \perp , \square , $\square\!\!$);
- новые виды scbs-предложений (с указанными выше разделителями);
- ряд новых разделителей, соответствующих различным операциям над множествами (\cup , \cap , \backslash , Δ);
- новые виды сложных (производных) идентификаторов, которые по определенным правилам (с помощью разделителей \cup , \cap , \backslash , Δ) конструируются из других идентификаторов. В состав сложных идентификаторов непосредственно могут входить другие идентификаторы. Но любой сложный идентификатор, в конечном счёте, сводится к простым идентификаторам, а также к некоторому набору разделителей и ограничителей (скобок разного вида). О мотивах введения сложных идентификаторов в язык SCBs см. в правиле 2.7.4 лаконичного изображения scbs-текстов (см. подраздел 2.7).

В пункте 3.3.11 будет рассмотрено то, как трактуются в графическом языке SCBg введенные выше понятия, отражающие различные соотношения между множествами. В частности, будет рассмотрено то, каким scbg-конструкциям соответствует рассмотренное выше “теоретико-множественное” расширение языка SCBs.

Упражнения к подразделу 3.1.

Упражнение 3.1.1. Существуют ли конечно-мощные, но ∞ -элементные множества?

Упражнение 3.1.2. Могут ли бесконечные множества быть полностью представленными?

Упражнение 3.1.3. Пусть некоторые два scb-узла являются синонимичными. Следует ли из этого, что они являются знаками равных множеств?

Упражнение 3.1.4. Пусть имеется два равных множества. Следует ли из этого, что знаки этих множеств являются синонимичными?

1.2. Понятие кортежа. Атрибуты элементов кортежа. Представление кортежей в языке SCB. Типология кортежей

1.2.1. Понятие кортежа и атрибута

Ключевые понятия: кортеж; атрибут; числовой атрибут; неориентированное множество.

Кортеж – это множество, у которого каждому вхождению каждого его элемента явно или неявно (по умолчанию) ставится в соответствие некоторый **атрибут**, указывающий роль этого вхождения элемента в рамках рассматриваемого кортежа. Формально атрибут вхождений элементов в кортежи – это множество знаков пар принадлежности, связывающих знаки кортежей с такими вхождениями их элементов, которые в рамках указанных кортежей выполняют некоторую одинаковую роль. **Кортеж** – это множество, для которого существенным является не только набор вхождений элементов в это множество, но и дополнительное явное указание роли (атрибута) в рамках этого множества хотя бы одного вхождения какого-либо его элемента. Кортежи также называют ролевыми структурами, упорядоченными наборами, упорядоченными множествами, ориентированными множествами, векторами, **n**-

ками. В частном случае атрибуты могут быть числовыми. В кортеже с числовыми атрибутами все вхождения его элементов нумеруются от 1 до некоторого ***n***. **Числовой атрибут** – это условный порядковый номер вхождения элемента в кортеж. Количество всех вхождений в состав кортежа всех его элементов будем называть **мощностью кортежа**. Элемент кортежа, имеющий в рамках этого кортежа атрибут ***ai***, будем называть ***ai*-элементом** (***ai*-компонентом**) этого кортежа.

Один и тот же объект может быть элементом разных кортежей. При этом в рамках разных кортежей этот объект может иметь разные атрибуты.

Множество, не являющееся кортежем, будем называть **неориентированным множеством**.

Принципиальным свойством атрибутов является то, что указание того, к какому атрибуту относится та или иная пара принадлежности, выходящая из знака кортежа, задает семантику соответствующего кортежа, поскольку каждый атрибут задает определенную роль вхождений элементов в рамках кортежей. Из этого, в частности, следует то, что в отличие от неориентированных множеств (т. е. множеств, не являющихся кортежами), кортежи с совпадающими элементами, но с несовпадающими атрибутами вхождений этих элементов считаются неравными кортежами.

Один и тот же элемент в рамках одного кортежа может выполнять сразу несколько ролей, что соответствует нескольким вхождениям этого элемента, принадлежащим разным атрибутам. Каждому атрибуту, как и любому другому множеству, можно поставить в соответствие знак этого атрибута. Знак атрибута есть не что иное, как относительное понятие, определяющее свойство какого-либо объекта, имеющее место по отношению к каким-то другим объектам. Примеры знаков атрибутов: “***быть сыном_***”, “***быть суммой_***”, “***быть слагаемым_***”.

При идентификации знаков атрибутов в языке SCB введем следующее правило: последним символом идентификатора знака атрибута в языке SCB должен быть символ подчеркивания. Примеры идентификаторов атрибутов приведены на scbs-тексте 3.2.1.1.

SCBs-текст 3.2.1.1. Примеры идентификации атрибутов

```

сын_ = быть сыном_;
дочь_ = быть дочерью_;
отец_ = быть отцом_;
мать_ = быть матерью_;
непосредственный потомок_ = быть непосредственным потомком_;
непосредственный предок_ = быть непосредственным предком_;
потомок_ = быть потомком_;
предок_ = быть предком_;
сумма_ = быть суммой_;
слагаемое_ = быть слагаемым_;
произведение_ = быть произведением_;
сомножитель_ = быть сомножителем_;
старший по возрасту_ = быть старшим по возрасту_;
max_ = быть максимальным числом_; /* для заданного множества чисел */
min_ = быть минимальным числом_; /* для заданного множества чисел */
1_ = быть первым компонентом кортежа_;
2_ = быть вторым компонентом кортежа_;

```

Примечание. Последние два атрибута и аналогичные им являются числовыми атрибутами или порядковыми числами. Они указывают условный номер вхождения элементов (компонентов) в состав соответствующих кортежей.

Для изображения знака атрибута на языке SCBg используется графический примитив \oplus , соответствующий изображению знака семейства дуг принадлежности (см. табл. 2.3.1 в подразделе 2.3). Примеры использования этого графического примитива см. в пункте 3.2.2.

1.2.2. Примеры кортежей и их представление в языках SCBg и SCBs

Ключевые понятия и идентификаторы ключевых scb-узлов:
классический кортеж; **кортеж с числовыми атрибутами;** **пара принадлежности;** **знак множества_** (**быть знаком множества_**); **элемент множества_** (**быть элементом множества_**); **кортеж** (**быть кортежем, не являющимся парой принадлежности**); **ориентированная пара** (**2-мощный кортеж**); **двойная линия со стрелкой** (графический примитив языка SCBg, изображающий знак ориентированной пары); **петля**; **атрибут**; **задаваемый по умолчанию**.

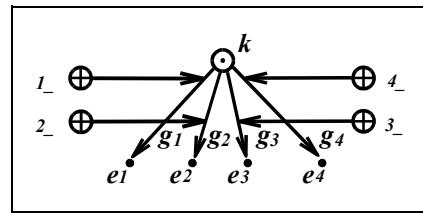
На scbg-текстах 3.2.2.1 – 3.2.2.2 и scbs-текстах 3.2.2.1 – 3.2.2.4 приведены примеры изображения 4-мощного (т.е. имеющего мощность, равную 4) **классического кортежа с числовыми атрибутами** без кратных элементов (т.е. при отсутствии многократного вхождения в состав кортежа хотя бы одного его элемента) на соответствующих модификациях языка SCB.

Приведем несколько вариантов изображения указанного кортежа на языке SCBg.

SCBg-текст 3.2.2.1. Вариант 1g изображения 4-мощного кортежа

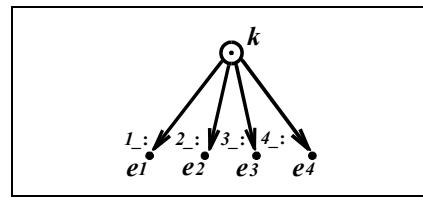
Здесь:

- **k** – знак кортежа;
- **e1, e2, e3, e4** – элементы этого кортежа.



SCBg-текст 3.2.2.2. Вариант 2g изображения 4-мощного кортежа

Идентификатор с двоеточием, приписываемый графическому изображению какого-либо scb-элемента (в данном случае – изображению scb-дуги) – это идентификатор scb-узла, из которого проведена scb-дуга в указанный scb-элемент.



Заметим, что идентификатор с двоеточием совсем не обязательно должен быть идентификатором знака атрибута (см. scbg-текст 2.4.4). Более того, идентификатор с двоеточием совсем не обязательно должен припisyваться изображениям только scb-дуги – это могут быть изображения scb-узлов, а также scb-элементов неуточняемого типа.

SCBs-текст 3.2.2.1. Вариант 1s изображения 4-мощного кортежа

```
/* Использование идентифицируемых дуг */
k > g1 > e1 ; k > g2 > e2 ; k > g3 > e3 ; k > g4 > e4 ;
1_- > g1 ; 2_- > g2 ; 3_- > g3 ; 4_- > g4 ;
```

SCBs-текст 3.2.2.2. Вариант 2s изображения 4-мощного кортежа

```
/* Использование неидентифицируемых дуг */
1_- > (k > e1) ; 2_- > (k > e2) ; 3_- > (k > e3) ; 4_- > (k > e4) ;
```

SCBs-текст 3.2.2.3. Вариант 3s изображения 4-мощного кортежа

```
/* Использование scb-узла, обозначающего кортеж */
k = (1_- : e1 , 2_- : e2 , 3_- : e3 , 4_- : e4) ;
```

SCBs-текст 3.2.2.4. Вариант 4s изображения 4-мощного кортежа

```
/* Использование scb-узла, обозначающего кортеж. Здесь числовые атрибуты подразумеваются по умолчанию */
k = ( e1 , e2 , e3 , e4 );
```

Кортеж будем называть **классическим кортежем** в том и только том случае, если он обладает следующими свойствами:

- 1) каждому вхождению элемента в состав кортежа (т.е. каждой scb-дуге, выходящей из знака кортежа) соответствует один и только один атрибут;
- 2) не существует двух различных вхождений в состав кортежа разных элементов (или одного и того же элемента), отмеченных одним и тем же атрибутом.

Введём ключевой scb-узел “**классический кортеж**”, обозначающий множество классических кортежей. Кортеж будем называть **классическим кортежем с числовыми атрибутами** в том и только в том случае, если:

- 1) кортеж является классическим;
- 2) в кортеже используются только числовые атрибуты от i_1 до n , где n – мощность кортежа, т.е. количество scb-дуг, выходящих из знака кортежа.

Аналогично введём ключевой scb-узел “**кортеж с числовыми атрибутами**”, обозначающий множество классических кортежей с числовыми атрибутами. Пара принадлежности, введенная нами в подраздел 2.1, является примером классического кортежа, который имеет мощность, равную 2 (т.е. является 2-мощным множеством), и которому соответствуют атрибуты “**знак множества**” и “**элемент множества**”.

Существенное отличие пар принадлежности от других кортежей, не являющихся парами принадлежности, заключается в том, что в языках SCB, SCBg и SCBs представление пар принадлежности (см. подраздел 2.3 и подраздел 2.5) и представление кортежей, не являющихся парами принадлежности, осуществляется принципиально разным образом. Связь знака пары принадлежности с элементами этой пары изображается инцидентностью соответствующих scb-элементов. В то время как связь знака кортежа, не являющегося парой принадлежности, с элементами этого кортежа изображается смежностью соответствующих scb-элементов, т.е. с помощью scb-дуги, соединяющей эти scb-элементы.

Введем ключевой scb-узел, обозначающий множество знаков всевозможных кортежей, не являющихся парами принадлежности. В языке SCBg этот узел можно изобразить либо с помощью графического примитива \odot , либо с помощью графического примитива Θ . Указанному scb-узлу припишем идентификатор “**кортеж**” (**быть кортежем, не являющимся парой принадлежности**). При этом введем правило 3.2.2.1 изображения на языке SCBg кортежа, не являющегося парой принадлежности.

Правило 3.2.2.1. Следующие изображения кортежа, не являющегося парой принадлежности, являются эквивалентными:

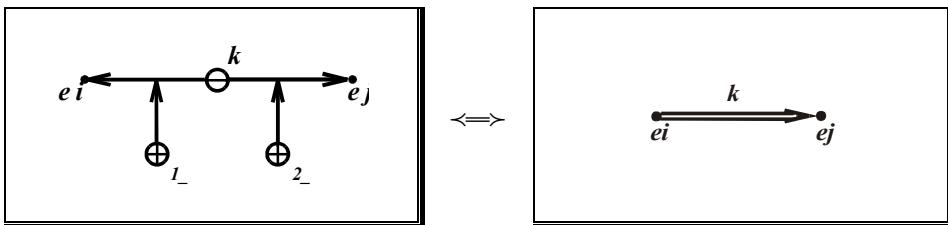


Таким образом, введен новый графический примитив Θ , который явно указывает принадлежность изображаемого узла ко множеству, обозначаемому scb-узлом с идентификатором “**кортеж**”. Иными словами, данный графический примитив явно задает тип изображаемого scb-узла как знака кортежа, не являющегося парой принадлежности.

Для более компактного изображения в языке SCBg 2-мощных классических кортежей (пар), не являющихся парами принадлежности, в языке SCBg имеется еще один графический примитив – **двойная линия** со стрелкой на одном конце. Такая двойная линия является изображением знака 2-мощного кортежа, не являющегося парой принадлежности. Знак 2-мощного кортежа будем называть **дугой**. Следовательно, двойная линия со стрелкой – это изображение дуги, не являющейся scb-дугой (знаком пары принадлежности). Таким образом, для языка SCBg можно ввести правило 3.2.2.2 ком-

пактного изображения 2-мощных классических кортежей (простых ориентированных пар, см. подраздел 2.4, scbg-текст 2.4.10).

Правило 3.2.2.2. Следующие изображения кортежа, не являющегося парой принадлежности, являются эквивалентными:

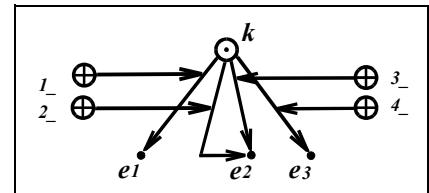


Рассмотрим несколько примеров изображения кортежей.

На scbg-тексте 3.2.2.3 и эквивалентном ему scbs-тексте 3.2.2.5 приведен пример изображения 4-мощного классического кортежа (четверки) с числовыми атрибутами и с кратными элементами (с многократным вхождением по крайней мере одного элемента).

С C B g - т е к с т 3 . 2 . 2 . 3 . Пример изображения 4-мощного классического кортежа с числовыми атрибутами и с кратными элементами

Здесь scb-элемент e_2 входит дважды в состав кортежа k (один раз под атрибутом 2_- , а второй раз – под атрибутом 3_-).



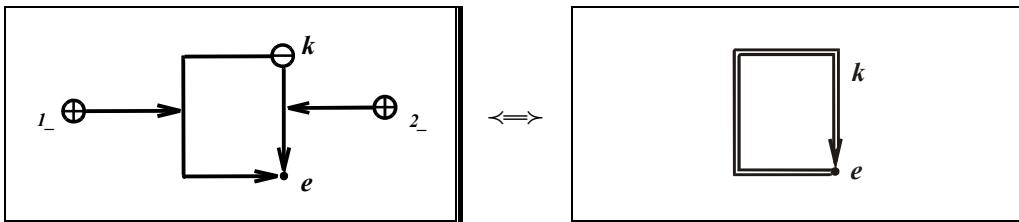
С C B s - т е к с т 3 . 2 . 2 . 5 . Пример изображения 4-мощного классического кортежа с числовыми атрибутами и с кратными элементами

```
 $k = (1_- : e1, 2_- : e2, 3_- : e2, 4_- : e3); /* Или k = (e1, e2, e2, e3); */$ 
```

Примечание. Подчеркнем, что в классическом кортеже допустимо многократное вхождение элементов.

На scbg-тексте 3.2.2.4 приведен пример изображения 2-мощного классического кортежа (пары) с кратным вхождением элемента. Такую пару будем называть **петлей**, множество всех петель и только петель обозначим ключевым scb-узлом “**петля**”.

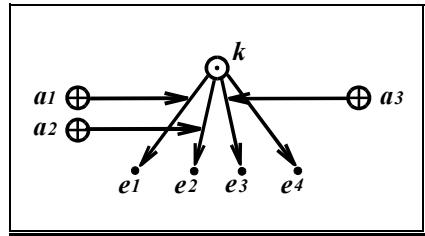
С C B g - т е к с т 3 . 2 . 2 . 4 . Пример изображения 2-мощного классического кортежа (пары) с кратным вхождением элемента



На scbg-тексте 3.2.2.5 приведен пример изображения кортежа с неявно заданным атрибутом. Такой атрибут будем называть **атрибутом по умолчанию**.

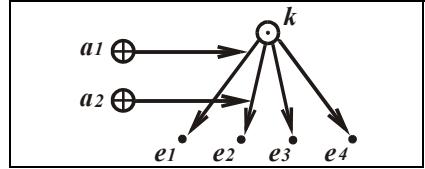
SCBg-текст 3.2.2.5. Пример изображения кортежа с атрибутом по умолчанию

Здесь отсутствие указания атрибута для вхождения scb-элемента e_4 в кортеж k означает не отсутствие атрибута у этого вхождения, а то, что этот неявно указываемый (т.е. указываемый по умолчанию) атрибут отличен от атрибутов a_1 , a_2 , a_3 .



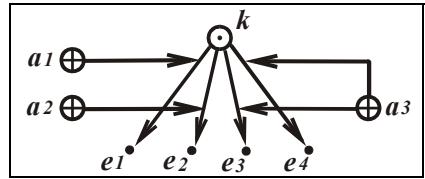
Примечание. Если несколько вхождений элементов в кортеж не имеют явно указываемого атрибута, то считается, что указанные вхождения имеют один и тот же задаваемый по умолчанию атрибут, отличающийся от всех остальных атрибутов, используемых в данном кортеже (см. scbg-текст 3.2.2.6).

SCBg-текст 3.2.2.6. Пример изображения кортежа, в котором несколько вхождений элементов отмечены атрибутом, задаваемым по умолчанию



SCBg-текст 3.2.2.7. Пример изображения кортежа, в котором несколько вхождений элементов имеют одинаковые атрибуты

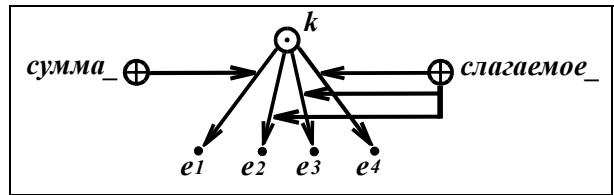
Здесь вхождения scb-элементов e_3 и e_4 в кортеж k имеют одинаковый атрибут a_3 .



На scbg-тексте 3.2.2.8 приведен пример изображения кортежа, связывающего набор некоторых чисел с их суммой. Это конкретный содержательный пример кортежа, в котором несколько вхождений элементов имеют одинаковые атрибуты.

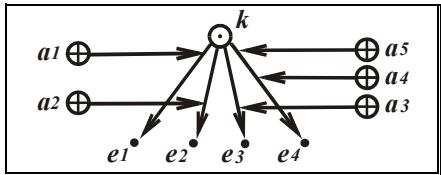
SCBg-текст 3.2.2.8. Пример изображения кортежа, связывающего набор некоторых чисел с их суммой

Очевидно, что противопоставлять друг другу различные слагаемые (например, нумеровать их) нет никакой необходимости.



SCBg-текст 3.2.2.9. Пример изображения кортежа, в котором по крайней мере одно вхождение какого-либо элемента имеет несколько различный атрибутов

Здесь вхождение элемента e_4 в кортеж k имеет два атрибута (атрибут a_4 и атрибут a_5).



1.2.3. Типология кортежей

Ключевые понятия: дуга (знак 2-мощного кортежа); кортеж, все компоненты которого имеют не более одного атрибута; кортеж, все компоненты которого имеют разные атрибуты; классический кортеж; неклассический кортеж; кортеж над предметами; кортеж над узловыми непредметными множествами; кортеж над системами множеств; числовой кортеж; метакортеж (кортеж над кортежами).

Поскольку кортежи являются множествами специального вида, их можно классифицировать по всем общим признакам классификации множеств, которые приведены в подразделе 2.3. Согласно этому, можно выделить:

1) по признаку кратности элементов:

- кортежи без кратных элементов;

-
- кортежи с кратными элементами (т.е. кортежи, у каждого из которых имеется по крайней мере один элемент, который входит в состав кортежа более, чем однократно) – см. пример на scfg-тексте 3.2.2.4;
- 2) по признаку рефлексивности:
- нерефлексивные кортежи;
 - рефлексивные кортежи (т.е. кортежи, каждый из которых включает в число элементов знак самого себя). Очевидно, что рефлексивные кортежи являются достаточно редким видом кортежей;
- 3) по мощности:
- 2-мощные кортежи (пары);
 - 3-мощные кортежи (тройки);
 - 4-мощные кортежи (четверки);
 - 5-мощные кортежи (пятерки);
 - и т.д.

Знаки 2-мощных кортежей (ориентированных пар) будем называть **дугами**.

Анализ различных вариантов соотношения вхождений элементов в кортежи и соответствующих им атрибутов позволяет выделить следующие классы кортежей:

- 1) по признаку наличия неявно указываемых атрибутов:
 - кортежи, у которых каждому вхождению элемента соответствует по крайней мере один явно указываемый атрибут;
 - кортежи, у которых имеется по крайней мере одно вхождение какого-либо элемента с неявно указываемым атрибутом (т.е. с атрибутом, указываемым по умолчанию);
- 2) по признаку количества атрибутов у компонентов кортежа:
 - кортежи, у которых каждому вхождению каждого элемента соответствует один и только один атрибут (явно или неявно указываемый);
 - кортежи, у которых имеется по крайней мере одно вхождение какого-либо элемента, которому соответствует несколько явно указываемых атрибутов;
- 3) по признаку однозначности атрибутов у компонентов кортежа:
 - кортежи, у которых не существует двух таких вхождений каких-либо элементов, которым соответствует один и тот же явно или неявно задаваемый атрибут (кортежи без симметрии);
 - кортежи, у которых имеется по крайней мере два таких вхождения каких-либо элементов, которым соответствует один и тот же явно или неявно задаваемый атрибут (кортежи с симметрией).

Важнейшим классом кортежей являются **классические кортежи**. Соответственно этому можно выделить класс **неклассических кортежей**, т.е. кортежей, не являющихся классическими.

По виду атрибутов можно выделить:

- кортежи с нечисловыми атрибутами;
- кортежи со смешанными (числовыми и нечисловыми) атрибутами.

По виду элементов кортежей можно выделить:

- кортежи, элементами которых являются знаки предметных множеств;
- кортежи, элементами которых являются знаки узловых непредметных множеств;
- кортежи, являющиеся знаками систем множеств;
- кортежи, элементами которых являются числа;
- кортежи, среди элементов которых имеются знаки кортежей (такие кортежи будем называть метакортежами).

Упражнения к пункту 3.2.3.

Упражнение 3.2.3.1. Могут ли классические кортежи быть неклассическими множествами? Если да, то приведите пример.

Резюме к подразделу 3.2

В заключение рассмотрения понятия кортежа и особенностей его использования в языке SCB необходимо сделать ряд примечаний.

Примечание 1. Почти всегда 2-мощные кортежи (пары) являются классическими, кроме тех случаев, когда какое-либо вхождение какого-либо элемента 2-мощного кортежа имеет несколько явно указываемых атрибутов, а также случаев, когда оба вхождения элементов в 2-мощный кортеж имеют одинаковые атрибуты.

Примечание 2. Если в классическом кортеже один из соответствующих ему атрибутов будет задаваться по умолчанию, то это не превращает указанный кортеж в неклассический.

Примечание 3. Любой кортеж, все атрибуты которого заданы явно, можно заменить на эквивалентный кортеж, в котором один из указанных атрибутов (любой) задается неявно (по умолчанию).

Примечание 4. Любой кортеж с нечисловыми или смешанными атрибутами можно заменить на эквивалентный кортеж с числовыми атрибутами, "пронумеровав" исходные атрибуты.

Примечание 5. Могут существовать кортежи, состоящие из одинаковых элементов, но при наличии хотя бы одного элемента, который входит в разные кортежи под разными атрибутами. Такие по-разному ориентированные кортежи (т.е. кортежи, являющиеся равными множествами, но с разным распределением ролей их элементов) будем называть **встречными кортежами**. Примерами встречных кортежей являются **встречные пары**, например, встречные пары принадлежности и соответствующие им встречные scb-дуги. Примерами встречных пар также являются:

- пара, указывающая наличие канала передачи информации из пункта **a** в пункт **b**, и пара, указывающая наличие канала передачи информации из пункта **b** в пункт **a** (далеко не все пункты имеют двустороннюю связь);
- пара, указывающая наличие гомоморфизма из структуры **a** в структуру **b**, и пара, указывающая наличие гомоморфизма из структуры **b** в структуру **a** (далеко не для каждой пары гомоморфных структур имеет место обратный гомоморфизм).

Примечание 6. Могут существовать кортежи, которые состоят из одинаковых элементов с одинаковыми атрибутами. Такие кортежи будем называть **равными** или **кратными кортежами**. Примерами кратных кортежей являются кратные пары, например, кратные пары принадлежности и соответствующие им кратные scb-дуги.

Примечание 7. Могут существовать вырожденные кортежи, в которых все вхождения элементов имеют одинаковые атрибуты.

Примечание 8. Могут существовать кортежи, в которых атрибуты вхождений его элементов однозначно задаются типами этих элементов. В таких кортежах атрибуты можно явно не указывать. В качестве примера таких кортежей можно привести 2-мощный кортеж с атрибутами:

- быть знаком точки, которая лежит на прямой, знак которой является другим компонентом кортежа;
- быть знаком прямой, на которой лежит точка, знак которой является другим компонентом кортежа.

Очевидно, что первым из указанных атрибутов могут обладать только геометрические точки, а вторым – только прямые. Следовательно, если множество "**быть точкой**" и множество "**быть прямой**" будут введены явно, то указанные выше атрибуты можно явно не вводить.

Примечание 9. С формальной точки зрения, каждый атрибут есть множество, каждый элемент которого является знаком пары принадлежности. При этом нет оснований считать ложным обратное утверждение. Другими словами, каждое семантически осмыслившее множество, состоящее из знаков пар принадлежности, можно считать атрибутом для соответствующих кортежей. Таким образом, любое множество при соответствующем рассмотрении можно трактовать как кортеж. Поэтому противопоставление неориентированных множеств и кортежей выглядит весьма условно. Неориентированное множество можно трактовать как множество, для элементов которого пока не выявлено различие в ролях этих элементов в рамках этого множества.

Примечание 10. В математической литературе обычно имеют дело с классическими кортежами, использующими числовые атрибуты и изображаемыми в этой литературе следующим образом:

(x₁, x₂, ..., x_n), где x₁ – 1-й компонент кортежа, x₂ – 2-й компонент, ..., x_n – n-й компонент.

1.3. Понятие отношения. Представление отношений в языке SCB. Типология отношений. Классические и неклассические отношения

1.3.1. Обобщение традиционной трактовки отношений

Ключевые понятия и идентификаторы ключевых scb-узлов:
отношение; нормализованное отношение; связка отношения; классическое отношение; отношение.

Каждое отношение общего вида, т.е. отношение, понимаемое в широком смысле слова, представляет собой некоторое семейство однотипных в каком-то смысле множеств. В состав отношений могут входить как неупорядоченные множества, так и кортежи. В частности, каждое отношение классического вида представляет собой множество классических кортежей, имеющих одинаковую арность и использующих одинаковые наборы атрибутов. Переходя от множеств, в частности, кортежей, входящих в состав отношения, к их нормализованным эквивалентам и переходя от трактовки отношения как семейства знаков множеств к его трактовке как семейства знаков множеств, получим нормализованный эквивалент отношения.

Нормализованное отношение – это множество знаков нормализованных множеств, каковыми, в частности, могут быть кортежи. Нормализованное отношение можно трактовать как иерархическую конструкцию, состоящую из следующих уровней:

- знак самого отношения;
- знаки нормализованных множеств, являющиеся элементами отношения (указанные множества могут быть как неупорядоченными множествами, так и кортежами);
- знаки атрибутов, используемых для указания роли элементов в кортежах, знаки которых являются элементами отношения;
- знаки множеств, являющиеся элементами множеств, знаки которых являются элементами отношения.

Множества, знаки которых находятся на четвёртом уровне указанной иерархической конструкции, могут быть либо нормализованными, либо ненормализованными, но тогда обязательно 1-мощными. Как уже отмечалось (см. подраздел 2.1), при нормализации множеств можно оперировать только одним видом ненормализованных множеств – унарными ненормализованными множествами, которые названы предметными множествами.

Особо подчеркнем то, что некоторые знаки, входящие в состав рассмотренной выше многоуровневой конструкции, могут принадлежать сразу нескольким уровням. Так, например, знак отношения может быть элементом кортежа или неориентированного множества, входящего в состав этого же отношения. Кроме того, знак кортежа или неориентированного множества, входящего в состав отношения, может быть элементом как самого этого кортежа или неориентированного множества (см. рефлексивное множество), так и другого кортежа или неориентированного множества, входящего в состав этого же отношения.

Входящие в состав отношения кортежи и неориентированные множества будем называть **связками** этого отношения. Другими словами, связка – это множество, знак которого принадлежит кому-либо нормализованному отношению. Термин "связка" здесь не случаен, т.к. каждое множество (в частности, кортеж), знак которого входит в состав нормализованного отношения, представляет собой описание некоторой конкретной связи между объектами, знаки которых являются элементами связки. При этом тип (семантика) указанной связи определяется отношением, элементом которого является знак соответствующей связки.

В языке SCBg для изображения ориентированных связок (т.е. связок, являющихся кортежами) чаще всего используется графический примитив Θ (см. пункт 3.2.2). Для изображения знаков неориентированных связок (т.е. связок, являющихся неориентированными множествами) введем еще один графический примитив \bullet .

Для сравнения рассмотренной широкой трактовки отношений приведём определение узкой (классической) трактовки отношений. Отношения, соответствующие такой трактовке, будем называть классическими.

Определение 3.3.1.1. Классическое отношение – это множество знаков нормализованных классических кортежей, имеющих одинаковую мощность и использующих одинаковые атрибуты.

Переход от узкой (классической) трактовки отношений к широкой трактовке – это допущение использования в качестве связок не только классических, но и неклассических кортежей, а также неориентированных кортежей, и неориентированных множеств, допущение неодинаковой мощности связок и допущение использования разных атрибутов в разных кортежах.

Примерами отношений общего вида можно считать множества, введенные нами в подразделах 2.1, 3.1 и обозначаемые следующими идентификаторами:

- **пара принадлежности** – это отношение, элементами которого являются знаки всевозможных пар принадлежности (в связи с этим вместо имени нарицательного “**пара принадлежности**” можно использовать эквивалентное имя собственное “**Отношение принадлежности**”);
- **узловое непредметное множество**;
- **система множеств**;
- **множество пар принадлежности**;
- **множество узловых множеств**;
- **множество без кратных элементов**;
- **множество с кратными элементами**;
- **рефлексивное множество**;
- **нерефлексивное множество**;
- **1-мощное множество, 2-мощное множество, 3-мощное множество** /* и т.д. */;
- **1-элементное множество, 2-элементное множество** /* и т.д. */;
- **конечное множество, бесконечное множество**;
- **кортеж**;
- **неориентированное множество**.

Введем специальный scb-узел с идентификатором “**отношение**”. Этот узел обозначает множество знаков всевозможных нормализованных отношений (нормализованность отношений в языке SCB подразумевается по умолчанию). Кроме того, в язык SCBg дополнительно введем еще один графический примитив  для изображения знаков конкретных отношений. Введение указанного графического примитива соответствует правилу:

Правило 3.3.1.1. Следующие изображения знака отношения являются эквивалентными:



Очевидно, что если подразумевать широкую трактовку понятия отношения, то множество с именем “**отношение**” также следует отнести к числу отношений, т.е. имеет место петля:



Для множества связок отношения и множества классических отношений введём соответственно ключевые scb-узлы “**связка отношения**” и “**классическое отношение**”.

Упражнения к пункту 3.3.1.

Упражнение 3.3.1.1. Приведите пример множества **s**, которое не является отношением, т.е. множество, для которого не выполняется условие:

отношение $\rightarrow s ;$

Упражнение 3.3.1.2. Существует ли отношение, являющееся рефлексивным множеством?

Упражнение 3.3.1.3. Существует ли отношение r , для которого имеет место следующее:

$r \rightarrow k ; k \rightarrow s ;$

Это означает, что связка k отношения r является множеством, одним из элементов которого является знак отношения r .

Упражнение 3.3.1.4. Возможна ли следующая конструкция:

отношение $\rightarrow r ; r \rightarrow a , b ; a \rightarrow s ;$

Здесь знак одной из связок отношения r является элементом другой связки этого же отношения.

1.3.2. Типология отношений на основе базовой типологии множеств

Ключевые понятия и идентификаторы ключевых scb-узлов:

ориентированное отношение; неориентированное отношение; частично ориентированное отношение; отношение с кратными связками; отношение без кратных связок; отношение со встречными связками; отношение без встречных связок; семейство множеств одинаковой мощности; семейство множеств неодинаковой мощности; бинарное отношение (семейство 2-мощных множеств); тернарное отношение (семейство 3-мощных множеств); арность отношения; двойная линия со стрелками на обоих концах (графический примитив языка SCBg, изображающий знак неориентированной пары).

Анализ принадлежности связок отношений к числу кортежей позволяет выделить следующие классы отношений:

- **ориентированные отношения** – отношения, все связки которых являются кортежами;
- **неориентированные отношения** – отношения, все связки которых являются неориентированными множествами;
- **частично ориентированные отношения** – отношения, среди связок которых встречаются как кортежи, так и неориентированные множества.

Введём ключевые scb-узлы “*ориентированное отношение*”, “*неориентированное отношение*” и “*частично ориентированное отношение*”, соответственно обозначающие множество ориентированных отношений, множество неориентированных отношений и множество частично ориентированных отношений.

Анализ наличия кратных вхождений элементов в связки отношения позволяет выделить следующие классы отношений:

- отношения, все связки которых являются множествами, имеющими кратное вхождение элементов;
- отношения, все связки которых не являются множествами, имеющими кратное вхождение элементов;
- отношения, некоторые связки которых являются множествами, имеющими кратное вхождение элементов.

Анализ наличия кратных связок, входящих в состав отношения, позволяет выделить следующие классы отношений:

- отношения, имеющие кратные (равные) связки, – множество таких отношений обозначим scb-узлом “**отношение с кратными связками**”;
- отношения, не имеющие кратных (равных) связок, – множество таких отношений обозначим scb-узлом “**отношение без кратных связок**”.

Примечание. Формально отношение не может быть множеством с кратным вхождением каких-либо элементов. Кратные связки отношения оформляются не путем многократного вхождения знака связки в состав отношения, а путем копирования соответствующей связки, т.е. путем включения знаков равных неориентированных множеств или равных кортежей.

Таким образом в scb-тексте могут встречаться равные множества или равные кортежи, но только в том случае, если их знаки являются элементами одного и того же отношения. Целесообразность такого решения обусловлена тем, что равные связки отношения могут иметь разные дополнительные характеристики (например, разные весовые характеристики соответствующих связей).

Кроме кратных (равных) связок могут существовать также встречные связки, т.е. связки, являющиеся встречными кортежами.

Напомним, что **встречные кортежи** – это кортежи, состоящие из одинаковых элементов, по крайней мере один из которых входит в разные (встречные) кортежи под разными атрибутами – см. подраздел 3.2.

Анализ наличия встречных связок, входящих в состав отношения, позволяет выделить следующие классы отношений:

- отношения, имеющие встречные связки, – множество таких отношений обозначим scb-узлом “**отношение со встречными связками**”;
- отношения, в которых встречные связки отсутствуют, – множество таких отношений обозначим scb-узлом “**отношение без встречных связок**”.

Анализ рефлексивности самих отношений позволяет выделить следующие классы отношений:

- отношения, все связки которых являются рефлексивными множествами (знаки таких отношений имеют петлевую пару принадлежности), – это хоть и экзотический, но вполне реальный класс отношений;
- отношения, не являющиеся рефлексивными множествами.

Анализ рефлексивности связок отношения позволяет выделить следующие классы отношений:

- отношения, все связки которых являются рефлексивными множествами;
- отношения, все связки которых не являются рефлексивными множествами;
- отношения, некоторые связки которых являются рефлексивными множествами.

Анализ мощности самих отношений позволяет выделить следующие классы отношений:

- конечные отношения (отношения, являющиеся конечными множествами);
- бесконечные отношения (отношения, являющиеся бесконечными множествами).

Анализ мощности связок отношения позволяет выделить следующие классы отношений:

- отношения со связками одинаковой мощности – множество таких отношений обозначим scb-узлом “**семейство множеств одинаковой мощности**” (см. подраздел 3.1);
- отношения со связками неодинаковой мощности (в каждом таком отношении существуют по крайней мере две связки, имеющие разную мощность) – множество таких отношений обозначим scb-узлом “**семейство множеств неодинаковой мощности**” (см. подраздел 3.1).

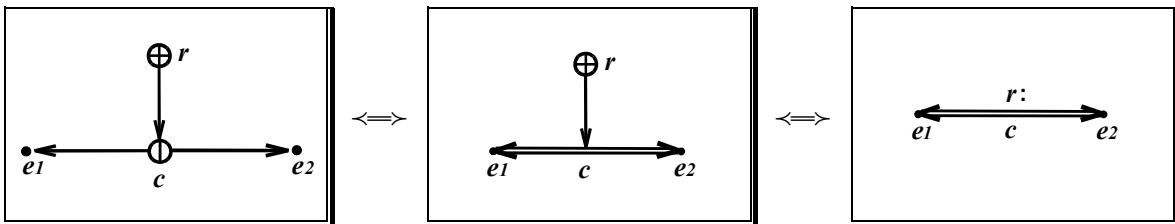
В свою очередь семейство отношений со связками одинаковой мощности разбивается на следующие подклассы:

- бинарные отношения – отношения, все связки которых являются 2-мощными множествами, т.е. мощность которых равна 2 (в частности, 2-мощными кортежами или ориентированными парами), – множество таких отношений обозначим scb-узлом “**бинарное отношение**”;
- тернарные отношения – множество таких отношений обозначим scb-узлом “**тернарное отношение**”;
- 4-арные отношения;
- и т.д.

В связи с вышесказанным отношениям со связками одинаковой мощности можно поставить в соответствие числовую характеристику – **арность отношения**.

Для наглядности изображения 2-мощных неориентированных связок в языке SCBg введём правило:

Правило 3.3.2.1. Следующие изображения 2-мощных неориентированных связок являются эквивалентными:



Здесь scb-узел **c**, обозначающий 2-мощную неориентированную связку, принадлежащую отношению **r**, вместе с выходящими из этого узла scb-дугами заменяется на жирную линию без стрелок. Знак 2-мощной неориентированной связки неуточняемого типа будем называть ребром. Таким образом, в язык SCBg для более наглядного изображения ребер вводится ещё один графический примитив – “жирная двойная линия со стрелками на обоих концах”. Напомним, что для наглядного изображения дуг, не являющихся scb-дугами (знаками пар принадлежности), в язык SCBg был введен специальный графический примитив – двойная линия со стрелкой на одном конце.

1.3.3. Типология отношений на основе типологии кортежей, входящих в состав отношения, а также анализа соотношения между кортежами

Идентификаторы ключевых scb-узлов: *схема отношения; атрибут по умолчанию; коммутативное отношение; коммутативность (метаотношение, связки которого связывают знак коммутативного отношения с коммутируемыми атрибутами).*

Определение 3.3.3.1. Схемой отношения **r** будем называть множество знаков всех тех и только тех атрибутов, которые используются в кортежах отношения **r**. Строгую формальную трактовку понятия схемы отношения как метаотношения, заданного на множестве всевозможных отношений, см. в пункте 3.3.13.

Примечание 1. Если в кортежах отношения используется атрибут, задаваемый по умолчанию, то в схеме этого отношения знак этого атрибута должен быть явно указан, а также явно отнесен к числу элементов специального множества, обозначаемого scb-узлом с идентификатором “**атрибут по умолчанию**”. Указанный scb-узел есть знак множества знаков всевозможных атрибутов, каждый из которых в текущий момент задаётся по умолчанию.

Примечание 2. В состав отношения **r** могут входить не только знаки кортежей, но и знаки неориентированных множеств.

Примечание 3. Атрибут, знак которого входит в схему отношения **r**, может использоваться не в каждом кортеже, который входит в состав отношения **r**.

Каждый атрибут есть не что иное, как бинарное ориентированное отношение, являющееся подмножеством базового для языка SCB отношения принадлежности, т. е. множества знаков всевозможных пар принадлежности.

В качестве примера схемы отношения приведем схему тернарного классического отношения сложения. Эта схема представляет собой множество, включающее в себя знаки следующих атрибутов:

- **сумма_(быть суммой_);**
- **слагаемое1_(быть 1-м слагаемым_);**

- ***слагающееся_ (быть 2-м слагаемым)***.

Примечание 4. Стого говоря, любое неориентированное множество можно “превратить” в кортеж, “подметив” какую-то особенность хотя бы одного из элементов этого множества по сравнению с другими его элементами. Поэтому, когда вводится какое-либо ориентированное отношение, необходимо явно перечислить те атрибуты, которые в этом отношении учитываются, т.е. необходимо для каждого неориентированного отношения явно указывать его схему. Это значит, что все атрибуты, не указанные в схеме отношения, при анализе этого отношения будут игнорироваться. Таким образом, могут существовать неориентированные отношения, включающие в себя кортежи (в этом случае атрибуты этих кортежей вообще не учитываются). Некоторые кортежи могут входить в разные отношения, в которых используются разные схемы (для 1-го отношения в этом кортеже будут учитываться одни атрибуты, а для 2-го отношения – другие).

Анализ включения атрибута, задаваемого по умолчанию, в состав схемы отношения позволяет выделить следующие классы отношений:

- отношения, использующие атрибут, задаваемый по умолчанию;
- отношения, не использующие атрибут, задаваемый по умолчанию.

Анализ того, в каждой ли связке отношения используются все атрибуты из схемы этого отношения, позволяет выделить следующие классы отношений:

- отношения, все связи которых используют одинаковые атрибуты;
- отношения, у которых имеются по крайней мере две связи с различными наборами атрибутов.

Анализ соотношения вхождений элементов в связи отношения и соответствующих им атрибутов позволяет выделить следующие классы отношений:

- 1)
 - отношения, в каждой связке которых каждому вхождению элемента соответствует один и только один атрибут;
 - отношения, в каждой связке которых существует по крайней мере одно вхождение какого-либо элемента, которому соответствует несколько атрибутов;
 - отношения, в некоторых связках которых каждому вхождению элемента соответствует один и только один атрибут, а в некоторых связках которых существует по крайней мере одно вхождение какого-либо элемента, которому соответствует несколько атрибутов;
- 2)
 - отношения, в каждой связке которых не существует двух таких вхождений элементов, которым соответствует один и тот же атрибут;
 - отношения, в каждой связке которых существуют по крайней мере два таких вхождения элементов, которым соответствует один и тот же атрибут;
 - отношения, в некоторых связках которых не существует двух таких вхождений элементов, которым соответствует один и тот же атрибут, а в некоторых связках которых существуют по крайней мере два таких вхождения элементов, которым соответствует один и тот же атрибут.

Определение 3.3.3.2. Множество ***r*** будем называть **коммутативным отношением** для атрибутов ***ai*** и ***aj*** в том и только том случае, если справедливы следующие высказывания:

- множество ***r*** является ориентированным отношением, в схему которого входят атрибут ***ai*** и атрибут ***aj***;
- кортеж отношения ***r*** содержит не более одного элемента под атрибутом ***ai*** и не более одного элемента под атрибутом ***aj***;
- если в кортеж отношения ***r*** входит элемент под атрибутом ***ai***, то в этот же кортеж входит элемент под атрибутом ***aj*** и наоборот;
- если в кортеж отношения ***r*** входят элемент ***xi*** под атрибутом ***ai*** и элемент ***xj*** под атрибутом ***aj***, то в отношение ***r*** будет входить кортеж, в который элемент ***xi*** входит под атрибутом ***ai***, а элемент ***xj*** – под атрибутом ***aj***. При этом все остальные элементы указанных кортежей совпадают и входят в эти кортежи под одинаковыми атрибутами.

Множеству коммутативных отношений сопоставим ключевой scb-узел “**коммутативное отношение**”.

Примечание 5. Если отношение r является коммутативным для атрибутов ai и aj , то от отношения r можно

перейти к эквивалентному отношению r^* путем:

- ликвидации любого из двух встречных кортежей, которые отличаются только двумя компонентами, имеющими атрибуты ai и aj причем ai – компонент одного из указанных кортежей совпадает с aj – компонентом другого кортежа;
- склеивания знаков атрибутов ai и aj .

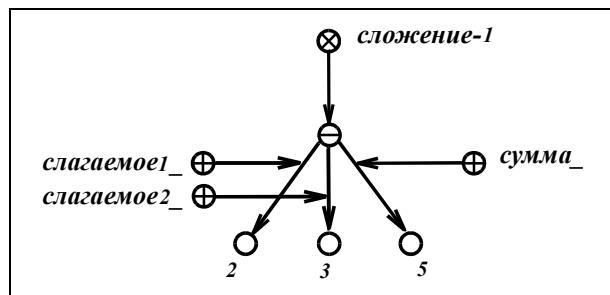
В результате этого эквивалентного преобразования сокращаются количество связок отношения и количество используемых атрибутов.

Приведём несколько вариантов представления отношения сложения.

SCB-текст 3.3.3.1. Вариант 1

изображения отношения сложения

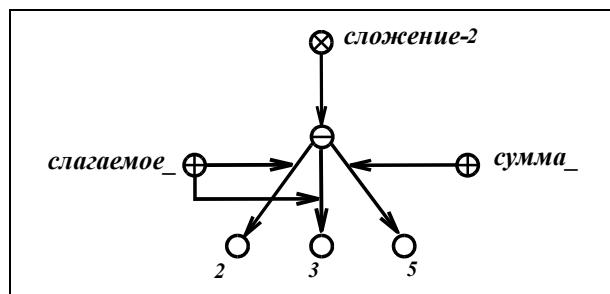
Это классическое тернарное коммутативное отношение с атрибутами “*слагаемое1_*”, “*слагаемое2_*”, “*сумма_*”.



SCB-текст 3.3.3.2. Вариант 2

изображения отношения сложения

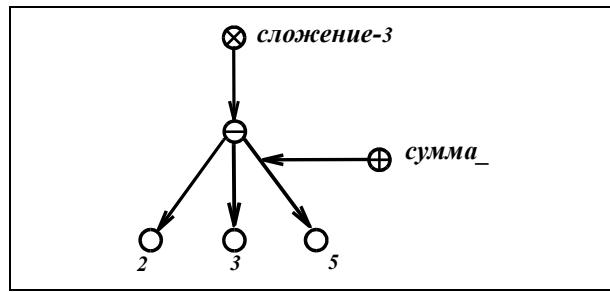
Это неклассическое тернарное ориентированное отношение, в каждом кортеже которого используются два атрибута: “*слагаемое_*”, “*сумма_*”.



SCB-текст 3.3.3.3. Вариант 3

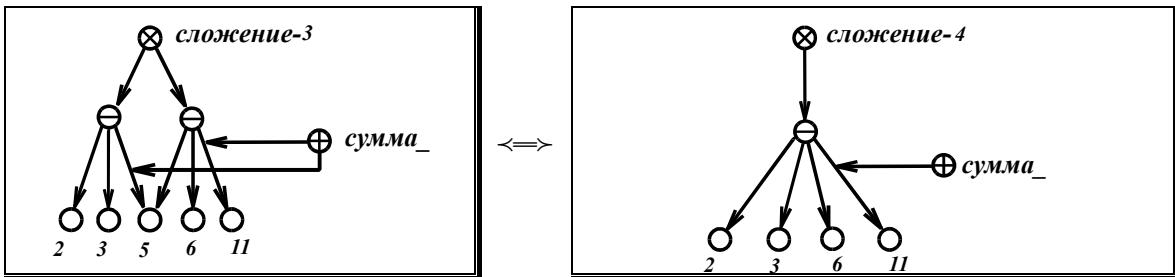
изображения отношения сложения

Это неклассическое тернарное ориентированное отношение, отличающееся от отношения “*сложение-2*” только тем, что в каждом его кортеже используются один атрибут по умолчанию и один атрибут явно (“*сумма_*”).

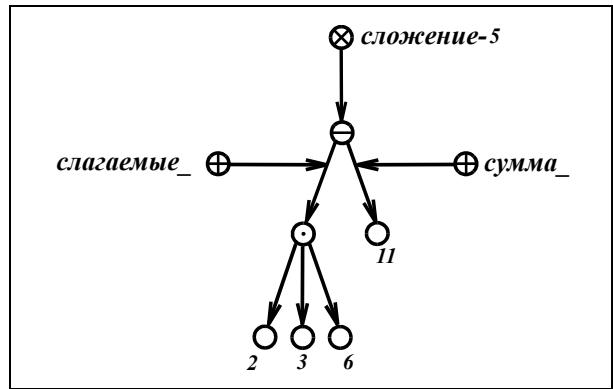


Очевидно, что для компактности изображения scb-текстов здесь целесообразнее задавать по умолчанию атрибут “*слагаемое_*”, а не атрибут “*сумма_*”, т.к. каждый кортеж рассматриваемого отношения имеет несколько компонентов с атрибутом “*слагаемое_*” и только один компонент с атрибутом “*сумма_*”.

SCB-текст 3.3.3.4. Вариант 4 изображения отношения сложения (неклассическое ориентированное отношение, кортежи которого имеют в общем случае различную мощность – от 3 и выше)



SCB-текст 3.3.3.5. Вариант 5 изображения отношения сложения (классическое бинарное отношение с атрибутами “*сумма_*” и “*слагаемые_*”)



Можно привести еще вариант 6 изображения отношения сложения в виде бинарного ориентированного отношения, отличающегося от отношения “*сложение-5*” тем, что в нем один из атрибутов задается по умолчанию.

1.3.4. Типология отношений на основе понятия проекции и понятия области определения

Идентификаторы ключевых scb-узлов: *область определения*; *домен* (*унарная проекция*); *проекция отношения*; *отношение с попарно непересекающимися доменами*; *отношение с совпадающими доменами*; *отношение над множествами*; *отношение над кортежами*; *метаотношение* (*отношение над отношениями*); *числовое отношение*; *геометрическое отношение*; *tempоральное отношение*.

Определение 3.3.4.1. Область определения отношения *r* – это множество всех тех и только тех объектов, которые являются элементами связок отношения *r*. В языке SCB понятию области определения ставится в соответствие отношение с именем “*область определения*”. Страговая формальная трактовка понятия области определения как метаотношения будет рассмотрена ниже в пункте 3.3.13.

Определение 3.3.4.2. Домен (*унарная проекция*) отношения *r* по атрибуту *ai* – это множество всех тех и только тех объектов, которые являются такими элементами связок отношения *r*, вхождения которых в эти связки имеют атрибут *ai*. В языке SCB понятию домена ставится в соответствие метаотношение с именем “*домен*”. См. пункт 3.3.13.

Примечание. Домен отношения по заданному атрибуту не требует того, чтобы связки указанного отношения имели только один элемент с указанным атрибутом. Таких элементов может быть сколько угодно.

Очевидно, что все элементы каждого домена отношения *r* являются также элементами области определения этого отношения.

Определение 3.3.4.3. Проекция отношения r по атрибутам a_1, a_2, \dots, a_m – это множество знаков всех тех и только тех кортежей, которые строятся из кортежей отношения r путём удаления всех вхождений элементов, которые имеют атрибуты, не совпадающие с указанными выше атрибутами a_1, a_2, \dots, a_m , а также путём последующего устранения кратности кортежей (из семейства кратных кортежей в указанном множестве кортежей должен оставаться один представитель). В языке SCB понятию проекции отношения ставится в соответствие метаотношение с именем “**проекция отношения**”.

Очевидно, что неунарная проекция заданного отношения r также является отношением, причём ориентированным отношением со схемой a_1, a_2, \dots, a_m .

Анализ соотношения доменов (унарных проекций) каждого отношения позволяет выделить следующие классы отношений:

- отношения, в каждом из которых домены являются попарно непересекающимися множествами, т.е. множествами, которые не имеют общих элементов (назовем такие отношения **отношениями с попарно непересекающимися доменами**);
- отношения, в которых имеются по крайней мере два домена с общими элементами;
- отношения, в каждом из которых все домены совпадают.

Примечание. Ориентированное отношение с попарно непересекающимися унарными проекциями можно заменить на эквивалентное неориентированное отношение путём замены явно указываемых атрибутов исходного отношения на явное указание принадлежности элементов связок отношения к соответствующей унарной проекции. Например, ориентированное отношение, связки которого имеют смысл «точка t лежит на прямой p » можно заменить на эквивалентное отношение, связки которого имеют смысл «геометрические объекты t и p инцидентны друг другу». Если при этом дополнительно будет указано к какому классу геометрических объектов (к классу точек или к классу прямых) относятся объекты t и p , то будет очевидно, какой из указанных объектов на каком находится (прямая не может лежать на точке!).

В соответствии с приведенной выше классификацией для обозначения конкретных типов отношений введем следующие ключевые scb-узлы: “**отношение с попарно непересекающимися доменами**”, “**отношение с совпадающими доменами**”.

Анализ того, какая теоретико-множественная связь имеет место между областью определения отношения и самим отношением, позволяет выделить следующие классы отношений:

- отношения, у которых знак каждой связки является элементом их области определения (каждое такое отношение является подмножеством собственной области определения). Элементами некоторых связок такого отношения являются знаки связок этого же отношения;
- отношения, у которых знаки некоторых (не всех) связок являются элементами их области определения (каждое такое отношение пересекается с собственной областью определения, но не является её подмножеством);
- отношения, знаки связок которых не входят в состав области определения этих отношений.

Анализ того, входит или нет знак отношения в состав собственной области определения, позволяет выделить следующие классы отношений:

- отношения, у которых некоторые связки содержат в качестве хотя бы одного из своих элементов знак самого этого отношения;
- отношения, у которых нет связок, содержащих в качестве элемента знак самого этого отношения.

Содержательный анализ области определения отношений позволяет выделить следующие классы отношений:

- **отношения над множествами** – в область определения каждого такого отношения входят знаки всевозможных множеств;
- **отношения над кортежами** – в область определения каждого такого отношения входят знаки всевозможных кортежей; при этом в состав каждой связки такого отношения входит знак по крайней мере одного кортежа, т.е. элементами рассматриваемых связок являются знаки других связок;
- **отношения над отношениями** того или иного типа (**метаотношения**) – в область определения каждого такого отношения входят знаки всевозможных отношений соответствующего типа; при этом в состав каждой связки такого отношения входит знак по крайней мере одного отношения;

- **числовые отношения** – отношения над числами и числовыми отношениями;
- **геометрические отношения** – отношения над геометрическими фигурами;
- **tempоральные отношения** – отношения, описывающие связи во времени;
- и др.

В соответствии с приведенной выше классификацией для обозначения конкретных типов отношений введем следующие ключевые scb-узлы: “**отношение над множествами**”, “**отношение над кортежами**”, “**метаотношение**” (и его синоним “**отношение над отношениями**”), “**числовое отношение**”, “**геометрическое отношение**”, “**tempоральное отношение**”.

Упражнения к пункту 3.3.4.

Упражнение 3.3.4.1. Запишите на языках SCBg и SCBs высказывание о том, что множество s является областью определения некоторого отношения r .

Упражнение 3.3.4.2. Запишите на языках SCBg и SCBs высказывание о том, что множество U является областью определения отношения “**область определения**”. Является ли указанное множество U универсальным множеством?

Упражнение 3.3.4.3. Все ли отношения (как классические, так и неклассические) имеют область определения? Если да, то как этот факт записывается на языках SCBg и SCBs?

1.3.5. Типология отношений на основе понятия функциональной зависимости

Идентификаторы ключевых scb-узлов: *функциональная зависимость; отношения без функциональных зависимостей; отношения с одной функциональной зависимостью; отношения с несколькими функциональными зависимостями; ключевая функциональная зависимость; ключ отношения; отношение без ключей; отношение с одним ключом; отношение с несколькими ключами; функция; декартово произведение; отношение без функций; отношение с одной функцией; отношение с несколькими функциями; взаимно однозначное бинарное отношение; алгебраическая операция.*

Определение 3.3.5.1. Будем говорить, что отношение r имеет **функциональную зависимость** атрибутов a_{u1}, \dots, a_{ut} от **атрибутов** a_{x1}, \dots, a_{xn} в том и только в том случае, если:

- указанные множества атрибутов не имеют общих элементов;
- все перечисленные атрибуты входят в схему отношения r ;
- в отношении r не существует двух таких кортежей, у которых бы совпадали элементы, имеющие атрибуты из множества $\{a_{x1}, \dots, a_{xn}\}$, но не совпадали элементы, имеющие атрибуты из множества $\{a_{u1}, \dots, a_{ut}\}$.

Другими словами, те элементы кортежа, входящего в отношение r , которые имеют атрибуты из множества $\{a_{x1}, \dots, a_{xn}\}$, однозначно определяют другие элементы этого же кортежа, имеющие атрибуты из множества $\{a_{u1}, \dots, a_{ut}\}$ [330] (*Мейер Д. 1987 книга Теория РБД*). Атрибуты из множества $\{a_{x1}, \dots, a_{xn}\}$ будем называть атрибутами аргументов функциональной зависимости, а атрибуты из множества $\{a_{u1}, \dots, a_{ut}\}$ – атрибутами результата функциональной зависимости. Минимальное число атрибутов аргументов и атрибутов результата равно 1.

Анализ наличия функциональных зависимостей в отношениях позволяет выделить следующие классы отношений:

- отношения без функциональных зависимостей;
- отношения с одной функциональной зависимостью;
- отношения с несколькими функциональными зависимостями.

Определение 3.3.5.2. Функциональную зависимость атрибутов a_{y1}, \dots, a_{ym} от атрибутов a_{x1}, \dots, a_{xn} в рамках отношения r будем называть **ключевой функциональной зависимостью** в том и только в том случае, если в схеме отношения r не существует ни одного атрибута, который бы не принадлежал либо множеству $\{a_{y1}, \dots, a_{ym}\}$, либо множеству $\{a_{x1}, \dots, a_{xn}\}$.

Множество атрибутов аргументов ключевой функциональной зависимости иногда называют **ключом заданного отношения**.

Из приведенного определения следует, что для любых двух различных кортежей ki и kj , входящих во множество r , существует атрибут axe из множества $\{a_{x1}, \dots, a_{xn}\}$ такой, что элемент кортежа ki с атрибутом axe не совпадает с элементом кортежа kj , имеющим тот же атрибут. Другими словами, в отношении r не существует двух таких кортежей, в которых бы совпадали элементы с атрибутами из множества $\{a_{x1}, \dots, a_{xn}\}$ и не совпадали элементы, имеющие остальные атрибуты [330] (*Майер Д. 1987 книга Теория РБД*). Это означает, что для кортежей, принадлежащих отношению r , элементы, имеющие атрибуты из множества $\{a_{x1}, \dots, a_{xn}\}$, однозначно определяют все остальные элементы кортежа, т.е. однозначно определяют весь кортеж.

Анализ наличия ключевых функциональных зависимостей в отношениях позволяет выделить следующие классы отношений:

- отношения без ключей;
- отношения с одним ключом;
- отношения с несколькими ключами.

Определение 3.3.5.3. Ключевую функциональную зависимость отношения r будем называть **функцией** в том и только в том случае, если:

- количество атрибутов результатов этой ключевой зависимости равно 1 – обозначим этот единственный атрибут результата функциональной зависимости через ay ;
- в каждом кортеже отношения r существует не более одного компонента с атрибутом ay ;
- проекция отношения r по атрибутам a_{x1}, \dots, a_{xn} в случае, если количество указанных атрибутов больше 1, представляет собой декартово произведение.

Примечание. Определение декартова произведения см. в [пункт 3.3.7](#).

Функцию n -арного отношения будем называть $(n - 1)$ -арной функцией.

Заметим, что каждая функциональная зависимость бинарного ориентированного отношения является ключевой функциональной зависимостью, а также функцией.

Анализ наличия функций в отношениях позволяет выделить следующие классы отношений:

- отношения, не имеющие функций;
- отношения, имеющие одну функцию;
- отношения, имеющие несколько функций.

Определение 3.3.5.4. Взаимно однозначным отношением будем называть бинарное ориентированное отношение, которому соответствуют две различные функции.

Определение 3.3.5.5. Функцию отношения r будем называть **алгебраической операцией** в том и только том в случае, если отношение r принадлежит классу отношений, у которых все домены совпадают (см. [пункт 3.3.4](#)).

1.3.6. Представление и типология классических отношений

Идентификаторы ключевых scb-узлов: *классическое отношение; неклассическое отношение.*

Понятие классического нормализованного отношения было дано выше (см. определение 3.3.1.1). Важной особенностью классического отношения является то, что его можно представить в виде таблицы, столбцы которой соответствуют используемым атрибутом, а строки – кортежам, входящим в состав отношения. Элементы кортежей в этой таблице представляются именами (идентификаторами) соответствующих объектов. Таблицы указанного вида являются основой для реляционных моделей баз данных [330] (*Майер Д. 1987гн-Теория РБД*). Поэтому эти таблицы иногда называют реляционными.

Каждое классическое отношение есть множество знаков классических кортежей, имеющих одинаковую мощность и одинаковые атрибуты.

Классическим отношениям противопоставляются **неклассические отношения**, в состав которых могут входить:

- как кортежи, так и неориентированные множества;
- множества разной мощности;
- как классические, так и неклассические кортежи;
- кортежи, использующие разные наборы атрибутов.

Для множества неклассических отношений введём ключевой scb-узел “**неклассическое отношение**”.

Классические отношения в математической литературе обычно называют просто отношениями. Неклассические отношения не являлись предметом серьезного анализа, т. к. они легко представимы в виде эквивалентных отношений классического вида. Но поскольку в интеллектуальных системах важное значение имеет не только факт представимости, но и удобство (в частности, компактность) представления, некоторые классические отношения в памяти интеллектуальных систем целесообразно представлять в виде эквивалентных неклассических отношений.

Упражнение 3.3.6.1. Построить реляционные таблицы фрагментов (подмножеств) некоторых отношений, например, отношения сложения, отношения умножения, генеалогических отношений, отношения, связывающего страны с их столицами, отношения, связывающего выполняемые проекты с их исполнителями, и т.д.

1.3.7. Отношения предельного вида

Идентификаторы ключевых scb-узлов: *булеан; множество всевозможных перестановок; множество сочетаний; шкала множеств; универсум; минимальное декартово произведение классического отношения; декартова степень (декартово произведение, все домены которого совпадают).*

Каждое конкретное отношение, относящееся к классу “**отношение предельного вида**”, представляет собой семейство знаков всевозможных множеств, удовлетворяющих какому-то конкретному требованию (свойству). К такому классу отношений, в частности, можно отнести:

- семейство знаков всевозможных классических кортежей, каждый из которых использует один и тот же (заданный) набор атрибутов, для которого фиксируется (задается), во-первых, набор атрибутов и, во-вторых, набор множеств, однозначно соответствующих указанному набору атрибутов. При этом каждый из указанных выше классических кортежей должен удовлетворять следующим требованиям: (1) использовать все те и только те атрибуты, которые указаны выше, и (2) каждый компонент кортежа должен являться элементом того из указанных выше множеств, которое соответствует атрибуту, которым отмечен этот компонент;
- множество всевозможных классических кортежей ("предельный" характер этого отношения заключается в том, что оно является объединением всевозможных классических отношений);
- множество всевозможных кортежей (необязательно классических) – это отношение представляет собой объединение всевозможных ориентированных отношений.

Список такого рода отношений можно продолжить, но кроме таких конкретных отношений предельного вида существует целый ряд классов отношений предельного вида. К таким классам отношений можно отнести следующие:

- декартовы произведения (прямые произведения);
- булеаны;
- отношения, каждое из которых является множеством всевозможных перестановок из заданного набора элементов;
- отношения, каждое из которых является множеством всевозможных сочетаний заданной мощности из заданного набора элементов;
- шкалы множеств;
- отношения, каждое из которых является универсумом, построенным на заданном множестве.

Определение 3.3.7.1. Отношение d является **декартовым произведением** в том и только в том случае, если:

- отношение d является классическим отношением с атрибутами a_1, a_2, \dots, a_n ;
- в состав отношения d входит каждый кортеж вида $(a_1 : x_{1i}, a_2 : x_{2i}, \dots, a_n : x_{ni})$, где x_{1i} есть элемент множества, являющегося доменом отношения d по атрибуту a_1 ; x_{2i} есть элемент множества, являющегося доменом отношения d по атрибуту a_2 и т.д., x_{ni} есть элемент множества, являющегося доменом отношения d по атрибуту a_n .

Из данного определения следует, что в состав отношения d не входят неориентированные множества (т.к. это отношение является классическим), а также никакие другие кортежи кроме тех, которые указаны выше. Добавление кортежей приведет либо к нарушению классического характера отношения d , либо к расширению множеств, являющихся его унарными проекциями. Множество отношений, являющихся декартовым произведением, будем обозначать ключевым scb-узлом “**декартово произведение**”.

Очевидно, что каждому классическому отношению можно поставить в соответствие целое семейство декартовых произведений, являющихся надмножествами этого классического отношения. При этом очевидно, что в указанном семействе декартовых произведений существует **минимальное декартово произведение**, т.е. такое декартово произведение, которое является подмножеством всех остальных декартовых произведений, входящих в это семейство. Для каждого классического отношения существует единственное такое минимальное декартово произведение имею одинаковые атрибуты и для каждого атрибута – совпадающие унарные проекции. Каждое декартово произведение, являющееся надмножеством заданного классического отношения, но не являющееся для этого классического отношения минимальным декартовым произведением, имеет по крайней мере одну унарную проекцию, которая является надмножеством соответствующей унарной проекции (т.е. проекции по тому же атрибуту) заданного классического отношения. Остальные унарные проекции рассматриваемого декартова произведения либо совпадают с соответствующими унарными проекциями заданного классического отношения, либо являются их надмножествами, но никогда не могут быть их подмножествами.

Заметим, что классическое отношение можно определить как отношение, для которого существует декартово произведение, подмножеством которого это отношение является.

Поскольку каждое декартово произведение однозначно задаётся его схемой (набором используемых атрибутов) и набором всех его унарных проекций, легко ввести соответствующие условные обозначения декартовых произведений. Пусть отношение d представляет собой бинарное декартово произведение, пусть a_1 и a_2 – атрибуты, используемые отношением d , и пусть x_1 есть домен отношения d по атрибуту a_2 . Введём следующее условное обозначение бинарного декартова произведения: $d = (x_1(a_1) \times x_2(a_2))$.

Тернарное декартово произведение будем обозначать следующим образом:

$(x_1(a_1) \times x_2(a_2) \times x_3(a_3))$.

Аналогичным образом обозначаются декартовы произведения любой другой арности. Указанные условные обозначения декартовых произведений разрешено использовать в языке SCBs. Заметим, что заданные множества x_1, x_2, \dots, x_n , являющиеся унарными проекциями декартова произведения, могут иметь между собой самые различные теоретико-множественные соотношения. В частности, все

эти множества могут совпадать. В этом случае декартово произведение называют ***n*-й декартовой степенью** заданного множества, где число ***n*** есть арность такого декартова произведения. Множество декартовых степеней будем обозначать scb-узлом “**декартова степень**”.

Рассмотрим классическое отношение, у которого:

- отсутствуют связи с кратными вхождениями элементов;
- все его связи являются встречными друг другу, т.е. являются кортежами, имеющими одинаковые элементы, но входящими в кортежи под различными атрибутами.

Примечание. Поскольку рассматриваемое отношение является классическим, кратные связи в нем отсутствуют.

Очевидно, что у такого отношения все элементы его области определения входят в состав каждого его кортежа. Очевидно также, что для заданной области определения существует несколько отношений, обладающих указанными выше свойствами. Но существует одно, которое можно считать отношением предельного вида и которое будем называть множеством всех перестановок из заданного набора элементов.

Определение 3.3.7.2. Отношение ***p*** будем называть **множеством перестановок** в том и только в том случае, если оно обладает следующими свойствами:

- является классическим отношением;
- не имеет связок с кратными вхождениями элементов;
- все его связи являются встречными друг другу (т.е. любые две связи отношения ***p*** являются встречными кортежами);
- каждый встречный кортеж любого кортежа, входящего в отношение ***p***, также входит в состав отношения ***p***.

Очевидно, что отношение, принадлежащее к классу множеств перестановок, однозначно определяется: (1) своей областью определения и (2) своей схемой, т.е. набором используемых атрибутов. Следовательно, для построения отношения рассматриваемого класса достаточно знать его область определения и его схему. При этом заметим, что мощность области определения для каждого отношения рассматриваемого класса совпадает с количеством используемых в нем атрибутов. Заметим также, что если для заданного канторовского множества ***s*** построить его ***n***-ную декартову степень (где ***n*** есть мощность множества ***s***) и после этого исключить все кортежи, имеющие кратные элементы, то полученное отношение будет принадлежать к классу множеств перестановок.

Определение 3.3.7.3. Рассмотрим семейство неориентированных отношений, не имеющих кратных связок и не имеющих связок с кратными вхождениями элементов. Для такого семейства отношений можно построить минимальное “отношение предельного вида”, являющееся подмножеством для всех отношений, удовлетворяющих указанным выше свойствам. Для этого достаточно (1) построить множество, являющееся результатом объединения областей определения всех отношений, входящих в состав заданного семейства неориентированных отношений, (2) для построенного множества построить семейство всевозможных его подмножеств. Указанное отношение предельного вида будем называть **булеаном**.

Определение 3.3.7.4. Рассмотрим семейство *m*-арных неориентированных отношений, не имеющих кратных связок и не имеющих связок с кратными вхождениями элементов. Для такого семейства отношений также можно построить минимальное отношение предельного вида, т.е. минимальное отношение, являющееся надмножеством для всех отношений, удовлетворяющих указанным выше свойствам. Для этого необходимо (1) построить множество, являющееся объединением областей определения всех отношений, входящих в состав заданного семейства неориентированных отношений, (2) для построенного множества построить семейство всевозможных его подмножеств, имеющих мощность, равную ***m***. Указанное отношение предельного вида будем называть **множеством сочетаний** из заданного множества по ***m*** элементов.

Для семейства множеств всевозможных перестановок, семейства множеств сочетаний и семейства булевых в языке SCB вводятся следующие ключевые узлы: “**множество перестановок**”, “**множество сочетаний**”, “**булеан**”.

Упражнение 3.3.7.1. Могут ли существовать разные классические отношения, у которых совпадают минимальные декартовы произведения? Если да, то приведите примеры.

1.3.8. Типология бинарных отношений и метаотношения над ними

Идентификаторы ключевых scb-узлов: *бинарное отношение; бинарное ориентированное отношение (классическое бинарное отношение); бинарное неориентированное отношение; бинарное отношение без функций; бинарное отношение с одной функцией; взаимно однозначное бинарное отношение (бинарное отношение с двумя функциями); рефлексивное бинарное отношение; иррефлексивное бинарное отношение; частично рефлексивное бинарное отношение; симметричное бинарное отношение; антисимметричное бинарное отношение; частично симметричное бинарное отношение; транзитивное бинарное отношение; антитранзитивное бинарное отношение; частично транзитивное бинарное отношение; отношение эквивалентности* (рефлексивное, симметричное и транзитивное); *отношение предпорядка* (рефлексивное и транзитивное); *отношение частичного порядка* (рефлексивное, антисимметричное и транзитивное); *отношение линейного порядка* (подкласс отношений частичного порядка).

Определение 3.3.8.1. Функциональные (однозначные) бинарные отношения – это бинарные отношения, для которых существует по крайней мере одна унарная функция, т.е. по крайней мере один атрибут, являющийся ключом (см. пункт 3.3.5).

К числу бинарных ориентированных отношений, использующих числовые атрибуты $1_{_}$ (быть 1-м компонентом кортежа) и $2_{_}$ (быть 2-м компонентом кортежа), относятся следующие уже рассмотренные выше базовые отношения языка SCB:

- отношение принадлежности (множество знаков всевозможных пар принадлежности);
- отношение непринадлежности (множество знаков всевозможных пар непринадлежности);
- отношение нечёткой принадлежности (множество знаков всевозможных пар нечеткой принадлежности);
- универсальное бинарное ориентированное отношение с атрибутами $1_{_}$ и $2_{_}$ (множество знаков всевозможных ориентированных пар неуточняемого типа).

Определение 3.3.8.2. Отношение r является **рефлексивным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением со схемой $\{ ai, aj \}$;
- 2) для каждого x из области определения отношения r кортеж $(ai : x, aj : x)$ принадлежит этому отношению.

Примером рефлексивного бинарного отношения: является отношение с именем “**включение множества**” (см. пункт 3.3.11).

Определение 3.3.8.3. Отношение r является **иррефлексивным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением;
- 2) в рамках отношения r не существует ни одного кортежа, элементы (компоненты) которого бы совпадали.

Определение 3.3.8.4. Отношение r является **частично рефлексивным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением;
- 2) в рамках отношения r существуют петли, т. е. такие кортежи, элементы (компоненты) которых совпадают (отношение r не является иррефлексивным бинарным отношением);
- 3) оно не является рефлексивным бинарным отношением.

Определение 3.3.8.5. Отношение r является **симметричным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением со схемой $\{ ai, aj \}$;
- 2) для каждого кортежа $(ai : x, aj : y)$ из отношения r справедливо, что кортеж $(ai : y, aj : x)$ также принадлежит этому отношению (x не совпадает с y).

Определение 3.3.8.6. Отношение r является **антисимметричным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением со схемой $\{ ai, aj \}$;
- 2) выполняется условие: если кортеж $(ai : x, aj : y)$ принадлежит отношению r и кортеж $(ai : y, aj : x)$ также принадлежит r , то $x = y$.

Определение 3.3.8.7. Отношение r является **частично симметричным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением со схемой $\{ ai, aj \}$;
- 2) существует, по крайней мере, один кортеж $(ai : x, aj : y)$, принадлежащий бинарному отношению r , для которого существует кортеж $(ai : y, aj : x)$, также принадлежащий отношению r ;
- 3) существует, по крайней мере, один кортеж $(ai : x, aj : y)$, принадлежащий бинарному отношению r , для которого не существует кортежа $(ai : y, aj : x)$, принадлежащего отношению r .

Определение 3.3.8.8. Отношение r является **транзитивным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением со схемой $\{ ai, aj \}$;
- 2) для любых x, y, z из области определения отношения r справедливо, что если кортежи $(ai : x, aj : y)$ и $(ai : y, aj : z)$ принадлежат отношению r , то и кортеж $(ai : x, aj : z)$ также принадлежит r .

Определение 3.3.8.9. Отношение r является **антитранзитивным бинарным отношением** в том и только в том случае, если:

- 1) оно является бинарным отношением со схемой $\{ ai, aj \}$;
- 2) для любых x, y, z из области определения отношения r справедливо, что если кортежи $(ai : x, aj : y)$ и $(ai : y, aj : z)$ принадлежат отношению r , то не найдётся кортежа $(ai : x, aj : z)$, также принадлежащего r .

Определение 3.3.8.10. **Частично транзитивные отношения** – это бинарные отношения, которые не являются ни транзитивными, ни антитранзитивными.

Определение 3.3.8.11. **Бинарные отношения эквивалентности** – это бинарные отношения, обладающие свойствами рефлексивности, симметричности, транзитивности.

Определение 3.3.8.12. **Бинарные отношения предпорядка** – это бинарные отношения, обладающие свойствами рефлексивности, транзитивности.

Определение 3.3.8.13. **Бинарные отношения частичного порядка** – это бинарные отношения, обладающие свойствами рефлексивности, антисимметричности, транзитивности.

Определение 3.3.8.14. Отношение r является **отношением линейного порядка** в том и только в том случае, если:

- оно является отношением частичного порядка с атрибутами ai и aj ;
- для любых двух разных элементов x и y из области определения отношения r имеет место принадлежность отношению r либо кортежа $(ai : x, aj : y)$, либо кортежа $(ai : y, aj : x)$. То есть любые два элемента из области определения отношения r сравнимы.

Всем вышеперечисленным типам отношений в языке SCB сопоставляются ключевые узлы, обозначающие, множества отношений соответствующего типа: “**рефлексивное бинарное**

отношение", "иррефлексивное бинарное отношение", "частично рефлексивное бинарное отношение", "симметричное бинарное отношение", "антисимметричное бинарное отношение", "частично симметричное бинарное отношение", "транзитивное бинарное отношение", "антитранзитивное бинарное отношение", "частично транзитивное бинарное отношение", "отношение эквивалентности", "отношение предпорядка", "отношение частичного порядка", "отношение линейного порядка".

Множество знаков всевозможных пар принадлежности, которое будем называть **отношением принадлежности**, является для языка SCB базовым отношением, с помощью которого в языке SCB осуществляется представление всех математических структур (множеств, кортежей, отношений). Таким образом, язык SCB осуществляет представление математических структур (в частности отношений) путем их сведения к отношению принадлежности. Оно является бинарным ориентированным отношением частично транзитивным, частично симметричным, частично рефлексивным. Отношение принадлежности относится к классу бинарных отношений, знаки пар которого представляются в языке SCB специальным образом – в виде scb-дуг.

Каждое семейство знаков пар принадлежности (в частности, каждый атрибут какого-либо отношения) также представляет собой для языка SCB не что иное, как специальным образом задаваемое бинарное ориентированное отношение, являющееся некоторым подмножеством отношения принадлежности. В языке SCBg знаки таких отношений изображаются графическим примитивом "кружок с плюсиком" –

1.3.9. Множество соответствий как метаотношение, заданное на множестве бинарных ориентированных отношений

Идентификаторы ключевых scb-узлов: *соответствие*; *отношение_*; *атрибут_*; *отображение*, *проекция_*; *сюръекция*; *однозначное соответствие* (функциональное соответствие); *аргумент_*; *взаимно однозначное соответствие* (функциональное отображение); *инъекция*; *биекция* (однозначная сюръекция, функциональная сюръекция); *взаимно однозначное отображение*; *взаимно однозначная сюръекция*.

Каждое конкретное **соответствие** между двумя множествами будем трактовать как тернарный кортеж, связывающий:

- 1) знак некоторого бинарного отношения – этому знаку приписывается атрибут “*отношение_*” (быть отношением; в данном случае речь идет только о бинарных отношениях);
- 2) знак пары, связывающей знак одного из заданных множеств со знаком соответствующего ему атрибута, используемого в указанном выше бинарном отношении, – в этой паре используется атрибут “*атрибут_*” (быть атрибутом), а второй атрибут задается по умолчанию;
- 3) знак аналогичной пары, связывающей знак второго из заданных множеств со знаком соответствующего ему атрибута.

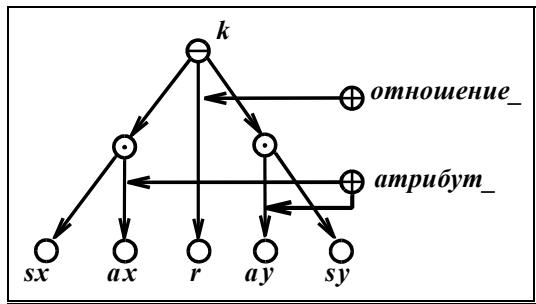
Множество знаков всевозможных конкретных соответствий есть не что иное, как одно из метаотношений, заданных на множестве всевозможных бинарных отношений (множество всевозможных бинарных отношений входит в область определения указанного метаотношения). Рассматриваемому метаотношению присвоим имя “*соответствие*” (быть соответствием).

Определение 3.3.9.1. Кортеж *k* задает **соответствие** между множеством *x* и множеством *y*, т.е. имеет место scb-конструкция:



в том и только в том случае, если:

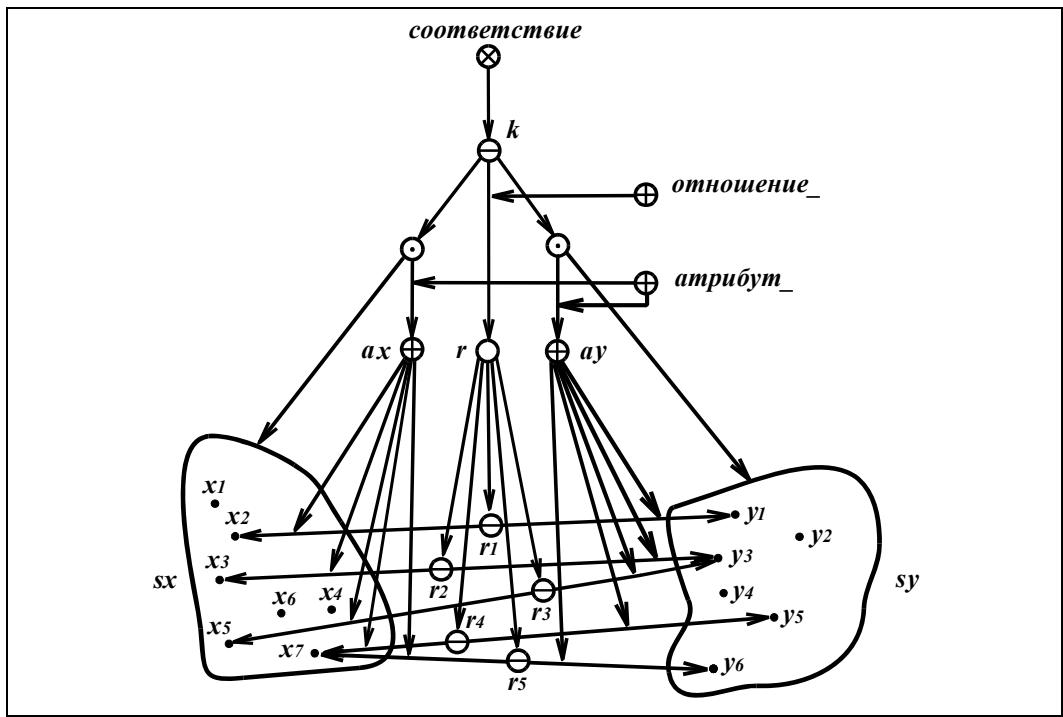
- 1) имеет место конструкция вида:



- 2) *r* – бинарное ориентированное отношение с атрибутами *ax* и *ay*;
k – знак конкретного соответствия между множествами *sx* и *sy*;
sx – множество прообразов;
sy – множество образов;
r – отношение соответствия.
- 3) $r \subseteq (sx(ax) \times sy/ay))$.

На scbg-тексте 3.3.9.1 приведён пример соответствия, взятый из [464] ([Таран Т.А. 1998гн-ОсновыДМ](#)).

SCBg-текст 3.3.9.1. Пример изображения кортежа отношения “*соответствие*”

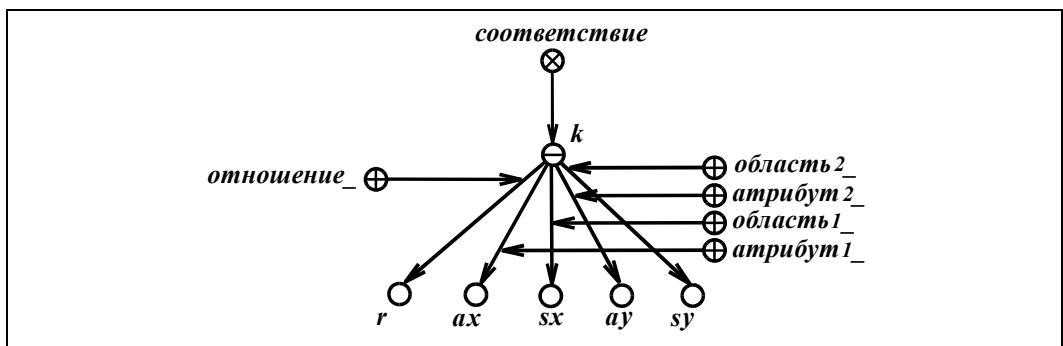


Примечание 1. В общем случае не все элементы множества ***sx*** и множества ***sy*** должны быть элементами области определения отношения ***r***. И соответственно этому, не все элементы множества ***sx*** должны быть элементами проекции отношения ***r*** по атрибуту ***ax*** (см. ***x1***, ***x4***, ***x6***), а также не все элементы множества ***sy*** должны быть элементами проекции отношения ***r*** по атрибуту ***ay*** (см. ***y2***, ***y4***).

Примечание 2. Одному элементу множества ***sx*** может соответствовать несколько элементов множества ***sy*** (см. элемент ***x1*** и пары ***r4*** и ***r5***). Аналогично этому одному элементу множества ***sy*** может соответствовать несколько элементов множества ***sx*** (см. элемент ***y3*** и пары ***r2*** и ***r3***).

Примечание 3. Множество ***sx*** и множество ***sy*** могут иметь общие элементы (в частности, одно из них может быть подмножеством другого) и даже могут совпадать.

SCBg-текст 3.3.9.2. Пример изображения кортежа отношения “*соответствие*” с использованием 5-мощного кортежа



Примечание. Кортеж отношения “*соответствие*” (“*быть соответствием*”) можно было бы сделать 5-мощным, явно включив в число его элементов знаки множеств ***sx*** и ***sy***, а также знаки атрибутов ***ax***

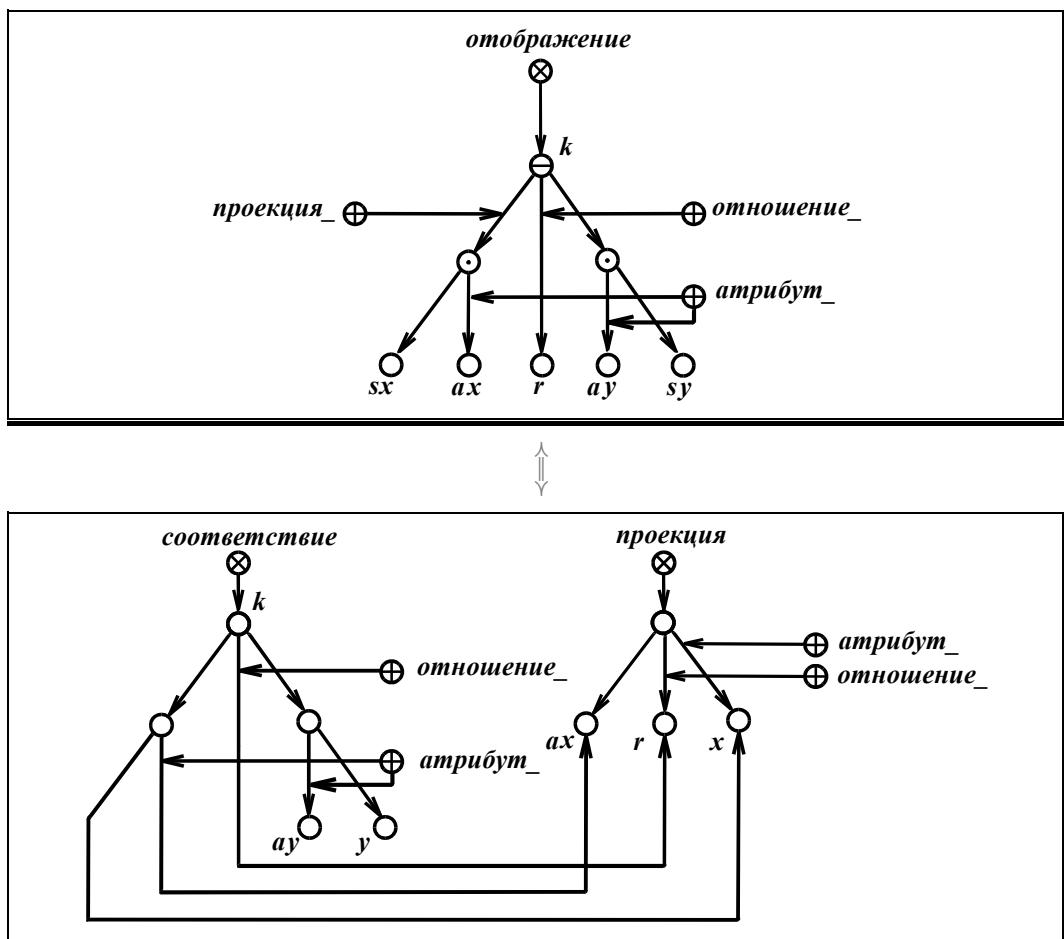
и *ay*. Но тогда пришлось бы вводить дополнительные атрибуты для увязывания *ax* с *sx* и *ay* с *sy*, например, как на scbg-тексте 3.3.9.2. Но у построенного таким образом отношения “**соответствие**” для каждого кортежа *k* будет существовать встречный – компоненты кортежа с атрибутами “**атрибут₁**” и “**область₁**” могут быть заменены местами с компонентами, имеющими атрибуты “**атрибут₂**” и “**область₂**”. Это обусловлено тем, что в соответствии между множествами *sx* и *sy* нет асимметрии.

Типология соответствий определяется следующими факторами:

- как соотносятся между собой множество *sx* и множество *sy* (равенство, включение, строгое пересечение, т.е. наличие как общих, так и не общих элементов, непересечение);
- как соотносятся множества *sx* и *sy* с унарными проекциями отношения *r* по соответствующим атрибутам (указанные множества могут совпадать с соответствующими унарными проекциями, а могут быть их подмножествами);
- имеются ли функциональные зависимости у отношения *r*, и если да, то сколько (одна или две).

Определение 3.3.9.2. Отображением множества *sx* на множество *sy* будем называть такое соответствие между множеством *sx* и множеством *sy*, в котором множество *sx* является унарной проекцией соответствующего бинарного отношения *r* по соответствующему атрибуту *ax*, а не надмножеством указанной унарной проекции (см. scbg-текст 3.3.9.3).

SCBg-текст 3.3.9.3. Запись отображения множеств на языке SCB



Примечание. Подчеркнём, что и с формальной стороны каждое конкретное отображение *k* является соответствием, т.е. метаотношение “**отображение**” (“быть отображением”) является подмножеством (частным случаем) метаотношения “**соответствие**” (“быть соответствием”). Правда, в кортежах метаотношения “**отображение**” используется дополнительный атрибут “**проекция_**” (“быть проекцией”), указывающий на область прообразов отображения.

Определение 3.3.9.3. Сюръекцией множества sx и множества sy будем называть такое соответствие между множеством sx и множеством sy , которое является одновременно отображением множества sx на множество sy , а также отображением множества sy на множество sx .

Определение 3.3.9.4. Однозначным соответствием (функциональным соответствием) из множества sx (с атрибутом ax) во множество sy (с атрибутом ay) будем называть такое соответствие между множеством sx (с атрибутом ax) и множеством sy (с атрибутом ay), у которого входящее в его состав бинарное ориентированное отношение r имеет ключ, каковым является атрибут ax . Это значит, что компонент с атрибутом ax в кортеже, принадлежащем отношению r , однозначно определяет этот кортеж и, следовательно, однозначно определяет компонент с атрибутом ay в указанном кортеже. Иными словами, каждому xi из множества sx соответствует не более одного yi из множества sy .

Определение 3.3.9.5. Взаимно однозначным соответствием множества sx и множества sy будем называть такое соответствие между множествами sx и sy , которое является одновременно функциональным соответствием как от множества sx во множество sy , так и от множества sy во множество sx .

Определение 3.3.9.6. Однозначным отображением (функциональным отображением) множества sx на множество sy будем называть такое соответствие между sx и sy , которое:

- является отображением множества sx на множество sy ;
- является однозначным соответствием из множества sx на множество sy .

Определение 3.3.9.7. Инъекцией множества x во множество y будем называть такое соответствие между x и y , которое:

- является отображением множества x на множество y ;
- является однозначным соответствием из множества y во множество x .

Определение 3.3.9.8. Однозначной сюръекцией (функциональной сюръекцией) из множества sx во множество sy будем называть такое соответствие между sx и sy , которое:

- является отображением sx на sy ;
- является отображением sy на sx ;
- является однозначным соответствием из sx в sy .

Примечание. Если соответствие является функциональной сюръекцией из sx в sy , то оно является сюръекцией и инъекцией множества sy (!) во множество sx (!).

Примечание. Функциональную сюръекцию из множества sx во множество sy называют также **биекцией** множества sy (!) во множество sx (!).

Определение 3.3.9.9. Взаимно однозначное отображение sx в sy – это:

- взаимно однозначное соответствие sx и sy ;
- отображение sx на sy .

Определение 3.3.9.10. Взаимно однозначная сюръекция – это:

- взаимно однозначное соответствие sx и sy ;
- сюръекция sx и sy .

Для каждого из вышеперечисленных классов соответствий в языке SCB вводятся ключевые узлы: “**сюръекция**”, “**функциональное соответствие**”, “**взаимно функциональное соответствие**”, “**функциональное отображение**” и т.д.

1.3.10. Типология тернарных отношений и метаотношения над ними

Ключевые понятия: тернарное отношение; коммутативное классическое тернарное отношение; ассоциативное тернарное отношение; тернарное отношение с одной функцией; тернарное отношение с двумя функциями; тернарное отношение с тремя функциями; коммутативное отношение; некоммутативное отношение; ассоциативное отношение; неассоциативное отношение; дистрибутивность.

Определение 3.3.10.1. Коммутативное классическое тернарное отношение – это отношение, у которого для каждого кортежа вида (x, y, z) имеется кортеж вида (y, x, z).

Примером коммутативного классического тернарного отношения является классический вариант отношения сложения (см. “[сложение-1](#)” в пункте 3.3.3). Общее определение коммутативного отношения (необязательно тернарного и необязательно классического) также приведено в [пункте 3.3.3](#).

Определение 3.3.10.2. Ассоциативное тернарное отношение – это отношение, у которого для каждой пары кортежей вида:

($a_1 : x, a_2 : y, a_3 : xy$), ($a_1 : xy, a_2 : z, a_3 : xyz$)

имеются кортежи вида:

($a_1 : y, a_2 : z, a_3 : yz$), ($a_1 : x, a_2 : yz, a_3 : yz$).

К числу метаотношений, заданных на множестве тернарных отношений, относится метаотношение “[дистрибутивность](#)” (см. пункт 3.3.13).

Упражнения к пункту 3.3.10.

Упражнение 3.3.10.1. Являются ли функции, соответствующие тернарным отношениям, операциями?

Упражнение 3.3.10.2. Может ли ассоциативное отношение быть некоммутативным?

Упражнение 3.3.10.3. Может ли коммутативное отношение быть неассоциативным?

1.3.11. Отношения над множествами

Идентификаторы ключевых scb-узлов: *включение множества, строгое включение множества, равенство множеств, эквивалентность множеств по совпадению элементов, пара пересекающихся множеств, объединение множеств, соединение множеств, разбиение множеств, пересечение множеств, разность множеств, симметрическая разность множеств.*

Каждое отношение, принадлежащее к классу отношений над множествами, характерно тем, что в его область определения входят знаки всевозможных множеств.

В подразделе 3.1 были рассмотрены основные соотношения между множествами, а также было рассмотрено описание этих соотношений в языке SCBs. Сейчас мы рассмотрим трактовку этих соотношений в графическом реляционном языке SCBg. Для этого необходимо всем основным теоретико-множественным соотношениям поставить в соответствие определенные отношения над множествами.

К числу бинарных отношений над множествами принадлежат отношения, которым поставим в соответствие следующие идентификаторы (здесь используется синтаксис языка SCBs):

- **включение множества**
 - = *быть включением одного множества /* подмножества */ в другое /* надмножество */*
 - = *Множество всевозможных ориентированных пар, каждая из которых связывает некоторое множество с его подмножеством*

-
- = теоретико-множественное включение;
 - строгое включение множества
 - = быть строгим включением одного множества /* собственного подмножества */ в другое /* надмножество */
 - = Множество всевозможных ориентированных пар, каждая из которых связывает некоторое множество с его собственным подмножеством
 - = теоретико-множественное строгое включение;
 - равенство множеств
 - = быть равными множествами
 - = Множество всевозможных неориентированных пар, каждая из которых связывает равные друг другу множества
 - = теоретико-множественное равенство;
 - эквивалентность множеств по совпадению элементов
 - = множества с одинаковыми элементами
 - = быть парой множеств, имеющих одинаковые элементы
 - = Множество всевозможных неориентированных пар, каждая из которых связывает множества с одинаковыми элементами;
 - пара пересекающихся множеств
 - = быть парой пересекающихся множеств
 - = Множество всевозможных неориентированных пар, каждая из которых связывает два пересекающихся множества;

Нельзя также забывать, что к числу бинарных отношений над множествами относятся следующие базовые для языка SCB отношения:

- **Отношение принадлежности**
 - = Множество всевозможных знаков пар принадлежности
 - = пара принадлежности
 - = Принадлежность;
- **Отношение непринадлежности**
 - = Множество всевозможных знаков пар не являющихся парами принадлежности
 - = Множество всевозможных знаков пар непринадлежности
 - = пара непринадлежности;

В языке SCB знак пары принадлежности называется scb-дугой и изображается (в языке SCBg) линией со стрелкой. В языке SCB знак пары непринадлежности будем называть **негативной scb-дугой** и изображать (в языке SCBg) перечеркнутой линией со стрелкой. Следует особо подчеркнуть то, что отношение непринадлежности является удобным средством представления (изображения) множеств, каждое из которых состоит из каких угодно элементов, кроме "неэлементов" некоторого заданного множества. Множество указанного вида будем называть отрицанием заданного множества, дополнением заданного множества до некоторого условного универсального множества или негативным множеством по отношению к заданному множеству. Итак, принадлежность некоторого элемента **x** множеству, которое является отрицанием (универсальным дополнением) множества **s**, изображается в языке SCB путем проведения пары непринадлежности из **s** в **x**. Таким способом, в частности, представляются и "отрицательные" отношения:

- не включение множества;
- не быть строгим включением множества /* не путать с нестрогим включением */;
- неравенство множеств;
- не быть множествами с одинаковыми элементами;
- пара непересекающихся множеств.

К числу тернарных отношений над множествами принадлежат отношения, которым поставим в соответствие следующие идентификаторы:

- **объединение множеств**
 = **объединение**
 = **множество ориентированных троек, каждая из которых связывает некоторые два множества с результатом их объединения**
 = **теоретико-множественное объединение**;
- **соединение множеств**
 = **множество ориентированных троек, каждая из которых связывает некоторые два множества с результатом их соединения**;
- **разбиение множества**
 = **разбиение**
 = **множество ориентированных троек, каждая из которых связывает некоторые два непересекающихся множества с результатом их объединения**;
- **пересечение множеств**
 = **множество ориентированных троек, каждая из которых связывает некоторые два множества с результатом их пересечения**
 = **теоретико-множественное пересечение**;
- **разность множеств**
 = **множество ориентированных троек, каждая из которых связывает некоторые два множества /* уменьшаемое и вычитаемое */ с их разностью**
 = **теоретико-множественная разность**;
- **симметрическая разность множеств**
 = **множество ориентированных троек, каждая из которых связывает некоторые два множества с их симметрической разностью**
 = **теоретико-множественная симметрическая разность**.

Подчеркнем, что некоторые из перечисленных отношений целесообразно обобщить, разрешив использование связок произвольной мощности. Так, например,

- отношение “**равенство множеств**” будем трактовать как множество неориентированных связок, каждая из которых представляет собой семейство из произвольного числа равных друг другу множеств;
- отношение “**множество с одинаковыми элементами**” будем трактовать как множество неориентированных связок, каждая из которых представляет собой семейство из произвольного числа множеств, имеющих по крайней мере один общий для них всех элемент;
- отношение “**объединение множеств**” будем трактовать как множество ориентированных связок, каждая из которых связывает некоторый набор из произвольного числа множеств с результатом их объединения;
- отношение “**соединение множеств**” будем трактовать как множество ориентированных связок, каждая из которых связывает некоторый набор из произвольного числа множеств с результатом их соединения;
- отношение “**разбиение множества**” будем трактовать как множество ориентированных связок, каждая из которых связывает некоторый набор из произвольного числа попарно непересекающихся множеств с результатом их объединения;
- отношение “**пересечение множеств**” будем трактовать как множество ориентированных связок, каждая из которых связывает некоторый набор из произвольного числа множеств с результатом их пересечения.

Кроме этого введем ещё одно отношение со связками различной мощности:

- отношение “**попарно непересекающиеся множества**”, которое будем трактовать как множество неориентированных связок, каждая из которых представляет собой семейство из произвольного числа попарно непересекающихся множеств. Заметим, что 2-мощный вариант связок рассматриваемого отношения является вырожденным и может быть выражен с помощью отношения непринадлежности, т.е. путем отрицания принадлежности этой связки к отношению “**пересекающиеся множества**”.

Ниже приводятся определения перечисленных отношений с использованием средств языка SCB, а также примеры использования этих отношений для записи соответствующих соотношений между множествами.

Определение 3.3.11.1. Определение отношения “*включение множества*”. Будем говорить, что $k \leftarrow \text{включение множества}$; в том и только в том случае, если:

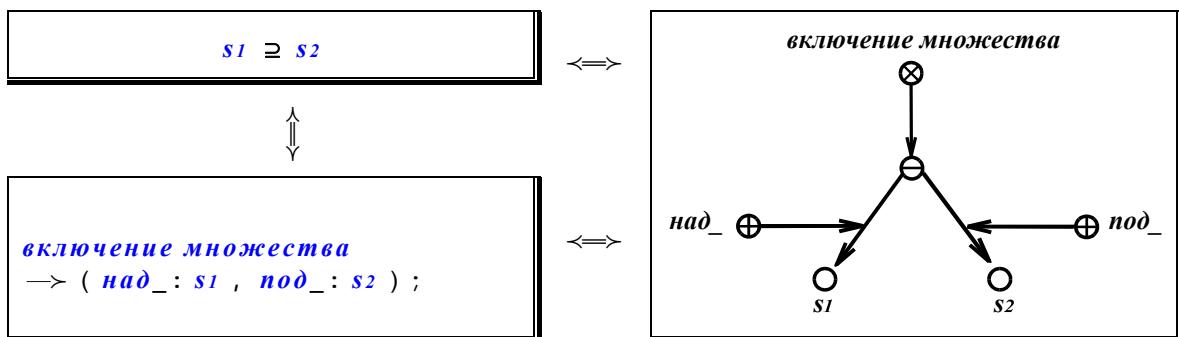
- $k = (\text{над_}: s_1, \text{под_}: s_2)$; , т.е. k – 2-мощный кортеж с атрибутом “*над_*” (быть надмножеством) и атрибутом “*под_*” (быть подмножеством);
- для каждой конструкции вида $s_2 \rightarrow x$; существует конструкция вида $s_1 \rightarrow x$; , т.е. каждый элемент множества s_2 является и элементом множества s_1 . Точнее, каждому вхождению какого-либо элемента во множество s_2 соответствует вхождение этого же элемента во множество s_1 .
- для каждого x количество конструкций вида $s_2 \rightarrow x$; не превышает количества конструкций вида $s_1 \rightarrow x$. То есть каждый элемент множества s_1 либо вообще не входит в число элементов множества s_2 , либо входит с меньшей кратностью, либо входит с равной кратностью (но не с большей кратностью).

В рамках кортежа, принадлежащего отношению “*включение множества*”, компонент с атрибутом “*над_*” будем называть надмножеством, а компонент с атрибутом “*под_*” – подмножеством.

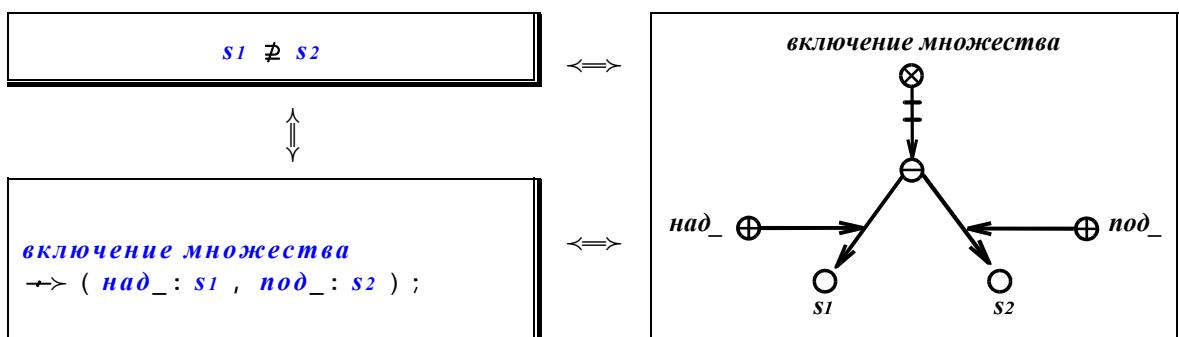
Отношение “*включение множества*” является: бинарным, ориентированным, классическим, без функций, рефлексивным, антисимметричным, транзитивным.

На scb-текстах 3.3.11.1 и 3.3.11.2 приведены примеры использования отношения “*включение множества*” для изображения соотношений включения и невключения множества, приведенных в таблице 3.1.1.

SCB-текст 3.3.11.1. Варианты изображения соотношения нестрогого включения множества s_2 во множество s_1



SCB-текст 3.3.11.2. Варианты изображения соотношения нестрогого невключения множества s_2 во множество s_1

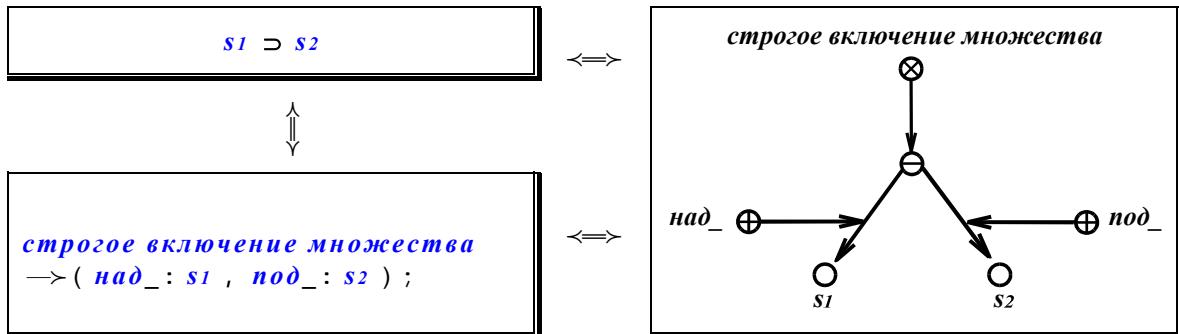


Определение 3.3.11.2. Определение отношения “*строгое включение множества*”. Будем говорить, что $k \leftarrow \text{строгое включение множества}$; в том и только в том случае, если:

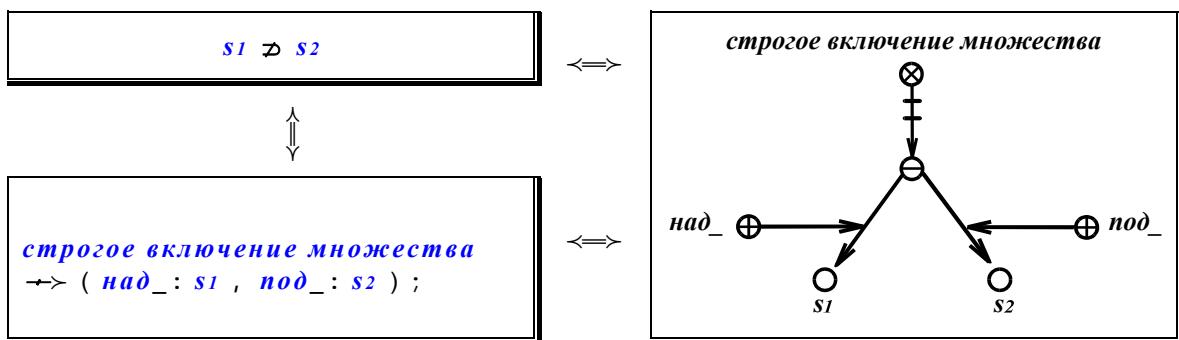
- $k = (\text{над_}: s_1, \text{под_}: s_2)$;
- $k \leftarrow \text{включение множества}$;
- существует по крайней мере один элемент x , для которого справедливо следующее:
 - $s_1 \rightarrow x$;
 - имеет место следующая дизъюнкция:
 - либо не существует конструкций вида $s_2 \rightarrow x$;
 - либо количество конструкций вида $s_1 \rightarrow x$; больше, чем количество конструкций вида $s_2 \rightarrow x$; (т.е. число позитивных scb-дуг, проведенных из s_1 в x , больше числа позитивных scb-дуг, проведенных из s_2 в x).

На scb-текстах 3.3.11.3 и 3.3.11.4 приведены примеры использования отношения “*строгое включение множества*” для изображения соотношений строгого включения и невключения множества, приведенных в табл. 3.1.1.

SCB-текст 3.3.11.3. Варианты изображения соотношения строгого включения множества s_2 во множество s_1



SCB-текст 3.3.11.4. Варианты изображения соотношения строгого невключения множества s_2 во множество s_1



Определение 3.3.11.3. Определение метаотношения “*семейство вложенных множеств*”. Будем говорить, что $k \leftarrow \text{семейство вложенных множеств}$; в том и только в том случае, если:

- множество k имеет по крайней мере два элемента;
- для каждой конструкции вида $k \rightarrow x_1, x_2$; имеет место
 - либо конструкция вида *включение множества* $\rightarrow (\text{над_}: x_1, \text{под_}: x_2)$;
 - либо конструкция вида *включение множества* $\rightarrow (\text{над_}: x_2, \text{под_}: x_1)$;

Таким образом, отношение “**семейство вложенных множеств**” представляет собой неориентированное отношение со связками разной мощности.

Определение 3.3.11.4. Минимальным в семействе вложенных множеств будем называть множество, которое включено во все остальные множества указанного семейства.

Определение 3.3.11.5. Определение отношения “**равенство множеств**”.

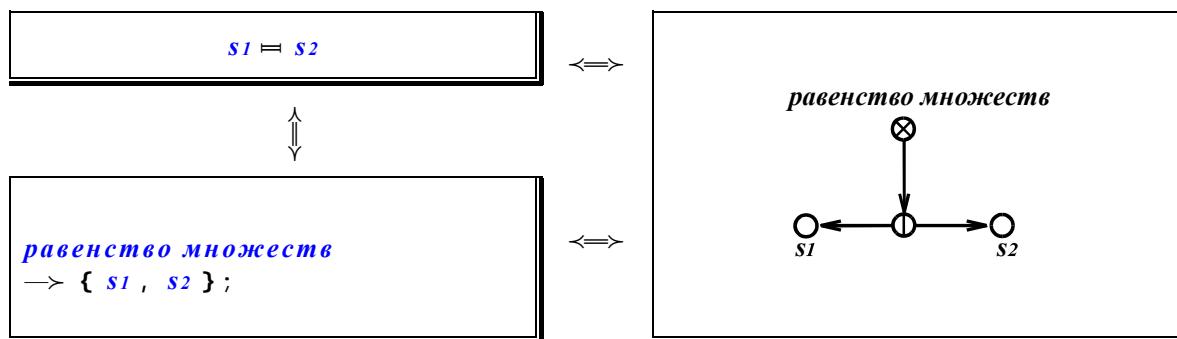
Будем говорить, что $k \leftarrow$ **равенство множеств**; в том и только в том случае, если:

- $k = \{ s_1, s_2, \dots, s_n \}$;
- для каждой пары множеств $\{ s_i, s_j \}$ из семейства множеств $\{ s_1, s_2, \dots, s_n \}$ имеет место следующее:
 - $(\text{над_} : s_i, \text{под_} : s_j) \leftarrow$ **включение множества**;
 - $(\text{над_} : s_j, \text{под_} : s_i) \leftarrow$ **включение множества**;

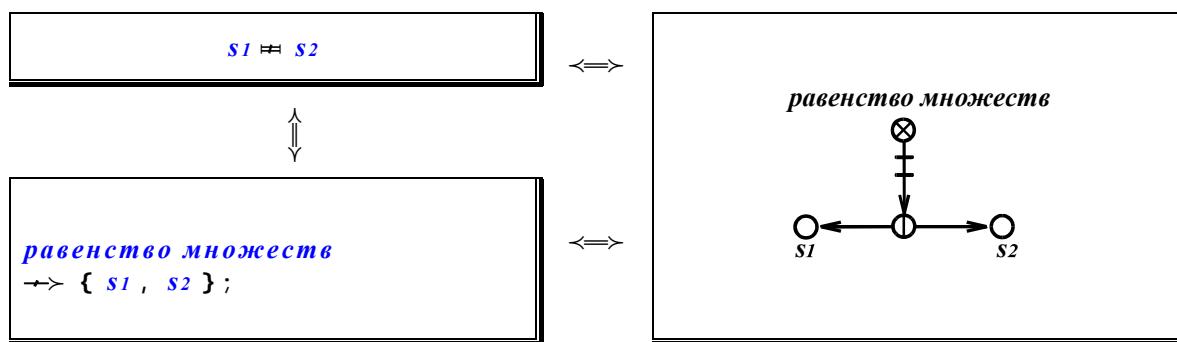
Иначе говоря, равные множества – это множества, состоящие из одинаковых элементов, каждый из которых имеет одинаковое количество вхождений во все указанные множества.

На scb-текстах 3.3.11.5 и 3.3.11.6 приведены примеры использования отношения “**равенство множеств**” для изображения соотношений равенства и неравенства множеств, приведенных в табл. 3.1.1.

SCB-текст 3.3.11.5. Варианты изображения равенства множеств s_2 и s_1



SCB-текст 3.3.11.6. Варианты изображения неравенства множеств s_2 и s_1



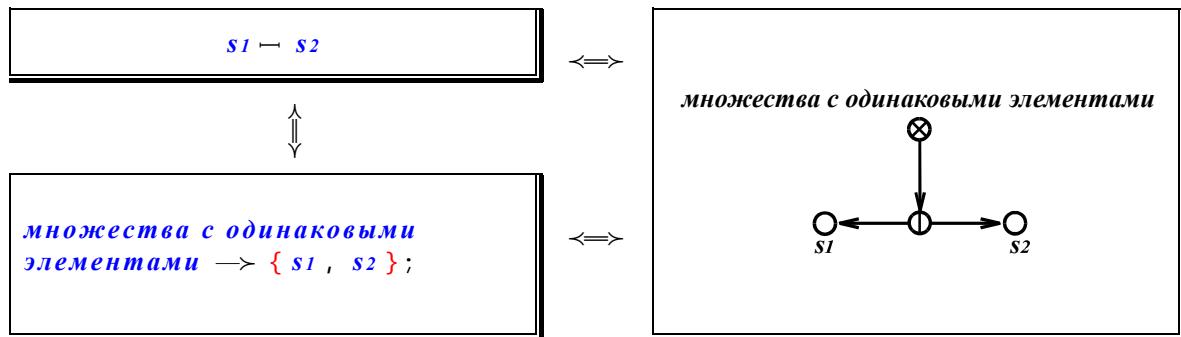
Определение 3.3.11.6. Определение отношения “**множества с одинаковыми элементами**”. Будем говорить, что $k \leftarrow$ **множества с одинаковыми элементами**; в том и только в том случае, если:

- $k = \{ s_1, s_2, \dots, s_n \}$;
- для каждой пары множеств $\{ s_i, s_j \}$ из семейства множеств $\{ s_1, s_2, \dots, s_n \}$ имеет место следующее:
 - для каждой конструкции вида $s_i \rightarrow x$; существует конструкция вида $s_j \rightarrow x$; т.е. каждый элемент множества s_i является также и элементом множества s_j (при этом кратность вхождения этого элемента в s_i и в s_j не учитывается);

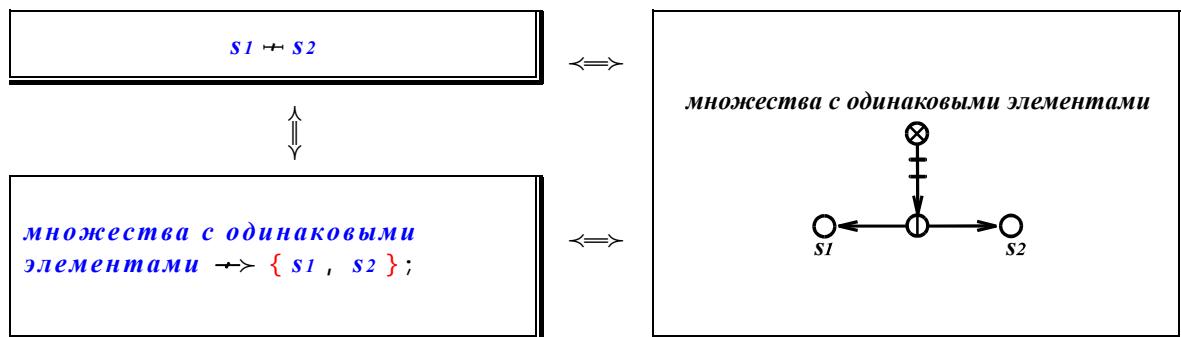
- для каждой конструкции вида $sj \rightarrow x$; существует конструкция вида $si \rightarrow x$;

На scb-текстах 3.3.11.7 и 3.3.11.8 приведены примеры использования отношения “*множества с одинаковыми элементами*” для изображения соответствующих соотношений, приведенных в таблице 3.1.1.

SCB-текст 3.3.11.7. Варианты изображения множеств с одинаковыми элементами



SCB-текст 3.3.11.8. Варианты изображения множеств с неодинаковыми элементами



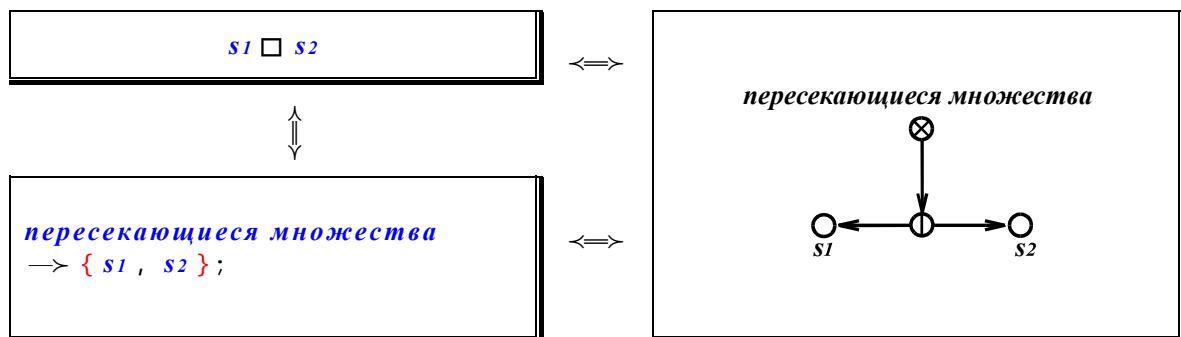
Определение 3.3.11.7. Определение отношения “*пересекающиеся множества*”.

Будем говорить, что $k \rightarrow$ *пересекающиеся множества*; в том и только в том случае, если:

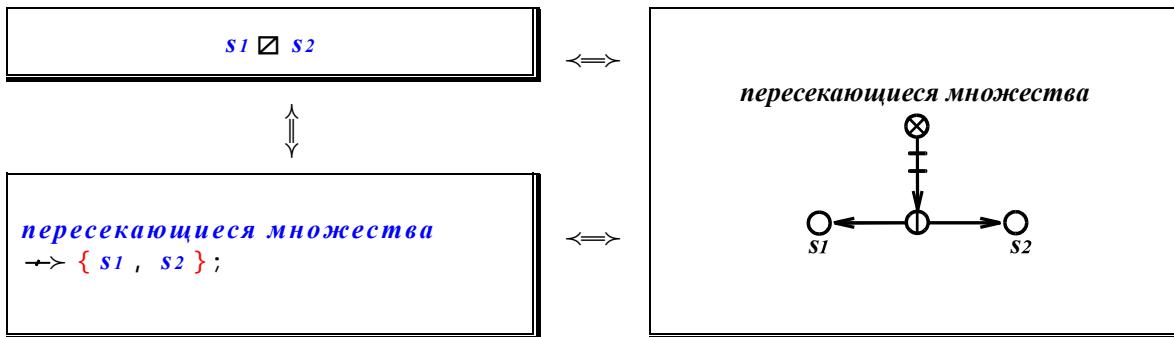
- $k = \{ s_1, s_2, \dots, s_n \}$;
- существует x такой, что для каждого множества s_i из семейства $\{ s_1, s_2, \dots, s_n \}$ имеет место конструкция $s_i \rightarrow x$;

На scb-текстах 3.3.11.9 и 3.3.11.10 приведены примеры использования отношения “*пересекающиеся множества*” для изображения соответствующих соотношений, приведенных в табл. 3.1.1.

SCB-текст 3.3.11.9. Варианты изображения пересекающихся множеств

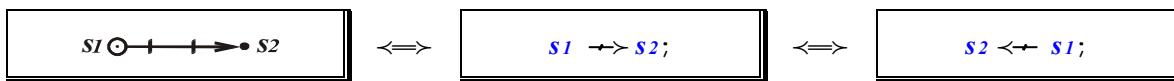


SCB-текст 3.3.11.10. Варианты изображения непересекающихся множеств



Определение 3.3.11.8. Определение отношения “*Отношение непринадлежности*”. Синонимичными идентификаторами этого отношения являются также “*пара непринадлежности*”, “*быть парой непринадлежности*” (см. подраздел 2.6). Будем говорить, что знак ориентированной (упорядоченной) пары (s_1, s_2) является элементом отношения *непринадлежности* в том и только в том случае, если не существует ни одной пары принадлежности вида $s_1 \rightarrow s_2$.

Пары отношения *непринадлежности* называются парами *непринадлежности* и изображаются scbg-конструкциями и scbs-конструкциями следующего вида:



См. также подразделы 2.3, 2.4, 2.5.

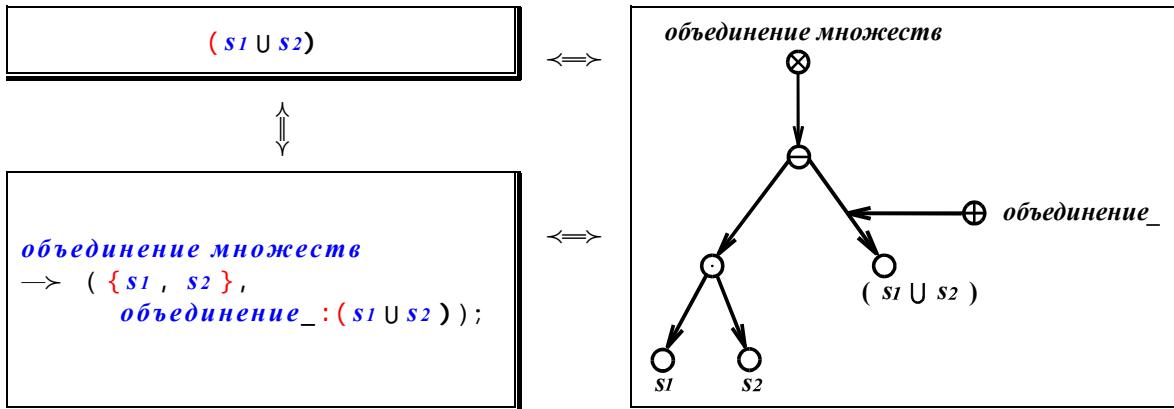
Определение 3.3.11.9. Определение отношения “*объединение множеств*”. Будем говорить, что $k \leftarrow$ *объединение множеств*; в том и только в том случае, если:

- $k = (\{s_1, s_2, \dots, s_n\}, \text{объединение_} : sm)$;
- для каждой конструкции вида $si \rightarrow e$; (где si есть элемент множества k , не отмеченный атрибутом “*объединение_*”) существует конструкция $sm \rightarrow e$;
- для каждой конструкции вида $sm \rightarrow e$; существует по крайней мере одно множество si из семейства $\{s_1, s_2, \dots, s_n\}$ такое, что $si \rightarrow e$;
- для каждой конструкции вида $sm \rightarrow e$; имеет место следующее:
пусть x_1 есть кратность вхождения элемента e во множество si (т.е. количество вхождений e в si), x_2 – количество вхождений e в s_2 и т.д., xn – количество вхождений e в s_n и пусть $xmax$ – максимальное из этих чисел,
тогда количество вхождений указанного элемента e во множество sm равняется $xmax$.

Примечание. Если элемент e не входит в число элементов множества si , то количество вхождений этого элемента в указанное множество считается равным нулю.

На scb-тексте 3.3.11.11 приведены примеры использования отношения “*объединение множеств*” для изображения соответствующего соотношения, приведенного в таблице 3.1.1.

SCB-текст 3.3.11.11. Варианты изображения отношения объединения множеств



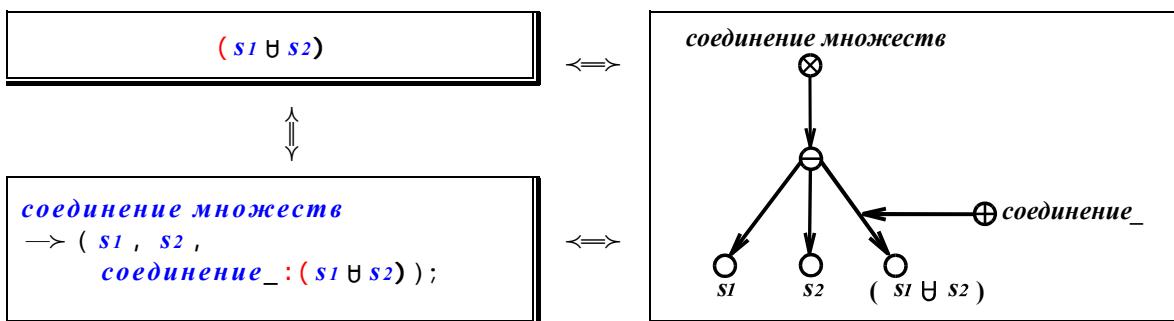
Определение 3.3.11.10. Определение отношения “*соединение множеств*”.

Будем говорить, что $k \leftarrow \text{соединение множеств}$; в том и только в том случае, если:

- $k = (s1, s2, \dots, sn, \text{соединение_}: sm)$;
- для каждой конструкции вида $si \rightarrow e$; (где si есть множество из семейства $\{s1, s2, \dots, sn\}$) существует конструкция $sm \rightarrow e$;
- для каждой конструкции вида $sm \rightarrow e$; существует множество si из семейства $\{s1, s2, \dots, sn\}$ такое, что $si \rightarrow e$;
- для каждого элемента e имеет место следующее:
пусть $x1$ – количество вхождений e в $s1$, $x2$ – количество вхождений e в $s2$ и т.д. xen – количество вхождений e в sn ,
тогда количество вхождений e в sm равно $\sum_{i=1}^n xi$.

На scb-тексте 3.3.11.12 приведены примеры использования отношения “*соединение множеств*” для изображения соответствующего соотношения, приведенного в табл. 3.1.1.

SCB-текст 3.3.11.12. Варианты изображения отношения соединения множеств



Определение 3.3.11.11. Определение отношения “*разбиение множеств*”.

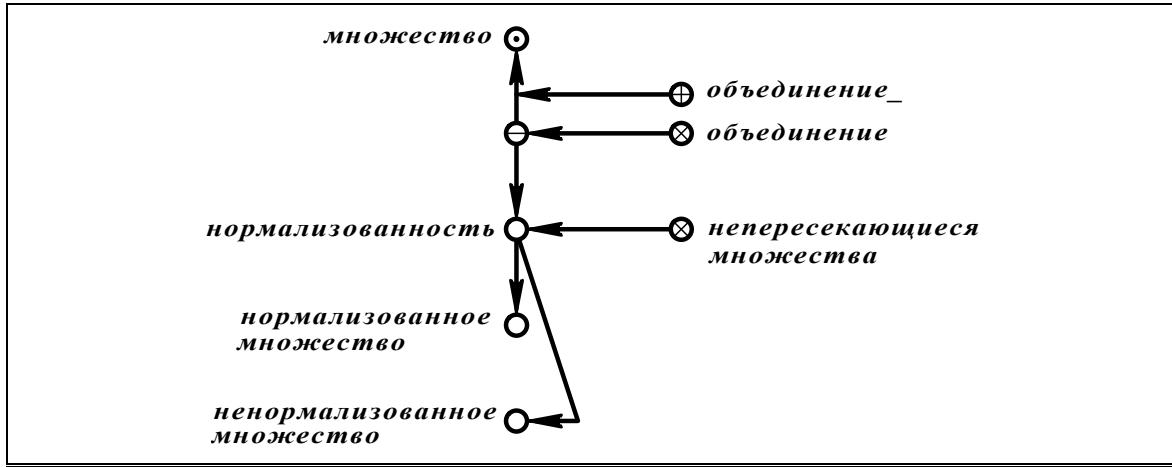
Будем говорить, что $k \leftarrow \text{разбиение множеств}$; в том и только в том случае, если:

- $k = (\{s1, s2, \dots, sn\}, \text{объединение_}: sm); k \leftarrow \text{объединение множеств};$
- для каждой пары множеств $\{si, sj\}$ из семейства $\{s1, s2, \dots, sn\}$ не существует e такого, что $si \rightarrow e; sj \rightarrow e$; , т.е. не существует общих элементов у множеств si и sj .

Отношение разбиения играет очень важную роль при представлении знаний – с его помощью осуществляется классификация самых различных понятий. При представлении отношения разбиения можно

явно не вводить знак этого отношения, а пользоваться знаком отношения объединения и знаком отношения “**непересекающиеся множества**” (см. scb-текст 3.3.11.13).

SCB-текст 3.3.11.13. Формальная запись на языке SCB типологии множеств, которая приведена в таблице 2.1.1

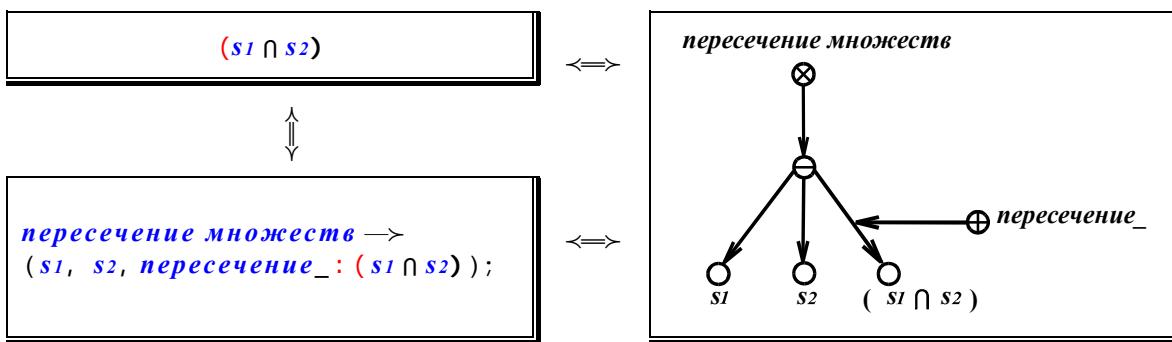


Определение 3.3.11.12. Определение отношения “**пересечение множеств**”. Будем говорить, что $k \leftarrow \text{пересечение множеств}$; в том и только в том случае, если:

- $k = (s_1, s_2, \dots, s_n, \text{пересечение_} : sm)$;
- для каждой конструкции вида $s_1, s_2, \dots, s_n \rightarrow e$; существует конструкция вида $sm \rightarrow e$;
- и наоборот: из $sm \rightarrow e$; следует $s_1, s_2, \dots, s_n \rightarrow sm$;
- для каждого e , являющегося элементом множества sm , имеет место следующее:
пусть x_1 – количество вхождений e в s_1 , x_2 – количество вхождений e в s_2 и т.д., xn – количество вхождений e в sn , $xmin$ – минимальное из перечисленных чисел,
тогда количество вхождений e в sm равно $xmin$.

На scb-тексте 3.3.11.14 приведены примеры использования отношения “**пересечение множеств**” для изображения соответствующего соотношения, приведенного в табл. 3.1.1.

SCB-текст 3.3.11.14. Варианты изображения отношения пересечения множеств



Определение 3.3.11.13. Определение отношения “**разность множеств**”.

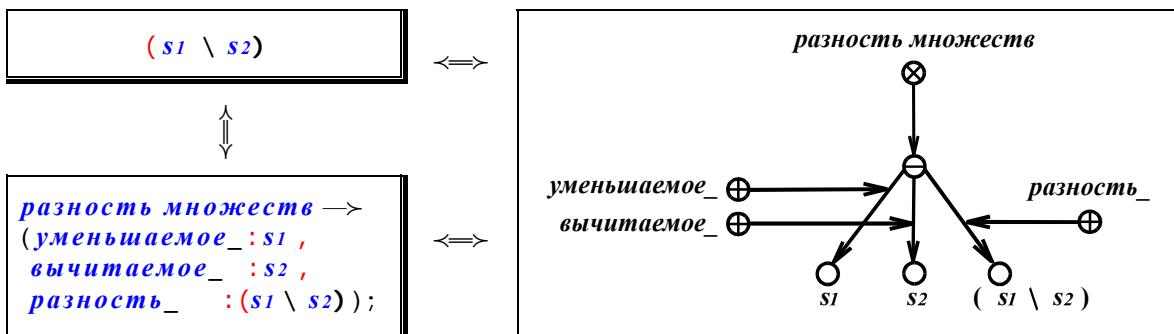
Будем говорить, что $k \leftarrow \text{разность множеств}$; в том и только в том случае, если:

- $k = (\text{уменьшаемое_} : s_1, \text{вычитаемое_} : s_2, \text{разность_} : s_3)$;
- для каждой конструкции вида $s_1 \rightarrow e; s_2 \rightarrow e$; существует одна и только одна конструкция вида $s_3 \rightarrow e$; т.е. каждый элемент множества s_1 , не являющийся элементом множества s_2 , входит в число элементов множества s_3 , причем указанный элемент входит в s_1 и в s_3 с одинаковой кратностью;

- каждый элемент e , который входит в s_1 с кратностью x_1 , а в s_2 – с кратностью x_2 и при этом $x_2 \geq x_1$, не входит во множество s_3 ;
- каждый элемент e , который входит в s_1 с кратностью x_1 , а в s_2 – с кратностью x_2 и при этом $x_2 < x_1$, входит во множество s_3 с кратностью $(x_1 - x_2)$;
- каждый элемент e , входящий в s_3 с кратностью x_3 , входит во множество s_1 с кратностью $(x_1 + x_2)$, где x_2 – кратность вхождения этого элемента во множество s_2 . Напомним при этом, что кратность, равная нулю, означает отсутствие вхождения элемента во множество.

На scb-тексте 3.3.11.15 приведены примеры использования отношения “**разность множеств**” для изображения соответствующего соотношения, приведенного в табл. 3.1.1.

SCB-текст 3.3.11.15. Варианты изображения отношения разности множеств

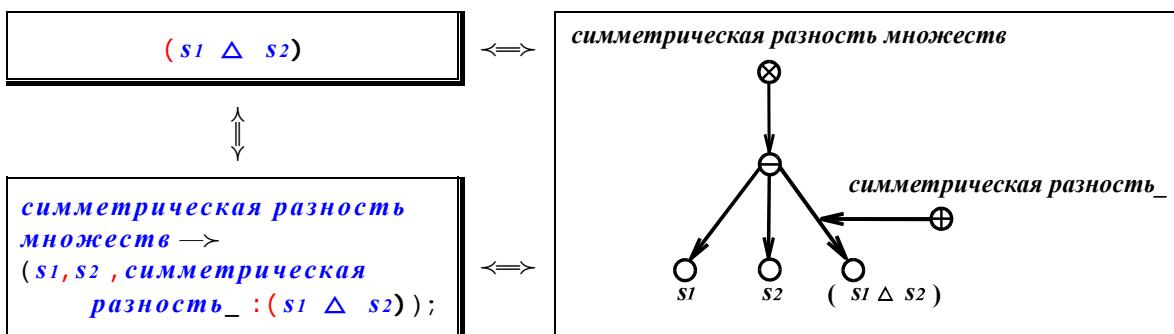


Определение 3.3.11.14. Определение отношения “**симметрическая разность множеств**”. Будем говорить, что $k \leftarrow$ **симметрическая разность множеств**; в том и только в том случае, если существует конструкция вида:

$$\begin{aligned} k &= (s_1, s_2, \text{разность_} : s_3); \\ (\{s_1, s_2\}, \text{объединение_} : sy) &\leftarrow \text{объединение множеств}; \\ (s_1, s_2, \text{пересечение_} : sp) &\leftarrow \text{пересечение множеств}; \\ (\text{уменьшающее_} : sy, \text{вычитающее_} : sp, \text{разность_} : s3) &\leftarrow \text{разность множеств}; . \end{aligned}$$

На scb-тексте 3.3.11.16 приведены примеры использования отношения “**симметрическая разность множеств**” для изображения соответствующего соотношения, приведенного в табл. 3.1.1.

SCB-текст 3.3.11.16. Варианты изображения отношения симметрической разности множеств



Определение 3.3.11.15. Определение отношения “**семейство попарно непересекающихся множеств**”. Будем говорить, что $k \leftarrow$ **семейство попарно непересекающихся множеств**; в том и только в том случае, если:

- $k = \{s_1, s_2, \dots, s_n\}$;

- для каждой пары множеств $\{si, sj\}$ из семейства множеств $\{s1, s2, \dots, sn\}$ не существует e такого, что $si \rightarrow e; sj \rightarrow e$; т.е. не существует общих элементов у множеств si и sj .

Заметим при этом, что имеет место синонимия следующих идентификаторов:

попарно непересекающиеся множества
 $=$ **семейство попарно непересекающихся множеств** ;

Примечание. Следует четко отличать следующие понятия:

- отношение синонимии (множество знаков пар синонимии, множество знаков пар равенства знаков);
- отношение эквивалентности множеств по набору элементов;
- отношение равенства множеств (множество знаков пар равенства множеств);
- отношение равенства кортежей;
- отношение равномощности множеств (отношение эквивалентности множеств по мощности);
- отношение эквивалентности множеств по количеству элементов.

Завершая рассмотрение отношений над множествами, приведем некоторые теоретико-множественные соотношения между введенными нами отношениями.

включение множества \supset **строгое включение множества** ;
строгое включение множества \subsetneq **равенство множеств** ;
равенство множеств \equiv **множества с одинаковыми элементами** ;
разбиение множества \sqsubset **объединение множеств** ;
разбиение множества \sqsubset **соединение множеств** ;
объединение множеств \sqcup **соединение множеств** .

Примечание. Могут ли пересекаться два ориентированных отношения, которые имеют разные схемы (разные наборы атрибутов)? Да, поскольку каждое отношение, "зная" свою схему, имеет возможность учитывать только свои атрибуты, т.е. атрибуты, входящие в его схему. Поэтому связки разных отношений, являющиеся равными множествами, целесообразно "склеивать" независимо от распределения атрибутов. При этом надо внимательно следить за теми атрибутами, которые являются общими для отношений, связки которых склеиваются. Связки, являющиеся равными множествами и даже равными кортежами, целесообразно сохранять только в рамках одного отношения (кратные связи, встречные связи).

1.3.12. Отношения над кортежами

Идентификаторы ключевых scb-узлов: включение кортежа, пересечение кортежей, равенство кортежей, встречные кортежи.

Определение 3.3.12.1. Определение отношения "**включение кортежа**". Кортеж ki включает в себя кортеж kj (т.е. является его **подкортежем**) в том и только в том случае, если каждый элемент кортежа kj входит в кортеж ki под тем же атрибутом.

Определение 3.3.12.2. Определение отношения "**пересечение кортежей**". Кортеж ke является результатом **пересечения кортежей** ki и kj в том и только в том случае, если каждый элемент кортежа ke входит в кортеж ki и кортеж kj под тем же атрибутом.

Определение 3.3.12.3. Определение отношения "**равенство кортежей**". Будем говорить, что кортеж ki **равен** кортежу kj в том и только в том случае, если:

- указанные кортежи являются равными множествами;
- существует взаимно однозначное соответствие, которое каждому вхождению элемента в кортеж ki ставит в соответствие вхождение этого же элемента, но в кортеж kj , причём с теми же атрибутами и, наоборот, каждому вхождению элемента в кортеж kj ставит в соответствие вхождение этого же элемента в кортеж ki , причём с теми же атрибутами.

Напомним, что вхождение элемента в кортеж может вообще не иметь атрибута, может иметь один атрибут, может иметь несколько атрибутов.

Равные кортежи будем также называть **кратными кортежами**. Частным случаем равных кортежей являются кратные пары принадлежности.

Определение 3.3.12.4. Определение отношения “**встречные кортежи**”. Кортежи, являющиеся равными множествами, но не являющиеся равными кортежами, будем называть **встречными** кортежами.

1.3.13. Отношения над отношениями

Ключевые понятия и идентификаторы ключевых scb-узлов: метаотношений, **схема отношения**, **отношение**, **атрибут по умолчанию**, **область определения**, **домен**, **проекция отношения**, **функциональная зависимость**, **ключевая функциональная зависимость**.

Определение 3.3.13.1. Будем считать, что отношение **r** относится к классу отношений над отношениями (к классу **метаотношений**) в том и только в том случае, если:

- в область определения отношения **r** входят либо знаки **всевозможных** отношений, либо знаки **всех** отношений некоторого класса;
- в состав каждой связки отношения **r** входит знак некоторого отношения.

Анализ области определения метаотношений позволяет выделить следующие классы таких отношений:

- отношения над всевозможными отношениями;
- отношения над классическими отношениями;
- отношения над ориентированными отношениями;
- отношения над неориентированными отношениями;
- отношения над бинарными ориентированными отношениями;
- отношения над бинарными неориентированными отношениями
- и т.д.

Выше при рассмотрении типологии отношений мы ввели целый ряд понятий, формальное уточнение которых требует введения **метаотношений**, соответствующих этим понятиям. К числу указанных понятий относятся такие понятия, как **схема отношения** (см. пункт 3.3.3), **проекция отношения** (см. пункт 3.3.4), **домен**, **функциональная зависимость**, **ключевая функциональная зависимость**, **функция**, **алгебраическая операция** (см. пункт 3.3.5). Каждому из перечисленных понятий поставим в соответствие метаотношение, которое в языке SCB будет обозначаться определенным scb-узлом. Таким образом будут введены scb-узлы со следующими идентификаторами:

- “**схема отношения**”;
- “**область определения**”;
- “**домен**”;
- “**проекция отношения**”;
- “**функциональная зависимость**”;
- “**ключевая функциональная зависимость**”;
- “**функция**”;
- “**алгебраическая операция**”.

Перейдем к более строгому определению этих понятий, приближая тексты самих определений к их записи на формальном логическом языке, который будет построен на основе языка SCB и подробно рассмотрен в разделе 5.

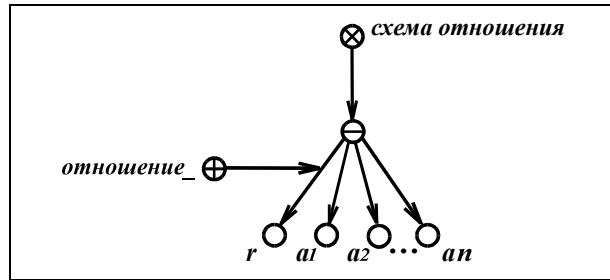
Определение 3.3.13.2. Определение метаотношения “**схема отношения**”, являющегося уточнением понятия схемы отношения (см. пункт 3.3.3). Будем говорить, что множество **k** принадлежит метаотношению “**схема отношения**”, т.е. имеет место конструкция **k** ← **схема отношения**; в том и только в том случае, если:

- множество k является кортежем;
- существует конструкция вида: $k \rightarrow \text{отношение}_r : ri$; где ri – знак множества, трактуемого в рамках кортежа как некое рассматриваемое ориентированное отношение, отношение_r – знак атрибута, указывающего на рассматриваемое отношение (в данном случае на отношение ri);
- все остальные элементы кортежа k отмечаются атрибутом, задаваемым по умолчанию, и трактуются как знаки атрибутов, используемых в кортежах ориентированного отношения ri , – указанные знаки атрибутов будем называть элементами схемы отношения ri ;
- в кортеже k перечислены все элементы схемы отношения ri , т.е. все атрибуты, используемые в этом отношении. При этом если в отношении ri используется атрибут, задаваемый по умолчанию, то он должен быть явно указан следующим образом:
 $k \rightarrow \text{атрибут по умолчанию}$; , где узел с идентификатором **"атрибут по умолчанию"** является знаком атрибута, задаваемого по умолчанию;
- никаких других элементов кортеж k не содержит.

Итак, уточнением понятия схемы отношения (см. пункт 3.3.3) является метаотношение с именем **"схема отношения"** (быть схемой отношения), связки которого являются кортежами, которые имеют мощность от 2 и выше и которые используют атрибут **"отношение_"** (быть отношением) и атрибут, задаваемый по умолчанию (см. scbg-текст 3.3.13.1). Примеры использования метаотношения **"схема отношения"** приведены на scbg-текстах 3.3.13.2 и 3.3.13.3.

SCBg-текст 3.3.13.1. Общий вид отношения **"схема отношения"**

Здесь r – знак некоторого отношения a_1, a_2, \dots, a_n – знаки всех тех и только тех атрибутов, которые используются в связках отношения r .

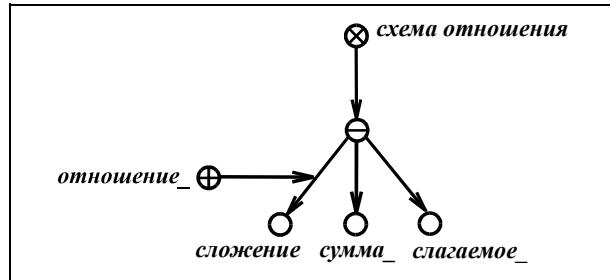


Примечание 1. Метаотношение **"схема отношения"** можно было бы определить и по-другому:

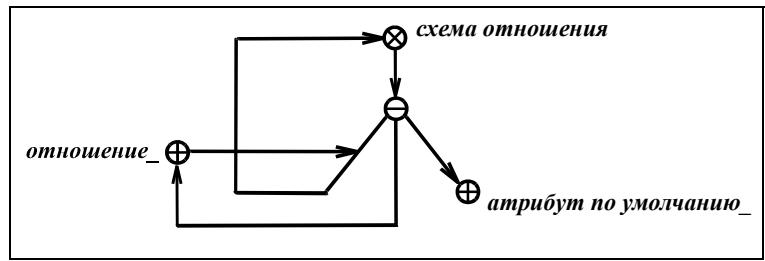
- без атрибута, задаваемого по умолчанию, т.е. с явным введением атрибута **"быть элементом схемы отношения_"**;
- трактовать это метаотношение как бинарное с явным введением знака схемы рассматриваемого отношения.

Примечание 2. Один и тот же кортеж может входить в состав нескольких отношений с разными схемами. Это означает, что в кортежах, принадлежащих заданному отношению, могут использоваться не только атрибуты, входящие в схему этого отношения. Но такие атрибуты в рамках указанного отношения не учитываются. Из всего этого следует также то, что атрибут, задаваемый по умолчанию, соответствует либо тем элементам кортежей, рассматриваемого отношения, которые не отмечены атрибутами, входящими в схему этого отношения (т.е. элементам кортежей, которые вообще никаким атрибутом не отмечены), либо тем элементам кортежей, которые отмечены только атрибутами, не входящими в схему рассматриваемого отношения.

SCBg-текст 3.3.13.2. Пример использования отношения **"схема отношения"** для отношения **"сложение"**



SCBg-текст 3.3.13.3. Схема метаотношения “*схема отношения*”



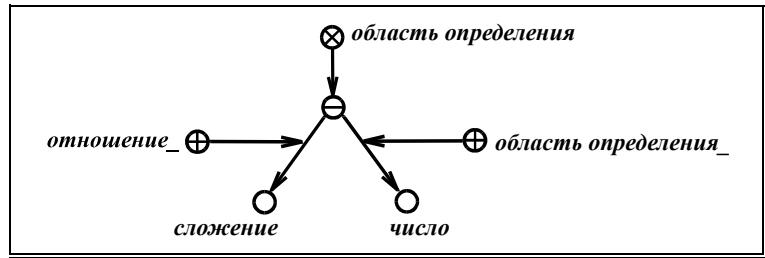
Определение 3.3.13.3. Определение метаотношения “*область определения*”, являющегося уточнением понятия области определения (см. в пункте 3.3.3). Будем утверждать, что

$k \leftarrow \text{область определения}$; в том и только в том случае, если:

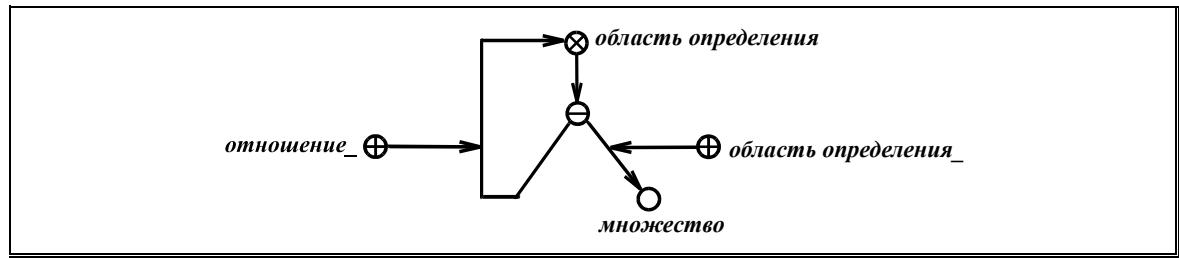
- $k = (\text{отношение_}: r, \text{область определения_}: p)$;
- для каждой конструкции вида $r \rightarrow c \rightarrow x$; существует конструкция $p \rightarrow x$; т.е. каждый элемент каждой связки отношения r является элементом множества p ;
- и наоборот, для каждой конструкции вида $p \rightarrow x$; существует конструкция $r \rightarrow c \rightarrow x$;

Примеры использования метаотношения “*область определения*” приведены на scbg-текстах 3.3.13.4 и 3.3.13.5.

SCBg-текст 3.3.13.4. Пример связи метаотношения “*область определения*”, которая означает, что областью определения отношения “*сложение*” является множество всевозможных чисел



SCBg-текст 3.3.13.5. Пример связи метаотношения “*область определения*”, которая означает, что областью определения отношения “*область определения*” является множество знаков всевозможных множеств (куда входят, в частности, и знаки отношений)



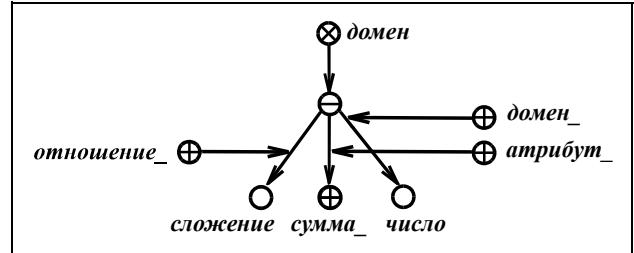
Определение 3.3.13.4. Определение метаотношения “*домен*”, являющегося уточнением понятия домена (унарной проекции). Будем утверждать, что $k \leftarrow \text{домен}$; в том и только в том случае, если:

- $k = (\text{отношение_}: r, \text{домен_}: p, \text{атрибут_}: a)$;
- для каждой конструкции вида $r \rightarrow c \rightarrow a : x$; существует конструкция $p \rightarrow x$; т.е. каждый элемент каждой связки отношения r , имеющий атрибут a , является элементом множества p ;
- и, наоборот, для каждой конструкции вида $p \rightarrow x$; существует конструкция $r \rightarrow c \rightarrow a : x$;
- $p \leftarrow \text{канторовское множество}$; т.е. множество p не содержит кратных элементов, следовательно, не существует такого y , что $p \rightarrow y, y$.

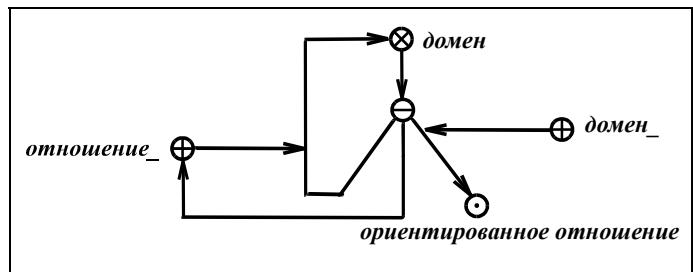
Примечание. Таким образом, для неориентированных отношений домен получить невозможно, т.к. в этих отношениях нет атрибутов.

Примеры использования метаотношения “**домен**” приведены на scbg-текстах 3.3.13.6 и 3.3.13.7.

SCBg-текст 3.3.13.6. Пример связи метаотношения “**домен**”, которая означает, что доменом отношения “**сложение**” по атрибуту “**сумма_**” является множество всевозможных чисел



SCBg-текст 3.3.13.7. Пример связи метаотношения “**домен**”, которая означает, что доменом отношения “**домен**” по атрибуту “**отношение_**” является множество знаков всевозможных ориентированных отношений

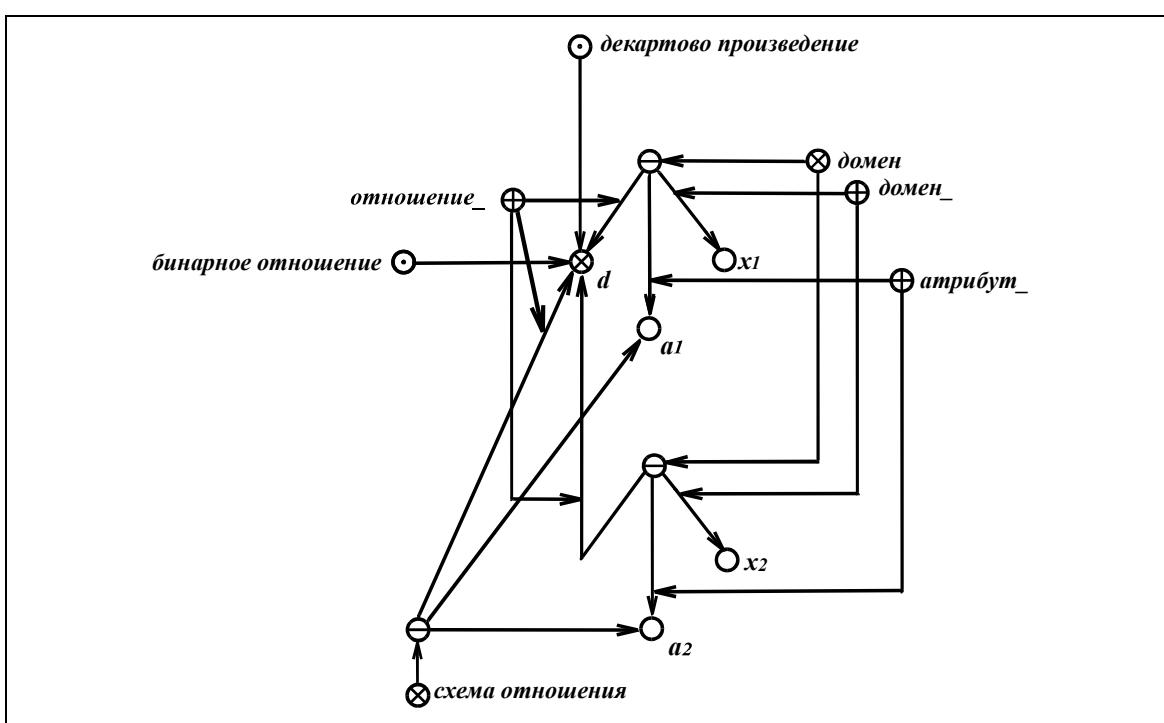


В пункте 3.3.7 в язык SCBs был введен способ условного обозначения декартовых произведений. С помощью метаотношения “**схема отношения**” и метаотношения “**домен**” можно построить scbg-конструкцию, эквивалентную указанному обозначению декартовых произведений (см. scbg-текст 3.3.13.8).

SCBg-текст 3.3.13.8. Пример представления декартова произведения с использованием метаотношений “**схема отношения**” и “**домен**”

$$(x_1(a_1) \times x_2(a_2)) = d;$$





Примечание. Декартово произведение вида ($x_1(a_1) \times x_2(a_2)$) можно трактовать как объединение всех возможных бинарных ориентированных отношений, в которых атрибуты a_1 и a_2 составляют схему этих отношений, а множества x_1 и x_2 являются доменами этих отношений соответственно по атрибуту a_1 и по атрибуту a_2 .

Определение 3.3.13.5. Определение метаотношения “**проекция отношения**”, являющегося уточнением понятия неунарной проекции (см. пункт 3.3.4).

Будем утверждать, что $k \leftarrow \text{проекция отношения}$; в том и только в том случае, если:

- 1) $k \rightarrow \text{отношение_r, проекция_p, атрибут_a1, атрибут_a2, \dots, атрибут_an}$;
- 2) для каждой конструкции вида $r \rightarrow c$; , в которой каждой конструкции $k \rightarrow \text{атрибут_ai}$; соответствует конструкция $c \rightarrow ai : x$; , (т.е. в кортеже c используются все атрибуты, перечисленные в кортеже k , существует одна конструкция вида:
 $p \rightarrow cp$; , для которой:
 - каждой конструкции вида $r \rightarrow c \rightarrow aj : y$; $k \rightarrow \text{атрибут_aj}$; соответствует конструкция $cp \rightarrow aj : y$; , (т.е. в кортеже cp включаются все компоненты кортежа c , имеющие атрибуты, перечисленные в кортеже k);
 - а также каждой конструкции вида $cp \succ gj \succ y$; соответствует конструкция $aj \rightarrow gj$; $r \rightarrow c \rightarrow aj : y$; $k \rightarrow \text{атрибут_aj}$; (т.е. в кортеже cp включаются только компоненты кортежа c , имеющие атрибуты, перечисленные в кортеже k);
- 3) и, наоборот, для каждой конструкции вида $p \rightarrow cp$; существует конструкция:
 $r \rightarrow c$; , для которой:
 - имеет место то, что каждой конструкции вида $r \rightarrow c \rightarrow aj : y$;
 $k \rightarrow \text{атрибут_aj}$; соответствует конструкция $cp \rightarrow aj : y$; (т.е. в кортеже cp включаются все компоненты кортежа c , имеющие атрибуты, перечисленные в кортеже k);
 - имеет место то, что каждой конструкции $cp \succ gj \succ y$; соответствует конструкция $aj \rightarrow gj$; $r \rightarrow c \rightarrow aj : y$; $k \rightarrow \text{атрибут_aj}$; (т.е. в кортеже cp включаются только компоненты кортежа c , имеющие атрибуты, перечисленные в кортеже k);

- 4) не существует такого z , что $z, z \leftarrow p$; (т.е. p не содержит кратных элементов и, следовательно, является канторовским множеством);
- 5) для каждой конструкции вида $z_1, z_2 \leftarrow p$; справедлива следующая конструкция:
 $\{z_1, z_2\} \leftarrow \text{равенство кортежей}$; (т.е. множество p представляет собой отношение, не имеющее кратных связок).

Определение 3.3.13.6. Определение метаотношения “функциональная зависимость”, менее строгое определение этого понятия см. в пункте 3.3.5. Будем утверждать, что $k \leftarrow \text{функциональная зависимость}$; в том и только в том случае, если:

- 1) $k \rightarrow \text{отношение_}: r$;
- 2) существует по крайней мере одна конструкция вида $k \rightarrow \text{аргумент_}: axi$;
- 3) существует по крайней мере одна конструкция вида $k \rightarrow \text{результат_}: ayi$;
- 4) не существует ни одной конструкции вида $k \rightarrow \text{аргумент_}: axi, \text{результат_}: axi$;
- 5) не существует ни одной конструкции вида
 $k \succ g \succ axi, \text{аргумент_}, \text{результат_} \rightarrow gj$;
- 6) для всех c_1 и c_2 имеет место следующая импликация:
 - если существует конструкция вида $r \rightarrow c_1, c_2$; , у которой для всех axi и xi имеет место импликация:
 - если имеет место конструкция вида $k \rightarrow \text{аргумент_}: axi; c_1 \rightarrow axi : xi$;
 - то имеет место конструкция вида $k \rightarrow \text{аргумент_}: axi; c_1 \rightarrow axi : xi$;
 - то для всех ayi и yi имеет место следующая импликация:
 - если имеет место конструкция вида $k \rightarrow \text{аргумент_}: ayi; c_1 \rightarrow ayi : yi$;
 - то имеет место конструкция вида $k \rightarrow \text{аргумент_}: ayi; c_1 \rightarrow ayi : yi$.

Определение 3.3.13.7. Определение метаотношения “ключевая функциональная зависимость”, менее строгое определение этого понятия см. в пункте 3.3.5.

Конструкция вида $k \leftarrow \text{ключевая функциональная зависимость}$;
эквивалентна конструкции:

$k \leftarrow \text{функциональная зависимость}; k \rightarrow \text{отношение_}: r$;

схема отношения $\rightarrow kc \rightarrow \text{отношение_}: r$; ,

для которой справедливо импликативное высказывание:

- если существует конструкция вида
 $kc \succ g \succ a; \text{отношение_} \rightarrow g$; ,
- то справедливо строгое дизъюнктивное высказывание:
 - либо существует конструкция вида
 $k \rightarrow \text{аргумент_}: a$;
 - либо существует конструкция вида
 $k \rightarrow \text{результат_}: a$;

Определение 3.3.13.8. Определение метаотношения “функция”.

Конструкция вида $k \leftarrow \text{функция}$;

эквивалентна конструкции:

$k \rightarrow \text{ключевая функциональная зависимость}; k \rightarrow \text{отношение_}: r$;

$\text{результат_} \rightarrow ay$; ,

для которой:

- существует единственная конструкция:
 - $k \rightarrow \text{результат_}: ai$; /* Для $ai = ay$ */
- имеет место импликация:
 - если дана конструкция вида $k \rightarrow c$; ,
 - то существует единственная конструкция $c \rightarrow ay : yi$;
- имеет место импликация:

- если дана конструкция вида

проекция → *kp* → *отношение_*: *r* ; *проекция_*: *p* ;

у которой:

- существует конструкция:

kp → *атрибут_*: *ai* ; *атрибут_*: *aj* ;

/* Т.е. существует по крайней мере проекция отношения с двумя атрибутами, по которым берется */
/* проекция отношения *r*. Следовательно, речь идет о неунарной проекции. */

- конструкция вида

kp → *атрибут_*: *ax* ;

эквивалентна конструкции:

kp → *аргумент_*: *ax* ;

/* Т.е. каждый атрибут, по которому берется проекция отношения *r*, является */

/* атрибутом-аргументом функциональной зависимости. И наоборот. */

- то справедлива конструкция вида

p ← *декартово произведение* ;

К числу метаотношений над классическими отношениями (т.е. к числу метаотношений, которые не распространяются на область всевозможных отношений), в частности, относятся:

- метаотношение “**минимальное декартово произведение**” – это метаотношение, которое является подмножеством метаотношения “**включение множества**” и которое связывает каждое классическое отношение с таким декартовым произведением, которое является его минимальным надмножеством (т.е. таким декартовым произведением, для которого не существует другого декартова произведения, которое было бы подмножеством первого и надмножеством рассматриваемого классического отношения);
- метаотношение “**дополнение до декартова произведения**” – это метаотношение, которое является подмножеством метаотношения “**разбиение множества**” и которое связывает каждое классическое отношение с его минимальным декартовым произведением и с разностью минимального декартова произведения и рассматриваемого классического отношения;
- метаотношение “**соединение отношений**” – это тернарное метаотношение, каждый кортеж которого связывает два классических отношения, имеющих один общий атрибут, с классическим отношением, каждый кортеж которого есть результат объединения одного из кортежей, принадлежащих первому исходному отношению, и одного из кортежей, принадлежащих второму исходному отношению (при этом в объединяемых кортежах компоненты, имеющие общий для исходных отношений атрибут, должны совпадать).

Очевидно, что для каждого классического отношения существует единственное минимальное декартово произведение и единственное дополнение до этого минимального декартова произведения.

Очевидно также, что если два соединяемых классических отношения будут являться соответственно *m*-арным и *n*-арным, то классическое отношение, полученное в результате их соединения по указанному (общему) атрибуту, будет (*m + n - 1*)-арным.

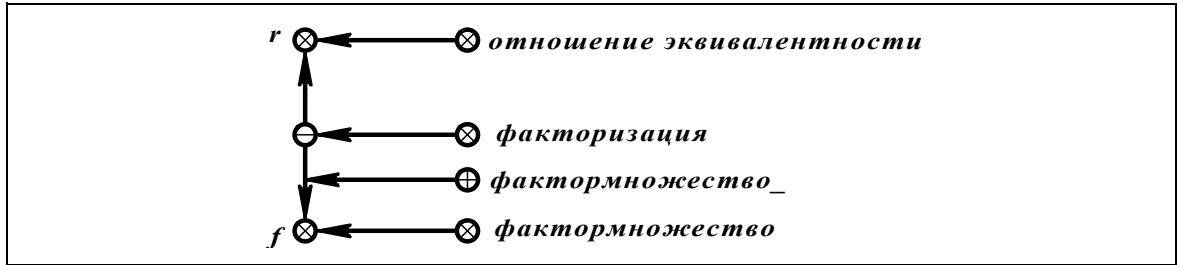
Очевидно также, что если взять проекцию отношения, полученного в результате соединения отношений *r1* и *r2*, по всем атрибутам, входящим в схему отношения *r1*, то получим само это отношение *r1*.

Метаотношения над бинарными классическими отношениями рассматривались выше. Перечислим некоторые из них:

- метаотношение “**произведение бинарных отношений**” – это метаотношение, каждый кортеж которого имеет вид (*{r1, a1}*, *{r2, a2}*, *произведение_*: *r3*), где *r1*, *r2*, *r3* – бинарные классические отношения с одинаковыми схемами *{a1, a2}*, при этом конструкция *r3* → (*a1 : x1, a2 : x2*) ; имеет место тогда и только тогда, когда существует *z* такой, что *r1* → (*a1 : x1, a2 : z*) ; *r2* → (*a1 : z, a2 : x2*) ; [464] (Таран Т.А.1998кн-ОсновДМ);
- метаотношение “**транзитивное замыкание**” [464] (Таран Т.А.1998кн-ОсновДМ);
- метаотношение “**соответствие**” и различные подмножества этого метаотношения.

Каждому отношению эквивалентности можно поставить в соответствие фактормножество, осуществляющее разбиение области определения этого отношения на классы эквивалентности. Связь между отношением эквивалентности r и соответствующим ему фактормножеством f задается с помощью бинарного ориентированного метаотношения, которому припишем идентификатор “**факторизация**”(см. scb-текст 3.3.13.9).

SCB-текст 3.3.13.9. Отношение “**факторизация**”



При этом имеют место следующие синонимичные соотношения идентификаторов:

фактормножество
= попарно непересекающиеся множества
= семейство попарно непересекающихся множеств ;

Семантически фактормножество представляет собой не что иное, как один из возможных признаков (свойств) классификации множества, являющегося областью определения соответствующего отношения эквивалентности (см. scb-текст 3.3.13.10).

SCB-текст 3.3.13.10. Классификации множества, являющегося областью определения соответствующего отношения эквивалентности



С формальной точки зрения каждое фактормножество можно трактовать как неориентированное отношение, связки которого имеют нефиксированную и часто достаточно большую мощность. Элементами каждой такой связки являются все те и только те объекты, которые в определенном смысле эквивалентны некоторому заданному объекту.

Упражнения к пункту 3.3.13.

Упражнение 3.3.13.1. Можно ли говорить о неунарной проекции неклассического отношения?

Упражнение 3.3.13.2. Можно ли говорить о функциональной зависимости для неклассического отношения?

Упражнение 3.3.13.3. Существует ли хотя бы одно отношение, которому соответствует несколько функциональных зависимостей (несколько ключей), или несколько функций, или несколько операций? Если да, то приведите примеры.

Упражнение 3.3.13.4. Чем отличаются:

- функциональная зависимость от ключевой зависимости;
- ключевая зависимость от функции;
- функция от операции?

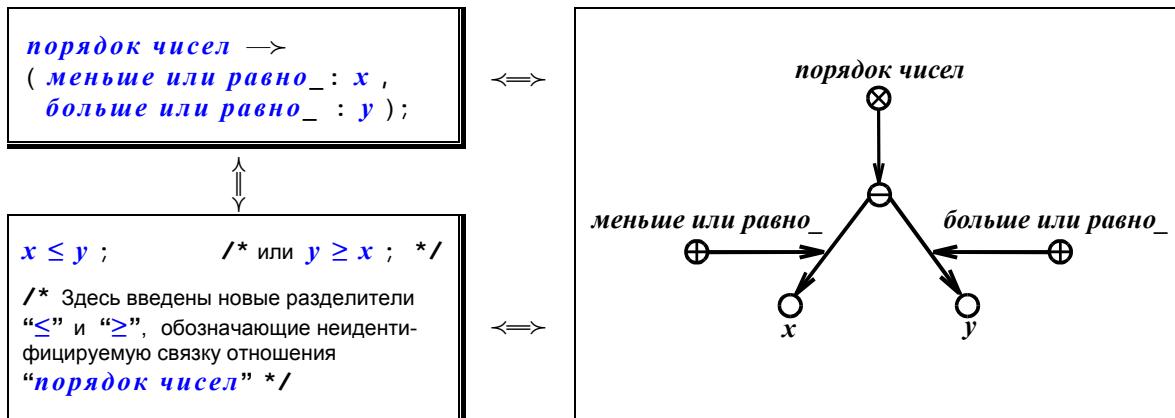
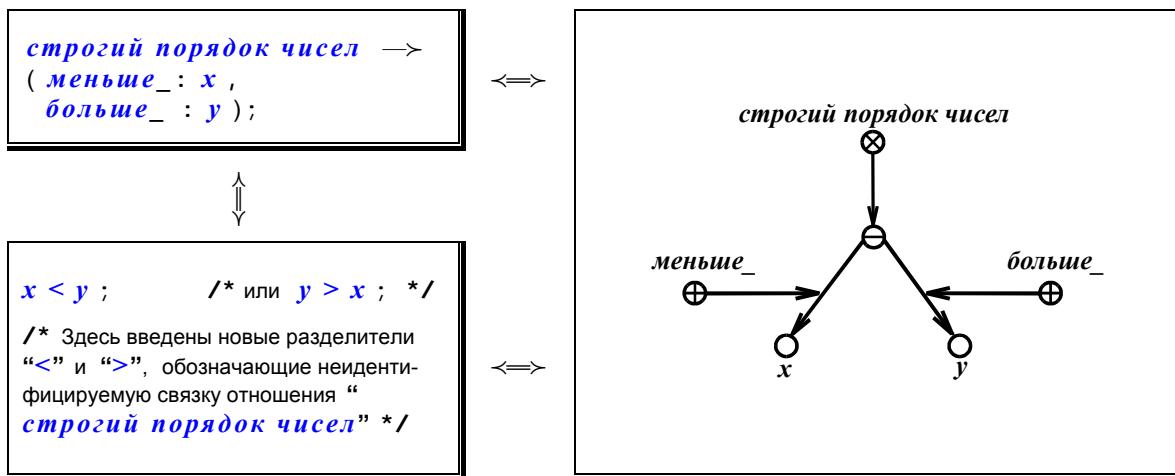
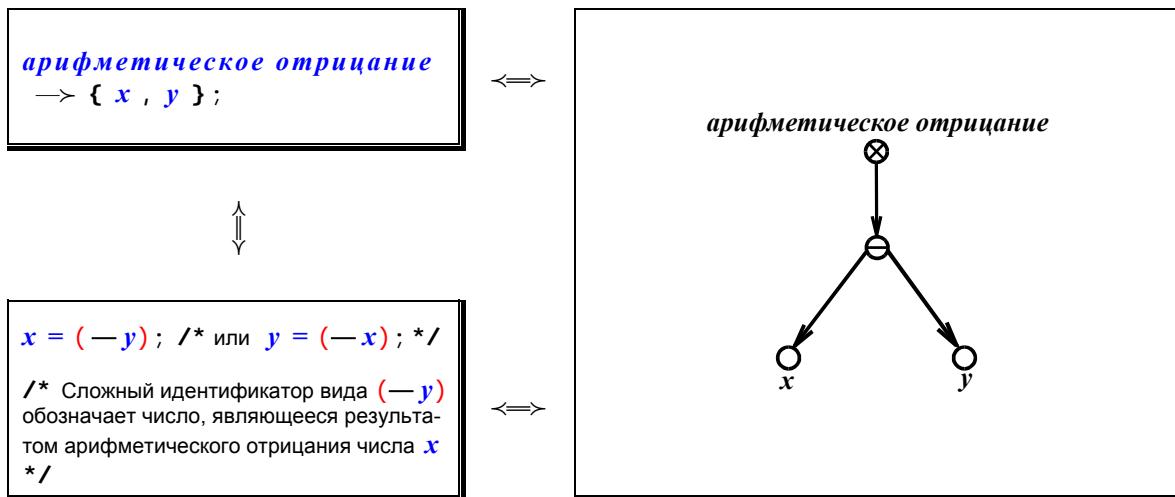
1.3.14. Числовые отношения

К числу отношений, заданных на множестве чисел, можно отнести следующие:

- “**порядок чисел**” – бинарное ориентированное отношение с атрибутом “**меньше или равно_**” и атрибутом “**больше или равно_**”;
- “**строгий порядок чисел**” – бинарное ориентированное отношение с атрибутом “**меньше_**” и атрибутом “**больше_**”;
- “**последовательность целых чисел**” – бинарное ориентированное антисимметричное отношение, использующее атрибут “**предшествующее_**” и атрибут “**следующее_**”, заданное на множестве целых чисел. Каждый кортеж этого отношения связывает два целых числа, первое из которых непосредственно предшествует второму, а второе – непосредственно следует за первым (т.е. является результатом прибавления единицы к первому числу);
- “**арифметическое отрицание**” – бинарное неориентированное отношение;
- “**числовой модуль**” – бинарное ориентированное отношение с атрибутом “**модуль_**” и с атрибутом, задаваемым по умолчанию;
- “**целая часть числа**” – бинарное ориентированное отношение с атрибутом “**целое_**” и с атрибутом, задаваемым по умолчанию;
- “**сложение**” – ориентированное отношение со связками разной мощности (но не менее 3) и с атрибутами “**сумма_**” и “**слагаемое_**” (последний атрибут может задаваться по умолчанию);
- “**умножение**” – ориентированное отношение со связками разной мощности (но не менее 3) и с атрибутами “**произведение_**” и “**сомножитель_**” (последний атрибут может задаваться по умолчанию);
- “**степень**” – классическое тернарное отношение с атрибутами “**корень_**” (основание степени), “**логарифм_**” (показатель степени), “**степень_**” (результат возведения в степень);
- “**факториал**” – классическое бинарное отношение с атрибутами “**факториал_**” и “**основание_**”;
- “**sin**” – классическое бинарное отношение с атрибутами “**sin_**” и “**arcsin_**”;
- “**cos**” – классическое бинарное отношение с атрибутами “**cos_**” и “**arccos_**”;
- “**tg**” – классическое бинарное отношение с атрибутами “**tg_**” и “**arctg_**”;
- “**ctg**” – классическое бинарное отношение с атрибутами “**ctg_**” и “**arcctg_**”.

Примечание. Следует четко отличать числовые отношения и действия, направленные на вычисление неизвестных чисел, связанных указанными отношениями между собой и с известными числами. В частности, элементарным таким действием является вычисление неизвестного числа, для которого существует кортеж одного из указанных отношений, связывающий вычисляемое число только с известными числами. При этом одному отношению может соответствовать несколько таких действий. Например, отношению “**сложение**” соответствуют действие сложения и действие вычитания, отношению “**степень**” – действие вычисления корня, действие вычисления логарифма, действие возведения в степень.

Рассмотрим возможные варианты изображения кортежей для некоторых из перечисленных отношений, вводя для их записи в языке SCBs новые разделители (см. scb-тексты 3.3.14.1 – 3.3.14.8) и новые виды сложных scb-идентификаторов.

SCB-текст 3.3.14.1. Варианты изображения отношения нестрогого порядка чисел**SCB-текст 3.3.14.2.** Варианты изображения отношения строгого порядка чисел**SCB-текст 3.3.14.3.** Варианты изображения отношения арифметического отрицания

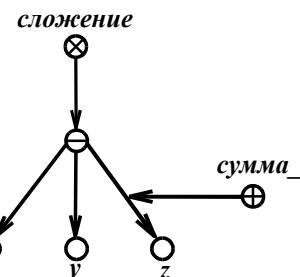
SCB-текст 3.3.14.4. Варианты изображения отношения арифметического сложения и вычитания

сложение \rightarrow
 $(x, y, \text{сумма_} : z);$

\iff

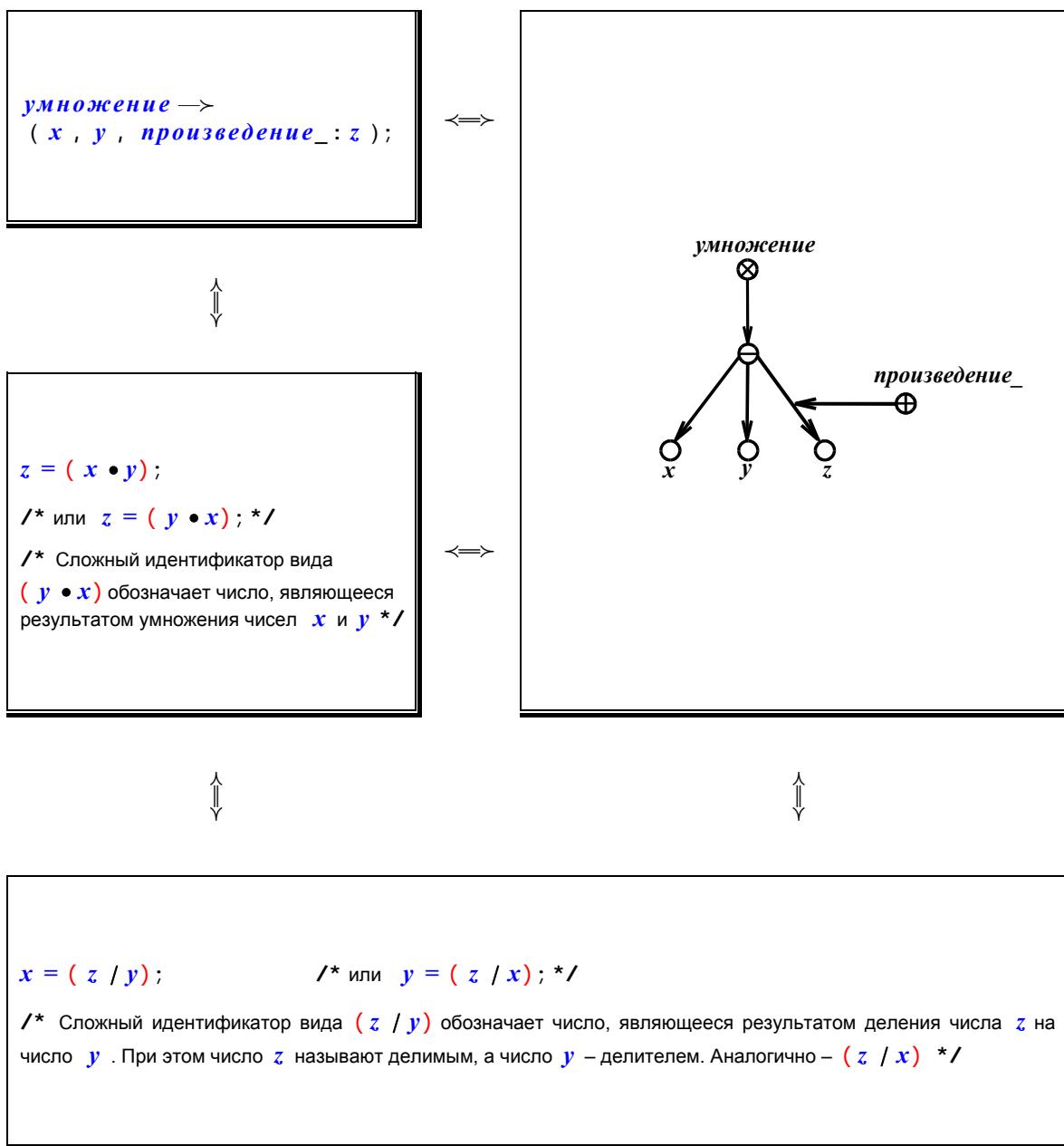
$z = (x + y);$
/* или $z = (y + x); */$
/* Сложный идентификатор вида
 $(y + x)$ обозначает число, являющее-
ся результатом сложения чисел x и y
*/

\iff



$x = (z - y);$ /* или $y = (z - x); */$
/* Сложный идентификатор вида $(z - y)$ обозначает число, являющееся результатом вычитания числа y из
числа z . При этом число y называют вычитаемым, а число z – уменьшаемым. Аналогично – $(z - x)$ */

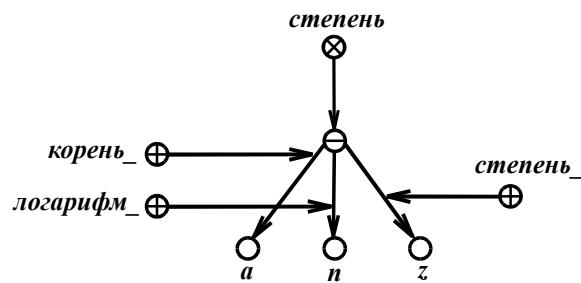
SCB-текст 3.3.14.5. Варианты изображения отношения арифметического умножения и деления



SCB-текст 3.3.14.6. Варианты изображения отношения “*степень*”

степень →
 (*корень_* : *a* ,
логарифм_ : *n* ,
степень_ : *z*) ;

↔↔

*z* = (*a* ↑ *n*) ;

/* Сложный идентификатор вида (*a* ↑ *n*) обозначает число, являющееся результатом возведения числа *a* в *n*-ую степень. При этом число *a* называют основанием степени, число *n* – показателем степени, а число *z* – степенью. В математической литературе рассматриваемый сложный идентификатор имеет вид *aⁿ*. */

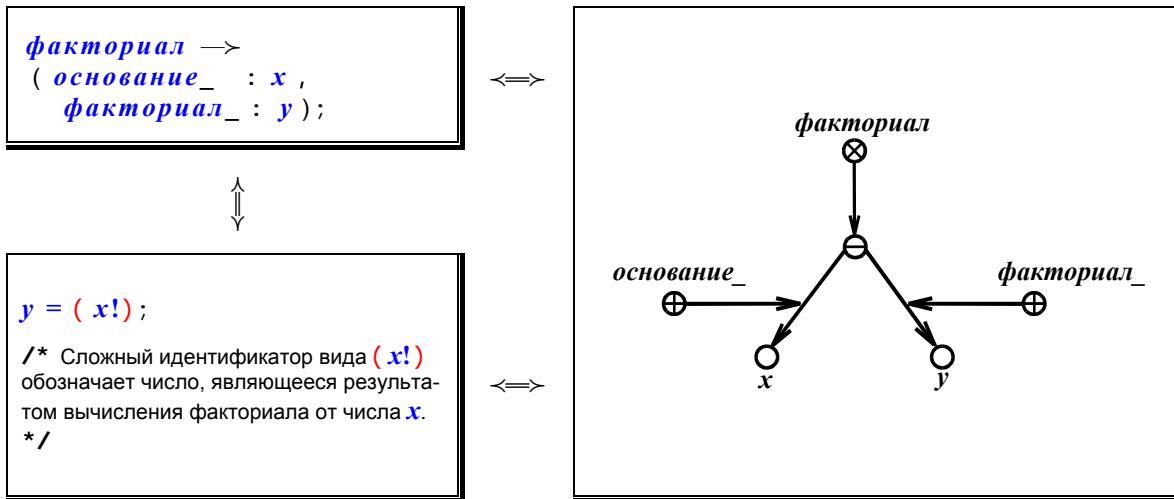
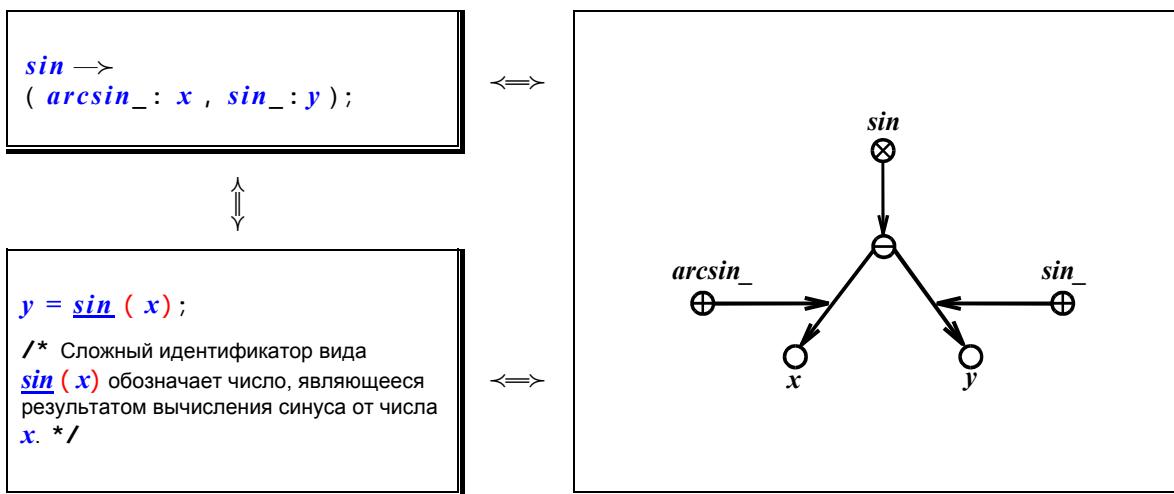
↔↔

a = (*n* √ *z*) ;

/* Сложный идентификатор вида (*a* √ *n*) обозначает число, являющееся результатом извлечения корня *n*-ой степени из числа *z*. При этом число *z* называют подкоренным числом, число *n* – показателем корня, а число *a* – корнем. В математической литературе рассматриваемый сложный идентификатор имеет вид *n√z*. */

n = log (*a* , *z*) ;

/* Сложный идентификатор вида log (*a* , *z*) обозначает число, являющееся результатом взятия логарифма числа *z* по основанию *a*. При этом число *a* называют основанием логарифма, число *z* – логарифмируемым числом, а число *n* – логарифмом. В математической литературе рассматриваемый сложный идентификатор имеет вид *log_az*. */

С С В -т е к с т 3 . 3 . 1 4 . 7 . Варианты изображения отношения “*факториал*”**С С В -т е к с т 3 . 3 . 1 4 . 8 .** Варианты изображения отношения “*sin*”

Примечание. Ключевые слова, используемые при построении сложных идентификаторов и совпадающих с идентификаторами соответствующих отношений, всегда подчёркиваются, чтобы их отличить от идентификаторов этих отношений. Следовательно, например, ключевое слово “*sin*” в сложном идентификаторе вида “*sin* (*x*)” и простой идентификатор “*sin*”, обозначающий соответствующее отношение, имеют разную семантику.

Формально вычисление неизвестного числа заключается в построении представления этого числа в той или иной системе счисления. Так, например, представление числа в десятичной системе счисления – это рассмотрение его как суммы чисел, каждое из которых является произведением десятичной цифры (т.е. натурального числа в диапазоне от *0* до *9*) на число, являющееся результатом возведения числа *10* в целую степень.

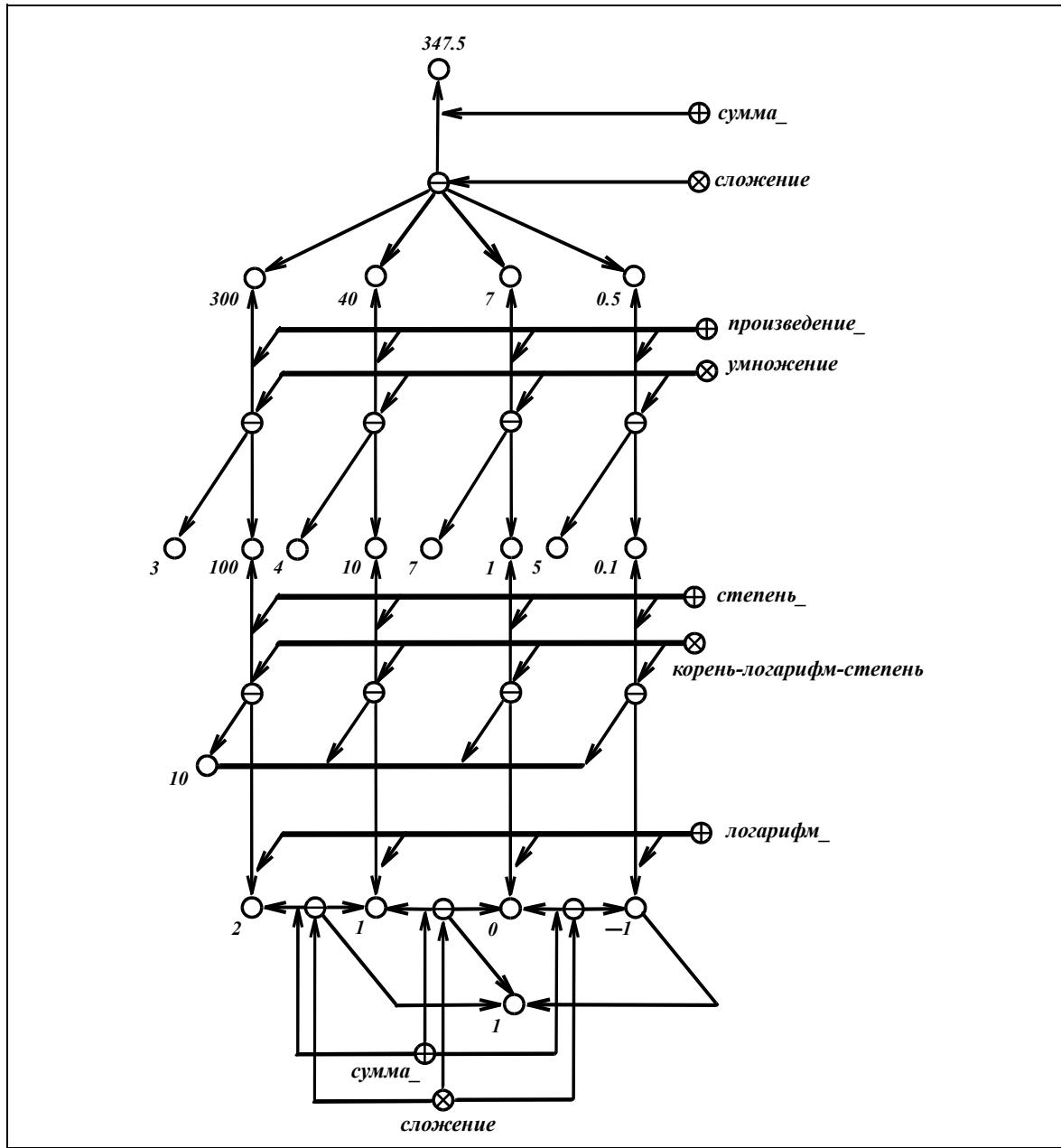
В качестве примера приведем представление на языке SCB числа *347.5* в десятичной системе счисления (см. scbg-текст 3.3.14.1). Напомним, что представление числа *x* в позиционной (в частности, в десятичной) системе счисления есть его разложение на слагаемые следующего вида

$$x = \sum_i x_i \times a^i, \text{ где}$$

- *i* – целое число,
- *a* – натуральное число, являющееся основанием (базисом) системы счисления (для десятичной системы – это число *10*),

- xi – натуральное число, находящееся в интервале $0 \leq xi \leq (a - 1)$.

SCB-текст 3.3.14.1. Представление числа 347.5 в десятичной системе счисления



Нетрудно заметить, что с позиции языка SCB суть представления чисел в той или иной системе счисления заключается в построении взаимно однозначного соответствия между представляемыми числами и такими scb-текстами (конструкциями), в которых используется конечное количество заранее известных (одних и тех же, зафиксированных) для каждой системы счисления scb-узлов. Указанные узлы будем называть ключевыми узлами соответствующей системы счисления. Для десятичной системы счисления такими ключевыми узлами являются:

- знаки натуральных чисел в интервале от 0 до 9 включительно (десятичные цифры);
- знак числа 10 (основание десятичной системы счисления);
- знаки отношений “**сложение**”, “**умножение**”, “**корень-логарифм-степень**”;
- знаки атрибутов “**сумма_**”, “**произведение_**”, “**корень_**”, “**логарифм_**”, “**степень_**”.

Итак, теоретико-множественная трактовка (на основе понятий языка SCB) десятичного представления заданного числа есть построение scb-конструкции указанного выше вида (конструкции, которая однозначно задает представляемое число) и привязка этой конструкции к перечисленным выше ключевым узлам. Привязка сводится к выявлению в построенной конструкции узлов, синонимичных ключевым узлам, и к последующему склеиванию таких синонимичных узлов.

Принципиальными здесь являются следующие обстоятельства:

- указанная привязка есть не что иное, как процедура ввода информации, являющейся десятичным представлением некоторого числа и записанной на языке SCB. Имеется в виду ввод информации в память системы, где хранящаяся там информация также представлена на языке SCB. То есть ввести информацию в такую память – это построить некоторую scb-конструкцию (scb-текст) и склеить некоторые ее scb-узлы с узлами, уже хранящимися в памяти;
- ключевые узлы, к которым осуществляется привязка вводимой информации заданного вида, должны быть заранее известны (!) и соответственно количество их должно быть конечным;
- после привязки вводимого scb-текста к ключевым узлам некоторые неключевые узлы вводимого текста могут оказаться синонимичными тем узлам, которые уже присутствуют в текущем состоянии памяти, и, следовательно, должны подлежать склеиванию с последними. Но принципиальным здесь является то, что такого рода синонимия может быть выявлена формальными средствами без участия субъекта, который инициирует ввод информации.

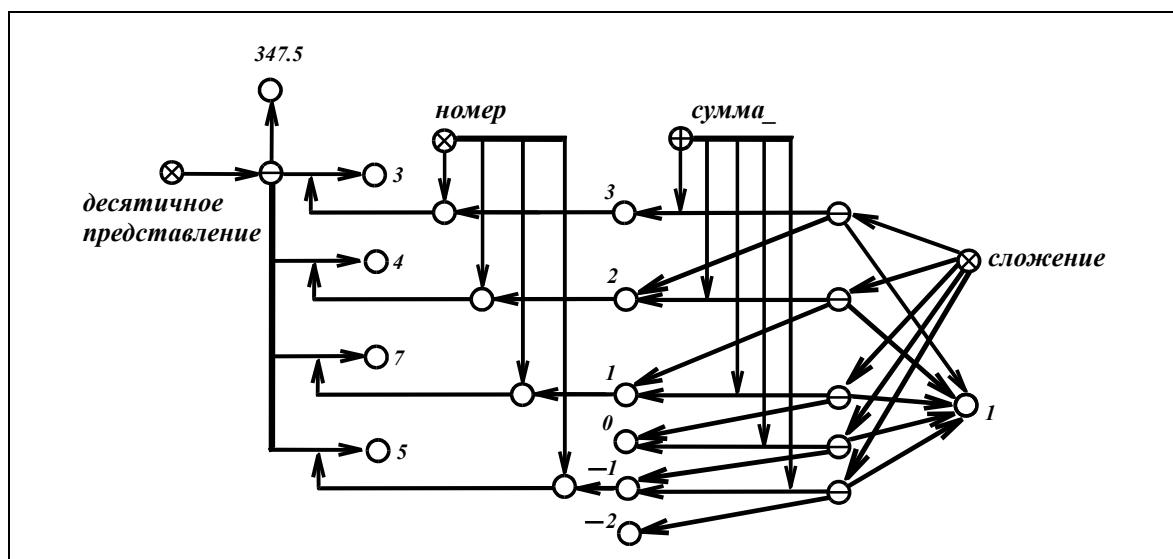
Примечание. В рамках отношения “**сложение**” можно выделить подмножество “**сложение десятичное**”, в кортежах которого все слагаемые имеют вид $(xi \bullet (10^i))$, где

- xi – натуральное число, находящееся в интервале $0 \leq xi \leq 9$;
- i – целое число.

Очевидно, что десятичное представление числа в языке SCB можно также трактовать как кортеж специального отношения “**десятичное представление**”, заданного на множестве десятичных цифр (т.е. натуральных чисел от 0 до 9) и использующего атрибуты, задающие номер соответствующего разряда в десятичном представлении.

В качестве примера на scbg-тексте 3.3.14.2 приведено представление числа **347.5** с помощью указанного специального отношения.

SCBg-текст 3.3.14.2. Представление числа **347.5** в десятичной системе счисления с помощью отношения “**десятичное представление**”

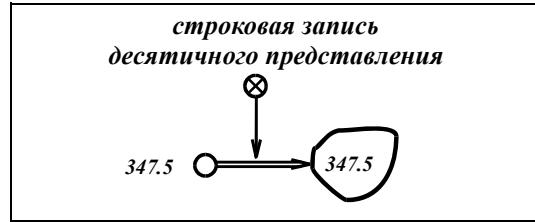


Примечание 1. Понятие порядкового номера расширяется на область всех целых чисел, т.к. порядковым номером может быть не только натуральное число, но и целое отрицательное число. При этом порядковый номер для отношения “**десятичное представление**” суть не что иное, как номер соответствующего десятичного разряда.

Примечание 2. Представляемое число здесь указывается с помощью атрибута, задаваемого по умолчанию.

Учитывая то, что десятичное представление чисел легко и однозначно изображается в виде цепочки символов (цепочки цифр), это представление можно изображать в виде содержимого scb-узла. При этом scb-узел, обозначающий число, и scb-узел, содержимое которого является строковым изображением десятичного представления этого числа, связаны отношением “*строковая запись десятичного представления*” (см. scbg-текст 3.3.14.3).

ScBg-текст 3.3.14.3. Представление числа *347.5* с использованием отношения “*строковая запись десятичного представления*”



Примечание. В данном случае строковая запись десятичного представления заданного числа (содержимое соответствующего scb-узла) совпадает с идентификатором того scb-узла, который является знаком этого числа. Но это совсем не обязательно. Идентификаторы чисел (числовых констант) могут формироваться достаточно произвольным образом, в частности, в тех случаях, когда десятичное представление числа в текущий момент не известно.

Числовые отношения можно условно разбить на следующие классы:

- отношения, областью определения которых является либо множество всевозможных чисел, либо множество чисел определенного класса (натуральных, целых, рациональных, действительных, комплексных);
- отношения, в область определения которых входят не только числа, но и множества чисел;
- отношения, заданные на множестве числовых отношений того или иного вида (такие отношения будем называть числовыми метаотношениями).

К числовым метаотношениям, в частности, относится метаотношение, связывающее две числовые функции, одна из которых является результатом дифференцирования другой.

Резюме к подразделу 3.3

В заключение рассмотрения понятия отношения и его трактовки в терминах языка SCB заметим, что рассмотренная выше типология отношений может быть расширена и модифицирована в соответствии с потребностями конкретного разработчика и конкретной предметной области, для которой формируются соответствующие отношения. В данном подразделе были рассмотрены лишь основные типы отношений, используемых практически в любой предметной области. При этом следует также заметить, что любое отношение на языке SCB может быть представлено различными способами. Расширение набора отношения производится путем добавления к набору существующих ключевых scb-узлов новых scb-узлов, обозначающих соответствующие отношения предметной области.

1.4. Представление реляционных структур в языке SCB. Типология реляционных структур. Классические и неклассические реляционные структуры

Ключевые понятия: реляционная структура, графовая структура, изоморфизм, гомоморфизм.

В подразделе 1.2 было рассмотрено понятие реляционной структуры. Данный подраздел посвящён уточнению этого понятия и рассмотрению способов представления реляционных структур на языке SCB.

1.4.1. Представление реляционных структур в языке SCB

Ключевые понятия: реляционная структура, сигнатурное отношение, сигнатурное множество, сигнатурный атрибут, первичный элемент реляционной структуры, вторичный элемент реляционной структуры, метаотношение, функция реляционной структуры, операция реляционной структуры.

Во 2-м разделе были рассмотрены такие математические структуры, как **множества, кортежи, отношения**. При этом нас интересовал:

- переход от классических (канторовских) множеств к неклассическим множествам (мультимножествам) и введение обобщенного понятия множества, включающего в себя как классические, так и неклассические множества;
- переход от классических кортежей к неклассическим и введение обобщенного понятия кортежа, включающего в себя как классические его варианты, так и неклассические;
- переход от классических отношений к неклассическим и введение обобщенного понятия отношения, включающего в себя как классические, так и неклассические его варианты.

В данном подразделе рассмотрим обобщение таких математических понятий, как алгебраическая система, алгебраическая модель, алгебра. Объединение перечисленных классов математических структур назовем **классической реляционной структурой**.

Каждая классическая реляционная структура (конструкция) включает в себя:

- некоторое семейство классических отношений, которое будем называть **сигнатурными отношениями** реляционной структуры (при этом некоторым отношениям из указанного семейства могут быть поставлены в соответствие функции или алгебраические операции с помощью соответствующих метаотношений);
- некоторое семейство специально выделенных множеств, которые будем называть **сигнатурными множествами** реляционной структуры (заметим, что сигнатурные множества в реляционной структуре могут отсутствовать);
- некоторое множество, которое будем называть **носителем реляционной структуры** и которое представляет собой нестрогое надмножество объединения всех указанных выше сигнатурных множеств реляционной структуры, а также областей определения всех указанных выше сигнатурных отношений, входящих в состав классической реляционной структуры.

Переход от классических реляционных структур к неклассическим и соответствующее этому введение обобщенного понятия реляционной структуры осуществляются по следующим направлениям:

- использование отношений, в область определения каждого из которых входят знаки связок этого же отношения и/или знаки связок других отношений этой же реляционной структуры, и/или знаки самих этих отношений (т.е. речь идет о связках, описывающих связи между другими связками, а также между отношениями);
- использование отношений, в область определения которых входят знаки реляционных структур (в частности, и знак той реляционной структуры, в состав которой эти отношения входят);
- использование в составе реляционной структуры не только классических, но и неклассических отношений;
- ослабление требования о том, чтобы сигнатурное множество реляционной структуры было подмножеством множества-носителя реляционной структуры. Для неклассической реляционной структуры на базе ее множества-носителя строится универсум, включающий в себя (1) все элементы

этого носителя, которые будем называть первичными элементами универсума реляционной структуры, и (2) вторичные элементы универсума этой реляционной структуры, которые представляют собой знаки всевозможных неориентированных множеств и кортежей, составленных из первичных и/или вторичных элементов реляционной структуры. После этого считается, что элементами реляционной структуры являются не все элементы ее сигнатурных множеств, а только те, которые оказались элементами универсума реляционной структуры – чаще всего первичными элементами;

- ослабление требования о том, чтобы область определения отношения реляционной структуры была подмножеством множества-носителя реляционной структуры (в число элементов неклассической реляционной структуры включаются не все связи ее неунарных отношений, а только те, которые оказались вторичными элементами универсума реляционной структуры).

Итак, в состав реляционной структуры общего вида входят:

- элементы множества-носителя (множества первичных элементов) реляционной структуры;
- знаки сигнатурных множеств реляционной структуры;
- знаки сигнатурных отношений реляционной структуры;
- знаки атрибутов, входящих в схемы сигнатурных отношений реляционной структуры (такие атрибуты будем называть сигнатурными атрибутами);
- знаки атрибутов, используемых для построения таких вторичных элементов реляционной структуры, которые являются элементами унарных отношений этой реляционной структуры, в том числе те элементы сигнатурных множеств реляционной структуры, которые являются элементами универсума этой структуры;
- те знаки связок сигнатурных отношений реляционной структуры, которые являются вторичными элементами универсума этой реляционной структуры;
- промежуточные вторичные элементы реляционной структуры – это элементы, которые не входят в число вышеперечисленных элементов реляционной структуры и являются элементами множеств, обозначаемых вышеуказанными вторичными элементами реляционной структуры (т.е. теми вторичными элементами универсума реляционной структуры, которые являются элементами сигнатурных множеств и отношений этой реляционной структуры). При этом, если промежуточный вторичный элемент реляционной структуры одним из своих элементов имеет вторичный элемент универсума этой реляционной структуры, не попавший в вышеназванные классы элементов реляционной структуры, то этот вторичный элемент универсума также приписывается к числу промежуточных вторичных элементов реляционной структуры;
- знаки атрибутов, используемых для построения промежуточных вторичных элементов реляционной структуры. Эти атрибуты также включаются в число сигнатурных атрибутов реляционной структуры.

Таким образом:

- все первичные элементы универсума реляционной структуры становятся первичными элементами самой реляционной структуры;
- не все вторичные элементы универсума реляционной структуры становятся вторичными элементами самой этой реляционной структуры (т.е. указанные понятия следует четко отличать);
- не все элементы сигнатурных множеств и отношений реляционной структуры становятся элементами этой реляционной структуры;
- не все элементы сигнатурных атрибутов реляционной структуры входят в состав этой реляционной структуры.

Каждая реляционная структура рассматривается как формальная (математическая) модель некоторой предметной области, а используемые в реляционной структуре знаки сигнатурных атрибутов – как "относительные" понятия соответствующей предметной области. При этом сигнатурные множества и отношения реляционной структуры есть не что иное, как "абсолютные" понятия указанной предметной области. При этом будем отличать сигнатурные элементы, соответствующие неопределяемым (исходным, базовым) понятиям, а также сигнатурные элементы, которые соответствуют определенным понятиям. Заметим при этом, что для одной реляционной структуры может существовать несколько вариантов разбиения используемых понятий на неопределяемые (основные, базовые) и определяемые.

Итак, семейство всех реляционных структур общего вида мы разбили на два класса:

- классические реляционные структуры,
- неклассические реляционные структуры.

Заметим при этом, что среди всевозможных неклассических реляционных структур наибольший интерес для нас представляют такие структуры, у которых в область определения их отношений входят знаки отношений, знаки атрибутов, знаки систем множеств (в том числе знаки реляционных структур). Такого рода реляционные структуры будем называть иерархическими реляционными структурами или сложноструктурированными реляционными конструкциями.

Особо подчеркнем то, что для интеллектуальных систем возможность работать с неклассическими, в частности с иерархическими реляционными структурами имеет принципиальное значение, так как по давляющее число предметных областей прикладных интеллектуальных систем носит именно такой характер.

Для строгого рассмотрения понятия реляционной структуры используются следующие введенные нами ранее понятия:

- универсум (над заданным множеством и заданным семейством атрибутов – см. пункт 1.2.1);
- отношение подчинения (результат транзитивного замыкания отношения принадлежности);
- отношение смежности (результат симметризации отношения принадлежности);
- отношение связности (транзитивное замыкание отношения смежности).

Уточнение понятия реляционной структуры общего вида проведем, опираясь на введенное нами в подразделе 2.1 понятие **системы множеств**. Каждую реляционную структуру будем трактовать как систему множеств, в которой тем или иным образом распределены роли между ее элементами. Напомним, что каждая конкретная система множеств представляет собой множество, элементами которого являются знаки множеств и в том числе знаки тех пар принадлежности, которые связывают между собой знаки множеств, входящих в состав системы множеств.

Перейдем к строгому определению понятия реляционной структуры (см. также пункт 1.2.1).

Определение 3.4.1.1. Реляционная структура – это такая система множеств, для которой каждому ее элементу дополнительно приписывается некоторая его роль в рамках этой системы множеств. Для указания роли элементов реляционных структур используются следующие атрибуты:

- **первичный элемент_** (= *primaryEl*) (быть первичным элементом реляционной структуры);
- **сигнатурный атрибут_** (= *attr*) (быть знаком атрибута, используемого в кортежах, знаки которых являются вторичными элементами реляционной структуры);
- **сигнатурное отношение_** (= *rel*) (быть знаком сигнатурного отношения реляционной структуры);
- **сигнатурное множество_** (быть знаком сигнатурного множества реляционной структуры).

Вторичный элемент реляционной структуры **G** – это знак множества, все элементы которого являются элементами структуры **G** и у которого элементами структуры **G** являются также знаки всех пар принадлежности, связывающих указанный выше знак множества с элементами этого множества. Очевидно также, что вторичными элементами реляционной структуры следует считать все входящие в неё знаки пар принадлежности, у которых соединяемые этими парами scb-элементы также являются элементами реляционной структуры. Если для какой-либо пары принадлежности указанные условия не выполняются, то знак этой пары в рамках соответствующей реляционной структуры может выполнять роль только ее первичного элемента.

Первичными элементами реляционной структуры могут быть знаки любых объектов (знаки пар принадлежности, знаки узловых множеств неуточняемого типа, знаки кортежей, знаки атрибутов, знаки отношений, знаки реляционных структур).

Таким образом, трактовка системы множеств как реляционной структуры "преобразует" систему множеств из неориентированного множества в кортеж и соответственно этому само понятие реляционной структуры можно трактовать как ориентированное отношение, схема которого состоит из вышеперечисленных атрибутов. Точнее говоря, отношение "**реляционная структура**" является ещё одним примером метатошений (см. пункт 3.3.13). В каждом кортеже, принадлежащем отношению "**реляционная структура**":

- существует по крайней мере один компонент с атрибутом "**первичный элемент_**";
- могут отсутствовать компоненты с атрибутом "**сигнатурный атрибут_**";
- существует по крайней мере один компонент с атрибутом "**сигнатурное отношение_**";

-
- могут отсутствовать компоненты с атрибутом “*сигнатурное множество*_”;
 - существует по крайней мере один вторичный несигнатурный элемент, который представляет собой компонент с атрибутом, задаваемым по умолчанию, т.е. существует по крайней мере один компонент без явно указанного атрибута;
 - не существует ни одного компонента, который был бы отмечен сразу несколькими атрибутами из числа вышеперечисленных;
 - для каждого компонента *si* с атрибутом, задаваемым по умолчанию, существует компонент *ri*, имеющий либо атрибут “*сигнатурное отношение*_”, либо атрибут “*сигнатурное множество*_”, по отношению к которому компонент *si* является подчиненным, т.е. связанным выходящей из *ri* парой неявно заданного отношения подчинения, которое является транзитивным замыканием подмножества отношения принадлежности, состоящего из только тех пар принадлежности, знаки которых являются компонентами структуры с атрибутом, задаваемым по умолчанию.

Термин “*реляционная структура*” можно трактовать как знак ориентированного неклассического отношения с кортежами разной мощности и со схемой, включающей в себя атрибуты: “*первичный элемент*_”, “*сигнатурный атрибут*_”, “*сигнатурное отношение*_” и “*сигнатурное множество*_”.

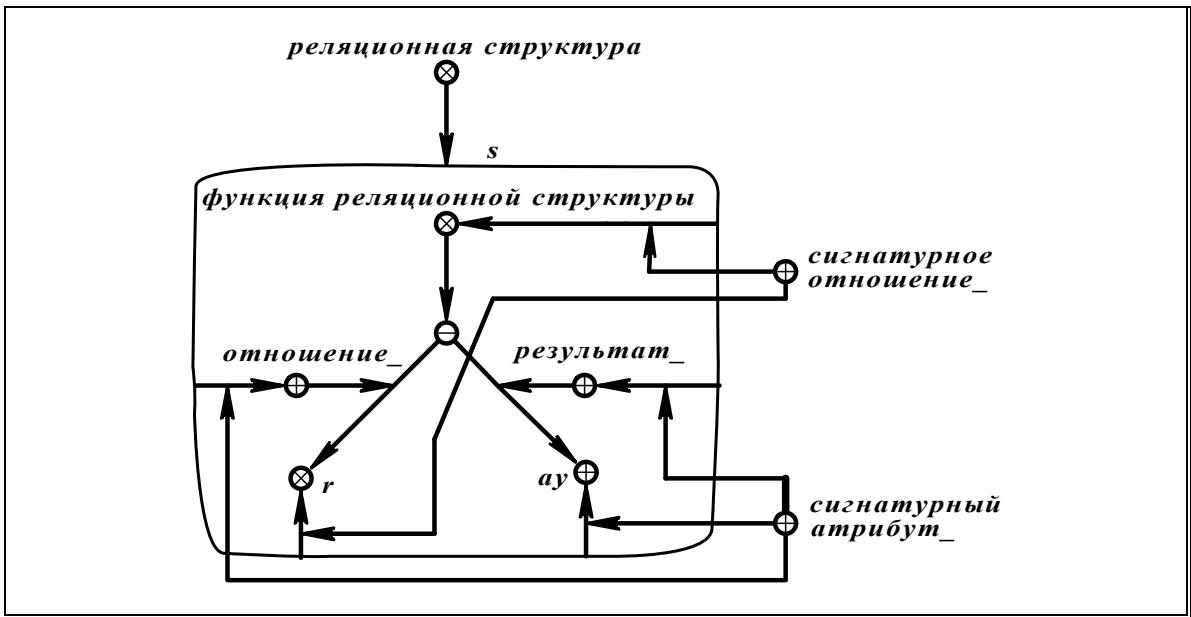
Примечание. Для уточнения роли некоторых компонентов реляционной структуры могут использоваться и другие атрибуты.

В общем случае не все элементы сигнатурного множества реляционной структуры должны быть элементами этой реляционной структуры. Следовательно, каждому сигнатурному множеству реляционной структуры ставится в соответствие его подмножество, состоящее из всех тех и только тех элементов сигнатурного множества, которые являются элементами реляционной структуры (чаще всего первичными элементами). Указанное подмножество будем называть **определяющим подмножеством сигнатурного множества** реляционной структуры.

В общем случае не все связи сигнатурного отношения реляционной структуры должны быть элементами этой реляционной структуры. Следовательно, каждому сигнатурному отношению реляционной структуры ставится в соответствие его подмножество, в состав которого входят все те и только те связи этого отношения, которые являются подмножествами этой реляционной структуры. Указанное подмножество сигнатурного отношения будем называть **определяющим подмножеством сигнатурного отношения** реляционной структуры.

Для уточнения вида **определяющих подмножеств сигнатурных отношений** реляционной структуры вводятся специальные сигнатурные метаотношения, используемые для указания роли компонентов реляционной структуры. К числу таких специальных сигнатурных отношений относятся метаотношение “*функция реляционной структуры*” и метаотношение “*алгебраическая операция реляционной структуры*”. Особо подчеркнем то, что указанные понятия ставятся в соответствие не самим сигнатурным отношениям, а их определяющим подмножествам в рамках указываемых реляционных структур. Поэтому эти понятия являются не абсолютными, а относительными для этих реляционных структур. Другими словами некоторое сигнатурное отношение некоторой реляционной структуры в целом может не быть функциональным, а его определяющее подмножество для указанной реляционной структуры может оказаться функциональным и наоборот.

Определение 3.4.1.2. Определение метаотношения “*функция реляционной структуры*”. Сигнатурное отношение *r* реляционной структуры *s* будем называть **функциональным отношением** этой реляционной структуры в том и только в том случае, если определяющему подмножеству указанного сигнатурного отношения указанной реляционной структуры соответствует некоторая функциональная зависимость, являющаяся функцией с атрибутом результата *ay* (см. пункт 3.3.5). В этом случае справедлива конструкция вида:



Примечание. Указание того, какая именно функциональная зависимость (а их может быть несколько) указываемого сигнатурного отношения нас интересует в рамках указываемой реляционной структуры, осуществляется с помощью специальных дополнительных атрибутов, вводимых в состав реляционной структуры, – атрибута “**результат_**”, атрибута “**отношение_**”. Указанные знаки атрибутов включаются в число элементов реляционной структуры (такие элементы реляционной структуры отмечаются атрибутом “**сигнатурный атрибут_**”). Эти атрибуты следует отличать от атрибутов, которыми отмечаются элементы реляционной структуры, т.е. указываются роли этих элементов в рамках реляционной структуры. К числу последних относятся: “**первичный элемент_**”, “**сигнатурное множество_**”, “**сигнатурное отношение_**”, “**сигнатурный атрибут_**”.

Определение метаотношения “**алгебраическая операция реляционной структуры**” (быть алгебраической операцией реляционной структуры) строится точно так же, как и определение метаотношения “**функция реляционной структуры**”.

Примечание. Следует отличать абсолютное понятие “**функция**” (см. пункт 3.3.5) от относительного понятия “**функция реляционной структуры**” (быть функцией данной реляционной структуры). Аналогично этому следует отличать понятие “**алгебраическая операция**” и понятие “**алгебраическая операция реляционной структуры**”.

Упражнения к пункту 3.4.1.

Упражнение 3.4.1.1. Согласно данному выше строгому определению реляционной структуры, могут ли первичные элементы реляционной структуры быть одновременно вторичными элементами реляционной структуры?

Упражнение 3.4.1.2. Может ли первый элемент реляционной структуры быть:

- знаком атрибута в кортежах, знаки которых являются вторичными элементами реляционной структуры;
- знаком сигнатурного отношения реляционной структуры;
- знаком сигнатурного множества реляционной структуры?

1.4.2. Типология реляционных структур

Ключевые понятия: алгебра, решетка, поле, кольцо, группа, алгебраическая модель, алгебраическая система.

Проведем типологию реляционных структур (см. также пункт 1.2.1). В соответствии с приведённым выше определением реляционной структуры, реляционные структуры можно разбить на два класса:

- неклассические реляционные структуры,
- классические реляционные структуры.

Особый класс реляционных структур – иерархические реляционные структуры.

Среди классических реляционных структур можно проследить следующую типологию:

- алгебраические системы,
 - алгебры,
 - алгебраическая структура с бинарными операциями (алгебраическая структура, каждая операция которой соответствует некоторому тернарному отношению)
 - алгебраическая структура с одной бинарной операцией
 - группойд
 - группойд с нейтральным элементом;
 - группойд без нейтрального элемента;
 - полугруппа
 - коммутативная полугруппа;
 - некоммутативная полугруппа;
 - полугруппа с правым сокращением;
 - полугруппа с двухсторонним сокращением;
 - группа
 - коммутативная группа (абелева группа);
 - алгебраическая структура с двумя бинарными операциями
 - кольцо
 - тело
 - поле;
 - решетка
 - дедекиндова решетка (модулярная решетка)
 - дистрибутивная решетка;
 - алгебраические модели
 - графовые структуры.

Упражнения к пункту 3.4.2.

Упражнение 3.4.2.1. Является ли алгебраическая система классической реляционной структурой?

Упражнение 3.4.2.2. Могут ли в алгебраической структуре разным отношениям быть со-поставлены алгебраические операции различной арности?

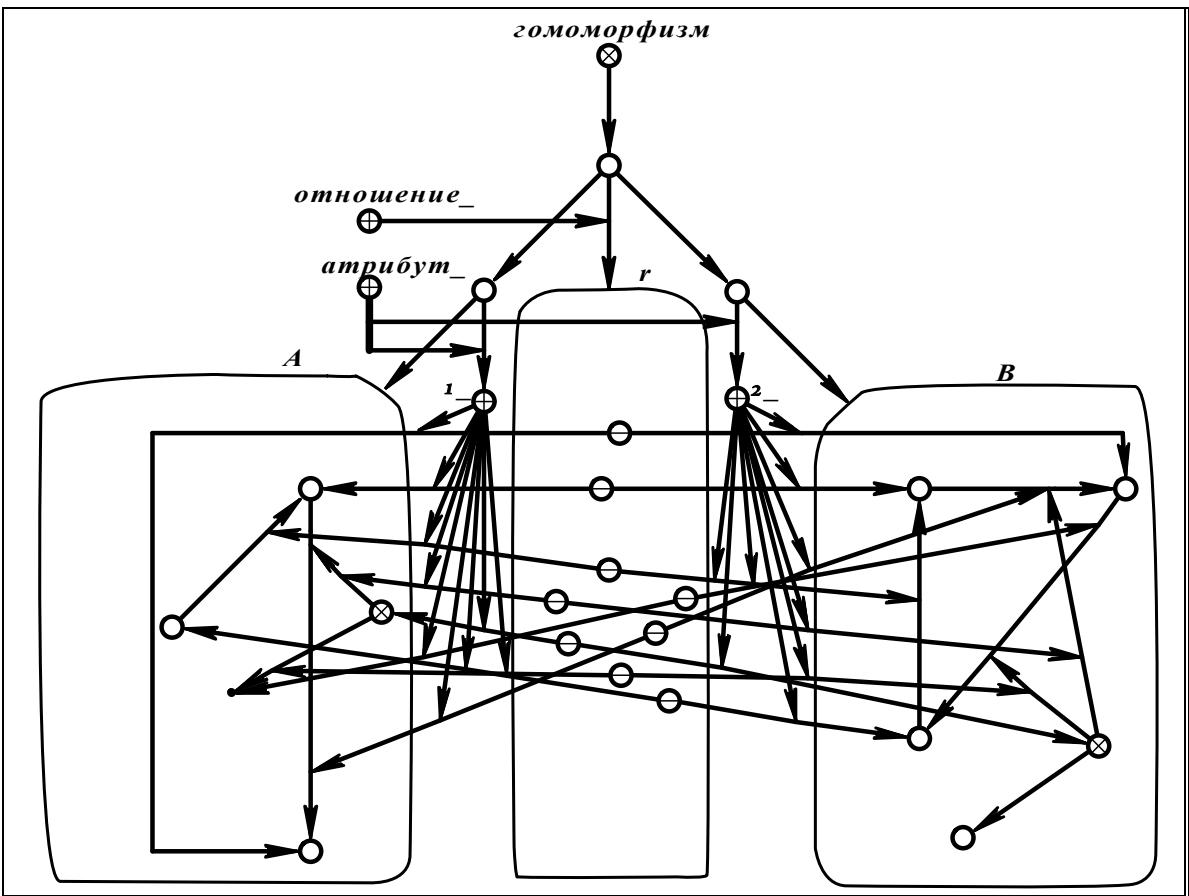
1.4.3. Отношения над реляционными структурами. Реляционные метаструктуры

Ключевые понятия и идентификаторы ключевых scb-узлов: реляционная метаструктура, *гомоморфизм*, *изоморфизм*, *автоморфизм*.

Основными отношениями над реляционными структурами являются:

- **отношение гомоморфизма,**
- **отношение изоморфизма,**
- **отношение автоморфизма (частный вид отношения изоморфизма).**

SCB-текст 3.4.3.1. Пример отношения гомоморфизма:



В приведённом примере реляционная структура *A* гомоморфна реляционной структуре *B* при соответствии гомоморфизма, заданного отношением *r*.

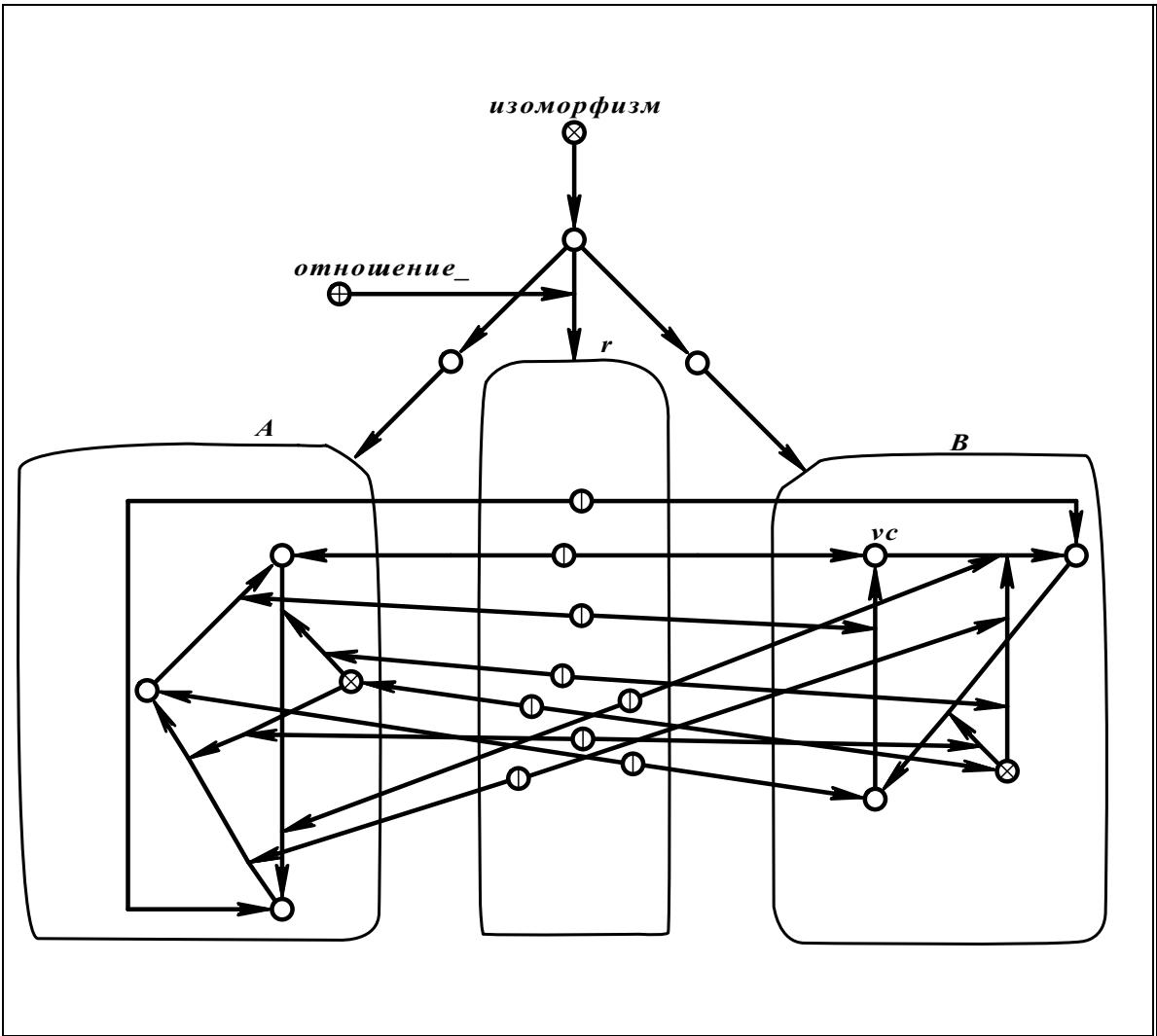
Реляционная структура *A* называется гомоморфной реляционной структуре *B*, тогда и только тогда, когда:

- 1) каждому первичному элементу реляционной структуры *A* однозначно соответствует первый элемент структуры *B*;
- 2) каждому сигнатурному множеству реляционной структуры *A* однозначно соответствует сигнатурное множество структуры *B*;
- 3) каждому сигнатурному отношению реляционной структуры *A* однозначно соответствует сигнатурное отношение структуры *B*;
- 4) каждому сигнатурному атрибуту реляционной структуры *A* однозначно соответствует сигнатурный атрибут структуры *B*;

- 5) кроме того: если элемент e структуры A включён во множество s в рамках этой структуры, т. е. существует дуга ($s \rightarrow e$), то однозначно соответствующий ему элемент e^* реляционной структуры B , должен быть включён во множество s^* , включённое в реляционную структуру B , причём s^* – элемент, однозначно соответствующий элементу s в рамках рассматриваемого отношения гомоморфизма, а также дуга ($s^* \rightarrow e^*$) включена в реляционную структуру B и также является элементом, однозначно соответствующим дуге ($s \rightarrow e$) в рамках рассматриваемого отношения гомоморфизма.

Реляционные структуры A и B называются изоморфными тогда и только тогда, когда реляционная структура A гомоморфна структуре B и реляционная структура B гомоморфна структуре A .

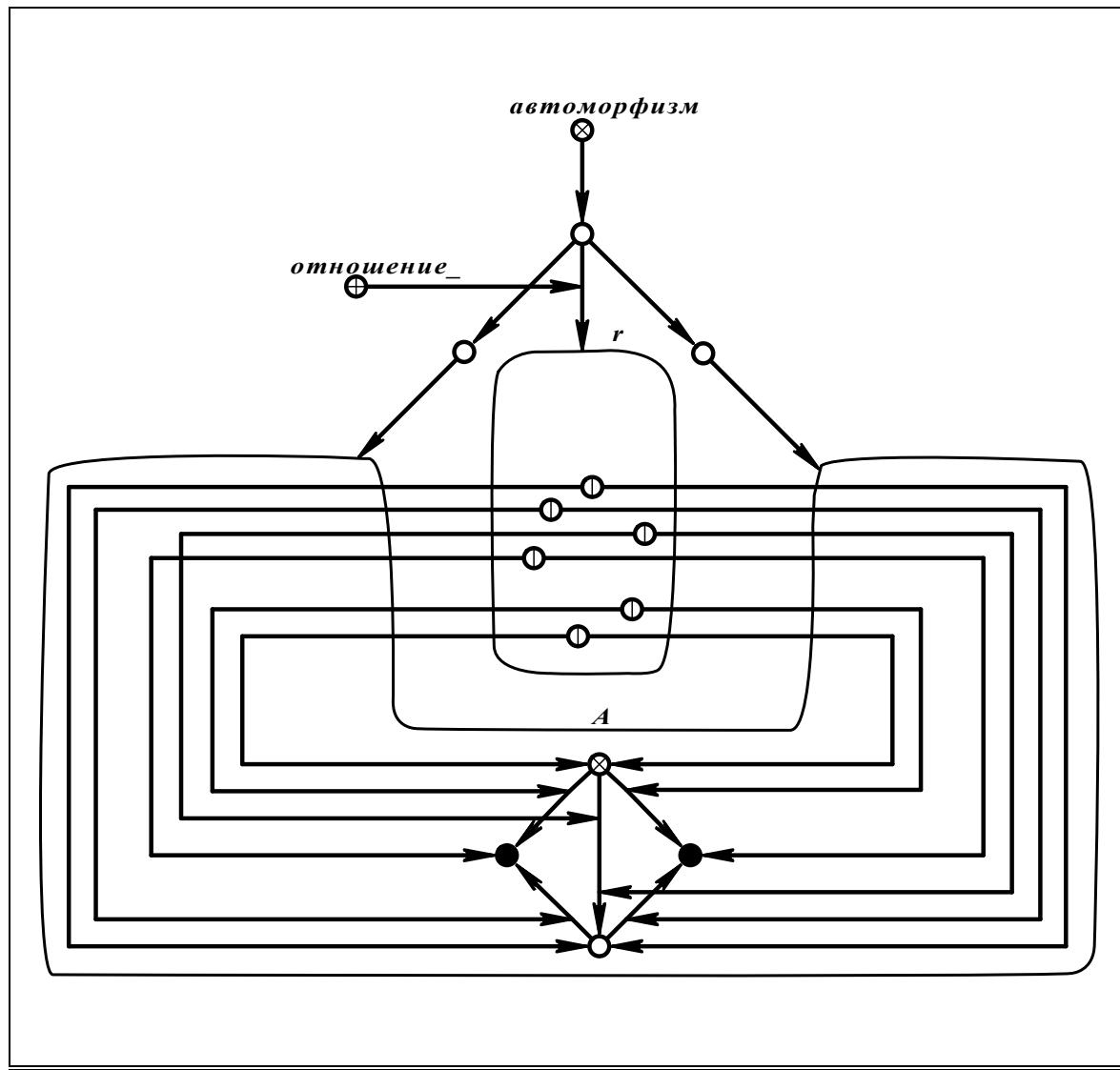
S C B g - текст 3 . 4 . 3 . 2 . Пример отношения изоморфизма



В приведённом примере реляционная структура A изоморфна реляционной структуре B при соответствии изоморфизма, заданного отношением r .

Частным случаем изоморфизма является автоморфизм, когда реляционная структура изоморфна сама себе. Выделив отношения гомоморфизма и изоморфизма, можно перейти к рассмотрению реляционных метаструктур, т.е. таких реляционных структур, в которых вышеперечисленные отношения играют роль сигнатурных, а первичными элементами являются реляционные структуры.

SCB-текст 3.4.3.3. Пример отношения автоморфизма

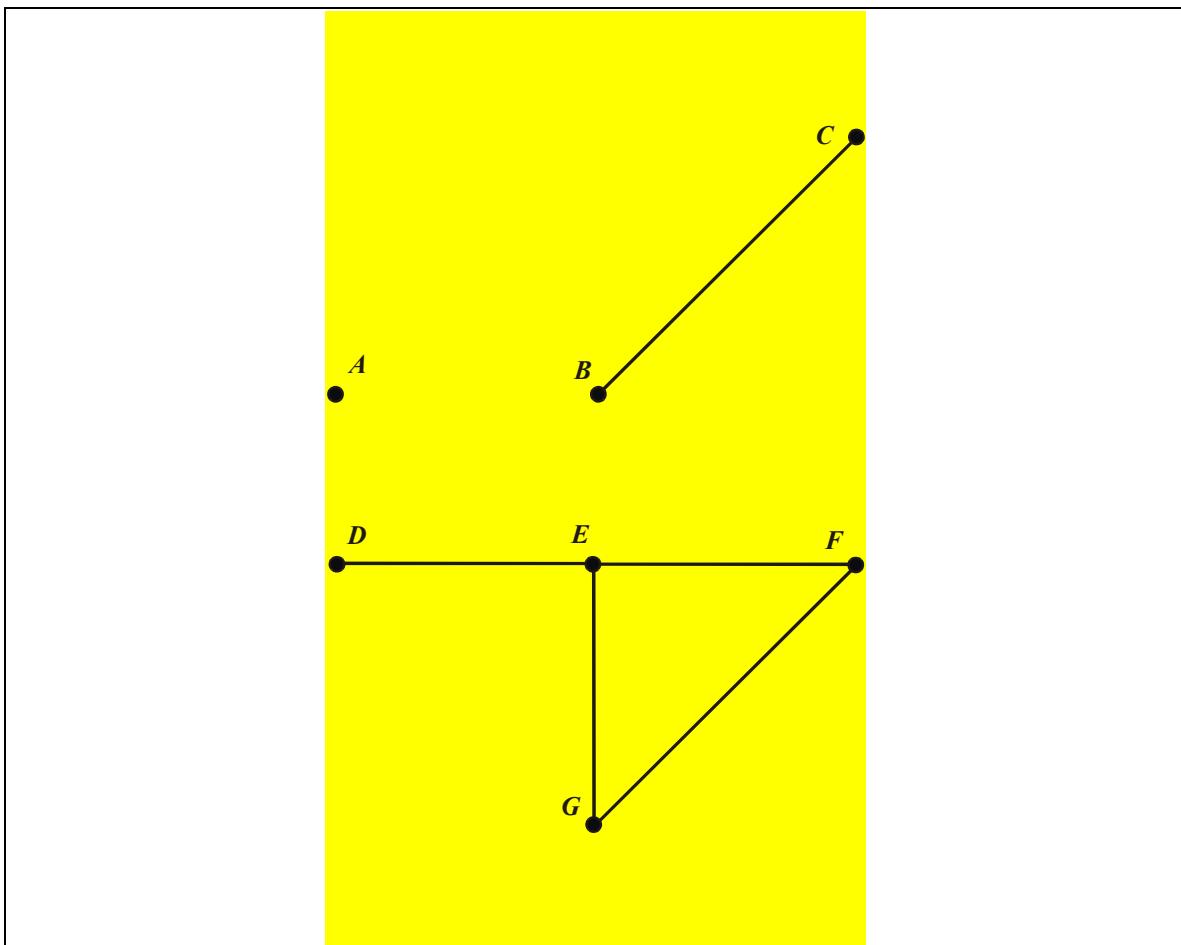


В приведённом примере реляционная структура A автоморфна при соответствии автоморфизма, заданного отношением r .

1.4.4. Графовые структуры и отношения над ними

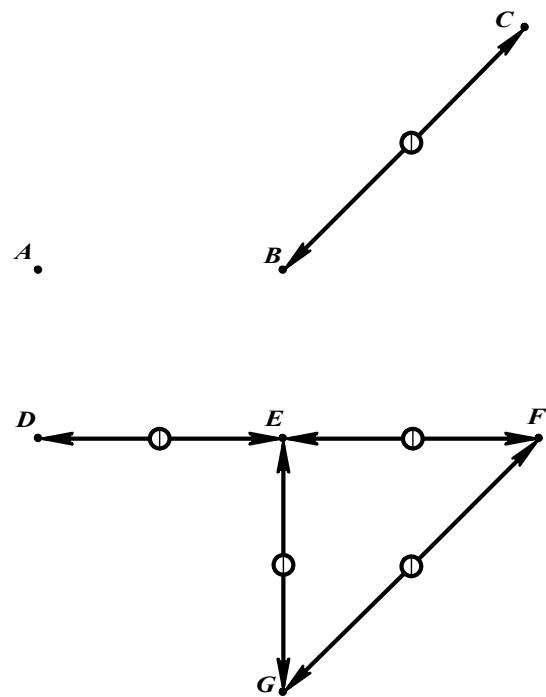
Ключевые понятия: графовая структура, связка инцидентности, ребро, вершина.

Графовая структура (граф) – реляционная структура, в которой все сигнатурные отношения являются бинарными (ориентированными либо неориентированными) отношениями смежности (см. подраздел 1.2). Связки этих отношений называют рёбрами графа. Первичные элементы графа называют вершинами. Приведём пример графовой структуры неориентированного графа в традиционном представлении:

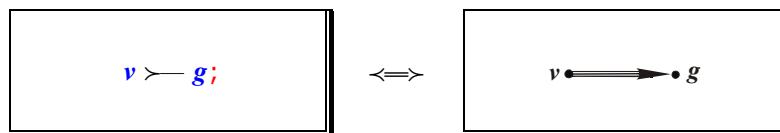


При представлении графа на языке SCB каждое неориентированное ребро графа представляется неориентированной бинарной связкой, а каждое ориентированное – ориентированной связкой.

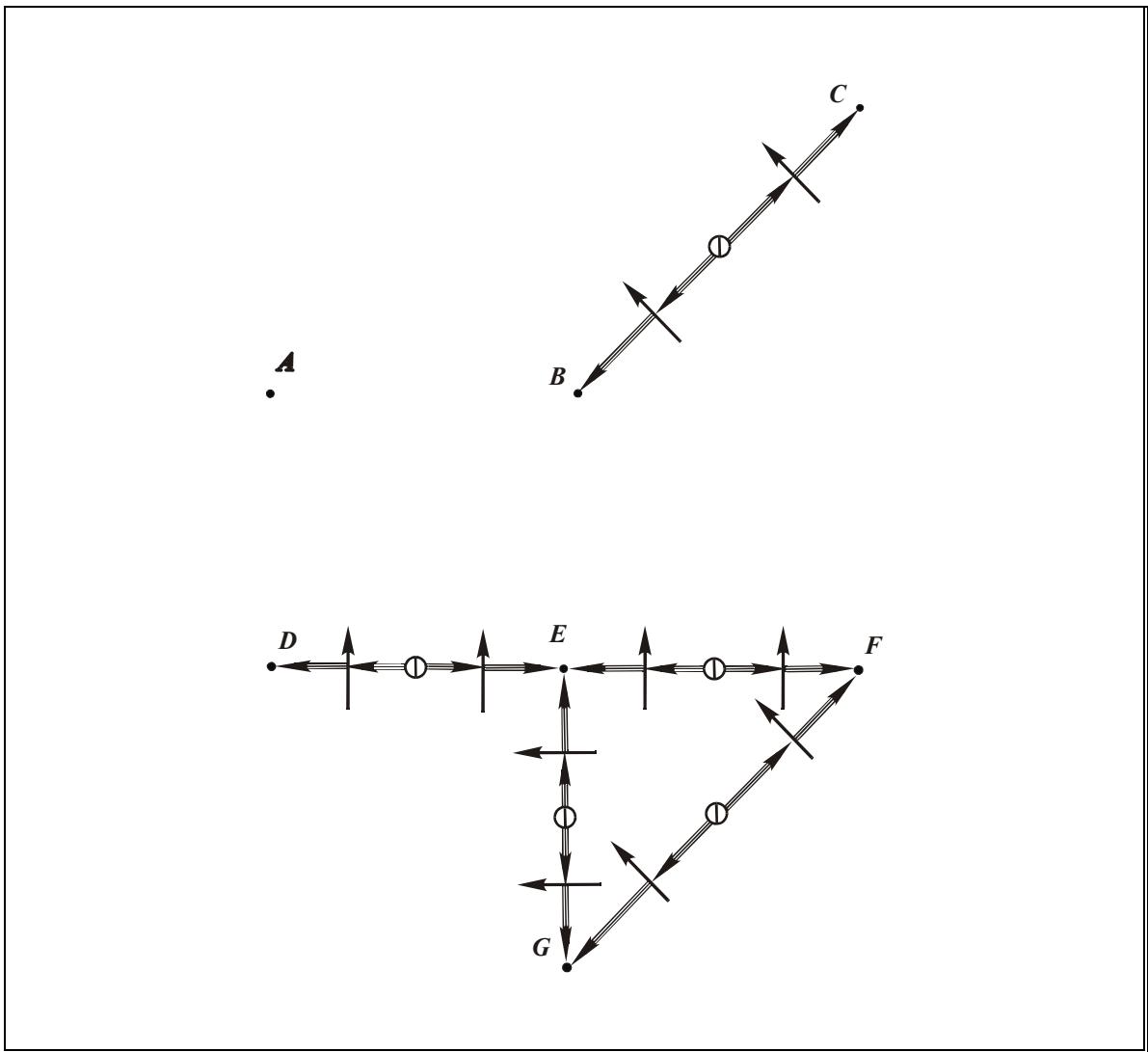
SCB-текст 3.4.4.1. Представление вышеприведённого графа на языке SCB:



Введём в графическую модификацию языка SCB изображение связки отношения инцидентности:



Используя связи отношения инцидентности, можно рассматриваемую нами графовую структуру представить и таким образом:



Над графовыми структурами, так же как и над реляционными структурами, определены отношения гомоморфизма, изоморфизма и автоморфизма.

Существует типология графовых структур (графов):

- ориентированные графы,
 - неориентированные графы,
 - смешанные графы;
-
- связные графы,
 - несвязные графы;
-
- циклические графы,
 - ациклические графы
- деревья
 - бинарные деревья,
 - тернарные деревья
- и т.п.

Выводы к разделу 3

Завершая раздел 3, подчеркнем, что на языке SCB возможно эффективное представление не только классических, но и неклассических математических структур – не только канторовских множеств, но и мульти множеств (множеств с повторениями), не только классических кортежей, но и кортежей неклассического вида с произвольным характером распределения атрибутов между компонентами кортежа, не только классических отношений, каждое из которых представляет собой множество классических кортежей одинаковой мощности и с одинаковыми атрибутами, но и отношений неклассического вида, не только классических реляционных структур (алгебраических систем), но и реляционных структур неклассического вида.

Особо отметим неограниченные возможности языка SCB для представления всевозможных метаструктур – метамножеств (множеств, элементами которых являются множества), метакортежей (кортежей, компонентами которых являются кортежи), метаотношений (отношений, в область определения которых входят отношения), реляционных метаструктур (реляционных структур, первичными и/или вторичными элементами которых являются знаки реляционных структур).

1. Ядро открытого семейства графовых языков представления знаний, построенных на теоретико-множественной основе

Разделы 2 и 3 были посвящены рассмотрению базового графового теоретико-множественного языка SCB, который является основой для разработки различных языков представления фактографических знаний различного вида. Средства указанного языка являются достаточными для представления любых предметных областей. Расширение семантической мощности языка SCB прежде всего должно быть направлено на обеспечение представления не только фактографических знаний, но и знаний, являющихся описанием логических свойств и закономерностей описываемых предметных областей. Для этой цели вводится язык более высокого уровня SC (Semantic Code), который является ядром открытого семейства графовых языков представления знаний, построенных на теоретико-множественной основе. Рассмотрению указанного языка и посвящен данный раздел.

Данный раздел может быть использован в качестве учебного пособия по дисциплинам «Математические основы искусственного интеллекта» и «Модели представления знаний, базы данных и СУБД» для студентов специальности «Искусственный интеллект».

1.1. Основные понятия, лежащие в основе логических языков

Ключевые понятия: знак множества (= константа), переменная, простая переменная, метапеременная, значение константы, значение переменной, значение метапеременной, область возможных значений переменной, область возможных значений метапеременной, переменная 1-го уровня, переменная 2-го уровня, логическая формула, связанная переменная логической формулы, свободная переменная логической формулы, пропозициональная переменная, высказывание (замкнутая логическая формула, логическая формула без свободных переменных), высказывательная форма (незамкнутая логическая формула, логическая формула со свободными переменными), истинное высказывание, ложное высказывание, формальная теория, формальная теория и субъект (отношение, связывающее формальные теории с соответствующими субъектами), предметная область формальной теории, атомарная логическая формула, атомарное высказывание (фактографический текст), фактографический язык, позитивная логическая формула, негативная логическая формула (отрицательная логическая формула), нечеткая логическая формула, неатомарная логическая формула, конъюнктивная логическая формула, дизъюнктивная логическая формула, строгая дизъюнктивная логическая формула (логическая формула исключающего ИЛИ), импликативная логическая формула, логическая формула эквивалентности, кванторная логическая формула, логическая формула о существовании, логическая формула о существовании и единственности, логическая формула о всеобщности, вхождение логической формулы (в состав другой логической формулы, но не обязательно непосредственно), высказывательная форма и релевантное высказывание.

Переменная – это принципиально отличающийся от **знака** семантически элементарный фрагмент текста. Если знак при теоретико-множественной трактовке семантики текстов обладает свойством обозначать некоторое множество (см. подраздел 2.1), то переменная обладает свойством принимать значения. Таким образом если каждому знаку ставится в соответствие множество, обозначаемое этим знаком, то каждой переменной ставится в соответствие область (множество) ее возможных значений. **Значением переменной** является какой-либо знак или какая-либо другая переменная. Образно говоря, значение переменной – это то, что можно вместо нее "подставить" (т.е. то, на что эту переменную можно заменить).

В дальнейшем нам будет удобно распространить свойство принимать значения и для знаков. При этом будем считать, что **область возможных значений** каждого знака представляет собой множество, состоящее из одного элемента, каковым является сам этот знак.

Каждой конкретной **переменной** ставится в соответствие:

- 1) множество всевозможных изображений этой переменной в различных текстах. Все эти изображения считаются синонимичными и семантически элементарными. Чаще всего синонимичные изображения переменных (т.е. различные изображения одной и той же переменной) в текстах представляются синтаксически подобными (одинаковыми) фрагментами текстов;
- 2) множество всевозможных значений этой переменной.

В зависимости от типа значений переменные разбиваются на два класса:

- **простые переменные**, значениями которых являются знаки множеств;
- **метапеременные**, значениями которых являются переменные.

Введение метапеременных в язык SC позволяет создавать на его основе логические метаязыки, позволяющие описывать свойства и закономерности самих логических формул и формальных теорий. Для этого, в частности, необходимо логические формулы и формальные теории трактовать как реляционные структуры (см. об этом в подразделе 5.4). Очевидно, что первичными элементами реляционных структур такого рода могут быть не только константные, но и переменные sc-элементы.

Следующим понятием, лежащим в основе логических языков, является понятие **логической формулы**.

Логической формулой будем называть соответствующим образом оформленный текст логического языка либо фрагмент этого текста.

Классификация **логических формул** по признаку наличия свободных переменных имеет следующий вид:

- **высказывание** (замкнутая логическая формула, логическая формула без свободных переменных, повествовательное предложение);
- **высказывательная форма** (незамкнутая логическая формула, логическая формула со свободными переменными).

Классификация **логических формул** по их внутренней структуре имеет следующий вид:

- **атомарная логическая формула** (логическая формула, не содержащая других логических формул);
- **неатомарная логическая формула** (логическая формула, в состав которой входят другие логические формулы);
 - **некванторная логическая формула**;
 - **конъюнктивная логическая формула**;
 - **дизъюнктивная логическая формула**;
 - **строгая дизъюнктивная логическая формула**;
 - **импликативная логическая формула**;
 - **логическая формула эквивалентности**;
 - **отрицательная логическая формула**;
 - **нечеткая логическая формула**;
 - **кванторная логическая формула**;
 - **логическая формула о существовании**;
 - **логическая формула о существовании и единственности**;
 - **логическая формула о всеобщности**.

На основании понятия логической формулы вводится специальный вид переменных – **пропозициональные переменные**, значениями которых являются знаки логических формул.

Логические формулы, относящиеся к классу **высказываний**, обладают истинностным свойством, т.е. являются либо **истинными высказываниями**, либо **ложными высказываниями**. Кроме того, высказывания могут классифицироваться по тому, известно ли в текущий момент значение истинностного свойства. По этому признаку высказывания делятся на **четкие высказывания** (т.е. высказывания, истинность или ложность которых установлена) и на **нечеткие высказывания**.

Специальным видом высказываний можно считать **формальные теории**, каждую из которых можно трактовать как априори истинное (с чьей-то точки зрения) конъюнктивное высказывание, т.е. как множество истинных высказываний, описывающих некоторую предметную область с точки зрения некоторого субъекта.

Особо отметим относительность истинностного свойства высказываний (зависимость от субъекта). Об истинности и ложности, четкости и нечеткости высказываний можно говорить только по отношению к конкретным формальным теориям, отражающим точки зрения различных субъектов. Так, например, одно и то же высказывание в одной формальной теории может быть истинным, а в другой формальной теории – ложным.

Логическую формулу, относящуюся к классу **высказывательных форм**, можно преобразовать в высказывание в результате замены входящих в формулу свободных простых переменных на некоторые константы и входящих в формулу свободных метапеременных на некоторые простые переменные.

1.2. Язык SC (Semantic Code), являющийся основой построения различных логических языков и языков представления знаний

Ключевые понятия: SC, sc-текст, sc-элемент, sc-узел, sc-дуга, sc-константа, sc-переменная, простая sc-переменная, sc-метапеременная, константный sc-узел, константная sc-дуга, переменный sc-узел, переменная sc-дуга.

Язык **SC** (Semantic Code) – это расширение языка SCB (Semantic Code Basic) путем включения в число текстовых элементов не только знаков множеств, но и переменных. Таким образом, элементы, входящие в состав **sc-текстов** (т.е. **sc-элементы**), делятся на следующие классы:

- **sc-константы** (константные sc-элементы; sc-элементы, являющиеся знаками множеств; sc-элементы, каждый из которых имеет одно значение, каковым является сам этот элемент; sc-элементы нулевого уровня; scb-элементы);
- **простые sc-переменные** (sc-элементы, значениями которых являются sc-константы; sc-элементы 1-го уровня; sc-переменные 1-го уровня);
- **sc-метапеременные** (sc-элементы, значениями которых являются sc-переменные; sc-элементы 2-го уровня).

В свою очередь, sc-переменные разбиваются на следующие подклассы:

- переменные, значениями которых являются знаки множеств неуточняемого типа;
- переменные, значениями которых являются знаки пар принадлежности;
- переменные, значениями которых являются знаки узловых множеств;
- метапеременные, значениями которых являются переменные, значениями которых являются знаки множеств неуточняемого типа;
- метапеременные, значениями которых являются переменные, значениями которых являются знаки пар принадлежности;
- метапеременные, значениями которых являются переменные, значениями которых являются знаки узловых множеств.

Язык SC, как и язык SCB, имеет две модификации – язык SCs (Semantic Code string) и язык SCg (Semantic Code graphical).

В языке SC scb-дугу будем называть **константной sc-дугой**.

В языке SC scb -узел будем называть **константным sc-узлом**.

Вводятся также ребра, указывающие на совпадения либо возможные совпадения значения некоторой переменной со значением другой переменной или константы.

Пояснение 4.2.1. Уточним семантику **sc-переменной**. Каждый scb-элемент (каждая sc-константа) является знаком какого-то конкретного множества (правда, об этом множестве может быть не все известно, как в случае scb-узлов неопределенного типа и scb-элементов неопределенного типа). В отличие от этого каждая sc-переменная является знаком не конкретного, а произвольного множества, принадлежащего какому-то дополнительно уточняемому семейству множеств. Знаки множеств, принадлежащих указанному семейству множеств, будем называть **значениями** соответствующей sc-переменной.

Пояснение 4.2.2. Понятие **sc-переменной** (переменного sc-элемента) следует четко отличать от понятия **неопределенного scb-элемента**, который относится к числу sc-констант и является знаком конкретного, но неизвестного в данный момент множества (в частности, унарного предметного множества). Примерами неопределенного sc-элемента являются: знак неизвестного числа, которое требуется вычислить на основании имеющейся о нем информации; знак неизвестного человека (например, преступника), личность которого требуется установить на основании некоторой имеющейся о нем информации. Каждому неопределенному sc-элементу ставится в соответствие область его возможных синонимов. Для этого вводится бинарное ориентированное отношение с

именем “**область возможных синонимов**” и с атрибутами, имеющими идентификаторы “**неопределенный знак**” и “**область возможных синонимов**”. Уточнение (вычисление) неопределенного scb-элемента часто осуществляется методом исключения и сводится к сужению области возможных его синонимов. Формальным результатом такого уточнения является склеивание неопределенного scb-элемента с одной из sc-констант, входящей в область возможных синонимов этого неопределенного sc-элемента. После чего неопределенный scb-элемент перестает быть неопределенным.

Особо подчеркнем то, что язык SC обладает такой семантической мощностью, которая позволяет создавать на его основе (в качестве подъязыков) языки представления информации (знаний) самого различного вида. Таким образом, для представления самых различный знаний вполне достаточно средств языка SC. Создание на базе языка SC каждого конкретного языка, обеспечивающего представление знаний того или иного вида, сводится к формированию некоторой системы понятий и соответствующих этим понятиям константных sc-узлов. Такие sc-узлы будем называть **ключевыми узлами** соответствующего подъязыка языка SC.

К числу таких подъязыков языка SC, в частности, относится рассматриваемый в разделе 5 язык SCL (Semantic Code Logic), специально предназначенный для записи логических формул и формальных теорий.

4.2. Язык SCg (Semantic Code graphical) – графическая модификация языка SC

Ключевые понятия: SCg , scg-текст, графический примитив, изображение sc-элемента .

Так как набор элементов языка SC по сравнению с языком SCB значительно образом расширен, то его графическая (нелинейная) модификация требует введения дополнительных графических примитивов для изображения различных типов sc-элементов. В связи с этим в языке SC приняты следующие соглашения:

- константные sc-элементы неуточняемого типа и константные sc-узлы изображаются кружочками (маленького и более крупного размера);
- переменные sc-элементы неуточняемого типа и переменные sc-узлы изображаются квадратами, ориентированными по вертикали и горизонтали;
- изображение sc-метапеременных отличается тем, что изображающий их квадрат повернут на 45 градусов;
- изображение sc-элементов неуточняемого типа от изображения sc-узлов отличается уменьшенным размером;
- изображения дуг и ребер, являющихся константами, простыми переменными и метапеременными, отличаются друг от друга тем, что константные дуги и ребра представлены сплошными линиями (того или иного вида), простые переменные – пунктирными линиями, а метапеременные – штрих-пунктирными линиями.

В соответствии с введенными соглашениями в табл. 4.2.1 приведены изображения sc-элементов для графической модификации языка SC. В число графических примитивов языка SC полностью входят графические примитивы языка SCBg, поскольку язык SCBg является подъязыком языка SCg. Алфавит графических примитивов, используемых для изображения основных типов scb-элементов в графическом языке SCBg приведен в табл. 2.3.1. Алфавит дополнительных графических примитивов языка SCBg приведен в табл. 2.4.1.

Таблица 4.2.1. Основные графические примитивы языка SC

Константы	Переменные	Мета-переменные	Пояснения
•	•	•	изображение sc-элемента неуточняемого типа
○	□	◇	изображение sc-узла неуточняемого типа
●	■	◆	обозначение предметного множества

Окончание табл. 4.3.1.

Константы	Переменные	Мета-переменные	Пояснения
\odot	\square	\diamond	обозначение узлового непредметного множества
\oplus	\blacksquare	\diamond	обозначение множества знаков пар принадлежности
\otimes	\boxtimes	\diamond	обозначение отношения
\ominus	\square	\diamond	обозначение ориентированной связки
\ominus	\square	\diamond	обозначение неориентированной связки
\odot	\diamond	\diamond	обозначение атомарной логической формулы
\Rightarrow	$= \Rightarrow$	$= \cdot \Rightarrow$	обозначение простой ориентированной пары с дополнительно уточняемой семантикой
\rightarrow	$- \rightarrow$	$- \cdot \rightarrow$	обозначение пары принадлежности
\rightarrow	$- \rightarrow$	$- \cdot \rightarrow$	обозначение пары непринадлежности
\rightarrow	$- \rightarrow$	$- \cdot \rightarrow$	обозначение пары нечеткой принадлежности
$=$	$= =$	$= \cdot =$	обозначение пары равенства значений
\leftrightarrow	$\leftarrow = \rightarrow$	$\leftarrow \cdot = \rightarrow$	обозначение неориентированной (неупорядоченной) пары с дополнительно уточняемой семантикой
			замкнутая линия, являющаяся обозначением множества sc-элементов, изображенных внутри этой линии
			шинная линия, являющаяся способом увеличения контактной зоны sc-узла
			линия, ограничивающая содержимое узла

4.3. Язык SCs (Semantic Code string) – линейная модификация языка SC

Ключевые понятия: SCs , scs-текст , scs-разделитель , scs-ограничитель .

Теперь перейдем к рассмотрению линейной (символьной) модификации языка SC – языка SCs (Semantic Code string). В табл. 4.3.1 перечислены разделители и ограничители этого языка.

В число разделителей и ограничителей языка SCs входят все разделители и ограничители языка SCBs. Кроме этого, к указанному перечню разделителей и ограничителей добавляются новые:

Таблица 4.3.1

Константные разделители	Переменные разделители	Метапеременные разделители	Пояснение				
;			разделитель scs-предложений				
,			разделитель идентификаторов sc-элементов в сложных scs-предложениях				
,			разделитель слов в простых идентификаторах sc-элементов				
<i>Begin</i>			признак начала scs-текста, в котором гарантировано отсутствие неявной омонимии				
<i>End;</i>			признак конца scs-текста, в котором гарантировано отсутствие неявной омонимии				
/*	*/		левый и правый scs-ограничители комментария (в scs-тексте)				
/*	*/		левый и правый scs-ограничители содержимого sc-узла				
→	←		scbs-связки инцидентности				
→	←	→→	←←	→→→	←←←	scs-связки принадлежности	
→→	←→	→→→	←←→	→→→→	←←←→	scs-связки непринадлежности	
→→	←→	→→→	←←→	→→→→	←←←→	scs-связки нечеткой принадлежности	
{	}	{·}	·}	{..}	..}	scs-ограничители сложного идентификатора неориентированного множества	
()	(·)	·)	(..)	..)	scs-ограничители сложного идентификатора кортежа	
[]	[·]	·]	[..]	..]	scs-ограничители сложного идентификатора системы множеств	
:		:		:	:	разделители атрибута и компонента кортежа	
:/		/:		:/:		scs-ограничители, неявно задающие дугу, выходящую из элемента, указанного внутри данных ограничителей, и входящего в sc-элемент, записанный непосредственно справа от правого ограничителя	
()					ограничители обозначения неидентифицируемой связки, ограничители аргументов функций	
⇒	⇐	⇒⇒	⇐⇐	⇒⇒⇒	⇐⇐⇐	связка, соответствующая ориентированной паре неуточняемого вида	
↔		↔↔		↔↔↔		связка, соответствующая неориентированной паре неуточняемого вида	

=	==	====	связка равенства значений
≠	≠≠	≠≠≠	связка неравенства значений

Продолжение табл. 4.4.1

Константные разделители	Переменные разделители	Метапеременные разделители	Пояснение
≈	≈≈	≈≈≈	связка нечеткого равенства значений

SCS-связки бинарных отношений над множествами:

≡	≡	≡	обозначение пары равенства множеств
≢	≢	≢	обозначение пары неравенства множеств
⊤	⊤	⊤	обозначение пары эквивалентности множеств по набору элементов
⊤	⊤	⊤	обозначение пары неэквивалентности множеств по набору элементов
□	□	□	обозначение пары пересекающихся множеств
□	□	□	обозначение пары непересекающихся множеств
⊂	⊃	⊂	связка строгого включения множеств
⊄	⊅	⊄	связка строгого невключения множеств
⊆	⊇	⊆	связка включения множеств
⊉	⊊	⊉	связка невключения множеств

SCS-связки бинарных отношений над числами:

<	>	<	>	<	>	связка строгого сравнения чисел
≤	≥	≤	≥	≤	≥	связка нестрогого сравнения чисел

имена унарных функций над числами:

—	обозначение операции арифметического отрицания
<u>abs</u>	обозначение функции взятия абсолютного значения
!	обозначение функции факториала
<u>exp</u>	обозначение показательной функции
<u>ln</u>	обозначение функции взятия натурального логарифма

<u>sin</u> <u>cos</u> <u>tg</u> <u>ctg</u>	обозначения тригонометрических функций
---	--

Окончание табл. 4.4.1

Константные разделители	Переменные разделители	Метапеременные разделители	Пояснение
SCS-связки бинарных функций над множествами:			
U	U	U	связка объединения множеств
⊕	⊕	⊕	связка соединения множеств
Π	Π	Π	связка пересечения множеств
\	\	\	связка разности множеств
△	△	△	связка симметрической разности множеств
×	×	×	связка декартова произведения
SCS-связки бинарных функций над числами:			
+	+	+	связка арифметического сложения
•	•	•	связка арифметического умножения
-	-	-	связка арифметического вычитания
/	/	/	связка арифметического деления
↑	↑	↑	связка возведения в степень
√	√	√	связка взятия корня
имена бинарных функций над числами:			
<u>log</u>	обозначение функции взятия логарифма		

4.4. Ключевые узлы графового языка SC

Ключевые понятия: **ключевой узел графового языка**, **ключевой узел языка SC**, **ключевой узел sc-подъязыка**.

Каждому линейному языку ставится в соответствие некоторое множество ключевых слов, знание смысла которых обеспечивает расшифровку произвольных текстов этого языка. К числу ключевых слов, в частности, относятся слова-разделители, слова-ограничители, слова-кванторы, слова-связки.

Аналогом ключевых слов в графовых языках являются **ключевые узлы**. В отличие от ключевого слова каждый ключевой узел имеет не более чем однократное вхождение в каждую языковую конструкцию.

Главным достоинством языка SC является то, что он представляет собой удобную основу для целого семейства языков, имеющих самое различное назначение и легко интегрируемых друг с другом. При этом построение каждого конкретного графового языка, являющегося подъязыком базового графового языка SC, в конечном счете сводится к формированию набора ключевых узлов, обозначающих основные понятия (в том числе и метапонятия), используемые в создаваемом языке. Некоторые

ключевые узлы, используемые во многих таких графовых языках, условно отнесем к ключевым узлам языка SC. Список этих ключевых узлов может быть легко расширен.

Во множестве ключевых узлов языка SC можно выделить, в частности, следующие группы узлов:

- 1) ключевые узлы, используемые для описания теоретико-множественных соотношений;
- 2) ключевые узлы, используемые для задания метаотношений, типологии отношений;
- 3) ключевые узлы, используемые для задания реляционной структуры, т.е. описания соотношения между исходной описываемой реляционной структурой и той однородной информационной конструкцией (т.е. константной sc-конструкцией с позитивными дугами), которая представляет указанную исходную реляционную структуру;
- 4) ключевые узлы, используемые для описания соотношений между реляционными структурами и для описания топологических свойств реляционных структур;
- 5) ключевые узлы, используемые для описания числовых соотношений;
- 6) ключевые узлы, используемые для описания всевозможных свойств (в том числе измеряемых свойств, таких как мощность множеств, точность неточных чисел, чёткость нечётких SC-дуг);
- 7) ключевые узлы, используемые для описания пространственных соотношений;
- 8) ключевые узлы, используемые для описания соотношений во времени (tempоральных соотношений);
- 9) ключевые узлы, используемые для описания отношений, заданных на множестве линейных информационных конструкций;
- 10) ключевые узлы, используемые для описания содержимого sc-узлов и идентификаторов sc-элементов;
- 11) ключевые узлы, используемые для записи команд просмотра и редактирования sc-конструкций, хранимых в графодинамической памяти.

Рассмотрим каждую группу ключевых узлов языка SC.

1-я группа ключевых узлов языка SC связана с понятием множества. Это понятие, как было отмечено в подразделе 2.1, является фундаментом денотационной семантики языка SC. Каждое множество задается:

- знаком (оболочкой) этого множества;
- набором элементов этого множества;
- подмножеством отношения принадлежности, связывающим знак этого множества со всеми (!) его элементами.

При этом не все элементы представляемого множества могут присутствовать в текущем состоянии перерабатываемой sc-конструкции и не все элементы представляемого множества, присутствующие в текущем состоянии sc-конструкции, могут быть явно указаны как элементы представляемого множества.

Типология множеств и соответствующие ключевые узлы рассмотрены в подразделе 3.1. Например, ключевой узел с scb-идентификатором “**множество**” обозначает универсальное нормализованное множество, т.е. множество, по отношению к которому все остальные нормализованные множества являются подмножествами.

К рассматриваемой группе ключевых узлов языка SC также относятся константные sc-узлы со следующими идентификаторами: **пара принадлежности**, **узловое непредметное множество**, **кортеж**, **включение множества**, **строгое включение множества**, **равенство множеств**, **эквивалентность множеств по совпадению элементов**, **пересекающиеся множества**, **Отношение принадлежности**, **Отношение непринадлежности**, **невключение множества**, **не быть строгим включением множества**, **неравенство множеств**, **не быть множествами с одинаковыми элементами**, **непересекающиеся множества**, **объединение множеств**, **соединение множеств**, **разбиение множеств**, **пересечение множеств**, **разность множеств**, **симметрическая разность множеств**.

Семантика этих ключевых узлов рассмотрена в подразделах 2.7 и 3.3.

2-я группа ключевых узлов языка SC связана с понятием отношения, т.е. содержит метаотношения и типологию отношений.

Ко 2-й группе ключевых узлов языка SC относятся константные sc-узлы со следующими идентификаторами: *отношение*, *схема отношения*, *отношение_*, *схема отношения_*, *область определения*, *домен*, *проекция*, *функция*, *минимальное декартово произведение*, *дополнение до декартова произведения*, *соединение отношений*, *произведение бинарных отношений*, *бинарное отношение*, *бинарное ориентированное отношение*, *классическое бинарное отношение*, *бинарное неориентированное отношение*, *бинарное отношение без функций*, *бинарное отношение с одной функцией*, *взаимно однозначное бинарное отношение*, *бинарное отношение с двумя функциями*, *рефлексивное бинарное отношение*, *иррефлексивное бинарное отношение*, *частично рефлексивное бинарное отношение*, *симметричное бинарное отношение*, *антисимметричное бинарное отношение*, *частично симметричное бинарное отношение*, *транзитивное бинарное отношение*, *антитранзитивное бинарное отношение*, *частично транзитивное бинарное отношение*, *отношение эквивалентности*, *отношение предпорядка*, *отношение частичного порядка*, *отношение линейного порядка*.

Семантика этих ключевых узлов рассмотрена в подразделе 3.3.

К 3-й группе относятся константные sc-узлы со следующими идентификаторами: *реляционная структура*, *первичный элемент_*, *сигнатурный атрибут_*, *сигнатурное отношение_*, *сигнатурное множество_*, *функция реляционной структуры*, *алгебраическая операция реляционной структуры*.

Семантика этих ключевых узлов рассмотрена в пункте 3.4.1.

Каждая конкретная реляционная структура, как уже отмечалось, в языке SC задается кортежем метаотношения *реляционная структура*. Типология реляционных структур определяется теоретико-графовыми ("топологическими") их свойствами (в частности, можно говорить о связных и несвязных реляционных структурах), а также алгебраическими свойствами (так, например, можно говорить о реляционных структурах, являющихся алгебраическими группами). Можно говорить о различных фрагментах реляционной структуры, удовлетворяющих тем или иным требованиям. Так, например, можно говорить о минимальном (в том или ином смысле) пути между заданными первичными элементами реляционной структуры. Можно говорить о различных соотношениях между реляционными конструкциями, например, о различных морфизмах (см. пункт 3.4.4.).

К 4-й группе ключевых узлов языка SC относятся константные узлы со следующими идентификаторами: *гомоморфизм*, *изоморфизм*, *автоморфизм*.

Семантика этих ключевых узлов была описана в пункте 3.4.

К 5-й группе ключевых узлов языка SC относятся константные sc-узлы со следующими идентификаторами: *порядок чисел*, *меньше или равно_*, *больше или равно_*, *строгий порядок чисел*, *меньше_*, *больше_*, *арифметическое отрицание*, *числовой модуль*, *модуль_*, *целая часть числа*, *целое_*, *сложение*, *сумма_*, *слагаемое_*, *умножение*, *произведение_*, *сомножитель_*, *sin*, *sin_*, *cos*, *cos_*, *tg*, *tg_*, *cgt*, *cgt_*, *arc_*, *степень*(=корень-логарифм-степень), *корень_*, *логарифм_*, *степень_*, *факториал*, *факториал_*, *основание_-*.

Также ключевыми узлами для десятичной системы счисления являются: знаки натулярных чисел в интервале от **0** до **9** включительно (десятичные цифры) и знак числа **10** (основание десятичной системы счисления). Десятичное представление числа в языке SCB можно также трактовать как кортеж специального отношения "**десятичное представление чисел**", заданного на множестве десятичных цифр (т.е. натулярных чисел от **0** до **9**) и использующего атрибуты, задающие номер соответствующего разряда в десятичном представлении. Учитывая то, что десятичное представление чисел легко и однозначно изображается в виде цепочки символов, это представление можно изображать в виде содержимого scb-узла. При этом scb-узел, обозначающий число, и scb-узел, содержимое которого является строковым изображением десятичного представления этого числа, связаны отношением "**строковая запись десятичного представления**".

Семантика этих ключевых узлов рассмотрена в пункте 3.3.14.

6-я группа ключевых узлов языка SC связана с понятием измеряемого **параметра** (свойства, признака, характеристики), в основе которого лежит идея описания всевозможных объектов как точек в n-мерном пространстве возможных значений различных параметров.

Каждому измеряемому параметру ставится в соответствие некоторое (чаще всего упорядоченное) множество, которое называется шкалой значений этого параметра. Значение некоторого параметра для некоторого конкретного объекта в языке SC задается проведением позитивной константной sc-дуги из sc-узла, представляющего элемент шкалы, в sc-узел, обозначающий класс всех объектов, имеющих соответствующее значение измеряемого параметра.

Существуют самые различные шкалы, соответствующие различным параметрам: числовые, нечисловые, непрерывные и дискретные. Простейшим примером такого многообразия шкал являются различные единицы измерения. Кроме этого можно противопоставлять абсолютные (реальные) шкалы и условные числовые шкалы, являющиеся способом формализации таких понятий, как "мало", "очень мало", "очень-очень мало", "много", "очень много" и т.д.

К 6-й группе ключевых узлов языка SC, в частности, относятся константные sc-узлы со следующими идентификаторами: *шкала измерения*, *номер*, *мощность множества* (=pwSet), *количество элементов* (=pwEl), *вес пары принадлежности* (=pwArc), *арность отношения*, *absScale*, *convScale*, *семейство множеств одинаковой мощности*, *килограмм*, *грамм*, *километр*, *метр*, *секунда*.

Семантика этих ключевых узлов рассмотрена в подразделе 6.1.

Ключевые узлы языка SC, относящиеся к 7-й группе ключевых узлов, обеспечивают описание всевозможных пространственных соотношений между объектами, таких как целое–часть, пространственная смежность, пространственное объединение, пространственное разбиение, пространственное пересечение.

Ключевые узлы языка SC, относящиеся к 8-й группе ключевых узлов, обеспечивают описание всевозможных темпоральных (временных) соотношений между процессами, таких как "быть этапом данного процесса", "начаться одновременно", "кончиться одновременно", "происходить во время выполнения данного процесса" (т.е. начаться не раньше и кончиться не позже), "начаться раньше", "кончиться позже", "происходить раньше" (в целом), "пересекаться во времени" (иметь одновременно выполняемые этапы), "непосредственно следовать за", "быть разбиением процесса на непересекающиеся во времени смежные этапы" и т.д.

Семантика темпоральных соотношений и соответствующих им **ключевых узлов** рассмотрена в пункте 6.2.

К 9-й группе ключевых узлов языка SC относятся константные узлы со следующими идентификаторами: *string*, *eqString*, *comprString*, *min_*, *max_*, *addString*, *symbol*.

Ключевой узел *string* является знаком множества знаков всевозможных строк символов (линейных текстов). Каждая такая строка в языке SC обозначается соответствующим константным sc-узлом.

Ключевой узел *eqString* является знаком симметричного отношения нефиксированной арности, каждая связка которого обозначает множество всех одинаковых символьных информационных конструкций. Например, запись вида: *eqString* → {*s*, *t*, *u*} ; означает, что sc-узлы *s*, *t*, *u* обозначают одинаковые строковые конструкции.

Ключевой узел *comprString* является знаком асимметричного отношения нефиксированной арности, выделяющего из множества строк минимальную и максимальную. Атрибутами отношения являются ключевые узлы *min_* и *max_*.

Ключевой узел *min_* является знаком атрибута, используемого в некоторых отношениях, указывающего на минимальный элемент. В частности, в рамках отношения *comprString* он указывает на минимальную строку.

Ключевой узел *max_* является знаком атрибута, используемого в некоторых отношениях, указывающего на максимальный элемент. В частности, в рамках отношения *comprString* он указывает на максимальную строку.

Ключевой узел *addString* является знаком асимметричного отношения нефиксированной арности соединения (конкатенации) строк, использующего атрибут *sum* для указания результата конкатенации и числовые атрибуты *1*, *2*, ... для указания порядка, в котором располагаются соединяемые строки. Например, запись вида *addString* $\rightarrow \{ 1 : s, 2 : t, sum_ : u \}$; означает, что строка, обозначенная узлом *u*, является результатом соединения строк *s* и *t*.

Ключевой узел *symbol* является знаком унарного отношения "быть символьной информационной конструкцией, состоящей из одного символа".

10-я группа ключевых узлов языка SC связана с понятием содержимого и с понятием идентификатора sc-элемента. Идентификатор sc-элемента и содержимое sc-элемента – это информационные конструкции, которые ставятся в соответствие указанному sc-элементу. Идентификаторы sc-элементов и содержимое sc-элементов вместе с текущим состоянием перерабатываемой sc-конструкции хранятся в памяти абстрактной sc-машины и используются при реализации ряда операций. Идентификаторы могут иметь sc-элементы любого вида (как узлы и дуги, так и элементы неопределенного типа, как константы, так и переменные). Разные sc-элементы обязаны иметь разные идентификаторы. Каждый идентификатор представляет собой некоторую строку символов.

Далеко не все хранимые в памяти sc-элементы должны иметь идентификаторы. Это требование предъявляется к ключевым узлам и ко всем хранимым sc-элементам, которые могут упоминаться при вводе новых sc-конструкций в память абстрактной sc-машины.

Содержимым могут обладать только константные предметные sc-узлы. При этом разные константные sc-узлы в принципе могут иметь одинаковое содержимое. Содержимое представляет собой информационную конструкцию произвольного вида. В частности, содержимым может быть любая символьная информационная конструкция, представление какого-либо числа в той или иной системе счисления. Далеко не все хранимые в памяти константные sc-узлы могут иметь содержимое. И далеко не для всех константных sc-узлов, у которых содержимое в принципе существует, это содержимое реально присутствует в текущем состоянии памяти, т.е. является вычисленным (сформированным) в текущий момент. Кроме того, содержимое sc-узла, независимо от того, сформировано оно или нет, может быть стационарным (фиксированным, не меняющимся в процессе переработки информации) и нестационарным (нефиксированным, меняющимся в процессе переработки информации). Узел с нестационарным содержимым – это аналог адресуемой области памяти традиционного компьютера, которая в процессе переработки информации меняет свое состояние.

К 10-й группе ключевых узлов языка SC относятся константные узлы со следующими идентификаторами: *formIdtf*, *existCont*, *formCont*, *fixCont*, *comprCont*, *ComprEqCont*, *EqCont*, *пояснение*, *текст_*, *wordDef*, *wordSem*, *word_*, *wordIncl*.

Ключевой узел *formIdtf* является знаком унарного отношения "быть sc-элементом, у которого в текущий момент времени имеется идентификатор". Проведение в некоторый sc-элемент константной негативной sc-дуги из узла *formIdtf* означает то, что у указанного sc-элемента идентификатор в текущий момент времени отсутствует.

Ключевой узел *existCont* является знаком унарного отношения "быть константным sc-узлом, у которого существует содержимое, но не обязательно в текущий момент времени". Проведение в некоторый sc-узел константной негативной sc-дуги из узла *existCont* означает то, что у указанного sc-узла содержимого в принципе быть не может.

Ключевой узел *formCont* является знаком унарного отношения "быть константным sc-узлом, у которого в текущий момент времени имеется содержимое". Проведение в некоторый sc-узел константной негативной sc-дуги из узла *formCont* означает то, что у указанного sc-узла содержимое в текущий момент времени отсутствует.

Ключевой узел *fixCont* является знаком унарного отношения "быть константным sc-узлом, содержимое которого является фиксированным и при этом не обязательно сформировано в текущий момент времени". Проведение в некоторый sc-узел, относящийся к классу *existCont*, константной

негативной sc-дуги из узла ***fixCont*** означает то, что у указанного sc-узла содержимое является нефиксированным, т.е. меняющимся в процессе переработки информации.

Ключевой узел *EqCont* является знаком симметричного отношения нефиксированной арности, каждая связка которого обозначает множество sc-узлов, имеющих равные содержимые.

Ключевой узел *пояснение* является знаком бинарного асимметричного отношения, связывающего множество sc-элементов любого вида с константным sc-узлом, содержимым которого является текст, представляющий собой пояснение указанного множества. Ключевой узел ***текст_*** является атрибутом донного отношения.

Ключевой узел *wordDef* является знаком бинарного асимметричного отношения, связывающего множество sc-элементов любого вида с константным sc-узлом, содержимым которого является текст, представляющий собой определение понятия, обозначаемого указанным sc-элементом. Ключевой узел ***текст_*** является атрибутом донного отношения. Семантика этих ключевых узлов более подробно рассмотрена в пункте 6.6.

Ключевой узел *text_* является знаком атрибута некоторых отношений, указывающего на константный sc-узел, содержимым которого является некоторый текст. Семантика этих ключевых узлов более подробно рассмотрена в пункте 6.6.

Ключевой узел *wordSem* является знаком бинарного асимметричного отношения, связывающего множество sc-элементов любого вида с константным sc-узлом, имеющим атрибут ***word_***, содержимым которого является текст, трактуемый как дополнительный (вспомогательный, альтернативный) идентификатор вышеуказанного sc-элемента.

Ключевой узел *word_* является знаком атрибута отношения ***wordSem***, указывающим на константный sc-узел, содержимым которого является слово или словосочетание, определяющее семантику некоторого понятия.

Ключевой узел *wordIncl* является знаком асимметричного отношения нефиксированной арности, которое описывает связь между вхождениями каких-либо терминов (термина) в какой-либо текст. Последним может быть раздел данного текста, либо библиографическая ссылка.

Семантика ключевых узлов, относящихся к 10-й группе также рассмотрена в пункте 6.6.

Ключевые узлы языка SC, относящиеся к 11-й группе ключевых узлов, обеспечивают запись:

- команд ассоциативного поиска и отображения sc-конструкций, соответствующих заданному образцу, который может иметь произвольный размер и произвольную конфигурацию;
- команд отображения содержимого указываемого sc-узла;
- команд удаления указываемого sc-элемента;
- команд ассоциативного поиска sc-конструкций, соответствующих заданному образцу, с последующим удалением тех sc-элементов, которые соответствуют указываемым элементам образца;
- команд ассоциативного поиска sc-конструкций, соответствующих другому заданному образцу (см. раздел 7).

4.5. Понятие sc-подъязыка. Семейство графовых языков, построенных на базе языка SC

Ключевые понятия: sc-подъязыки, семейство абстрактных sc-машин (информационных машин, внутренними языками которых являются различные sc-подъязыки), семейство sc-моделей (формальных моделей, реализуемых на различных абстрактных sc-машинах).

Понятие семейства sc-подъязыков, т.е. языков, являющихся подъязыками графового языка SC (см. определение 4.6.1), понятие семейства абстрактных sc-машин, т.е. абстрактных информационных машин, внутренними языками которых являются различные sc-подъязыки (см. определение 4.7.1), а также понятие семейства sc-моделей, т.е. формальных моделей, реализуемых на различных

абстрактных sc-машинах (см. определение 4.8.1), являются ключевыми понятиями всей данной работы. Это обусловлено тем, что:

- sc-подъязыки являются удобным средством семантического представления текстов самых различных языков (как символьных, так и графовых);
- абстрактные sc-машины являются удобным средством рассмотрения на семантическом уровне самых различных способов организации переработки информации;
- sc-модели являются удобным средством семантического представления самых различных формальных моделей и удобным средством приведения различных формальных моделей к общему виду.

Поэтому все рассматриваемые в данной работе графовые языки относятся к классу sc-подъязыков, все рассматриваемые абстрактные машины – к классу абстрактных sc-машин, а формальные модели – к классу sc-моделей.

Важнейшим достоинством семейства всевозможных sc-подъязыков, семейства абстрактных sc-машин и семейства формальных sc-моделей является то, что любой язык, любую абстрактную машину и любую формальную модель можно достаточно легко представить в виде эквивалентного sc-подъязыка, эквивалентной абстрактной sc-машины, эквивалентной формальной sc-модели. Такая уникальная возможность приведения к общему виду самых различных языков, абстрактных машин и формальных моделей создает все необходимые предпосылки для эффективной интеграции любых языков, абстрактных машин и формальных моделей, так как интегрировать различные подъязыки языка SC, различные абстрактные sc-машины, различные формальные sc-модели существенно проще, чем интегрировать произвольные языки, произвольные абстрактные машины и произвольные формальные модели.

Итак, sc-подъязыки являются удобным средством уточнения денотационной семантики самых различных (в первую очередь графовых) языков, а абстрактные sc-машины являются удобным средством уточнения операционной семантики всевозможных (и в первую очередь графовых) языков.

Введенные нами понятия sc-конструкции, sc-подъязыка, абстрактной sc-машины и формальной sc-модели дают возможность уточнить такие понятия, как:

- эквивалентность информационных конструкций, языков, абстрактных машин и формальных моделей;
- интеграция информационных конструкций, языков, абстрактных машин и формальных моделей;
- интерпретация абстрактных машин и формальных моделей.

Графовый язык SC рассматривается нами как ядро целого семейства графовых языков, являющихся подъязыками языка SC и называемых в соответствии с этим sc-подъязыками.

Будем называть **sc-подъязыком** такой графовый язык, каждая конструкция которого является sc-конструкцией.

Благодаря тому, что язык SC обладает неограниченной семантической мощностью, семейство графовых языков, создаваемых на его базе, заполняет абсолютно весь семантический спектр языков, т.е. в число sc-подъязыков входят и sc-подъязыки, являющиеся языками программирования (т.е. языками представления программ) и sc-подъязыки, являющиеся языками представления данных, перерабатываемых программами, и sc-подъязыки, являющиеся языками представления фактурографических высказываний, и sc-подъязыки, являющиеся языками представления логических высказываний (т.е. знаний о свойствах и закономерностях предметной области). Примеры таких языков см. в разделах 5, 6.

Синтаксис и семантика графовых языков опираются на понятие ключевого узла. У графовых языков, конструкции которых имеют метки (унарные отношения, заданные на множестве элементов конструкции), ключевую роль могут иметь также и метки. У языка SC набор меток фиксирован, поэтому основную роль в синтаксисе и семантике языков, построенных на базе языка SC, выполняет определенный набор ключевых узлов языка SC, знание семантики которых обеспечивает определение смысла (прочтение, расшифровку, понимание) любой конструкции такого языка. Следовательно, определение денотационной семантики ключевых узлов графового языка, сделанное на каком-либо метаязыке (в качестве которого может быть использован и естественный язык), составляет основу описания денотационной семантики каждого графового языка. При этом обратим внимание на то, что среди ключевых узлов графового языка отсутствуют аналоги разделительных и ограничительных ключевых лексем символьных языков. Разделителям и ограничителям трудно поставить в

соответствие какую-либо денотационную семантику, поскольку они используются только как средство структуризации линейных символьных конструкций.

Итак, каждый конкретный sc-подъязык имеет свои семантические и синтаксические особенности в дополнение к базовым семантическим и синтаксическим особенностям самого языка SC (см. подразделы 4.2 – 4.4). При этом дополнительные (к языку SC) семантические особенности каждого конкретного sc-подъязыка в полной мере могут быть описаны путем описания денотационной семантики всех ключевых узлов этого sc-подъязыка, точнее, ключевых узлов, вводимых в дополнение к ключевым узлам базового языка SC. Подчеркнем, что все ключевые узлы языка SC входят в число ключевых узлов каждого графового языка, построенного на его базе. Таким образом, создание на базе языка SC каждого конкретного sc-подъязыка, по существу, сводится к определению набора всех ключевых узлов создаваемого языка.

Перейдем к рассмотрению вопросов, связанных с интеграцией sc-конструкций и с интеграцией sc-подъязыков. **Интеграция двух sc-конструкций** – это их конкатенация, предполагающая склеивание синонимичных sc-элементов, принадлежащих разным интегрируемым sc-конструкциям. Напомним, что в рамках каждой sc-конструкции (в том числе и той, которая является результатом интеграции) неявная синонимия sc-элементов запрещена. Таким образом, основная проблема интеграции sc-конструкций заключается в поиске синонимичных sc-элементов, подлежащих склеиванию. Интегрировать различные языки – значит, привести к некоторому общему виду конструкции интегрируемых языков и обеспечить их совместное хранение и совместную переработку в памяти одной абстрактной машины. Приведение различных языков к общему виду может быть осуществлено путем построения sc-подъязыков, эквивалентных этим языкам. Интеграция различных sc-подъязыков сводится 1) к склеиванию ключевых узлов, принадлежащих разным sc-подъязыкам и оказавшихся синонимичными, и 2) к определению правил (операций) склеивания неключевых узлов, принадлежащих конструкциям разных sc-подъязыков и являющихся синонимичными. При этом возможна и более тонкая интеграция sc-подъязыков, заключающаяся в таком эквивалентном изменении интегрируемых sc-подъязыков, которое приводит к максимально возможному количеству синонимичных (и соответственно склеиваемых) ключевых узлов. Такая тонкая интеграция sc-подъязыков заключается в сближении семантически близких ключевых узлов, принадлежащих разным sc-подъязыкам, и приводит к сокращению общего количества ключевых узлов того sc-подъязыка, который является результатом интеграции. Очевидно, что из двух семантически эквивалентных sc-подъязыков лучшим следует считать тот, который имеет меньшее количество ключевых узлов.

4.6. Понятие абстрактной sc-машины

Ключевые понятия и идентификаторы ключевых узлов: абстрактная sc-машина, sc-память, sc-операция, микропрограмма sc-машины, интерпретация sc-машины, интеграция sc-машин, *node*, *const*, *arc*, *elem*, *var*, *pos*, *neg*, *fuz*, s-блокировка, e-блокировка, g-блокировка.

Абстрактная sc-машина (графодинамическая абстрактная машина, ориентированная на переработку sc-конструкций) представляет собой абстрактную графодинамическую параллельную асинхронную информационную машину, у которой внутренним языком является графовый язык SC и в графодинамической памяти в виде sc-конструкций хранится полная информация, необходимая не только для операций абстрактной sc-машины, но и для всех ее микропрограмм, т.е. все вспомогательные информационные конструкции, необходимые для реализации микропрограмм абстрактной sc-машины, также должны быть представлены в памяти этой машины.

Память абстрактной sc-машины, которую будем также называть **sc-памятью**, обеспечивает хранение текущего состояния перерабатываемой sc-конструкции. Никаких других информационных конструкций sc-память не содержит (точнее говоря, любые информационные конструкции, не являющиеся sc-конструкциями, могут храниться в sc-памяти, но только в том случае, если каждая из этих информационных конструкций является содержимым соответствующего константного sc-узла). Таким образом, sc-память представляет собой динамическую (меняющуюся во времени) sc-конструкцию и относится к классу графовых структурно-перестраиваемых (т.е. графодинамических) запоминающих сред. Переработка информации в sc-памяти сводится к следующему:

- генерации новых sc-узлов, которым сразу приписываются метка *node* и одна из меток, принадлежащих множеству *{ const , var }*;
- генерации новых sc-элементов неопределенного типа, которым сразу приписываются метка *elem* и метка из множества *{ const , var }*;

- генерации новых sc-дуг, каждая из которых проводится между двумя имеющимися sc-элементами; сгенерированной sc-дуге сразу приписываются метка *arc*, а также метки из множеств *{pos, neg, fuz}* и *{const, var}*;
- удалению (стиранию) имеющихся sc-элементов (sc-узлов, sc-элементов неопределенного типа, sc-дуг);
- формированию, удалению и модификации содержимого sc-узлов;
- формированию, удалению и модификации идентификаторов sc-элементов.

Следует подчеркнуть то, что в процессе переработки sc-конструкций метки sc-элементов (к числу которых относятся *node*, *arc*, *elem*, *const*, *var*, *pos*, *neg*, *fuz*) меняться не могут в том смысле, что изменение метки по существу означает замену этого элемента на другой элемент, имеющий совершенно другую денотационную семантику.

Итак, в sc-памяти хранятся:

- sc-элементы, каждому из которых приписываются соответствующая метка и его принадлежность к конкретному типу (константа или переменная, а для дуг дополнительно – позитивная, негативная или нечёткая). При этом для sc-дуги дополнительно указываются соединяемые ею sc-элементы;
- информационные конструкции, каждая из которых является содержимым некоторого обязательно указываемого sc-узла;
- символьные информационные конструкции, каждая из которых является идентификатором некоторого обязательно указываемого sc-элемента.

Операции абстрактной sc-машины будем также называть **sc-операциями**, операциями над sc-конструкциями, операциями над sc-памятью. Разные абстрактные sc-машины имеют разные системы операций. Следовательно, понятие абстрактной sc-машины определяет целый класс (семейство) абстрактных машин, имеющих одинаковую организацию памяти, но разные системы операций.

Каждой sc-операции взаимно однозначно соответствует **микропрограмма**, определяющая операционную семантику этой sc-операции, т.е. уточняющая то, какое преобразование эта операция осуществляет над sc-памятью. Микропрограммы операций абстрактной sc-машины, как и других абстрактных машин являются демоническими, т.е. автономными, активными, самоинициируемыми программами. Демонические микропрограммы могут вызывать другие микропрограммы, которые являются уже пассивными, т.е. инициируемыми только в результате явного обращения к ним из других микропрограмм либо (в случае рекурсивной микропрограммы) из нее же самой. Таким образом, множество микропрограмм абстрактной sc-машины (как, впрочем, и любой другой абстрактной машины) может быть условно разбито на множества демонических микропрограмм и на множество пассивных микропрограмм. Преобразования, выполняемые пассивными микропрограммами абстрактной машины, будем условно называть базовыми преобразованиями этой машины.

Следует отметить, что одинаковая организация памяти для разных абстрактных sc-машин и запрет их микропрограммам использовать информационные конструкции, хранимые не в sc-памяти, создают все необходимые предпосылки для создания общего для всех sc-машин языка микропрограммирования. В качестве такого языка предлагается язык SCP (Semantic Code Programming) [411] ([ПрогРАМ-2001кн](#)). В свою очередь, наличие языка микропрограммирования, общего для всех sc-машин, создает необходимые предпосылки для создания универсальной sc-машины, которая обеспечивает интерпретацию любой абстрактной sc-машины. В качестве такой универсальной sc-машины предлагается SCP-машина, рассматриваемая в [411] ([ПрогРАМ-2001кн](#)).

Будем говорить, что sc-машина *Cj* **интерпретирует** sc-машину *Ci* в том и только в том случае, если:

- в памяти sc-машины *Cj* содержатся все sc-конструкции, перерабатываемые в памяти sc-машины *Ci*;
- в памяти sc-машины *Cj* дополнительно хранятся все микропрограммы sc-машины *Ci*, представленные соответственно в виде sc-конструкций. Микропрограммы машины *Ci* в машине *Cj* становятся уже не микропрограммами, а хранимыми в памяти программами;
- в памяти sc-машины *Cj* дополнительно хранятся все вспомогательные данные, необходимые для реализации микропрограмм машины *Ci*;
- система операций sc-машины *Cj* обеспечивает реализацию любой хранимой в ее памяти микропрограммы sc-машины *Ci*.

Графодинамическая абстрактная sc-машина является весьма перспективным (как в теоретическом, так и практическом плане) уточнением понятия абстрактной машины, так как хорошие метаязыковые возможности языка SC и его открытость дают основания претендовать на роль мощного средства реализации самых различных формальных моделей, использующих самые различные языки. Но особенно важно то, что абстрактные sc-машины являются мощным средством описания операционной семантики самых различных языков (языков программирования, языков представления знаний). Для того чтобы описать операционную семантику какого-либо языка с помощью абстрактной sc-машины, необходимо:

- 1) разработать способ отображения произвольных конструкций описываемого языка в семантически эквивалентные sc-конструкции;
- 2) для каждой операции абстрактной машины, определяющей операционную семантику описываемого языка, построить эквивалентную ей операцию абстрактной sc-машины, описывающую определенный класс преобразований sc-конструкций, эквивалентных соответствующим конструкциям описываемого языка.

Другими словами, речь идет о моделировании на абстрактной sc-машине всевозможных абстрактных машин, но при взаимно однозначном отображении множества операций моделируемой абстрактной машины во множество операций абстрактной sc-машины. Последнее обстоятельство, обусловленное открытым характером операций абстрактной sc-машины, является принципиально важным. Принципиально важной является также возможность разработки общих принципов кодирования конструкций всевозможных языков с помощью конструкций языка SC, являющихся специальным видом однородных семантических сетей. Такие общие принципы сводятся к разработке системы ключевых sc-узлов, используемых при представлении конструкций любых языков. Наличие общих принципов кодирования конструкций всевозможных языков на языке SC обеспечивает не только совершенствование методов описания семантики этих языков, но и упрощение интеграции абстрактных машин, определяющих операционную семантику самых различных языков. Наличие таких общих принципов кодирования должно привести к тому, чтобы общность семантики различных языков проявлялась бы не только в совпадении некоторых ключевых sc-узлов в соответствующих им абстрактных sc-машинах, но и в совпадении некоторых операций этих абстрактных sc- машин.

Элементарные процессы абстрактной sc-машины выполняются параллельно и асинхронно над общей для них памятью (sc-памятью). Если области памяти, которые обрабатывают ("захватывают") элементарные процессы, не пересекаются, то эти процессы никак не влияют друг на друга. Взаимодействие элементарных процессов, обрабатывающих пересекающиеся области памяти, осуществляется следующим образом. При параллельном выполнении элементарных процессов над общими фрагментами sc-памяти действия, выполняемые над каждым хранимым в памяти sc-элементом, выполняются строго последовательно. Каждый элементарный процесс блокирует элементы обрабатываемой им sc-конструкции (хранимой в памяти) с помощью различных типов блокировки (**s-блокировки**, **e-блокировки** и **g-блокировки** – см. пояснения 4.6.2.1 – 4.6.2.3).

Пояснение 4.7.1. Блокировка sc-элемента, имеющая тип **s-блокировки** sc-элемента, запрещает другим элементарным процессам удалять этот элемент, а точнее, приписывать ему e-блокировку до тех пор, пока эта s-блокировка с указанного sc-элемента не будет снята. Следовательно, каждый элементарный процесс sc-машины, попытавшийся e-блокировать sc-элемент, s-блокированный другим элементарным процессом, прерывается и переходит в состояние ожидания снятия s-блокировки с указанного sc-элемента. Прерванный процесс продолжится после того, как он дождется указанной ситуации. Таким образом, sc-элемент, s-блокированный элементарным процессом абстрактной sc-машины, – это sc-элемент, существовавший до этого элементарного процесса и сохраняемый им до своего завершения. Один и тот же sc-элемент может быть s-блокирован несколькими элементарными процессами.

Пояснение 4.7.2. SC-элемент, **e-блокированный** элементарным процессом абстрактной sc-машины, – это sc-элемент, существовавший до этого элементарного процесса и удаляемый им на завершающей стадии своего выполнения. sc-элемент может быть e-блокирован только одним элементарным процессом. Для всех остальных элементарных процессов e-блокированный sc-элемент "невидим", т.е. для них этого sc-элемента уже не существует.

Пояснение 4.7.3. SC-элемент, **g-блокированный** элементарным процессом абстрактной sc-машины, порожден этим процессом и на завершающей стадии выполнения процесса либо удаляется (в случае, если это вспомогательный sc-элемент), либо освобождается от g-блокировки. SC-элемент может быть g-блокирован только одним элементарным процессом. Для всех остальных элементарных процессов g-блокированный sc-элемент "невидим", т.е. для них этого sc-элемента еще не существует.

Рассмотренные принципы разрешения конфликтов между параллельными элементарными процессами sc-машины, выполняемыми над общей sc-памятью, могут быть также использованы и для разрешения конфликтов между абстрактными машинами, параллельно работающими над общей памятью. Это означает, что проблемы интеграции абстрактных sc-машин фактически не существует – после интеграции sc-подъязыков, соответствующих интегрируемым sc-машинам, достаточно просто объединить множества операций этих машин.

Такая легкая интегрируемость абстрактных sc-машин и обуславливает высокий уровень их гибкости (открытости, модифицируемости, наращиваемости).

Каждой sc-операции однозначно соответствует также некоторое множество sc-узлов, хранимых в памяти абстрактной sc-машины и являющихся константами всех микропрограмм, используемых для реализации указанной sc-операции (сюда входят соответствующая этой sc-операции демоническая микропрограмма, микропрограммы, к которым она обращается, микропрограммы, к которым обращаются эти микропрограммы, и т.д.). Заметим, что константами микропрограмм абстрактной sc-машины могут быть только константные sc-узлы, т.е. таковыми не могут быть ни переменные sc-узлы, ни sc-дуги любого вида, ни sc-элементы неопределенного типа. Множество всех sc-узлов, являющихся константами всех микропрограмм, используемых для реализации некоторой sc-операции, будем называть ключевыми узлами этой sc-операции. Образно говоря, это те sc-узлы, к которым соответствующая sc-операция как бы "привязывается". SC-узел, хранимый в памяти абстрактной sc-машины и являющийся ключевым по крайней мере для одной из ее операций, будем называть ключевым узлом этой sc-машины.

Очевидно, что все ключевые узлы абстрактной sc-машины должны входить в число ключевых узлов внутреннего языка этой машины. Заметим при этом, что ключевые узлы внутреннего языка абстрактной sc-машины кроме ключевых узлов самой машины могут включать в себя и другие узлы. То есть понятие ключевого узла графового языка несколько шире понятия ключевого узла графодинамической машины, работающей на этом графовом языке.

Поскольку разные абстрактные sc-машины отличаются разным набором операций, то и наборы ключевых узлов у них в общем случае будут разными. Следовательно, строго говоря, разные абстрактные sc-машины работают на разных внутренних языках, имеющих разные наборы ключевых узлов, но являющихся при этом подъязыками одного и того же базового языка – языка SC.

Итак, для всего семейства абстрактных sc-машин фиксируются:

- ядро внутренних языков этих машин – язык SC;
- принципы организации памяти, обеспечивающей хранение и переработку sc-конструкций. Память всех sc-машин является графодинамической (структурно-перестраиваемой) и ассоциативной, поддерживающей ассоциативный доступ по произвольному образцу (по образцовой sc-конструкции произвольного размера и произвольной структуры);
- параллельный, асинхронный характер реализации элементарных процессов;
- принципы разрешения конфликтов между элементарными процессами, параллельно выполняемыми над общей sc-памятью;
- принципы построения микропрограмм, в частности, запрещающие использование микропрограммами каких-либо вспомогательных данных, не хранимых в sc-памяти в виде соответствующих sc-конструкций.

Таким образом, в рамках всего семейства абстрактных sc-машин не фиксируются ни конкретный подъязык языка SC (разные sc-машины в качестве своего внутреннего языка могут использовать разные подъязыки языка SC), ни набор операций и соответствующих им микропрограмм (разные sc-машины могут иметь разный набор операций).

Общим для всех абстрактных sc-машин является только небольшой набор специальных операций, обеспечивающих просмотр и редактирование sc-конструкций, хранимых в графодинамической памяти. Ключевые узлы языка SC, используемые для записи соответствующих команд просмотра и редактирования, рассмотрены в [подразделе 2.6](#).

Перечисленные выше общие принципы организации абстрактных sc-машин дают возможность существенно упростить решение проблемы интеграции абстрактных машин и формальных моделей (абстрактные машины, относящиеся к классу sc-машин, очевидно, существенно проще интегрировать, чем разнородные абстрактные машины), а также решение проблемы интерпретации абстрактных машин и формальных моделей (очевидно, что для интерпретации абстрактной sc-машины, особенно сложной, проще использовать абстрактную машину, также относящуюся к классу sc-машин).

4.7. Понятие формальной sc-модели

Ключевые понятия: формальная модель, формальная sc-модель.

Общее определение формальной модели приведено в пункте 1.1.1 (определение 1.1.).

Формальную модель, задаваемую языком, относящимся к классу sc-подъязыков, и абстрактной машиной, относящейся к классу абстрактных sc-машин, будем называть **формальной sc-моделью**.

Открытый (гибкий) характер формальных sc-моделей является важнейшим их достоинством и определяется легкой интегрируемостью формальных sc-моделей, которая, в свою очередь, обусловлена легкой интегрируемостью sc-подъязыков, sc-конструкций и абстрактных sc-машин. Интеграция формальных sc-моделей сводится к интеграции их языков (sc-подъязыков), к интеграции их аксиом (начальных sc-конструкций) и к интеграции их абстрактных машин (sc-машин). От интеграции формальных sc-моделей можно перейти к интеграции произвольных формальных моделей, которая сводится к следующим этапам:

- 1) построение sc-подъязыков, семантически эквивалентных языкам, используемым в интегрируемых формальных моделях, т.е. приведение указанных языков к общему каноническому виду – к некоторому способу представления текстов этих языков в виде sc-конструкций;
- 2) интеграция построенных sc-подъязыков с возможной корректировкой набора ключевых узлов для каждого из этих языков;
- 3) построение абстрактных sc-машин, семантически эквивалентных абстрактным машинам, используемым в интегрируемых формальных моделях;
- 4) интеграция построенных абстрактных sc-машин с возможной корректировкой набора ключевых узлов в каждой из этих машин;
- 5) построение sc-конструкций, семантически эквивалентных аксиомам интегрируемых формальных моделей и оформленных на указанных выше sc-подъязыках;
- 6) интеграция построенных sc-конструкций.

Выводы к разделу 1

Отличие языка SC от его базового подмножества (языка SCB) сводится:

- к расширению понятия scb-элемента путем включения в число элементов текста не только знаков множеств, но и простых переменных, значениями которых являются знаки множеств, а также метапеременных, значениями которых являются простые переменные. При этом значениями простых переменных могут быть знаки не только узловых множеств (в том числе знаки кортежей), но и знаки пар принадлежности;
- к расширению понятия множества путем включения в число возможных элементов множества не только знаков множеств, но и переменных.

1. Представление логических формул и формальных теорий в памяти графодинамических ассоциативных машин

Трактовка атомарных логических формул как множеств, элементами которых являются элементарные составляющие логических формул (константы и переменные), явное введение знаков для всех логических формул, входящих в состав логического текста, и, наконец, трактовка неатомарных (сложных) логических формул как множеств, элементами которых являются знаки логических формул, которые входят в состав этих неатомарных логических формул, – все это позволяет использовать фактографический язык SCB для представления логических формул и для описания соотношений между ними. Единственная особенность такого использования языка SCB заключается в том, что здесь приходится иметь дело не только с множествами, элементами которых являются знаки множеств (т.е. нормализованными множествами), но и с множествами, среди элементов которых встречаются переменные, которые, строго говоря, знаками множеств не являются.

Логические языки, построенные на базе языка SCB на основе указанных выше принципов, с полным основанием можно считать логическими языками теоретико-множественного или реляционного типа, т.е. языками, в основе которых лежит теоретико-множественный способ трактовки структуры логических формул. Рассмотрим один из вариантов такого логического языка. Назовем его SCL (Semantic Code Logic).

Данный раздел может быть использован в качестве учебного пособия по дисциплинам «Математические основы искусственного интеллекта» и «Логические основы интеллектуальных систем» для студентов специальности «Искусственный интеллект».

1.1. Принципы построения графового логического языка SCL (Semantic Code Logic) на теоретико-множественной основе

В данном подразделе рассматриваются специальные отношения и атрибуты, обеспечивающие представление в языке SCL неатомарных логических формул и формальных теорий. Неатомарная логическая формула в языке SCL трактуется как неориентированное множество или кортеж (ориентированное множество), в состав которого входят знаки логических формул, из которых состоит эта неатомарная логическая формула. Таким образом, для записи неатомарных логических формул вполне можно использовать фактографический язык SCB. В этом и заключается суть языка SCL. Система ключевых понятий, а следовательно, система ключевых узлов и синтаксис языка SCL не является единственно возможным способом построения логического языка на базе языка SCB.

Формальный логический язык SCL построен на базе языка SC как его подъязык путем фиксации определенного набора специальных ключевых узлов, т.е. узлов, семантика которых должна быть априори известна и согласована. Перечислим основные ключевые узлы языка SCL:

atExpr, conj, disj, alt, impl, if_, then_, eqExpr, negExpr, fuzExpr, pwFuzExpr, exist, fix_, pwExist, all, existAtExpr, allImpl, allEqExpr, theory, union_.

Каждая логическая формула (как атомарная, так и неатомарная) входит (в качестве компонента) в состав какой-либо формулы. В конечном счете такой неатомарной логической формулой является сама формальная теория, трактуемая как априори истинное конъюнктивное высказывание, т.е. как множество истинных высказываний различного вида. Тип логической формулы задается ключевым множеством, которое содержит все формулы такого типа. Здесь следует выделить группы таких ключевых множеств:

- Первая группа множеств, определяющих тип логической формулы, разбивает логические формулы по их структурному виду соответственно, включает себя следующие множества:
 - **множество атомарных** логических формул, которое будем обозначать ключевым узлом “**atExpr**” (быть атомарной логической формулой, каждая из которых представляет собой множество, состоящее из (1) знаков узловых множеств, (2) знаков пар принадлежности, (3) переменных, значениями которых являются знаки узловых множеств, (4) переменных, значениями которых являются знаки пар принадлежности, (5) метапеременных (если логическая формула относится к метатеории));
 - **множество конъюнктивных** логических формул, которое будем обозначать ключевым узлом “**conj**” (быть конъюнктивной формулой);

- множество дизъюнктивных формул, которое будем обозначать ключевым узлом “*disj*” (быть нестрогой дизъюнктивной формулой);
- множество альтернативных формул, которое будем обозначать ключевым узлом “*alt*” (быть строгой дизъюнктивной формулой – логической формулой исключающего ИЛИ);
- множество импликативных формул, которое будем обозначать ключевым узлом “*impl*” ;
- множество логических формул об эквивалентности, которое будем обозначать ключевым узлом “*eqExpr*” ;
- множество негативных логических формул, каждая из которых трактуется как 1-мощное множество (синглтон), элементом которого является отрицаемая логическая формула (т.е. формула, на которую действует логическое отрицание). Множество негативных логических формул обозначается ключевым узлом “*negExpr*” .
- Вторая группа множеств используется для определения типа кванторных логических формул и соответственно включает в себя следующие множества:
 - множество формул о существовании, которое будем обозначать ключевым узлом “*exist*” ;
 - множество формул о всеобщности, которое будем обозначать ключевым узлом “*all*” .

Кроме множеств, указывающих тип самой логической формулы, введем атрибуты, указывающие роль формулы, входящей в состав неатомарной формулы, представляющей собой кортеж из знаков других логических формул. К таким атрибутам относятся:

- “*if_*” (быть посылкой импликативной логической формулы);
- “*then_*” (быть следствием импликативной логической формулы);
- “*fix_*” (быть множеством фиксируемых элементов, т.е. тех переменных, которые связываются каким-либо квантором в рамках кванторной формулы – при этом в это множество разрешается включать уже выше зафиксированные, т.е. уже связанные переменные и даже константы).

Специальным видом неатомарных формул являются формальные теории. Формальная теория задается путем причисления ее знака ко множеству с именем (идентификатором) “*theory*”, которое представляет собой множество знаков всевозможных формальных теорий. А поскольку каждая формальная теория есть кортеж, множество *theory* можно трактовать как ориентированное отношение. В состав формальной теории обязательно входит знак описываемой этой теорией реляционной структуры под атрибутом “*union_*”.

1.2. Запись логических формул с использованием стилизованного естественного языка

Для записи логических формул на языке, близком к естественному, будем использовать такие варианты (шаблоны) этих формулировок, от которых легко можно перейти к формальному логическому языку. Заметим, что в этих шаблонах используются тексты языка SCg или языка SCs для записи атомарных логических формул. А сам стилизованный естественный язык используется для представления семантической структуры изображаемых (записываемых) неатомарных логических формул. Перечислим эти шаблоны.

Здесь прямоугольниками обозначаются тексты, построенные по одному из перечисленных шаблонов и представляющие собой запись различных логических формул.

Имеет место **конъюнкция** следующих формул:

-
-

/* Запись конъюнктивной формулы */

Имеет место **дизъюнкция** следующих формул:

-
-

/* Запись дизъюнктивной формулы */

Имеет место **строгая дизъюнкция** следующих формул:

-
-

/* Запись строгой дизъюнктивной формулы */

Имеет место **эквивалентность** следующих формул:

-
-

/* Запись формулы об эквивалентности */

Имеет место **импликация** следующих формул:

- **если**
- **то**

/* Запись импликативной формулы */

Существует конструкция вида:

[< атомарная формула >] ;

/* Запись формулы о существовании, в которой квантор существования действует на атомарную формулу */

Существует конструкция вида:

[< атомарная формула >] ;

у которой:

/* Запись формулы о существовании, в которой квантор существования действует на неатомарную формулу */

Для всех значений переменных:

[< атомарная формула >] ;

/* Запись формулы о всеобщности */

Преобразование перечисленных вариантов записи позитивных логических формул в негативные осуществляется добавлением в самом начале текста отрицания “**не**”.

Одной из наиболее часто используемых логических формул об эквивалентности является определение, которые выглядят следующим образом:

Имеет место **эквивалентность** следующих формул:

- Существует конструкция вида:

`[_x <— s ;]`;

- 

`/* Запись определения множества с именем "s" */`

Здесь прямоугольником изображается текст, являющийся формулировкой критерия, которому должны удовлетворять все элементы определяемого множества и только они, т.е. формулировкой критерия принадлежности произвольного элемента `_x` определяемому множеству `s`.

1.3. Язык SCLs (Semantic Code Logic string) – формальный линейный логический язык классического типа, использующий язык SCs для записи атомарных логических формул

Логический язык SCL является языком теоретико-множественного типа, в основе которого лежит трактовка логических связок и кванторов через понятия множества, кортежа, атрибута, отношения, т.е. трактовка формальных теорий и неатомарных логических формул как реляционных структур над высказываниями.

Язык SCL является подъязыком языка SC и имеет две модификации:

- линейную модификацию – язык SCLs (Semantic Code Logic string);
- графическую модификацию – язык SCLg (Semantic Code Logic graphical).

Основное отличие языка SCLs от классического логического языка заключается в способе записи атомарных логических формул. Атомарная логическая формула в языке SCLs – это текст языка SCs, ограниченный квадратными скобками. Неатомарные (сложные) логические формулы в языке SCLs строятся точно так же, как и в классическом логическом языке.

Если `bi` и `bj` есть scs-формулы, а переменная `xi` является свободной переменной логической формулы `bi`, то логическими формулами также являются конструкции вида:

<code>(¬ bi)</code>	– логическая формула, являющаяся отрицанием формулы <code>bi</code> ;
<code>(bi & bj)</code>	– конъюнктивная формула, являющаяся конъюнкцией формул <code>bi</code> и <code>bj</code> ;
<code>(bi ∨ bj)</code>	– дизъюнктивная формула;
<code>(bi bj)</code>	– строгая дизъюнктивная формула;
<code>(bi —> bj)</code>	– импликативная формула;
<code>(bi <=⇒ bj)</code>	– логическая формула об эквивалентности;
<code>(Эxi bi)</code>	– логическая формула о существовании;
<code>(Э!xi bi)</code>	– логическая формула о существовании и единственности;

-
- ($\exists n/xi\ bi$) – логическая формула о существовании n значений xi , удовлетворяющих формуле bi ;
- ($\exists >n/xi\ bi$) – логическая формула о существовании более чем n значений xi , удовлетворяющих формуле bi ;
- ($\exists \geq n/xi\ bi$) – логическая формула о существовании не менее чем n значений xi , удовлетворяющих формуле bi ;
- ($\exists <n/xi\ bi$) – логическая формула о существовании менее чем n значений xi , удовлетворяющих формуле bi ;
- ($\exists \leq n/xi\ bi$) – логическая формула о существовании менее чем n значений xi , удовлетворяющих формуле bi ;
- ($\exists /n_1, n_2/xi\ bi$) – логическая формула о существовании n значений переменной xi , удовлетворяющих формуле bi , где $n_1 \leq n \leq n_2$;
- ($\exists /n_1<, n_2/xi\ bi$) – логическая формула о существовании n значений переменной xi , удовлетворяющих формуле bi , где $n_1 < n \leq n_2$;
- ($\exists /n_1<, < n_2/xi\ bi$) – логическая формула о существовании n значений переменной xi , удовлетворяющих формуле bi , где $n_1 < n < n_2$;
- ($\exists /n_1, < n_2/xi\ bi$) – логическая формула о существовании n значений переменной xi , удовлетворяющих формуле bi , где $n_1 \leq n < n_2$;
- ($\forall xi\ bi$) – логическая формула о всеобщности, т.е. о том, что каждое значение переменной xi удовлетворяет формуле, т.е. "превращает" эту логическую формулу в истинное высказывание после соответствующей подстановки.

Приведем в табл. 5.3.1. перечень разделителей языка SCLs, которые используются для записи неатомных высказываний.

Таблица 5.3.1. Специальные разделители логического языка SCLs

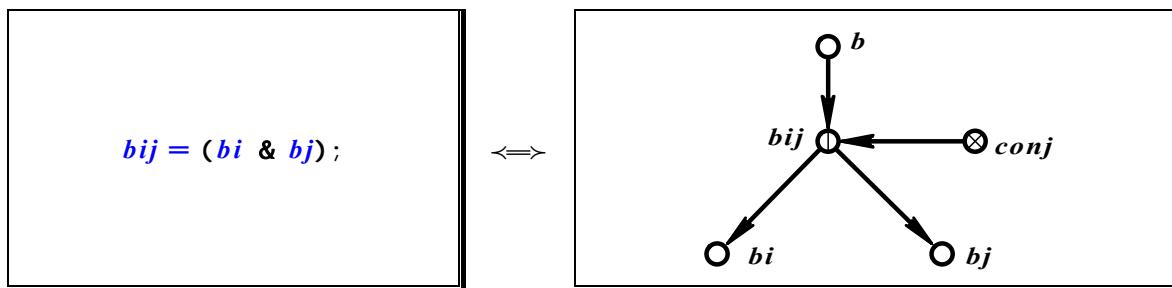
Разделитель	Комментарий
¬	символ логического отрицания
&	связка конъюнкции
∨	связка дизъюнкции
	связка строгой дизъюнкции
→	связка импликации
↔	связка эквиваленции
∃	квантор существования
∃!	квантор существования и единственности
∀	квантор всеобщности

/	разделитель (ограничитель) диапазона в формулах существования определенного количества значений
<	разделители, указывающие на включение границы диапазона
>	разделители, указывающие на не включение границы диапазона
‘	разделитель значений границ диапазона в формулах существования определенного количества значений

1.4. Язык SCLg (Semantic Code Logic graphical) – графический вариант изображения текстов языка SCL

Рассмотрим представление конкретных типов логических формул на языке SCLg путем "перевода" соответствующих scls-конструкций на язык SCLg (см. scl-тексты 5.4.1 – 5.4.6).

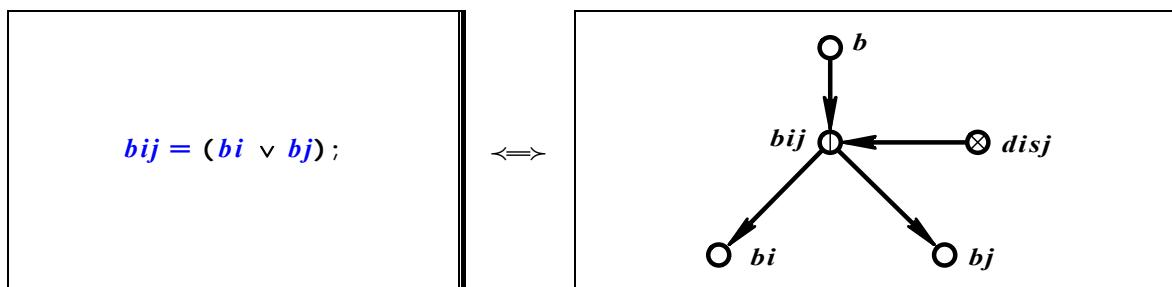
SCL - текст 5.4.1. Представление конъюнктивной логической формулы



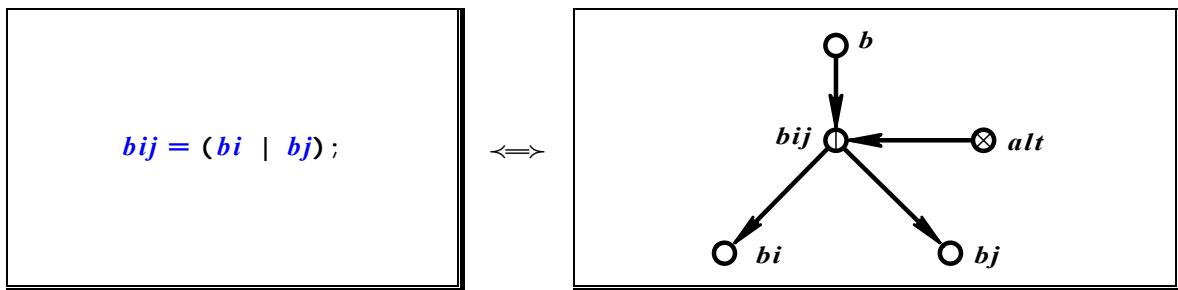
Здесь b есть неатомарная логическая формула дополнительно уточняемого вида, в состав которой формула bij непосредственно входит.

Примечание. В языке SCL допустимо существование вырожденных конъюнктивных логических формул, состоящих из одного компонента.

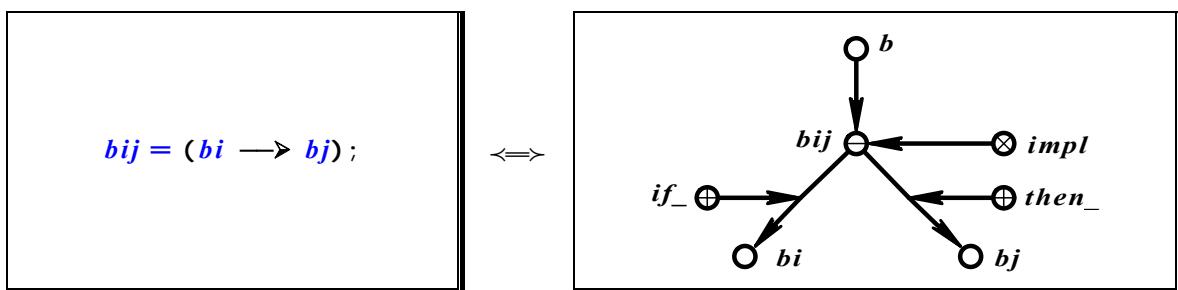
SCL - текст 5.4.2. Представление дизъюнктивной логической формулы



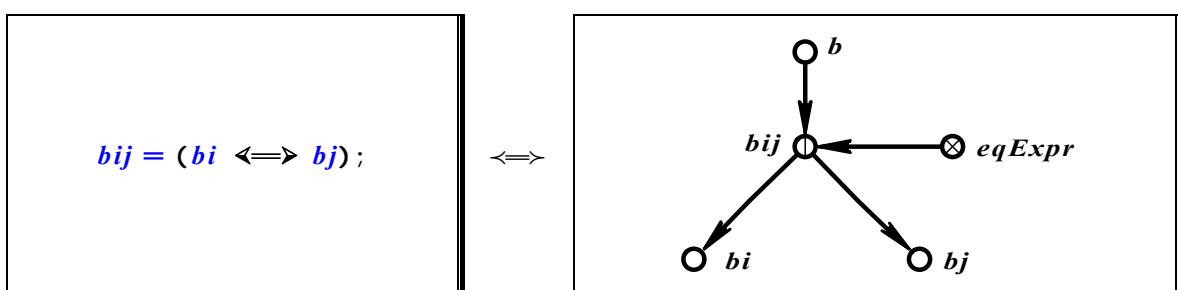
SCL - текст 5.4.3. Представление альтернативной логической формулы



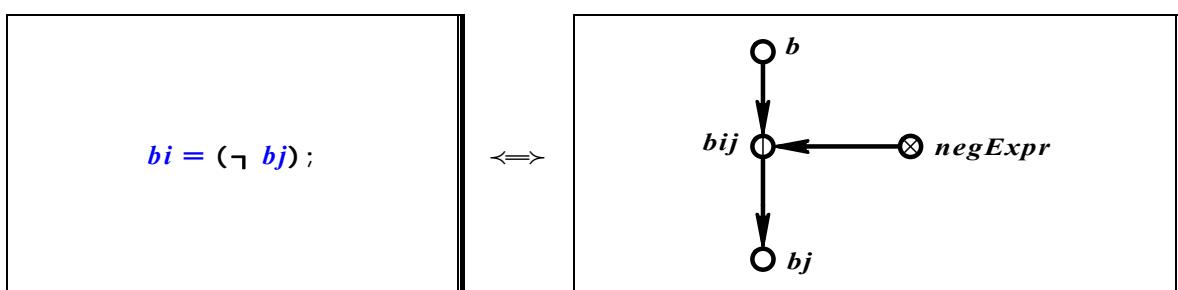
SCL-текст 5.4.4. Представление импликативной логической формулы



SCL-текст 5.4.5. Представление логической формулы об эквивалентности

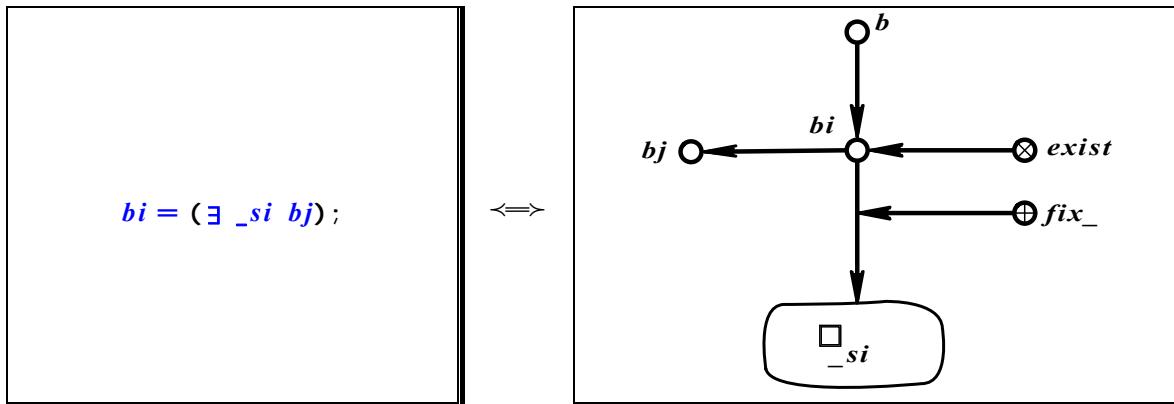


SCL-текст 5.4.6. Представление негативной логической формулы

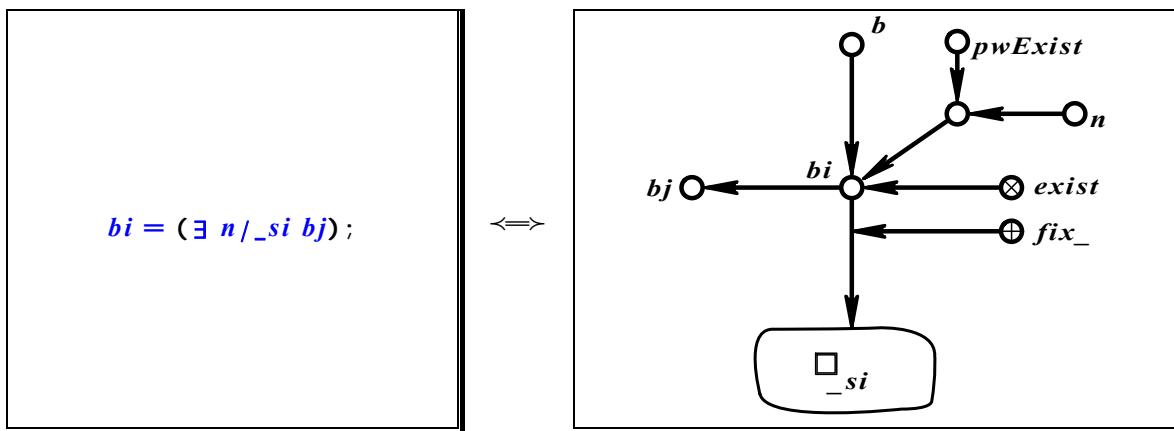


На нижеприведённых scl-текстах примеры записи кванторных логических формул.

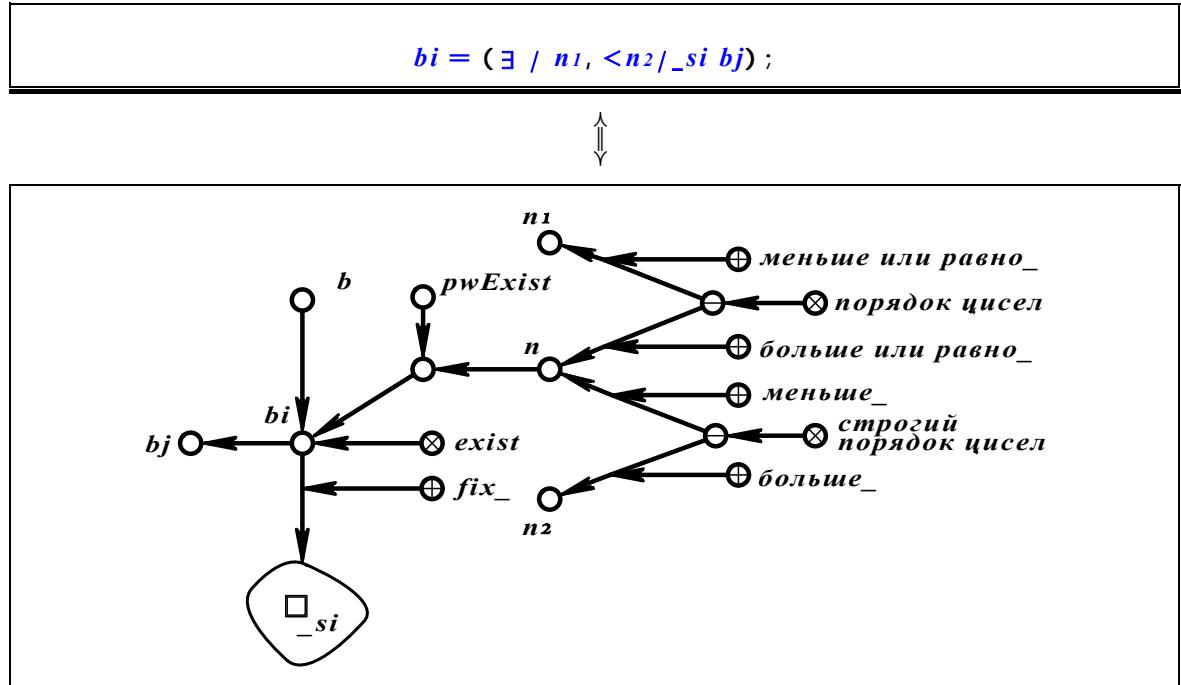
S C L - т е к с т 5 . 4 . 7 . Представление формулы о существовании



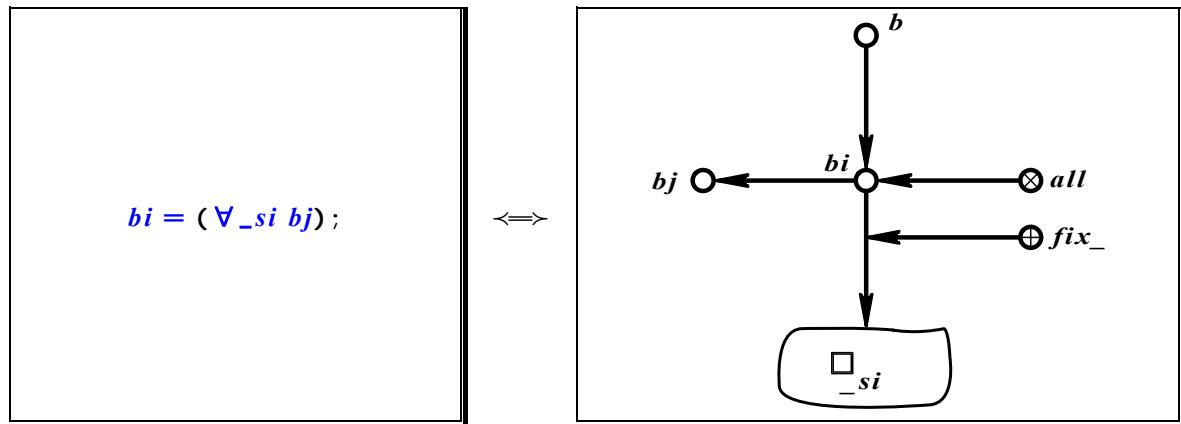
S C L - т е к с т 5 . 4 . 8 . Представление "числовой" формулы о существовании



SCL-текст 5.4.9. Представление "числовой" формулы о существовании



SCL-текст 5.4.10. Представление формулы о всеобщности

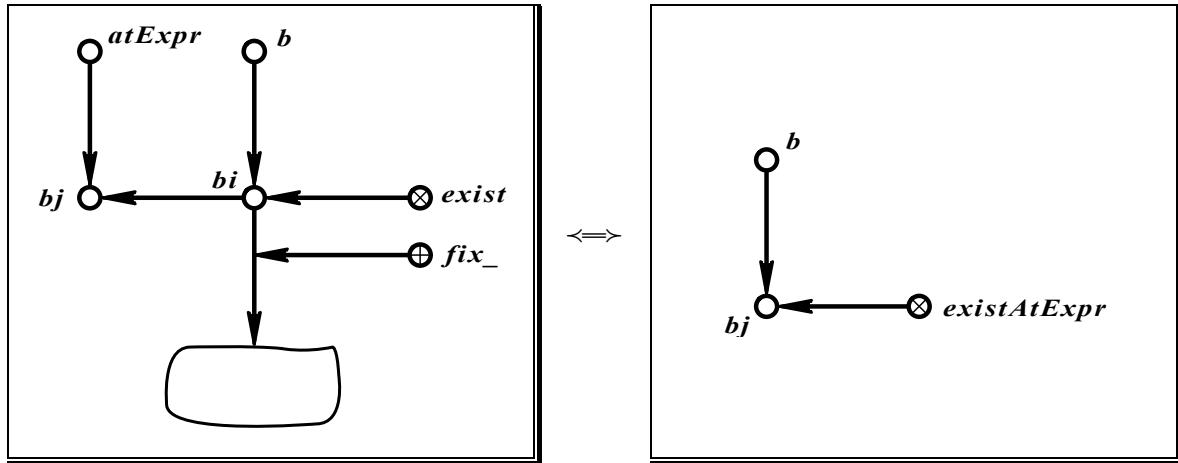


В случае, если квантор существования навешивается на атомарную формулу, а квантор всеобщности – на импликативную формулу, то для таких кванторных формул используется более лаконичный способ их записи и неявное указание связываемых переменных (см. scl-тексты 5.4.11 и 5.4.12).

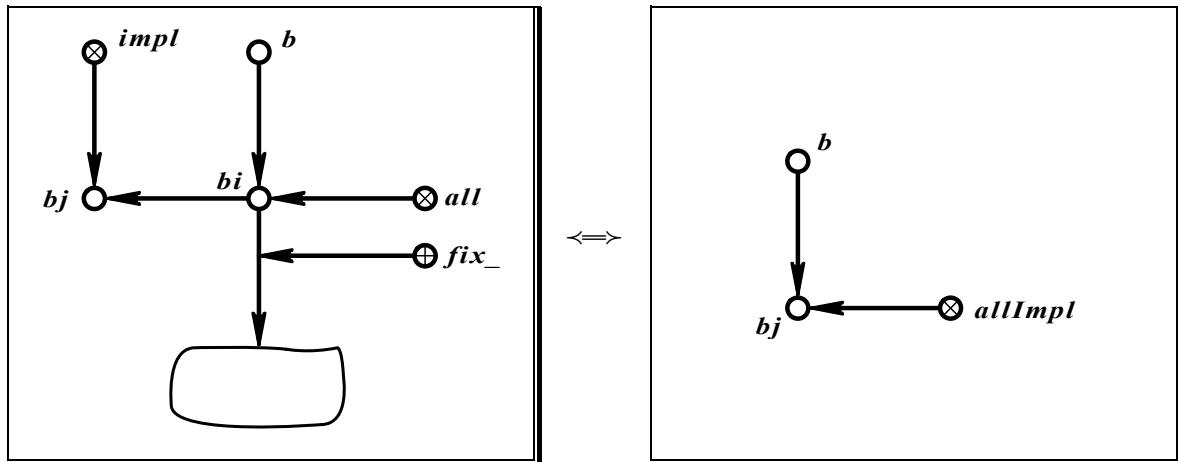
Правила неявного связывания переменных в кванторных формулах сводятся к следующему:

- переменные связываются сверху вниз (если какая-либо переменная связана в рамках некоторой кванторной формулы, то она считается связанный в рамках всех логических формул, которые входят в состав этой кванторной формулы);
- если формула **bj** отнесена к классу **existAtExpr**, то эта формула трактуется как кванторная формула о существовании, в которой связываются все переменные, которые являются элементами множества **bj** и которые не были связаны выше;
- если формула (**bi** —> **bj**) отнесена к классу **allImpl**, то эта формула трактуется как кванторная формула о всеобщности, в которой связываются все переменные, которые входят в состав как формулы **bi**, так и формулы **bj** и которые не были связаны выше.

SCL-текст 5.4.11. Приведение к более лаконичному виду формулы о существовании



SCL-текст 5.4.12. Приведение к более лаконичному виду формулы о всеобщности



Примечание. Поскольку каждый текст языка SCLg является текстом языка SCg и соответственно языка SC и поскольку язык SC, кроме графического варианта изображения текстов (языка SCg), имеет также абсолютно эквивалентный ему символьный вариант (язык SCs), то от sclg-текстов достаточно легко перейти к их эквивалентному символьному представлению на языке SCs. При этом такое символьное представление логических высказываний не следует путать с рассмотренным выше языком SCLs, который является результатом компромисса между реляционным логическим языком SCL и логическими языками классического типа.

1.5. Примеры записи логических формул на предложенных логических языках

Приведём несколько примеров записи логических высказываний:

- 1) на естественном языке;
- 2) на стилизованном естественном языке по приведенным выше "шаблонам";
- 3) на языке SCLs, максимально приближенном к классическому логическому языку;
- 4) на графическом языке SCLg.

При мер 5.5.1. Формальная запись следующих эквивалентных высказываний:

- 1) каждое (всякое, любое) классическое отношение является ориентированным;

2) если x есть классическое отношение, то x является также и ориентированным отношением.

Продолжение примера 5.5.1. Запись на стилизованном естественном языке с применением языка SCs для записи структуры атомарных логических формул:

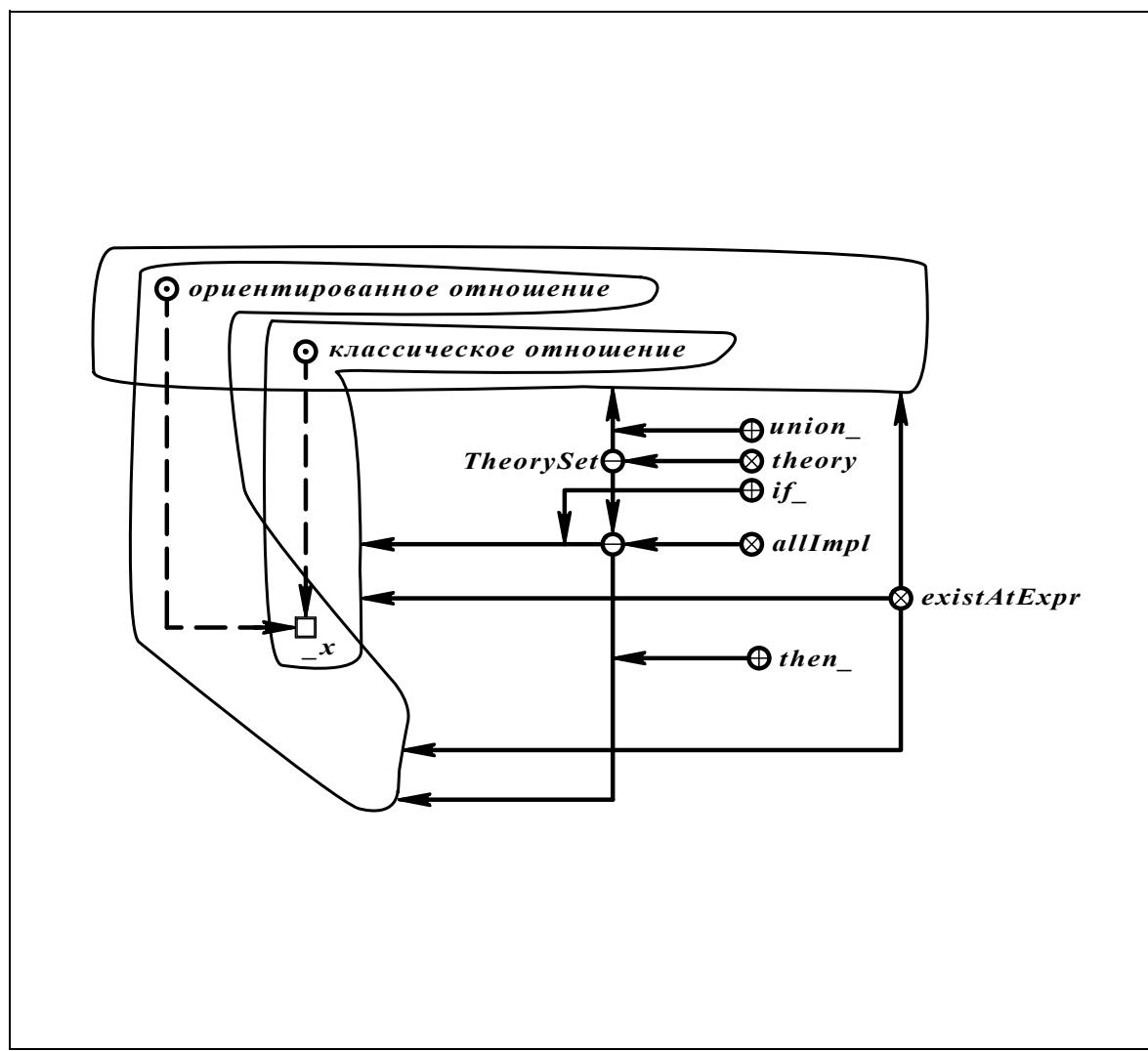
Для всех $_x$ имеет место **импликация** следующих формул:

- если существует $[_x \leftarrow \text{классическое отношение}]$,
- то существует $[_x \leftarrow \text{ориентированное отношение}]$.

Продолжение примера 5.5.1. Запись на языке SCLs:

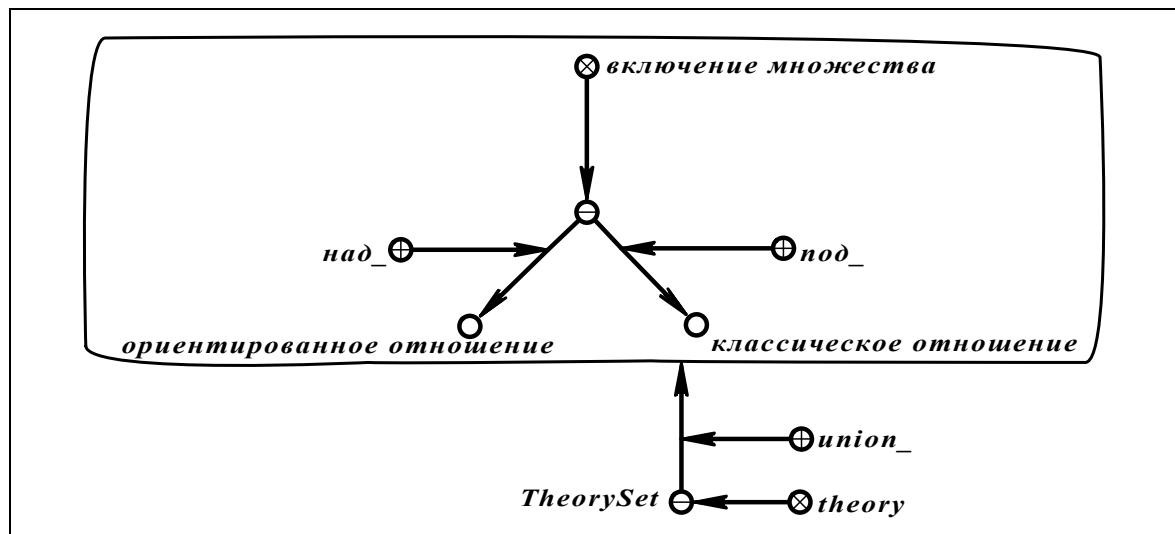
TheorySet → $\forall _x ([_x \leftarrow \text{классическое отношение}] \rightarrow [_x \leftarrow \text{ориентированное отношение}])$;

Продолжение примера 5.5.1. Запись на языке SCLg:



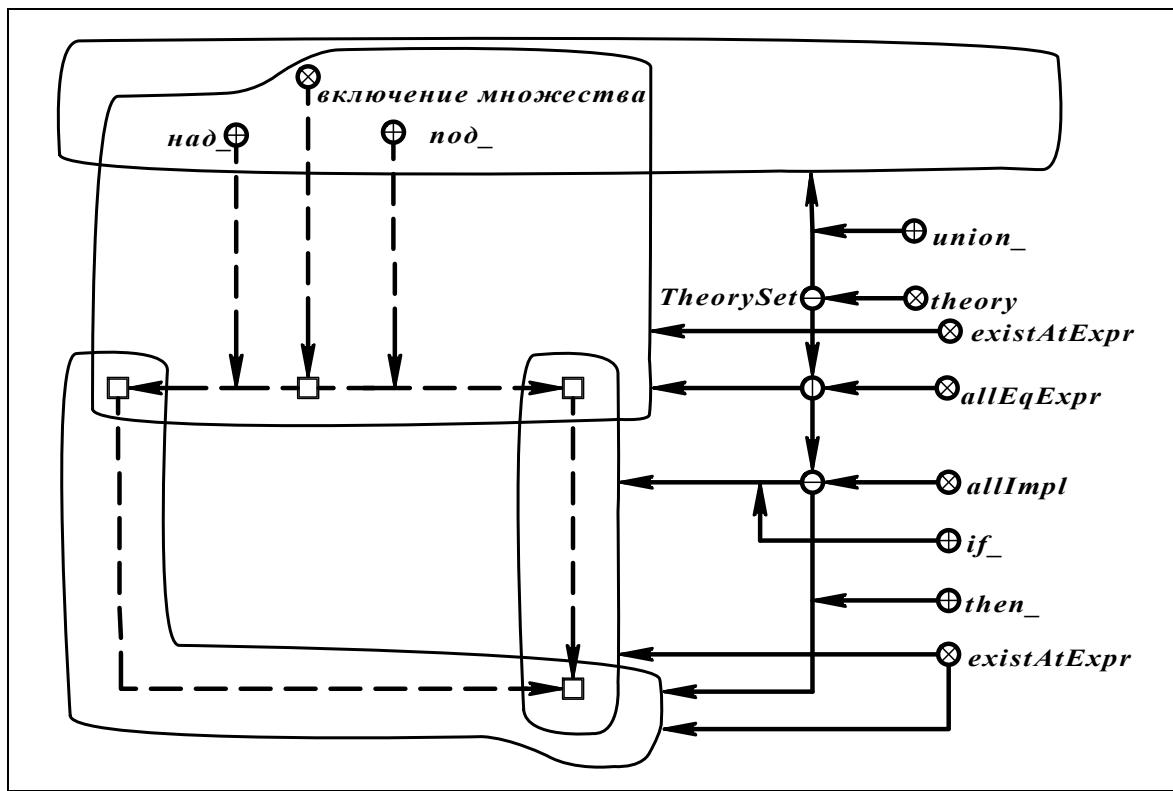
Пример 5.5.2. Высказывание, приведенное в примере 5.5.1, эквивалентно высказыванию о том, что множество “*классическое отношение*” является нестрогим подмножеством по отношению ко множеству “*ориентированное отношение*”.

Продолжение примера 5.5.2. Запись на языке SCLg (см. также пункт 3.3.11):



Пример 5.5.3. Формальная запись высказывания, которое следует из определения понятия “*включение множества*” (см. пункт 3.3.11) и из которого следует эквивалентность высказывания, приведенного в примере 5.5.1, и высказывания, приведенного в примере 5.5.2.

Продолжение примера 5.5.3. Запись на языке SCLg:



Пример 5.5.4. Запись высказывания:

Не существует ни одного классического отношения, не являющегося ориентированным.

Продолжение примера 5.5.4. Запись на стилизованном естественном языке:

Не существует конструкции вида:

[_x << **классическое отношение** ;] ,

у которой:

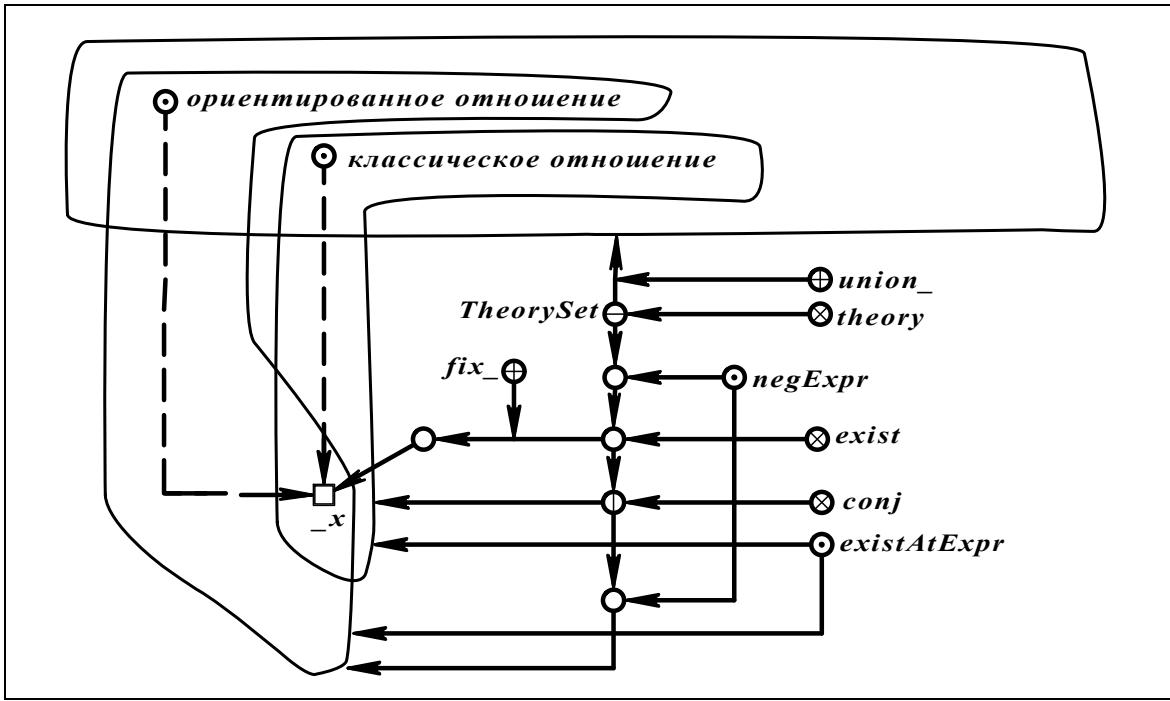
не существует конструкции вида:

[_x << **ориентированное отношение** ;] .

Продолжение примера 5.5.4. Запись на языке SCLs:

TheorySet_ → $\neg \exists _x ([_x \ll \text{классическое отношение} ;] \ \& \ \neg [_x \ll \text{ориентированное отношение} ;]) ;$

Продолжение примера 5.5.4. Запись на языке SCLg:



Пример 5.5.5. Очевидно, что высказывание, приведенное в примере 5.5.1, и высказывание, приведенное в примере 5.5.4, являются эквивалентными. Очевидно также, что такого рода эквивалентность имеет место для любых логических формул сходной структуры:

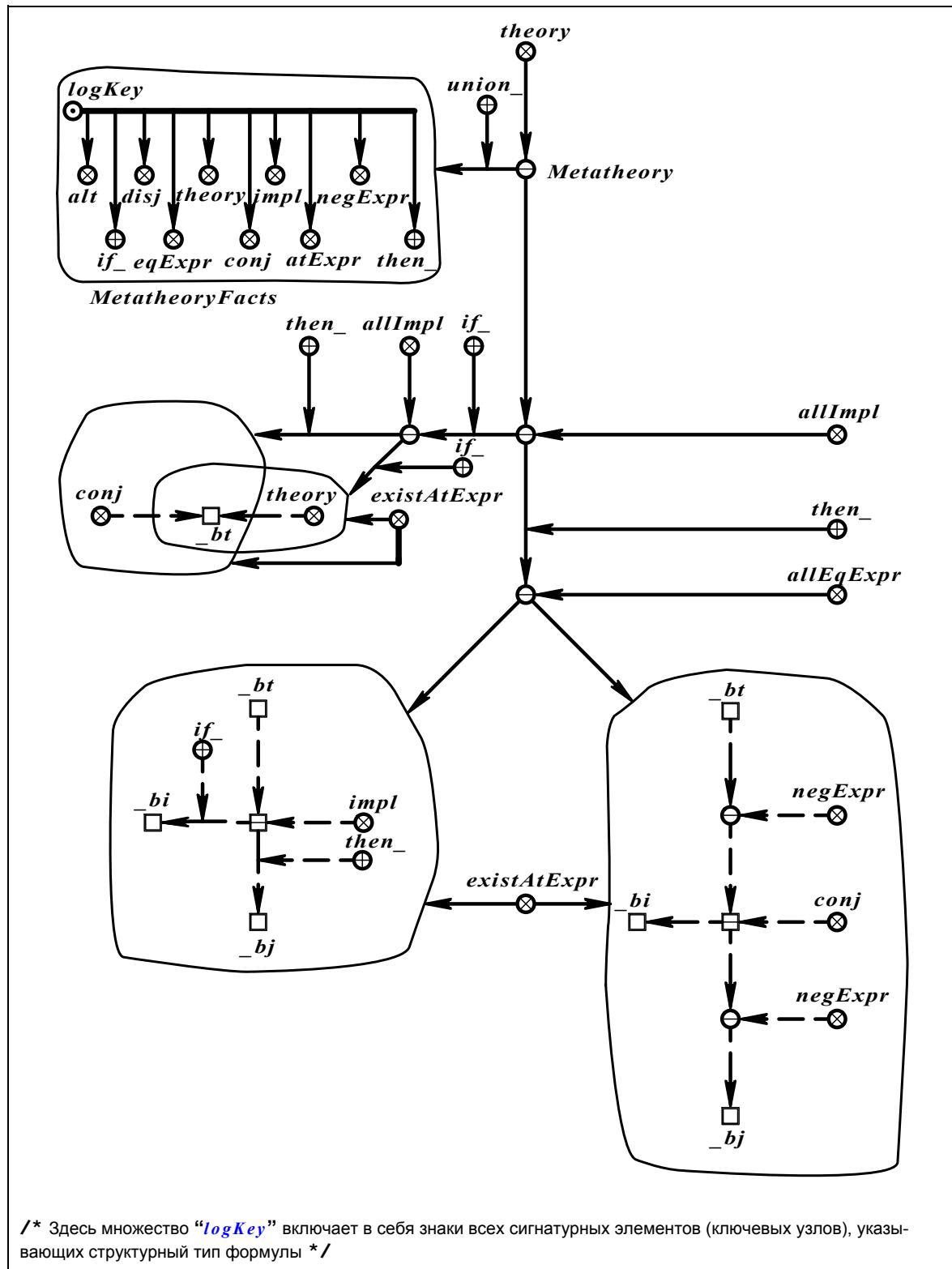
$$(bi \rightarrow bj)$$

\Leftrightarrow

$$\neg (bi \& \neg bj)$$

Рассмотрим то, как эта закономерность записывается в рамках формальной метатеории, для которой описывается совокупность всевозможных формальных теорий, представленных на языке SCL. Существенным здесь является то, что сама метатеория может быть представлена также на языке SCL, поскольку любая формальная теория, представленная на языке SCL, представляет собой реляционную структуру специального вида, а сам язык SCL ориентирован на описание произвольных реляционных структур. Таким образом, единство языка и метаязыка в предлагаемых графодинамических моделях проявляется не только на уровне языка SCB, но и на уровне языка SCL.

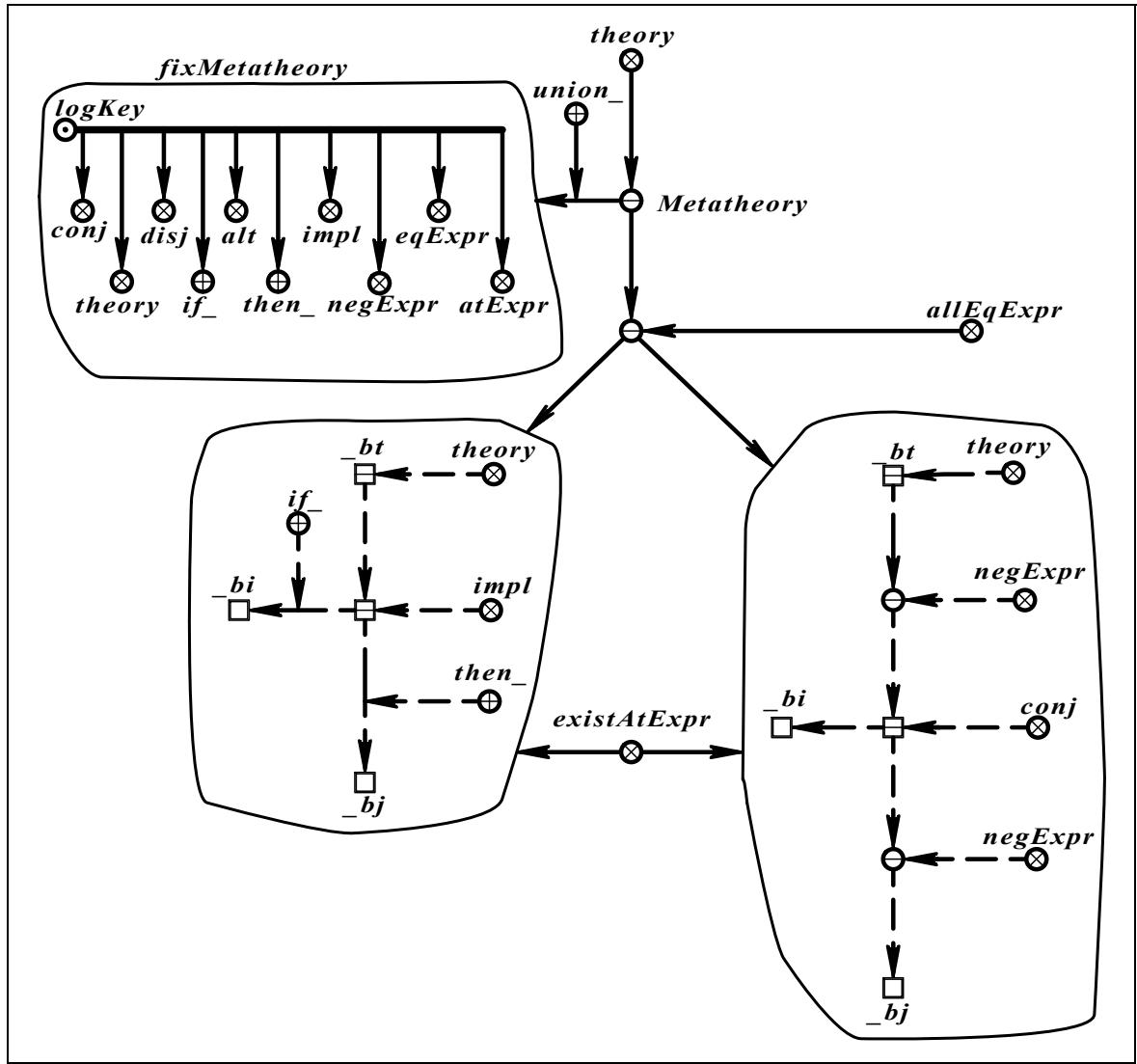
Продолжение примера 5.5.5. Запись на языке SCLg (вариант 1):



/* Здесь множество “*logKey*” включает в себя знаки всех сигнатурных элементов (ключевых узлов), указывающих структурный тип формулы */

Очевидно, что приведенный *sclg*-текст можно переписать по-другому – в виде следующих двух высказываний об эквивалентности, входящих в состав формальной метатеории.

Продолжение примера 5.5.5. Запись на языке SCLg (вариант 2):



Пример 5.5.6. Формальная запись следующих эквивалентных высказываний:

- 1) некоторые тернарные отношения являются классическими;
- 2) существуют тернарные отношения, являющиеся классическими;
- 3) существует по крайней мере одно отношение являющееся как тернарным, так и классическим.

Продолжение примера 5.5.6. Запись на стилизованном естественном языке и SCs:

Существует конструкция вида:

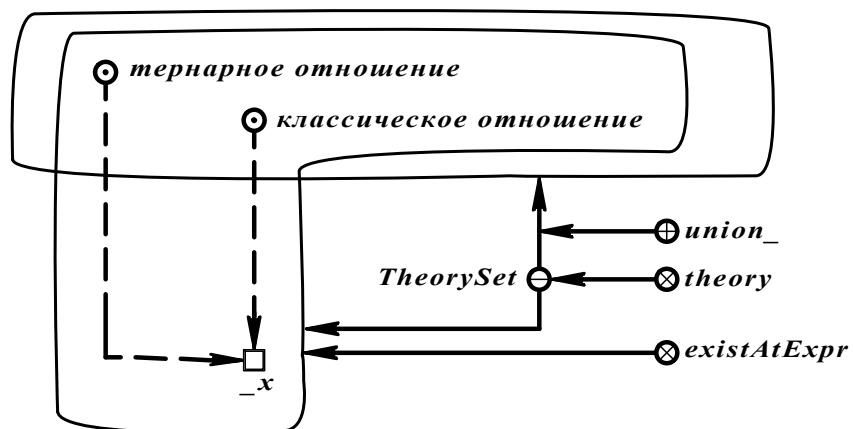
[_x << **тернарное отношение** , **классическое отношение** ;] .

Примечание. Множество, элементы которого удовлетворяют данному условию, есть пересечение множества “**тернарное отношение**” и множества “**классическое отношение**”.

П р о д о л ж е н и е п р и м е р а 5 . 5 . 6 . Запись на языке SCLs:

TheorySet → $\exists _x$
 $[_x \Leftarrow \text{тернарное отношение} ; \text{классическое отношение} ;]$

П р о д о л ж е н и е п р и м е р а 5 . 5 . 6 . Запись на SCLg:



П р и м е р 5 . 5 . 7 . Формальная запись следующих эквивалентных высказываний:

- 1) существуют отношения, являющиеся тернарными, но не являющиеся классическими;
- 2) существует по крайней мере одно отношение, которое относится к классу тернарных отношений, но не относится к классу классических отношений.

П р о д о л ж е н и е п р и м е р а 5 . 5 . 7 . Запись на стилизованном естественном языке и SCs:

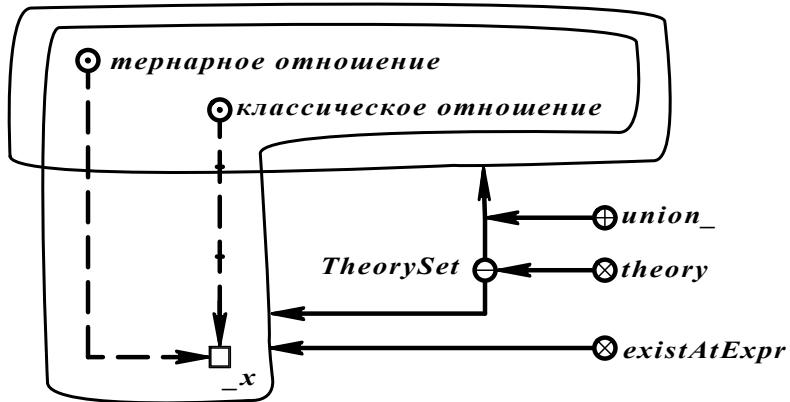
Существует конструкция вида:
 $[_x \Leftarrow \text{тернарное отношение} ; _x \Leftarrow \text{классическое отношение} ;]$.

Примечание. Множество, элементы которого удовлетворяют данному условию, есть результат вычитания множества “классическое отношение” из множества “тернарное отношение”.

П р о д о л ж е н и е п р и м е р а 5 . 5 . 7 . Запись на языке SCLs:

TheorySet → $\exists _x [_x \Leftarrow \text{тернарное отношение} ; _x \Leftarrow \text{классическое отношение} ;]$

Продолжение примера 5.5.7. Запись на языке SCLg:



Пример 5.5.8. Эквивалентная запись высказывания, приведенного в примере 5.5.7.

Продолжение примера 5.5.8. Запись на стилизованном естественном языке с использованием языка SCs:

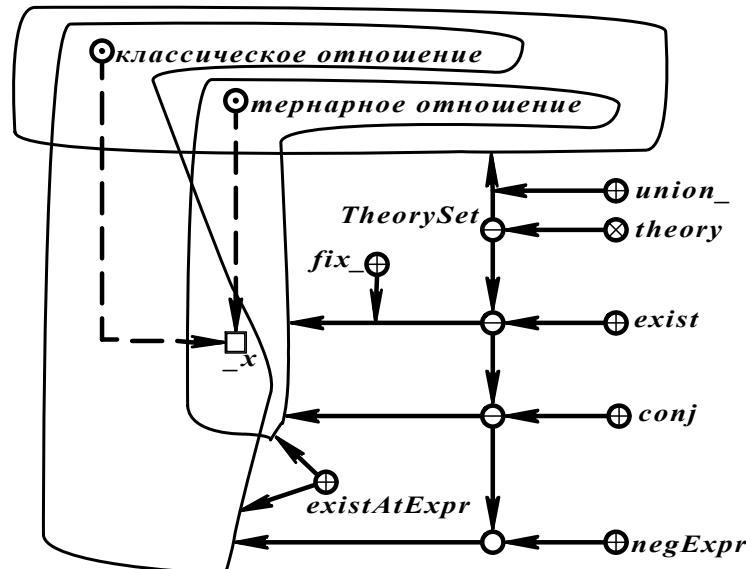
Существует конструкция вида:
`[_x << тернарное отношение ;]`

для которой
 не существует конструкции вида:
`[_x << классическое отношение ;] .`

Продолжение примера 5.5.8. Запись на языке SCLs:

`TheorySet → ∃ _x ([_x << тернарное отношение ;]
 & ¬ [_x << классическое отношение ;]) ;`

П р о д о л ж е н и е п р и м е р а 5 . 5 . 8 . Запись на языке SCLg:



Примечание. Логическая структура данного высказывания отличается от структуры высказывания, приведенного в примере 5.5.4, только тем, что здесь конъюнктивное высказывание является позитивным.

П р и м е р 5 . 5 . 9 . Формальная запись следующих эквивалентных высказываний:

- 1) существуют отношения, не являющиеся ни классическими, ни тернарными;
- 2) существуют отношения, каждое из которых не является классическим и не является тернарным.

П р о д о л ж е н и е п р и м е р а 5 . 5 . 9 . Запись на стилизованном естественном языке с использованием языка SCs:

Существует конструкция вида:

[_x <<— **отношение** ;]

_x <<— **классическое отношение** , **тернарное отношение** ;] ;

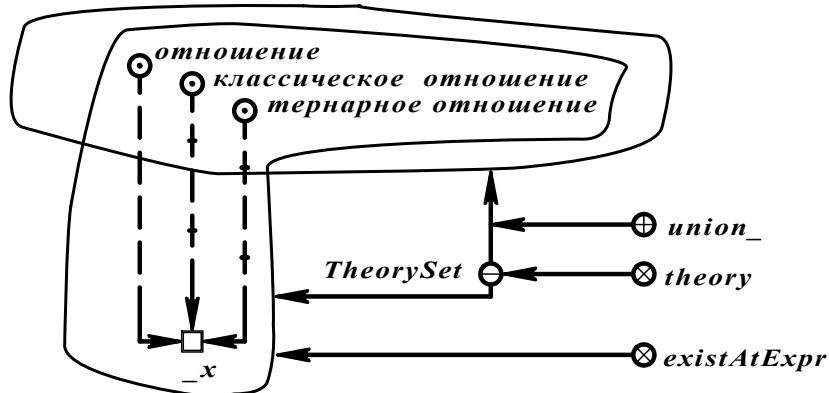
Логическая структура этого высказывания аналогична высказыванию, рассмотренному в примере 5.5.7. Отличие здесь заключается только в количестве негативных дуг.

Примечание. Множество, элементы которого удовлетворяют данному условию, представляет собой результат вычитания множества “(**классическое отношение** \sqcap **тернарное отношение**)” из множества “**отношение**”.

Продолжение примера 5.5.9. Запись на языке SCLs:

```
TheorySet → ∃ _x [ _x <-- отношение ; _x <-- классическое отношение ,  
тернарное отношение ; ] ;
```

Продолжение примера 5.5.9. Запись на языке SCLd:



Примечание. В соответствии с правилом замены негативной дуги на негативное атомарную формулу (см. пример 5.6.2) от высказывания, приведенного в примере 5.5.9, легко перейти к целому ряду эквивалентных высказываний.

Пример 5.5.10. Формальная запись следующих эквивалентных высказываний:

- 1) не существует ни одного классического отношения, которое являлось бы булевом;
- 2) не существует ни одного булеана, который был бы классическим отношением.

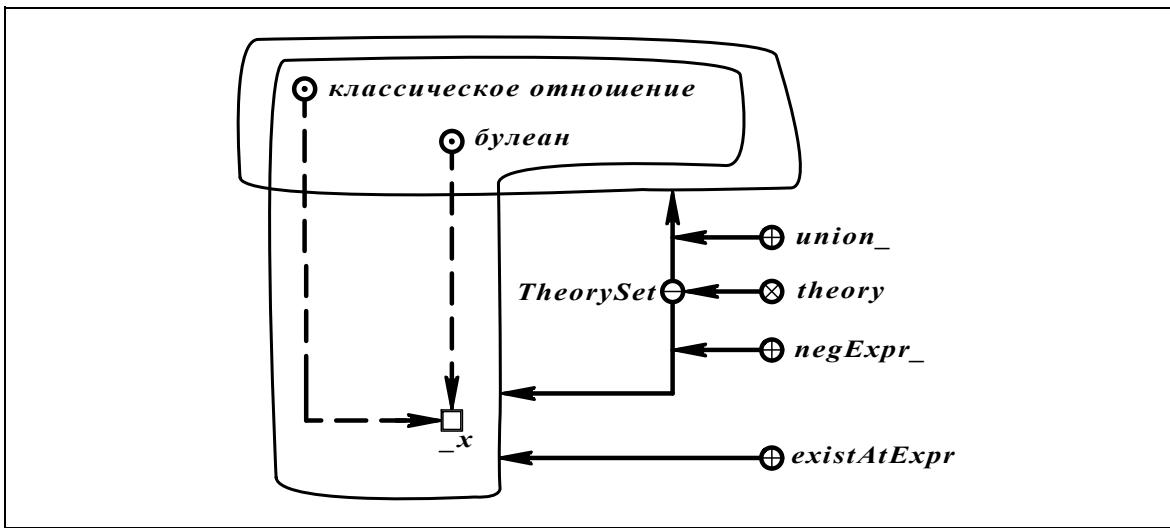
Продолжение примера 5.5.10. Запись на стилизованном естественном языке с использованием языка SCs:

```
Не существует конструкция вида:  
[ _x <-- классическое отношение , булев ; ]
```

Продолжение примера 5.5.10. Запись на языке SCLs:

```
TheorySet → ¬ ∃ _x [ _x <-- классическое отношение , булев ; ] ;
```

Продолжение примера 5.5.10. Запись на языке SCLg:



Примечание. От рассматриваемого высказывания можно перейти к целому ряду эквивалентных высказываний в соответствии с правилом преобразования негативных конъюнктивных формул в импликативные (см. пример 5.5.5). Примерами таких эквивалентных высказываний являются:

- 1) каждое классическое отношение не является булевом;
- 2) каждый булев не является классическим отношением.

Заметим при этом, что атомарное высказывание в языке SCL является вырожденным случаем конъюнктивного высказывания и может быть представлено в виде эквивалентной конъюнкции атомарных высказываний.

Завершая рассмотрение высказывания, приведенного в примере 5.5.10, заметим, что теоретико-множественная трактовка этого высказывания заключается в том, что пересечение множества “классическое отношение” и множества “булеан” не содержит элементов, т.е. является пустым множеством.

Пример 5.5.11. Варианты записи высказываний на естественном языке:

- 1) некоторые отношения в состав своей области определения включают некоторые тернарные классические отношения (но, возможно, не все тернарные классические отношения и, возможно, не только тернарные классические отношения).

Примечание. Примером такого отношения является метаотношение “функциональная зависимость”, поскольку:

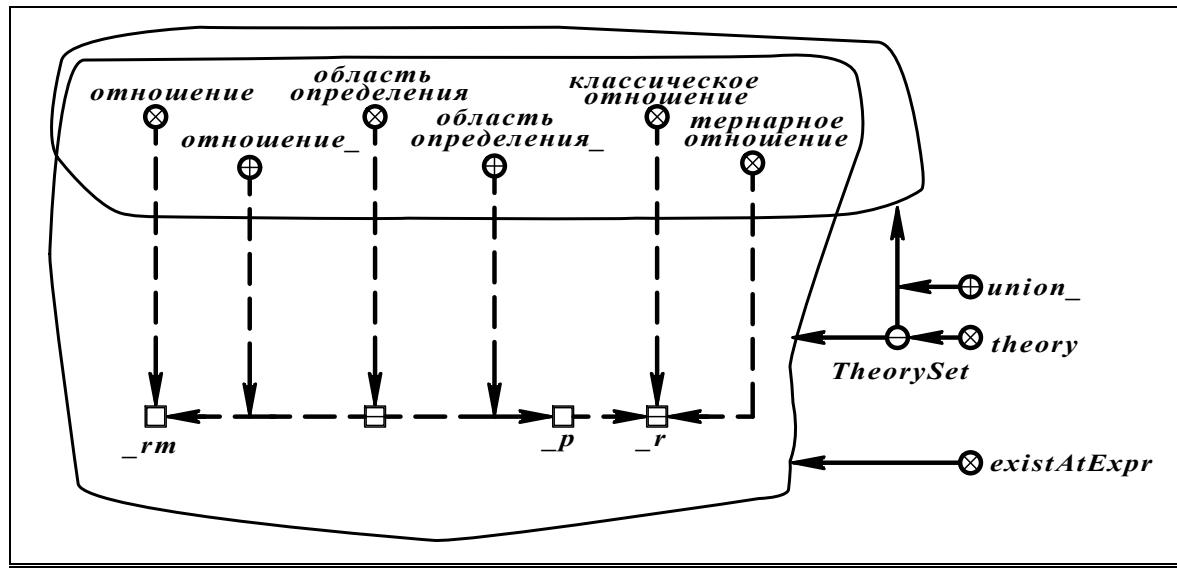
- не все тернарные классические отношения входят в область определения этого метаотношения, а только те, которые имеют функциональную зависимость;
- кроме некоторых тернарных классических отношений, в область определения метаотношения “функциональная зависимость” входят также некоторые неклассические отношения, некоторые бинарные отношения, некоторые четырехарные отношения и т.д.

- 2) существуют отношения *rm* и тернарное классическое отношение *r* такие, что *r* является одним из элементов области определения отношения *rm*.

Продолжение примера 5.5.11. Запись на стилизованном естественном языке и с использованием языка SCs:

<p>Существует конструкция вида:</p> <p>[<i>rm</i> << <i>отношение</i> ; <i>p</i> —> <i>r</i> ; <i>область определения</i> —>> (<i>отношение_</i> :: <i>rm</i> , <i>область определения_</i> :: <i>p</i> ·) ; <i>rm</i> << <i>классическое отношение</i> , <i>тернарное отношение</i> ;] .</p>

Продолжение примера 5.5.11. Запись на SCLg:



П р и м е р 5 . 5 . 1 2 . Варианты записи высказывания на естественном языке:

- 1) некоторые отношения в состав своей области определения включают все бинарные ориентированные отношения, но возможно не только их.

Примечание. Примером такого отношения "**соответствие**", в область определения которого кроме **всевозможных** (для любого бинарного ориентированного отношения можно построить семейство кортежей метаотношения "**соответствие**") бинарных ориентированных отношений входят и другие объекты – множества, не являющиеся бинарными ориентированными отношениями.

- 2) существует по крайней мере одно отношение такое, что каждое бинарное ориентированное отношение входит в состав его области определения.

Продолжение примера 5.5.12. Запись на стилизованном естественном языке и с использованием языка SCs:

Существует конструкция вида:

[*r* ʌn *отношение* ;]

у которой:

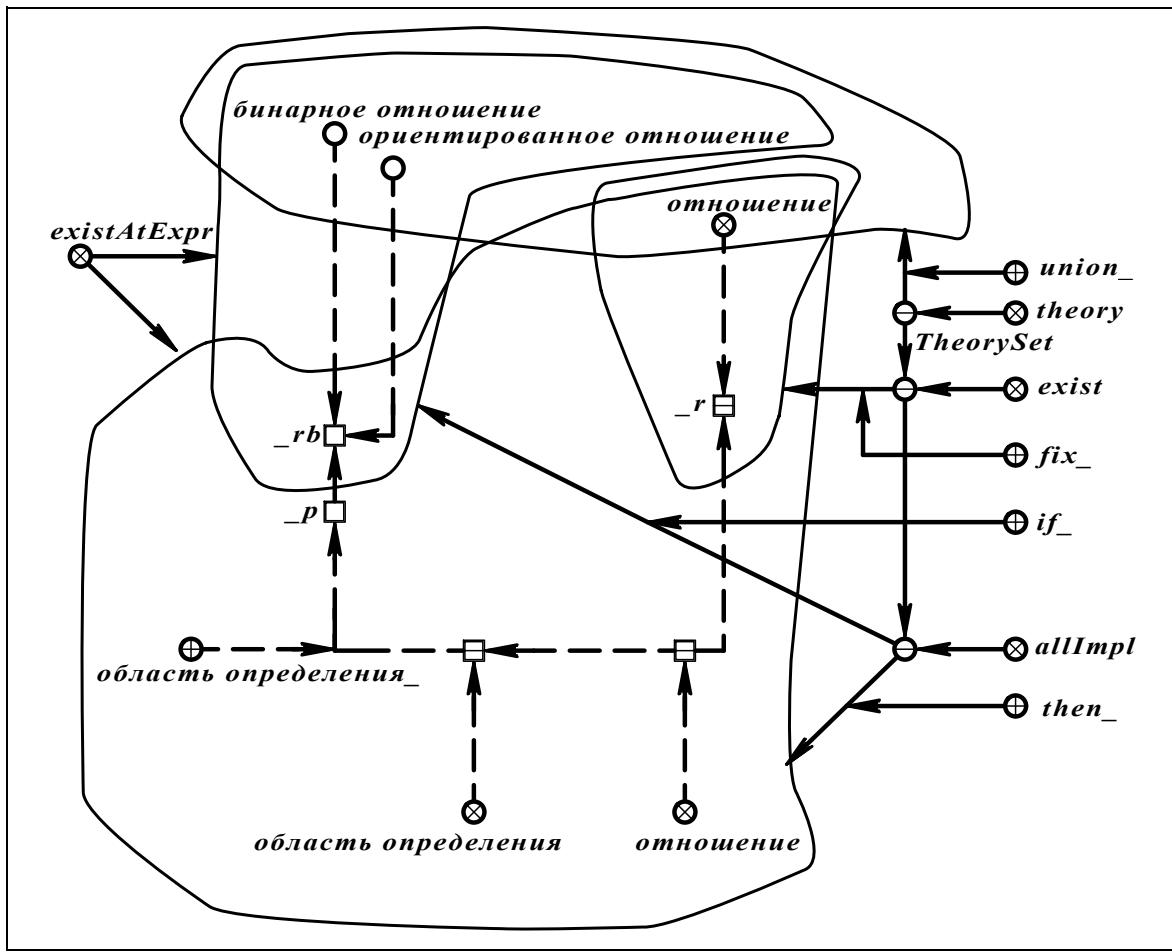
для каждого r имеет место импликация следующих логических формул:

- если существует конструкция вида:

[*rb* \Leftarrow бинарное отношение , ориентированное отношение ;]

- то существует конструкция вида:

П р о д о л ж е н и е п р и м е р а 5 . 5 . 1 2 . Запись на SCLg:



П р и м е р 5 . 5 . 1 3 . Формальная запись следующих эквивалентных высказываний:

- 1) некоторые отношения в состав своей области определения включают все бинарные ориентированные отношения и только их;
- 2) существует по крайней мере одно отношение, у которого:
 - все бинарные ориентированные отношения являются элементами его области определения,
 - и наоборот все элементы его области определения являются бинарными ориентированными отношениями.

Примечание. Примерами таких отношений являются “транзитивное замыкание” и “произведение бинарных отношений”.

Примечание. Из определения понятия равенства множеств (см. пункт 3.3.11) следует, что область определения отношения, которое удовлетворяет сформулированным выше требованиям, является множеством, равным множеству всевозможных бинарных ориентированных отношений.

Продолжение примера 5.5.13. Запись на стилизованном естественном языке и с использованием языка SCs:

Существует конструкция вида:

[_r << **отношение** ;
область определения →> (· **отношение_::_r** , область определения_::_p ·)]

у которой для каждого _rb имеет место **эквивалентность** следующих логических формул:

- **существует** конструкция вида:

[_rb << **бинарное отношение** , **ориентированное отношение** ;]

- **существует** конструкция вида:

[_rb << _p ;]

Пример 5.5.14. Формальная запись следующих эквивалентных высказываний:

- 1) не существует ни одного арифметического отношения, которое бы включало какие-либо геометрические фигуры в состав своей области определения;
- 2) не существует ни одной геометрической фигуры, которая была бы элементом области определения какого-либо арифметического отношения.

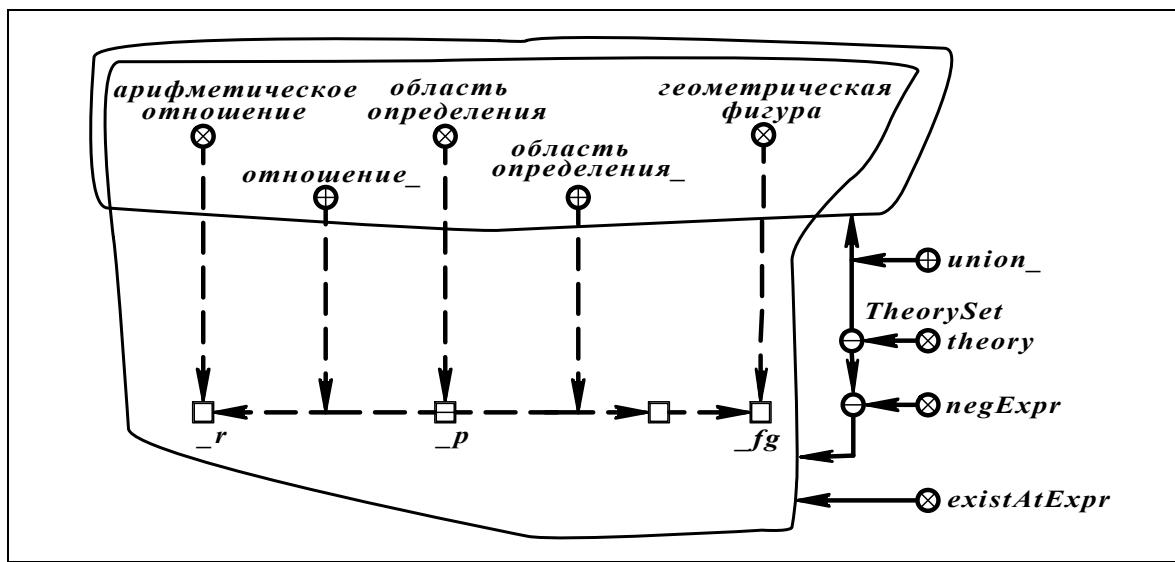
Продолжение примера 5.5.14. Запись на стилизованном естественном языке и с использованием языка SCs:

Не существует конструкции вида:

[_r << **арифметическое отношение** ;
область определения →> (· **отношение_::_r** , область определения_::_p ·) ;
_p →> _fg << **геометрическая фигура** ;]

Примечание. Логическая структура данного высказывания отличается от логической структуры высказывания, приведенного в примере 5.5.11, тем, что в первом случае высказывание о существовании является негативным, а во втором – позитивным.

Продолжение примера 5.5.14. Запись высказывания на языке SCLg:



Пример 5.5.15. Определения метаотношения “**унарная проекция**” (см. пункт 3.3.13).

Продолжение примера 5.5.15. Запись на языке SCLs:

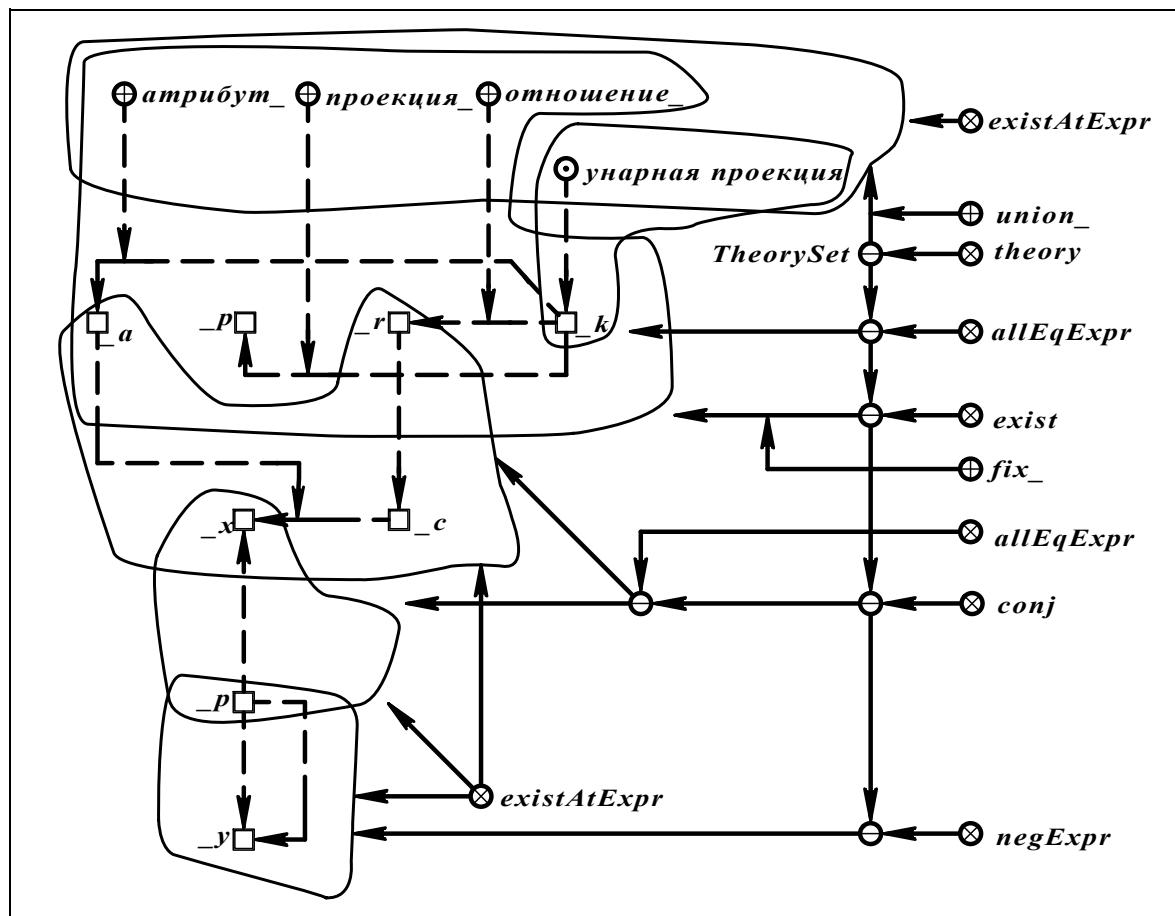
TheorySet →

```

 $\forall \_k ([\_k \Leftarrow\!\!-\!\!< \text{унарная проекция} ; ])$ 
 $\Leftrightarrow \exists \_r , \_p , \_a ([\_k == (\cdot \text{ отношение}\_:: \_r , \text{ проекция}\_:: \_p ,$ 
 $\text{ атрибут}\_:: \_a \cdot)]$ 
 $\& \forall \_x (\exists \\_c [\_r \rightarrow\!\!-\!\!> \_c \rightarrow\!\!-\!\!> \_a :: \_x]$ 
 $\Leftrightarrow [\_p \rightarrow\!\!-\!\!> \_x ; ])$ 
 $\& \neg \forall \_y [\_p \rightarrow \_y , \_y ; ]$ 
 $)$ 
)

```

Продолжение примера 5.5.15. Запись на языке SCLg:



Пример 5.5.16. Определение отрезка

Продолжение примера 5.5.16. Варианты записи на естественном языке:

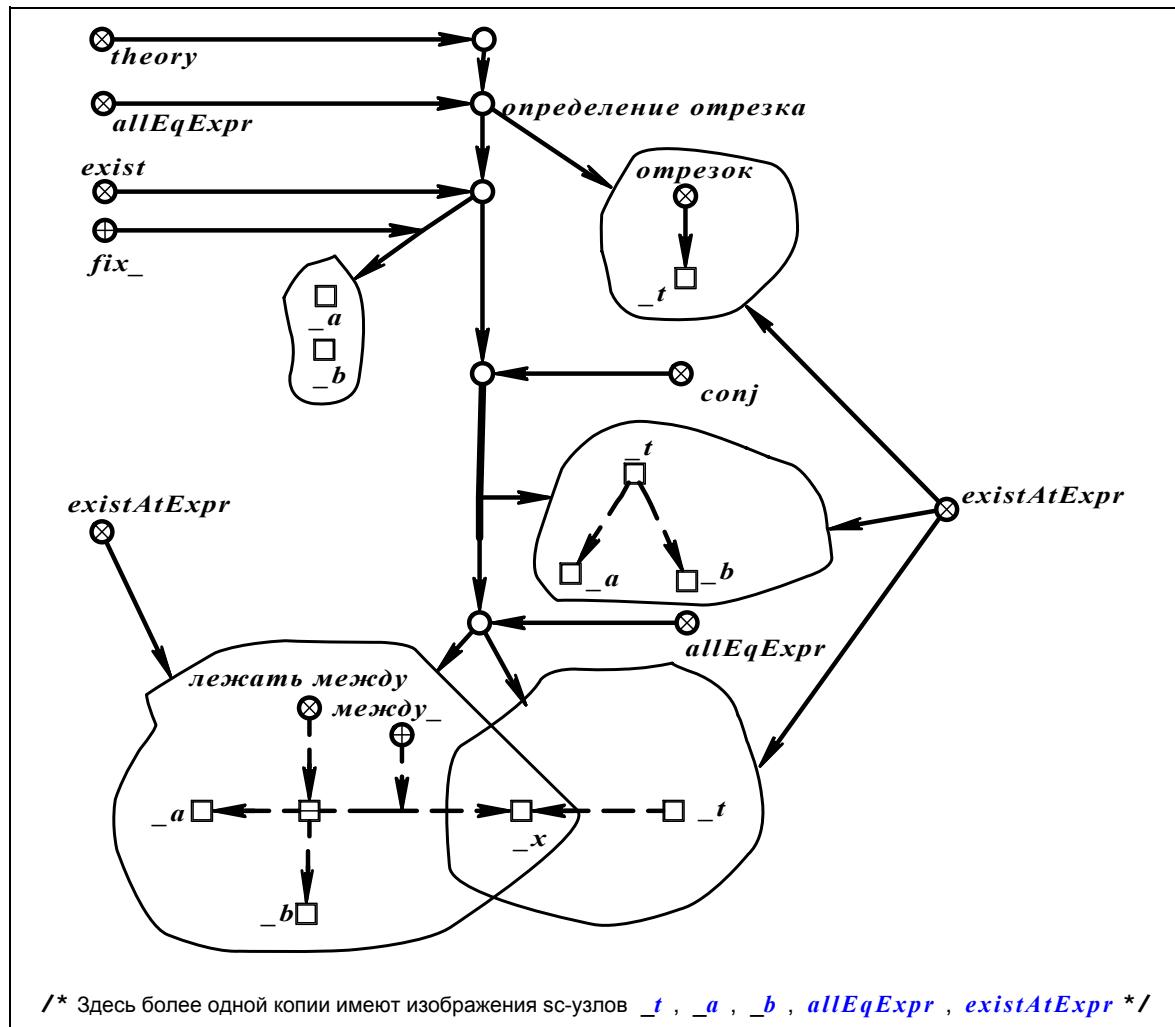
- 1) отрезок – это множество всех тех и только тех точек, которые лежат между двумя заданными.
- 2) будем говорить, что *t* есть отрезок, в том и только в том случае, если существуют *a* и *b* такие, что:
 - *a* и *b* являются элементами множества *t*;
 - для каждого *x* справедливо следующее:
 - если *x* есть элемент множества *t*, не совпадающий с *a* и *b*,
 - то *x* лежит между *a* и *b* и наоборот.

П р о д о л ж е н и е п р и м е р а 5 . 5 . 1 6 . Запись определения отрезка на стилизованном естественном языке и языке SCs:

Для всех значений переменной *t* имеет место эквивалентность следующих логических формул:

- Существует конструкция вида:
`[отрезок —>> t ;]`
- Существует конструкция вида:
`[t —>> a , b ;]`
 для которой имеет место эквивалентность следующих логических формул:
 - существует конструкция вида:
`[t —>> x , a , b ;] ; /* включение b в состав этой конструкции переменных a и b означает то, что значение переменной x не должно совпадать со значениями переменных a и b */`
 - существует конструкция вида:
`[лежать между —>> (· a , между : x , b ·) ;]`

Продолжение примера 5.5.16. Запись определения отрезка на языке SCLg:



Продолжение примера 5.5.16. Запись определения отрезка на языке SCLs:

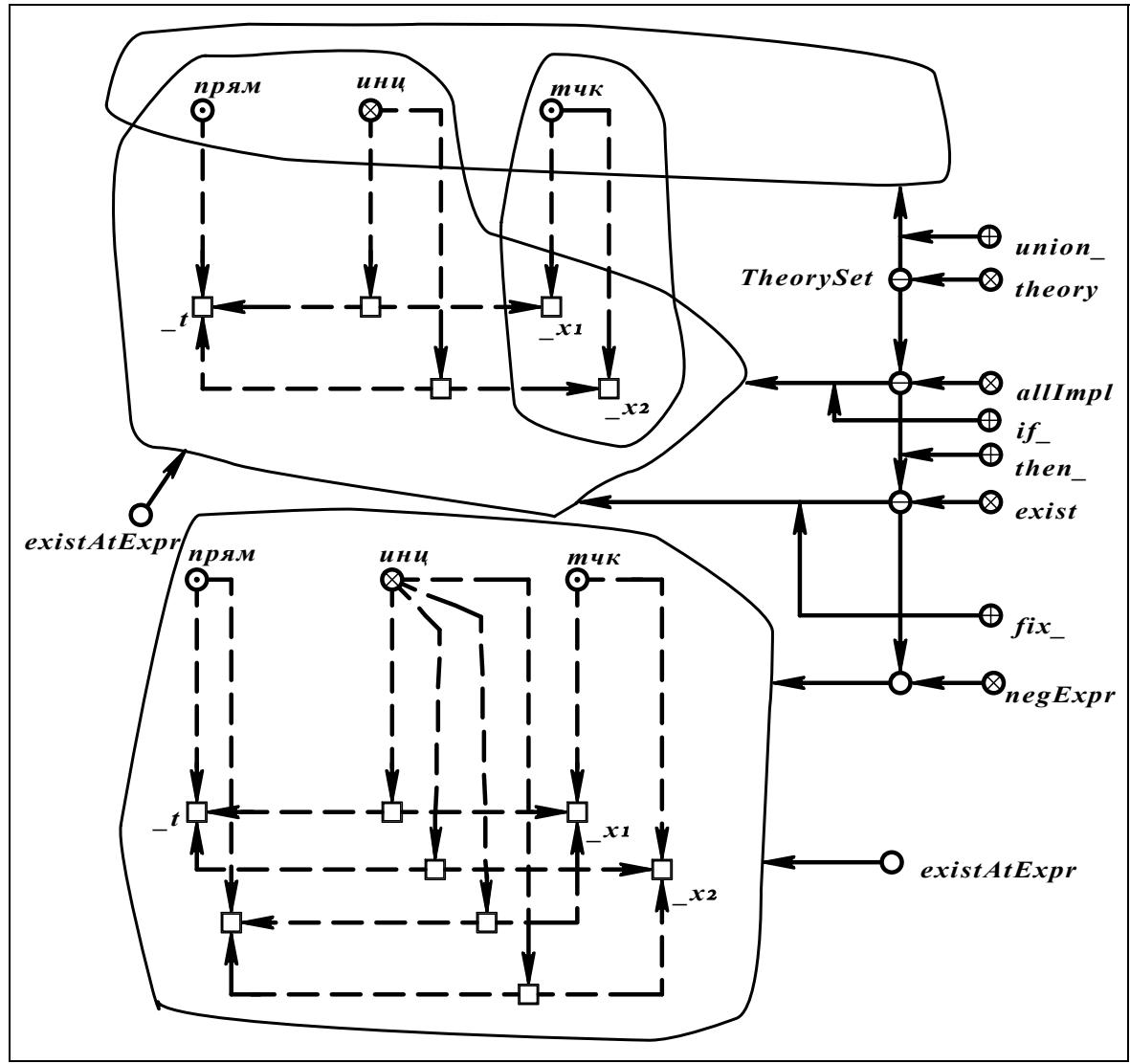
```

TheoryGeo ->
  ∀ _t ( [ отрезок ->> _t ; ]
  <=> ∃ _a , _b ( [ _t ->> _a , _b ; ]
  & ∀ _x ( [ _t ->> _x , _a , _b ; ]
  <=> [ лежать между -> (· _a , между_ _x , _b ·) ;
  )
  )
)
;
```

Пример 5.5.17. Запись аксиомы геометрии Евклида о существовании прямой, инцидентной двум точкам:

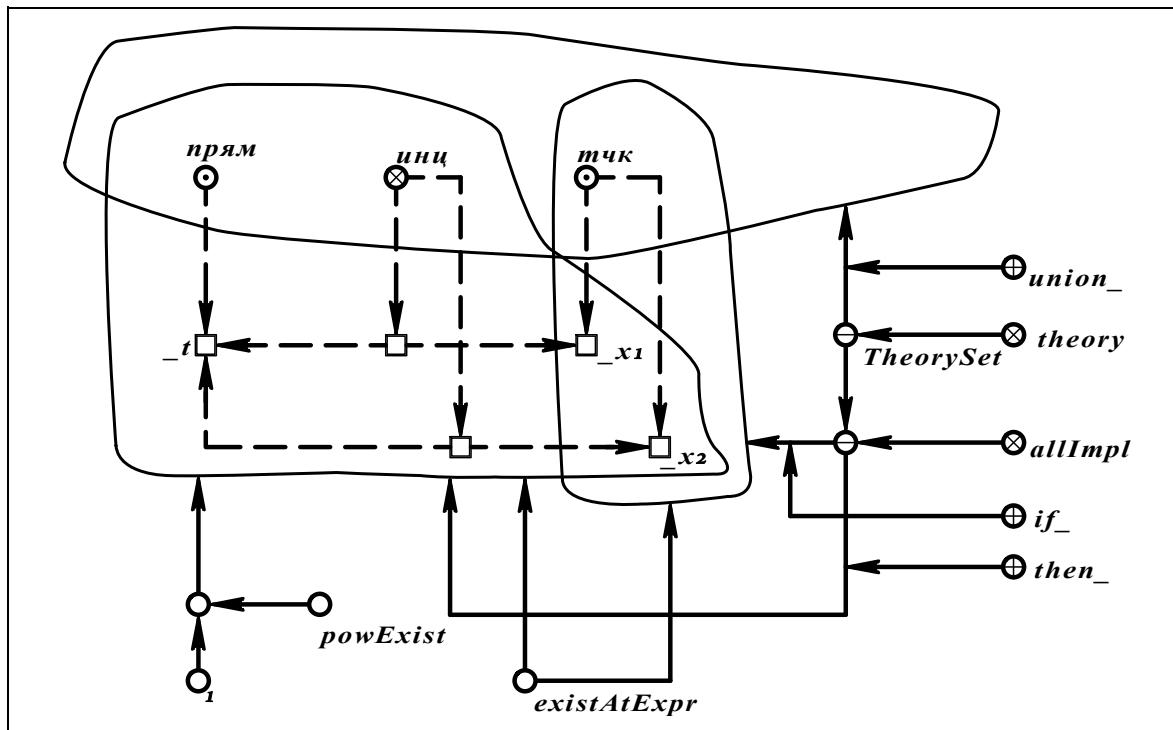
Для каждой пары точек существует одна и только одна инцидентная им прямая.

Продолжение примера 5.5.17. Запись аксиомы на языке SCLg:



Примечание. Квантор существования и единственность в SCLg задается явно (ключевой узел **exist**), но его можно свести к неявно задаваемому квантору существования. Приведем такого рода запись рассматриваемой аксиомы.

Продолжение примера 5.5.17. Запись аксиомы на языке SCLg (вариант 2)



Продолжение примера 5.5.17. Запись аксиомы на языке SCLs (вариант 2):

```
TheoryGeo -> ∀ _x1, _x2 ( [ тчк ->> _x1 , _x2 ; ]
    -> ∃ ! _t [ прям ->> _t ;
        инц ->> { _t , _x1 ; · } , { _t , _x2 ; · } ; ]
    ) ;
```

Пример 5.5.18. Определение множества всех множеств, которые не являются элементами самих себя [100; 99] ([Вilenkin N.Y. 1969 кни-РасскоМ](#); [Вilenkin N.Y. 1980 кни-СоврeOШКМ](#)).

Такое множество в пункте 3.1 мы называли множеством всевозможных нерефлексивных множеств и поставили ему в соответствии идентификатор “[нерефлексивное множество](#)”. В некоторых работах, например [100] ([Вilenkin N.Y. 1969 кни-РасскоМ](#)), нерефлексивные множества называют ординарными, а рефлексивные соответственно – экстраординарными.

К строгой формулировке определения необходимо подходить весьма аккуратно, чтобы не привнести в него внутреннюю противоречивость, приводящую к тому, что называется антиномиями (парадоксами) теории множеств. Противоречие здесь может возникнуть при рассмотрении вопроса о том, является ли само определяемое множество элементом самого себя. Поэтому самым логичным способом предотвратить внутреннюю противоречивость рассматриваемого определения – это разбить его на две части:

- часть определения, формулирующая критерий принадлежности к определяемому множеству всех тех и только тех множеств, которые не совпадают с определяемым множеством;
- часть определения, которая дополнительно указывает либо факт принадлежности, либо факт не-принадлежности определяемого множества самому себе.

Продолжение примера 5.5.18. Запись этого определения на стилизованном естественном языке с использованием языка SCs:

Имеет место **эквивалентность** следующих логических формул:

- **существует** конструкция вида:

[_s <-- **нерефлексивное множество** ;] ;

/* Из этой атомарной формулы следует то, что значение переменной _s не может совпадать sc-узлом, имеющим идентификатор “**нерефлексивное множество**”, т. е. не может совпадать со знаком определяемого множества. */

- **существует** конструкция вида:

[_s <--> _s ; **нерефлексивное множество** ;] ;

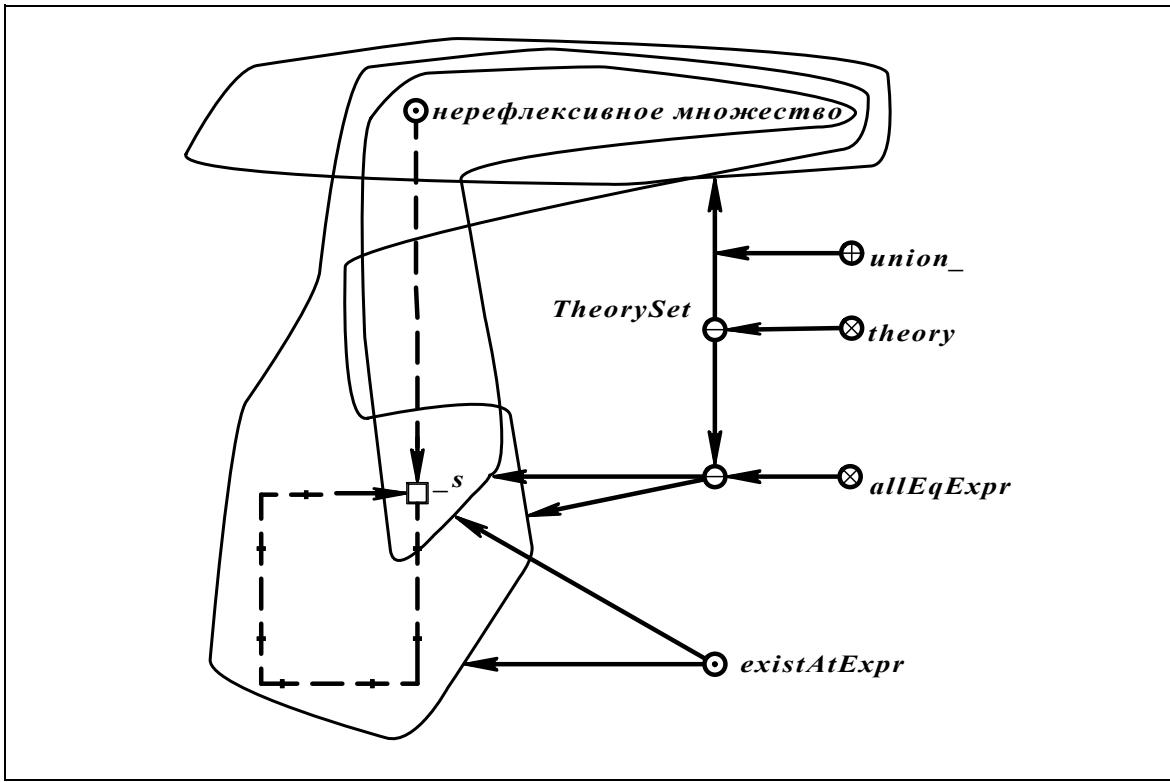
/* Включение в данную атомарную формулу sc-узла с идентификатором “**нерефлексивное множество**” означает, что значение переменной _s не должно совпадать со знаком определяемого множества. Другими словами, под _s подразумевается знак любого другого множества */

Попробуем сформулировать приведённое здесь определение на естественном языке:

Множество _s, не являющееся множеством всех нерефлексивных множеств, является элементом множества всех нерефлексивных множеств в том и только в том случае, если это множество _s не является элементом самого себя.

Таким образом в этом определении речь идёт только о тех множествах, которые не совпадают со множеством всех нерефлексивных множеств (т. е. с определяемым множеством). При таком определении вопрос о том, является ли множество всех нерефлексивных множеств элементом самого себя, остаётся открытым. То есть этому определению не противоречит ни утверждение о том, что множество всех нерефлексивных множеств является элементом самого себя, ни утверждение о том, что множество всех нерефлексивных множеств не является элементом самого себя. Итак, причина возникновения по крайней мере некоторых видов противоречий, которые называют парадоксами теории множеств, не во внутренней противоречивости самой теории множеств, а в некорректных, внутренне противоречивых формулировках некоторых утверждений.

П р о д о л ж е н и е п р и м е р а 5 . 5 . 1 8 . Запись определения на языке SCLg:



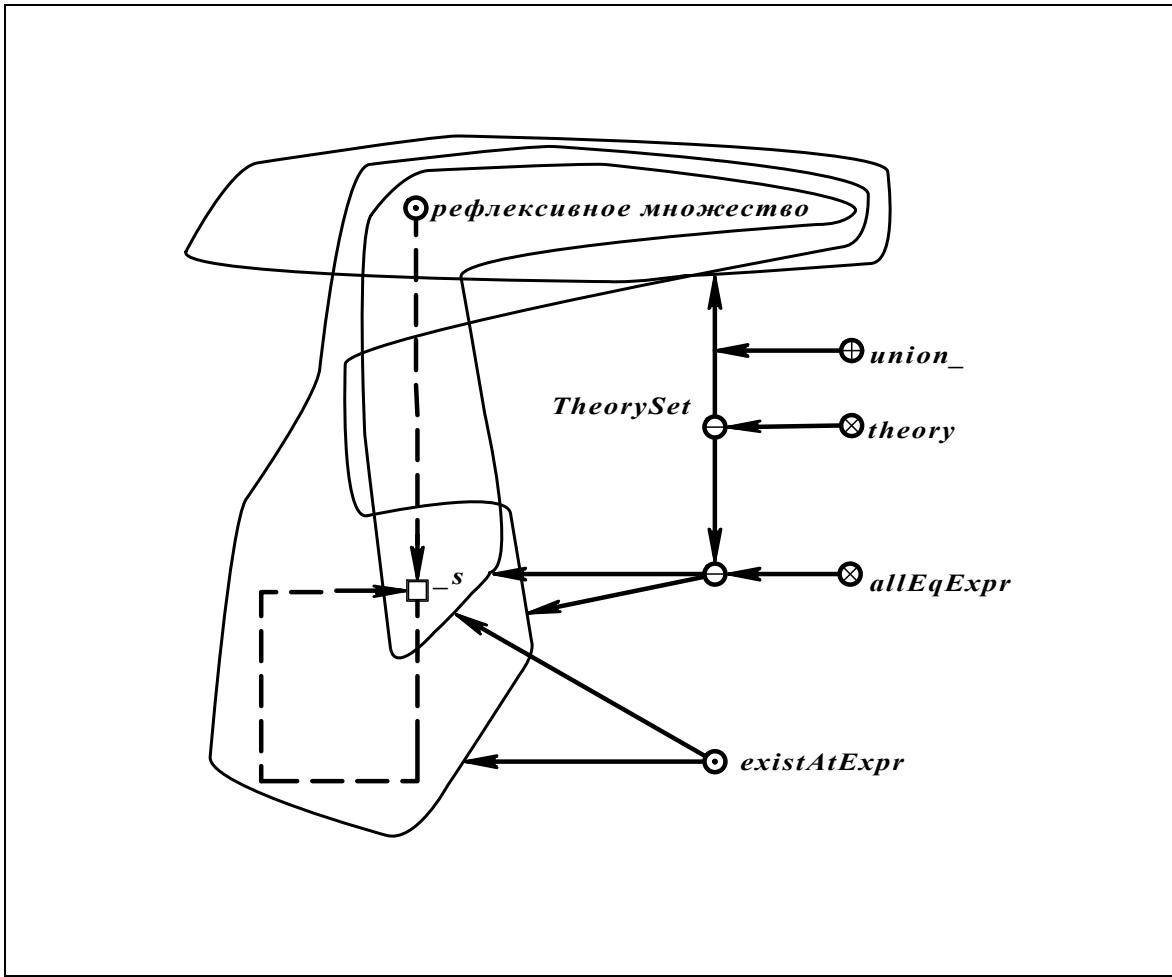
П р о д о л ж е н и е п р и м е р а 5 . 5 . 1 8 . Запись на языке SCLs:

```
forall _s  ([_s <-- нерафлексивное множество ;]
             <=>
             [_s <<+ _s ; нерафлексивное множество ;]);
```

П р и м е р 5 . 5 . 1 9 . Определение множества всех множеств, которые являются элементами самих себя. В пункте 3.1 такое множество мы называли множеством всевозможных рефлексивных множеств и поставили ему в соответствие идентификатор “**рефлексивное множество**”.

Очевидно, что определение очень похоже на предыдущее. Поэтому ограничимся его записью на языке SCLg.

Продолжение примера 5.5.19. Запись определения на языке SCLg:



Пример 5.5.20. Формальная запись следующего высказывания. Человека t будем называть брадобреем для группы лиц s , в состав которой входит и человек t , в том и только в том случае, если человек t бреет каждого человека x , принадлежащего группе лиц s , если этот человек не бреет себя сам.

Очевидно, что приведённая формулировка некорректна, т. к. к противоречию приводит попытка дать ответ на вопрос "бреет ли брадобрей сам себя". См. [100] ([Виленкин Н.Я. 1969 кни-РасскОМ](#)). Переформулируем рассматриваемое высказывание в целях устранения в нём внутренней противоречивости.

Человека t будем называть брадобреем для группы лиц s , в состав которой входит и человек t , в том и только в том случае, если человек t бреет каждого человека x , не совпадающего с t , принадлежащего группе лиц s , если этот человек не бреет себя сам.

При таком определении брадобрея ответ на вопрос "бреет ли брадобрей сам себя" может быть как положительным, так и отрицательным. То есть существуют два типа таких брадобреев:

- брадобреи, которые сами себя бреют;
- брадобреи, которые сами себя не бреют.

Продолжение примера 5.5.20. Запись этого определения на стилизованном естественном языке с использованием языка SCs:

Для всех значений переменных $_s$, $_t$ имеет место эквивалентность следующих логических формул:

- существует конструкция вида:

[$_s \rightarrow\!\!> \text{брадобрей} ::_t ;$]

- для всех значений переменных $_x$ имеет место эквивалентности следующих логических формул:

• существует конструкция вида:

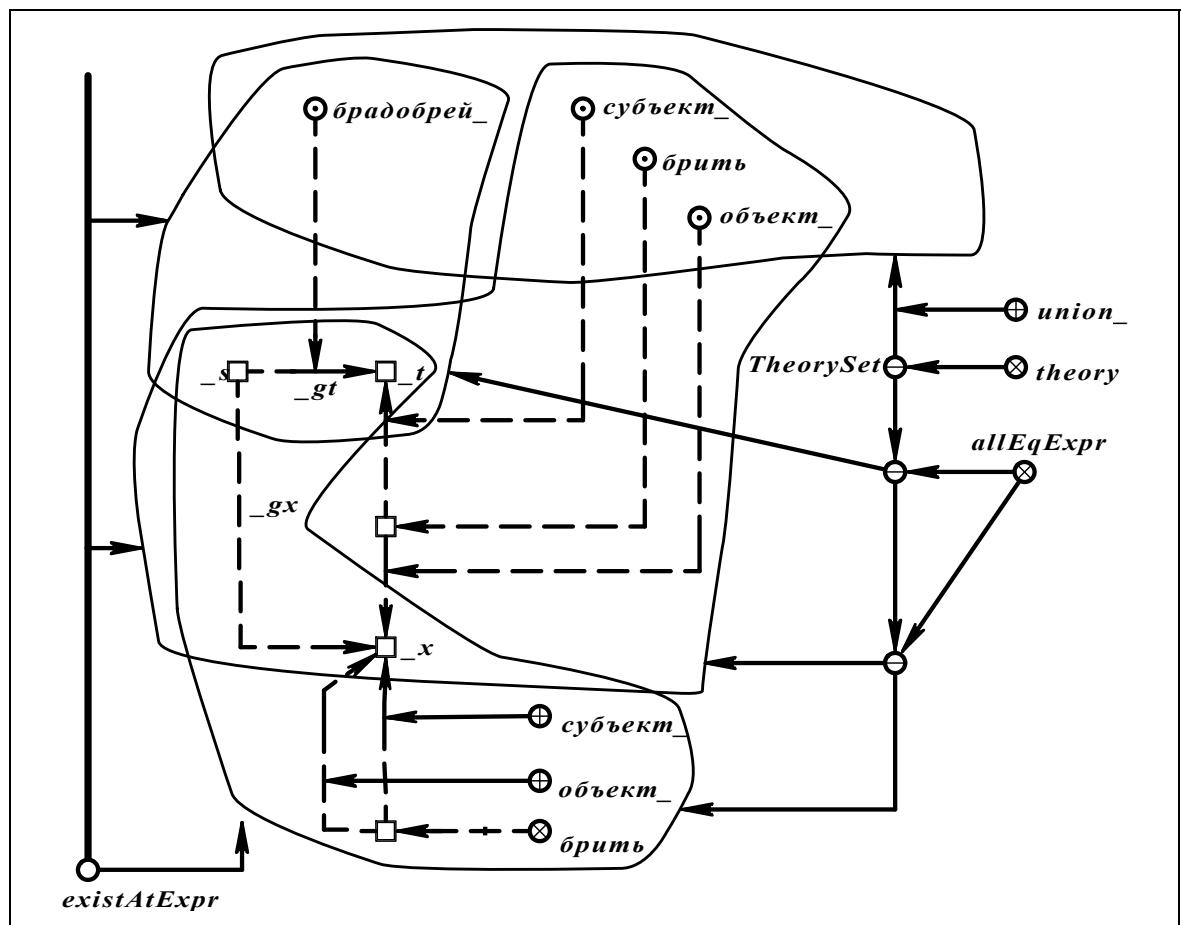
[$\text{брить} \rightarrow\!\!> (\cdot \text{субъект} ::_t \text{ объект} ::_x \cdot) ; _s \rightarrow\!\!> _x ::_t ;$]

• существует конструкция вида:

[$\text{брить} \leftarrow\!\!> (\cdot \text{субъект} ::_x \text{ объект} ::_x \cdot) ; _s \rightarrow\!\!> _x ::_t ;$]

Здесь атрибут *субъект* (субъект действия) указывает на человека, который бреет, а атрибут *объект* (объект действия, то, на что действие направлено) указывает на человека, которого бреют.

Продолжение примера 5.5.20. Запись определения на языке SCLg:



Продолжение примера 5.5.20. Запись на языке SCLs:

$\forall _s, _t ([_s \succ -gt \succ -t ; \text{брадобрей} \rightarrow\!\!> -gt ;]$
 $\Leftarrow\Rightarrow$

$$\begin{aligned}
 & (\forall _x \ [\text{брить} \rightarrow \cdot (\text{субъект_} :: _t, \text{ объект_} :: _x) ; \\
 & \quad _s \succ _gx \succ _x ; _s \succ _gt \succ _t ;] \\
 & \Leftrightarrow \\
 & [\text{брить} \rightarrow \cdot (\text{субъект_} :: _x, \text{ объект_} :: _x) ; \\
 & \quad _s \succ _gx \succ _x ; _s \succ _gt \succ _t ;] \\
 &) \\
 &)
 \end{aligned}$$

Пример 5.5.21. Определение изоморфизма систем множеств (см. пункт 3.4.3).

Продолжение примера 5.5.21. Запись этого определения на стилизованном естественном языке с использованием языка SCs:

Для всех значений переменной _k имеет место эквивалентность следующих логических формул:

- существует конструкция вида:

[_k $\Leftarrow\Rightarrow$ изоморфизм систем множеств ;]

- существует конструкция вида:

[_k $\Leftarrow\Rightarrow$ взаимно однозначная сюръекция ;]

_k == (· (· _sx, amp_ :: _ax ·), (_sy, amp_ :: _ay ·), отни_ :: _r ·) ;]

для которой имеет место коньюнкция следующих логических формул:

- имеет место эквивалентность следующих логических формул:

- существует конструкция вида:

[_sx $\rightarrow\rightarrow$ _gx, _ex ; _gx \succ _ex ; _gx $\Leftarrow\Rightarrow$ пара принадлежности ;
_r $\rightarrow\rightarrow$ (· _ax :: _gx, _ay :: _gy ·), (_ax :: _ex, _ay :: _ey ·) ;]

- существует конструкция вида:

[_sy $\rightarrow\rightarrow$ _gy, _ey ; _gy \succ _ey ; _gy $\Leftarrow\Rightarrow$ пара принадлежности ;
_r $\rightarrow\rightarrow$ (· _ax :: _gx, _ay :: _gy ·), (_ax :: _ex, _ay :: _ey ·) ;]

- имеет место эквивалентность следующих логических формул:

- существует конструкция вида:

[_sx $\rightarrow\rightarrow$ _uy, _gx ; _uy \succ _gx ; _uy $\Leftarrow\Rightarrow$ узловое множество ;
_gx $\Leftarrow\Rightarrow$ пара принадлежности ;
_r $\rightarrow\rightarrow$ (· _ax :: _uy, _ay :: _uy ·), (· _ax :: _gx, _ay :: _gy ·) ;]

- существует конструкция вида:

[_sy $\rightarrow\rightarrow$ _uy, _gy ; _uy \succ _gy ; _uy $\Leftarrow\Rightarrow$ узловое множество ;
_gy $\Leftarrow\Rightarrow$ пара принадлежности ;
_r $\rightarrow\rightarrow$ (· _ax :: _uy, _ay :: _uy ·), (· _ax :: _gx, _ay :: _gy ·) ;]

Продолжение примера 5.5.21. Запись на языке SCLs:

$\forall _k ([_k \Leftarrow\Rightarrow \text{изоморфизм систем множеств}]$

$\Leftrightarrow \exists _sx, _ax, _sy, _ay, _r$

([_k $\Leftarrow\Rightarrow$ взаимно однозначная сюръекция ;]

_k == (· (· _sx, amp_ :: _ax ·), (_sy, amp_ :: _ay ·), отни_ :: _r ·) ;]

& $\forall _gx, _ex, _gy, _ey$

([_sx $\rightarrow\rightarrow$ _gx, _ex ; _gx \succ _ex ;]

```

 $_gx \Leftarrow \text{пара принадлежности} ;$ 
 $_r \rightarrowtail (\_ax :: _gx, \_ay :: _gy) ,$ 
 $(\cdot \_ax :: \_ex, \_ay :: \_ey) ; ]$ 
 $\Leftrightarrow$ 
 $[ \_sy \rightarrowtail \_gy , \_ey ; \_gy \succ \_ey;$ 
 $\_gy \Leftarrow \text{пара принадлежности};$ 
 $\_r \rightarrowtail (\_ax :: \_gx, \_ay :: \_gy) ,$ 
 $(\cdot \_ax :: \_ex, \_ay :: \_ey) ; ] )$ 

&  $\forall \_gx, \_ex, \_gy, \_ey$ 
 $( [ \_sx \rightarrowtail \_yx, \_gx; \_yx \succ \_gx;$ 
 $\_yx \Leftarrow \text{узловое множество};$ 
 $\_gx \Leftarrow \text{пара принадлежности};$ 
 $\_r \rightarrowtail (\cdot \_ax :: \_yx, \_ay :: \_yx) ,$ 
 $(\cdot \_ax :: \_gx, \_ay :: \_gy) ; ]$ 
 $\Leftrightarrow$ 
 $[ \_sy \rightarrowtail \_vy, \_gy; \_vy \succ \_gy;$ 
 $\_vy \Leftarrow \text{узловое множество};$ 
 $\_gy \Leftarrow \text{пара принадлежности};$ 
 $\_r \rightarrowtail (\cdot \_ax :: \_vy, \_ay :: \_vy) ,$ 
 $(\cdot \_ax :: \_gx, \_ay :: \_gy) ; ]$ 
 $)$ 
 $)$ 
 $);$ 

```

Пример 5.5.22. Определение понятия “группа” (см. пункт 3.4.2).

Продолжение примера 5.5.22. Запись этого определения на стилизованном естественном языке с использованием языка SCs (для записи структуры атомарных формул).

Для всех значений переменной $_G$ имеет место эквивалентность следующих логических формул:

- существует конструкция вида:

$[\text{группа} \rightarrowtail _G;]$

- существует конструкция вида:

$[\text{алгебраическая структура с одной бинарной операцией} \rightarrowtail _G ;$

$_G \rightarrowtail \text{сигнатурное отношение} _r ,$

$\text{атрибут}_1 :: \text{аргумент}_1 ,$

$\text{атрибут}_2 :: \text{аргумент}_2 ,$

$\text{атрибут}_3 :: \text{результат} ;$

$\text{алгебраическая операция} \rightarrowtail$

$(\text{отношение} :: _r , \text{результат} :: \text{результат}) ;]$

для которой имеет место конъюнкция следующих логических формул:

- для всех $_a$, $_b$ имеет место импликация следующих логических формул:

- если существует конструкция вида

$[_G \rightarrowtail \text{первичный элемент} _a ,$
 $\text{первичный элемент} _b ;]$

- то существует конструкция вида

$[_G == [_r \rightarrow [\text{аргумент-1} :: _a , \text{аргумент-2} :: _b , \text{результат} :: _c] ; \cdot]]$

- для всех $_a, _b, _c, _d$ имеет место импликация следующих логических формул (аксиома ассоциативности):

- если существует конструкция вида:

$[_G == [_r \rightarrow [\text{аргумент-1} :: _a , \text{аргумент-2} :: _bc , \text{результат} :: _d] , [\text{аргумент-1} :: _b , \text{аргумент-2} :: _c , \text{результат} :: _bc] ; \cdot]]$

- то существует конструкция вида

$[_G == [_r \rightarrow [\text{аргумент-1} :: _ab , \text{аргумент-2} :: _c , \text{результат} :: _d] , [\text{аргумент-1} :: _a , \text{аргумент-2} :: _b , \text{результат} :: _ab] ; \cdot]]$

- существует конструкция вида (аксиома о существовании нейтрального элемента):

$[_G \rightarrow [\text{первичный элемент} :: _e]]$

для которой имеет место импликация следующих логических формул:

- если существует конструкция вида:

$[_G \rightarrow [\text{первичный элемент} :: _a]]$

- то существует конструкция вида

$[_G == [_r \rightarrow [\text{аргумент-1} :: _e , \text{аргумент-2} :: _a , \text{результат} :: _a] , [\text{аргумент-1} :: _a , \text{аргумент-2} :: _e , \text{результат} :: _a] ; \cdot]]$

- для всех $_a, _b$ имеет место импликация следующих логических формул (аксиома о левом делении):

- если существует конструкция вида

$[_G \rightarrow [\text{первичный элемент} :: _a , \text{первичный элемент} :: _b]]$

- то существует конструкция вида

$[_G == [_r \rightarrow [\text{аргумент-1} :: _x , \text{аргумент-2} :: _a , \text{результат} :: _b] ; \cdot]]$

- для всех $_a, _b$ имеет место импликация следующих логических формул (аксиома о правом делении):

- если существует конструкция вида

$[_G \rightarrow [\text{первичный элемент} :: _a , \text{первичный элемент} :: _b]]$

- то существует конструкция вида

$[_G == [_r \rightarrow [\text{аргумент-1} :: _a , \text{аргумент-2} :: _x , \text{результат} :: _b] ; \cdot]]$

Упражнения к подразделу 5.5.

Упражнение 5.5.1. Запишите на SCLs определение полугруппы:

Полугруппа – это реляционная структура \mathbf{G} , у которой:

- отсутствуют сигнатурные множества;
- имеются два сигнатурных отношения, одно – основное (ri), другое вспомогательное (“**алгебраическая операция**”)

- имеется только один кортеж метаотношения “**алгебраическая операция**” вида:
 $(\text{отношение_} : ri, \text{аргумент_} : ai, \text{аргумент_} : aj, \text{результат_} : ar,)$
- пересечение множеств G и ri представляет собой тернарное классическое отношение со схемой $\{ai, aj, ar\}$ и алгебраическую операцию с аргументами $\{ai, aj\}$.
- в G имеется 6 элементов с атрибутом “**атрибут**”: три основных (ai, aj, ar) и три вспомогательных ($\text{отношение_}, \text{атрибут_}, \text{результат_}$).
- для каждой структуры вида $(x_1 \circ x_2) \circ x_3 \Rightarrow x_1 \circ (x_2 \circ x_3)$ (ассоциативность).

Упражнение 5.5.2. Запишите утверждение о том, что из любого набора sc-элементов можно построить множество. Это утверждение разбивается на два следующих утверждения:

- из любого sc-элемента можно построить его синглтон (1-мощное множество, содержащее этот sc-элемент);
- из любого множества и любого sc-элемента можно построить другое множество, состоящее из элементов заданного множества с добавлением указанного sc-элемента (если этот sc-элемент входит в число элементов заданного множества, то увеличивается на единицу число вхождений этого sc-элемента).

Упражнение 5.5.3. Аналогичным образом запишите утверждение о том, что из любого набора sc-элементов и любого набора атрибутов можно построить кортеж с любой комбинацией распределения атрибутов по его компонентам.

Упражнение 5.5.4. Запишите утверждение о том, что существует множество, для которого существует sc-элемент, не являющийся элементом этого множества.

Упражнение 5.5.5. Запишите утверждение о том, что существует по крайней мере одна пара кратных пар принадлежности.

Упражнение 5.5.6. Запишите утверждение о том, что существует по крайней мере одна пара принадлежности, для которой не существует кратной ей пары принадлежности.

Упражнение 5.5.7. Запишите утверждение о том, что существует по крайней мере одна петля принадлежности (т. е. существуют множества содержащие знак самого себя в качестве своего элемента).

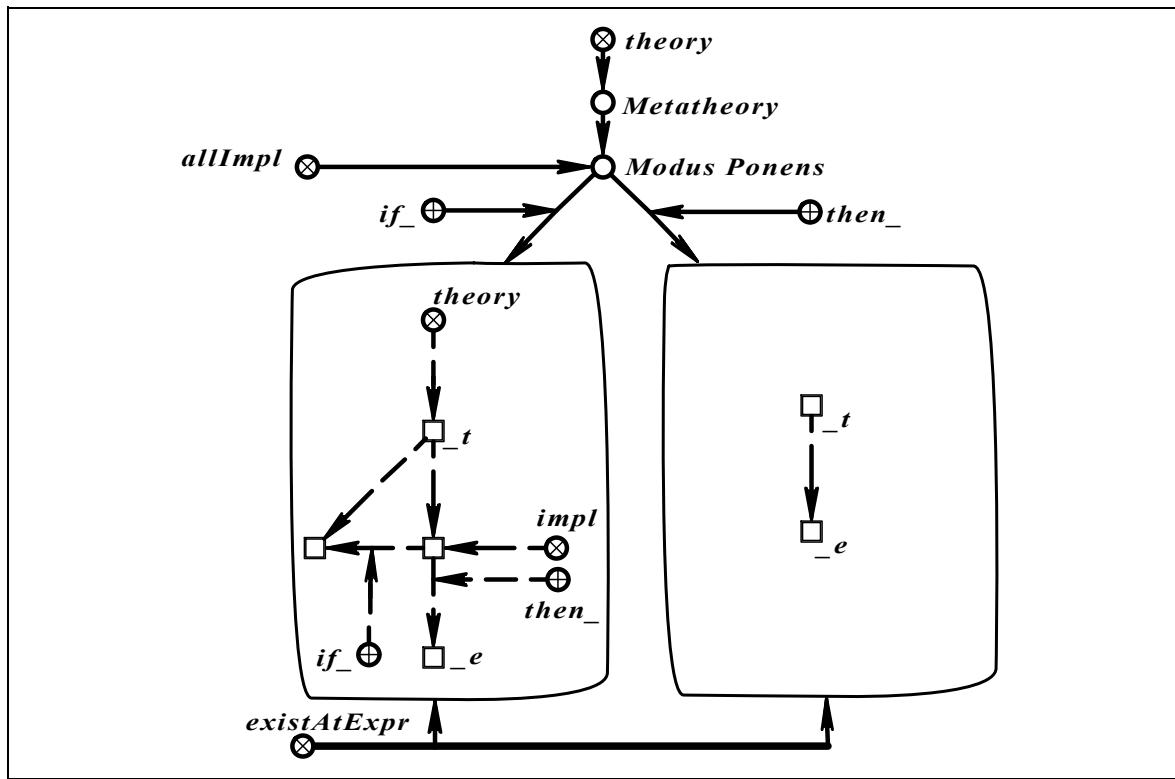
Упражнение 5.5.8. Запишите утверждение о том, что существует множества, не содержащие знак самого себя в качестве своего элемента.

1.6. Формальная метатеория и её представление на языке SCL

По аналогии с примером 5.5.5 можно привести еще целый ряд высказываний, которые описывают общие свойства всевозможных формальных теорий, каждая из которых описывает ту или иную предметную область, которая формально трактуется как некоторая реляционная структура. Свойства всевозможных формальных теорий описываются в рамках специальной метатеории (**Metatheory**), для которой совокупность всевозможных формальных теорий является описываемой предметной областью.

Приведем примеры записи общих логических закономерностей, имеющих место для всех формальных теорий. Описания этих закономерностей входят в состав формальной метатеории, которая описывает свойства всевозможных формальных теорий. Очевидно, в высказываниях этой метатеории присутствуют не только простые переменные, но и метапеременные.

П р и м е р 5 . 6 . 1 . Описание правила логического вывода **Modus ponens**

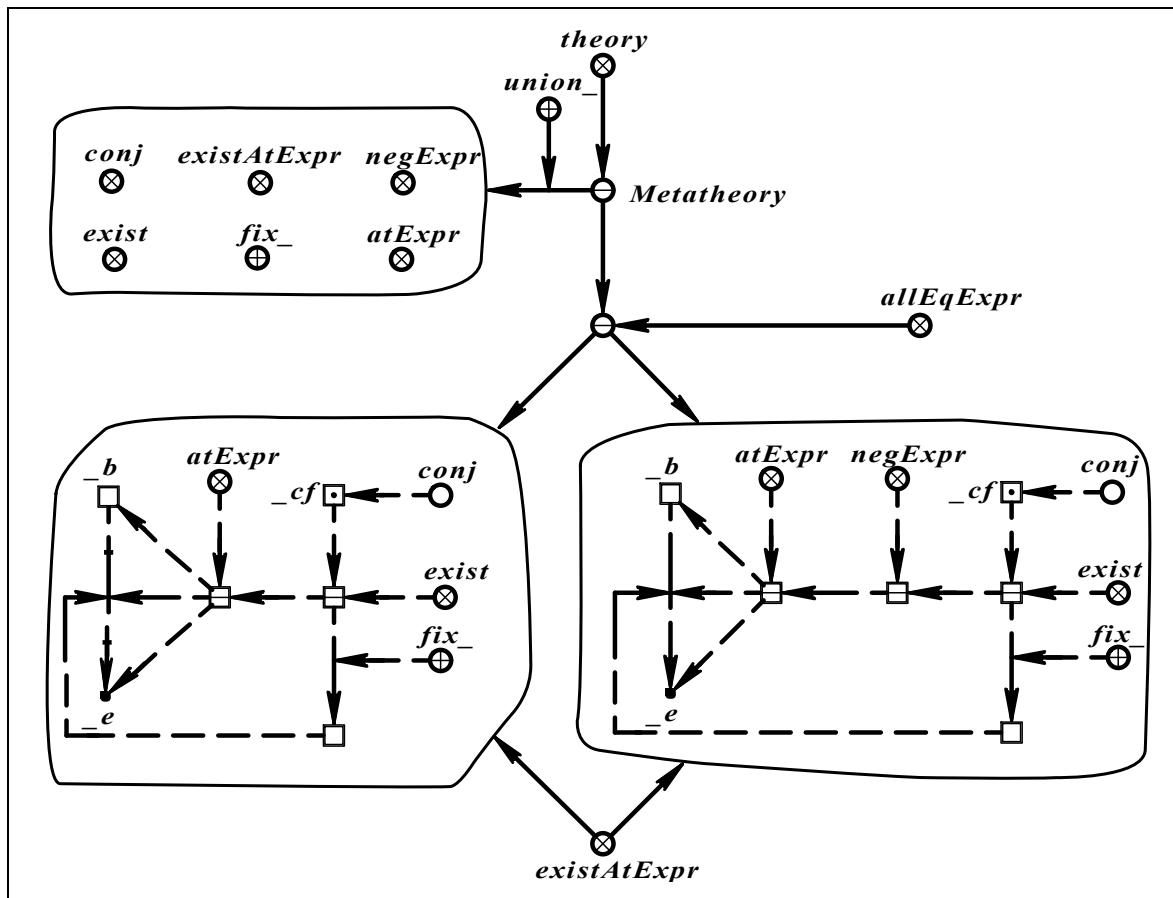


П р и м е р 5 . 6 . 2 . Рассмотрим в качестве примера правило замены негативной константной дуги на эквивалентное негативное высказывание. Итак, рассмотрим запись в рамках метатеории высказывания о том, что дуга непринадлежности, входящая в состав атомарного высказывания, эквивалентна негативному атомарному высказыванию, в состав которого входят:

- узел, из которого выходит указанная дуга непринадлежности;
- элемент, в который эта дуга входит;
- дуга принадлежности, проведенная из указанного узла в указанный элемент.

Из этого метавысказывания следует, что дуга непринадлежности, входящая в число фиксируемых элементов конъюнктивного высказывания, может быть преобразована в соответствующее негативное атомарное высказывание, входящее в состав того же конъюнктивного высказывания. Следовательно, дугу непринадлежности можно рассматривать просто как лаконичный способ записи соответствующего негативного атомарного высказывания.

Продолжение примера 5.6.2. Запись на языке SCLg:



В данном sclg-тексте используется понятие “**absent**”, являющееся знаком множества, состоящего из sc-элементов, которые должны отсутствовать. В случае, если указанные элементы присутствуют, они должны быть удалены (ликвидированы). Таким образом, понятие “**absent**” является одним из средств, обеспечивающих описание различных преобразований sc-текстов.

Упражнения к подразделу 5.6.

Упражнение 5.6.1. Запишите на SCLg определение метаотношения “**быть парой логических формул, одна из которых непосредственно входит в состав другой**”.

Упражнение 5.6.2. Запишите на SCLg определение метаотношения “**быть парой логических формул, одна из которых входит в состав другой**”.

Упражнение 5.6.3. Запишите на SCLg определение бинарного метаотношения, каждая пара которого связывает атомарные scl-формулы, одна из которых является частной по отношению ко второй.

Упражнение 5.6.4. Запишите на SCLg определение истинного высказывания о существовании, в котором квантор существования действует на атомарную scl-формулу. Это высказывание, для которого существует изоморфный подграф в описываемой реляционной структуре.

Упражнение 5.6.5. Запишите на SCLg определение истинного высказывания о существовании, имеющего произвольный вид.

Упражнение 5.6.6. Запишите на SCLg истинного высказывания о всеобщности, в котором квантор всеобщности действует на импликативное высказывание, в котором оба компонента являются атомарными scl-формулами.

Упражнение 5.6.7. Запишите на SCLg общую логическую закономерность (закон отрицания отрицания): $\neg \neg b \equiv b$

Выходы к разделу 5

В данном разделе показано, что на базе языка SC (Semantic Code), который является достаточно простым расширением фактографического языка SCB (путем добавления переменных и введения множеств, элементами которых являются переменные), можно построить логический язык, тексты которого представляют собой не что иное, как представление реляционных структур определенного вида. И точно так же, как в языке SCB, мы легко переходим от реляционных структур к реляционным метаструктурам, в языке SC мы легко переходим от логических формул и формальных теорий к логическим метаформулам и формальным метатеориям.

1. Типология знаний и языки представления знаний в графодинамических ассоциативных машинах

В разделе 4 был рассмотрен и описан графовый язык SC, являющийся основой представления знаний в графодинамических машинах. В разделе 5 был описан и рассмотрен язык SCL, являющийся sc-подъязыком, и предназначенный для представления знаний в виде формальных теорий, состоящих из логических высказываний, описывающих свойства стационарных реляционных структур. Раздел 6 посвящён рассмотрению способов представления на языках SC и SCL различных видов знаний, имеющих предметную и прикладную специфику. В этом разделе будет рассмотрено представление на языке SC различных шкал измерения и измеряемых величин, представление информации и описание закономерностей динамических систем, способы описания информационных целей в графодинамических ассоциативных машинах, принципы представления нейросетевых моделей и гипертекстовых информационных конструкций, которые в силу свойств языка SC при таком представлении приобретают уникальные свойства.

Данный раздел может быть использован в качестве учебного пособия по дисциплинам «Модели представления знаний, базы данных и СУБД» и «Нейросетевые модели и нейрокомпьютеры» для студентов специальности «Искусственный интеллект».

1.1. Представление знаний, связанных с понятием измерения

Ключевые понятия и идентификаторы ключевых узлов: число, *номер*, *шкала измерения*, *измерение*, *pwSet*, *pwArc*, *pwEl*, *pwFuzExpr*.

Рассмотрим одну из возможных (но не единственную возможную (!)) теоретико-множественную трактовку семантики чисел. Следует отметить, что изначально число появилось как результат измерения какого-либо параметра (свойства, характеристики) у некоторого объекта. Таким образом, процедура измерения – это установление некоторого соответствия между множеством исследуемых объектов (исследуемых на предмет анализа определенного параметра, свойства, характеристики) и некоторым множеством чисел. Каждый измеряемый параметр (свойство) с формальной точки зрения трактуется как множество всех тех и только тех объектов, которые этим свойством обладают. Конкретное число – это множество знаков всевозможных классов эквивалентности элементов с одинаковым значением измеряемого параметра. Каждому измеряемому параметру поставлено в соответствие множество классов эквивалентности, являющееся фактор-множеством для множества элементов, обладающих этим измеряемым параметром. Если каждое число есть множество и если известны элементы этого множества, то можно ввести знаки числа и связать их парами принадлежности с элементами обозначаемых ими множеств. Процедуры измерения одного и того же параметра могут быть различными (разными могут быть единицы измерения, разными могут быть точки привязки к числовой шкале).

Соответственно этому во множестве знаков всех пар принадлежности, выходящих из знаков чисел, выделяются подмножества, каждому из которых соответствует некоторая конкретная процедура измерения. Такие подмножества дуг принадлежности, выходящих из знаков чисел, будем называть конкретными шкалами измерения. Соответственно этому введём ключевой scb-узел с идентификатором **“шкала измерения”**, множество знаков всевозможных шкал измерения. Заметим при этом, что конкретная шкала (процедура) измерения может использоваться при измерении нескольких параметров.

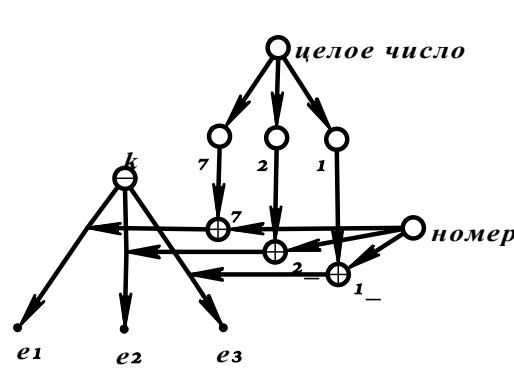
К числу измеряемых параметров, в частности, относятся:

- номер (порядковый номер элемента кортежа);
- мощность множества (количество пар принадлежности, выходящих из знака множества);
- количество элементов (количество элементов множества);
- вес пары принадлежности;
- арность отношения;
- масса;
- температура;
- величина плоского узла;
- расстояние, длина, площадь, объём;
- влажность;
- давление;

- сила электрического тока, напряжение, сопротивление, индуктивность, ёмкость;
- местоположение (координаты на местности);
- отметка времени;
- длительность во времени;
- скорость, ускорение.

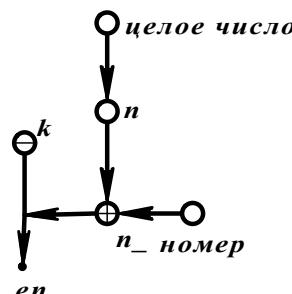
Рассмотрим параметр “**номер**” (порядковый номер элемента кортежа). Измеряемым объектом здесь является пара принадлежности, проведенная из знака кортежа. Замена числовых атрибутов (**1**, **2**, ...) на числа (!), указывающие порядковые номера, дает возможность манипулировать этими номерами, как любыми другими числами. Напомним, что числовые атрибуты, строго говоря, числами не являются. Итак, числовой атрибут **номер** есть множество знаков всех тех и только тех пар принадлежности, которые выходят из знаков кортежей и входят в элементы кортежей, имеющие в рамках этих кортежей порядковый номер **номер**.

Приведем пример записи результата "измерения" порядкового номера элемента в кортеже.



Эта конструкция означает, что в рамках кортежа **k** элемент **e1** имеет 1-й номер, элемент **e2** – 2-й номер, а элемент **e3** – 3-й номер. Заметим, что числа, указывающие номера элементов в кортежах, могут быть не только натуральными (т.е. положительными целыми числами), но и отрицательными целыми числами. Кроме того, номер элемента кортежа может быть нулевым. Следовательно, числа, указывающие номера элементов в кортежах, в общем случае относятся к классу целых чисел, а не к классу натуральных чисел. В качестве примера см. представление чисел в позиционных системах счисления.

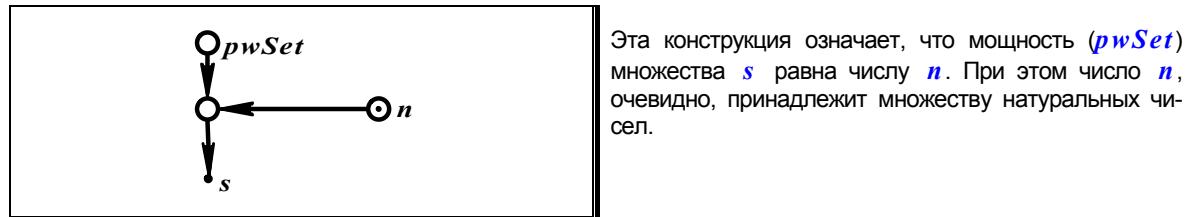
Нетрудно заметить, что между понятием порядкового номера элемента в кортеже и понятием числового атрибута имеет место следующее соотношение.



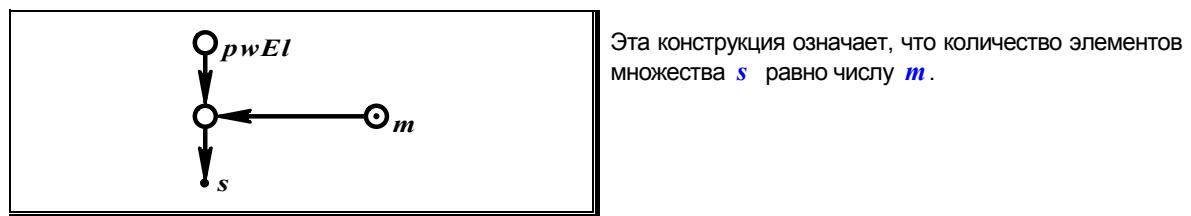
Такая "замена" числовых атрибутов на числа дает возможность описывать соотношения между номерами элементов кортежей с использованием всего многообразия числовых отношений.

Примечание. Далеко не в каждом кортеже используется нумерация его элементов. То есть элементы далеко не каждого кортежа обладают свойством иметь порядковый номер.

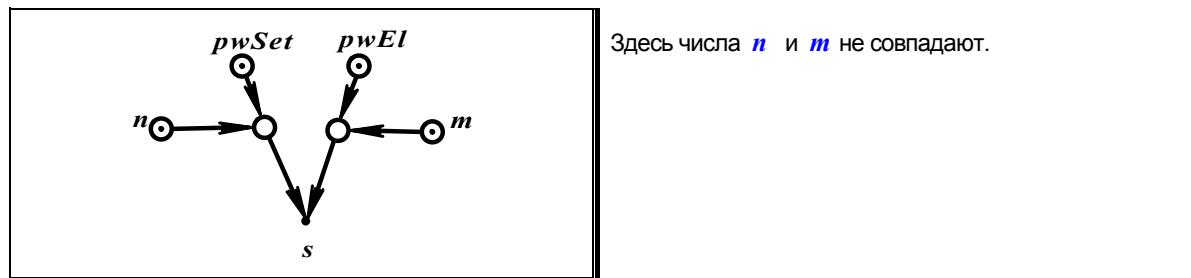
Параметр “**мощность множества**” будем также идентифицировать синонимичными идентификатором “***pwSet***” (power set). Приведем пример.



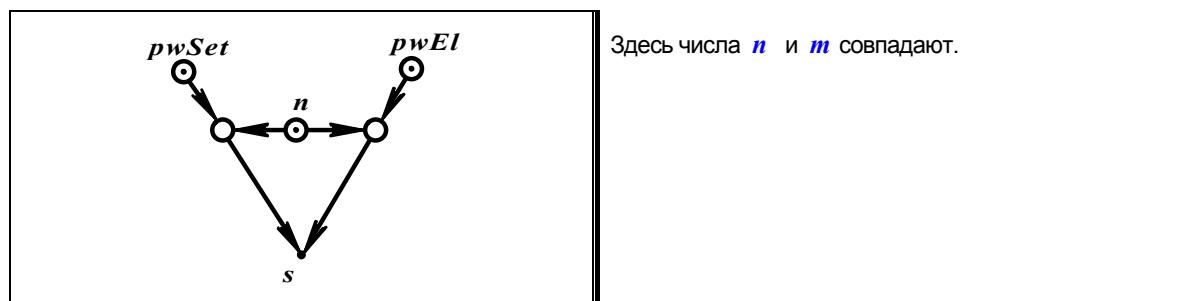
Параметр “**количество элементов**” (количество элементов множества) будем также идентифицировать синонимичным идентификатором “***pwEl***” (power elements). Приведем пример.



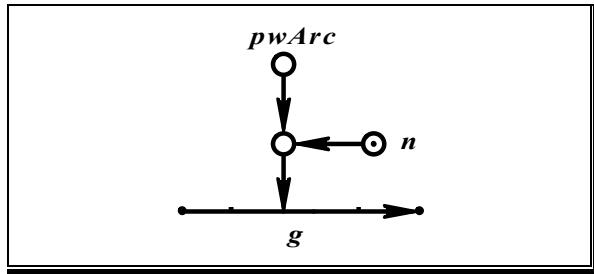
Если множество ***s*** имеет многоократное вхождение каких-либо элементов, то имеет место следующая конструкция.



Если же множество ***s*** не имеет кратных элементов (т.е. является канторовским), то имеет место следующая конструкция.



Рассмотрим параметр “**вес пары принадлежности**” (сила пары принадлежности, мощности scb-дуги, вес scb-дуги), который будем также идентифицировать синонимичным идентификатором “***pwArc***”. Приведем пример.

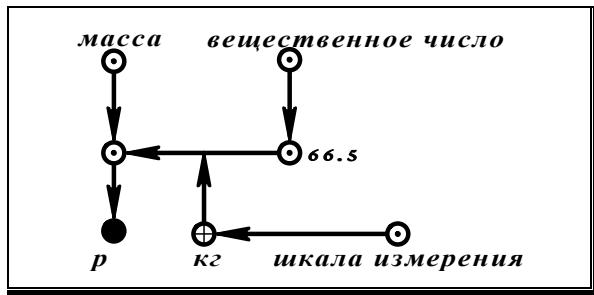


Эта конструкция означает, что вес пары принадлежности g равен числу n .

При этом:

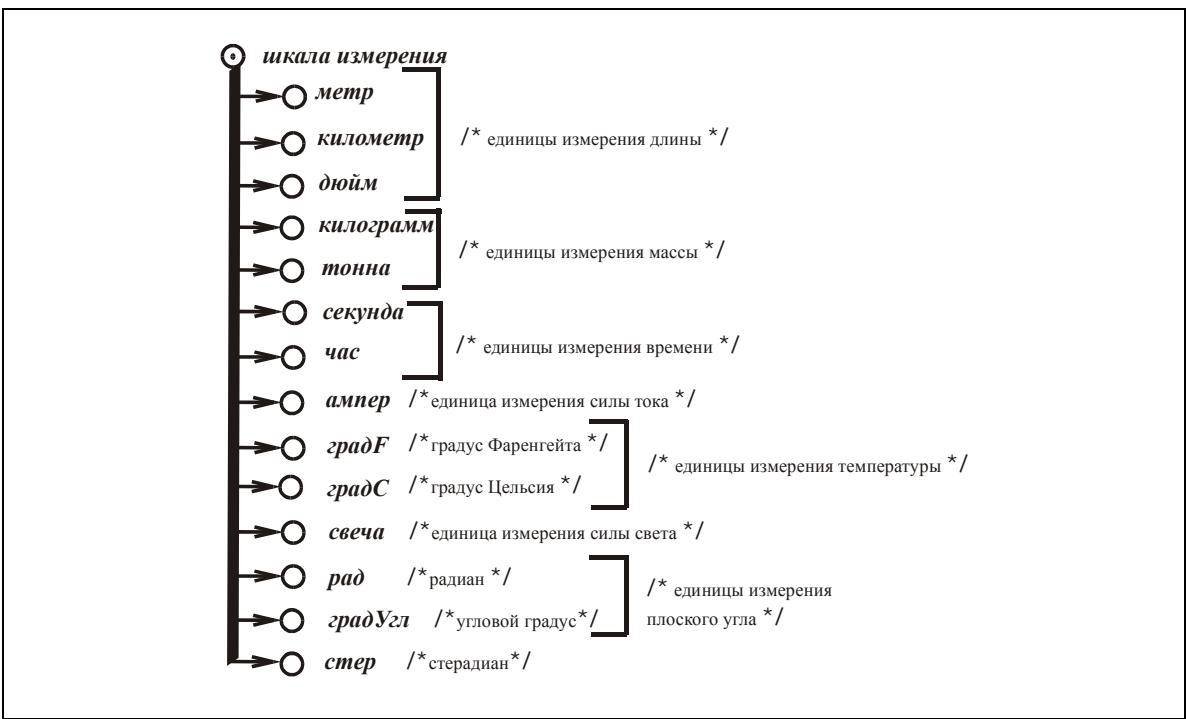
- если $n = 0$, то scb-дуга g негативна;
- если $n = 1$, то scb-дуга g позитивна;
- если $0 < n < 1$, то scb-дуга g считается нечетной с весом n (здесь число n также будем называть степенью нечеткости, степенью достоверности, степенью размерности дуги g);
- если n – целое число, большее 1, то scb-дуга g считается позитивной n -кратной дугой (здесь число n будем также называть кратностью дуги g).

Приведем пример записи результата измерения массы физического тела.



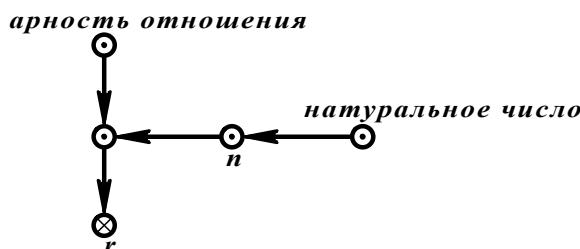
Эта конструкция означает, что масса физического тела p равна 66.5 кг.

Здесь введено ключевое понятие “**шкала измерения**” (быть шкалой измерения), которая обозначает множество знаков всевозможных шкал измерения. Заметим при этом следующее. То, что называют единицами измерения, есть простейший вид шкал измерения.



Примечание. Далеко не для всех измеряемых параметров (измеряемых характеристик) необходимо дополнительно указывать шкалу измерения. Это необходимо только тогда, когда измеряемому параметру соответствует несколько (!) шкал измерения. Примерами параметров, каждому из которых соответствует единственная (!) шкала измерения, являются параметры: *pwSet*, *pwEl*, *pwArc*.

Приведём пример записи результата "измерения" арности отношения.

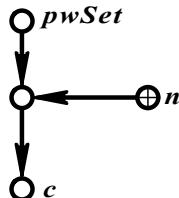


Эта конструкция означает, что отношение *r* является *n*-арным отношением, т.е. представляет собой семейство *n*-арных множеств (множеств, мощность которых равна *n*). Заметим, что далеко не каждое отношение обладает свойством иметь арность. Этим свойством обладают те и только те отношения, каждое из которых представляет собой семейство множеств одинаковой мощности. То есть понятие "**арность отношения**" и понятие "**семейство множеств одинаковой мощности**" являются синонимами.

Нетрудно заметить также, что приведенная выше scb-конструкция эквивалентна утверждению о том, что для каждого элемента с множества *r*, т.е. для каждой конструкции вида



имеет место конструкция вида

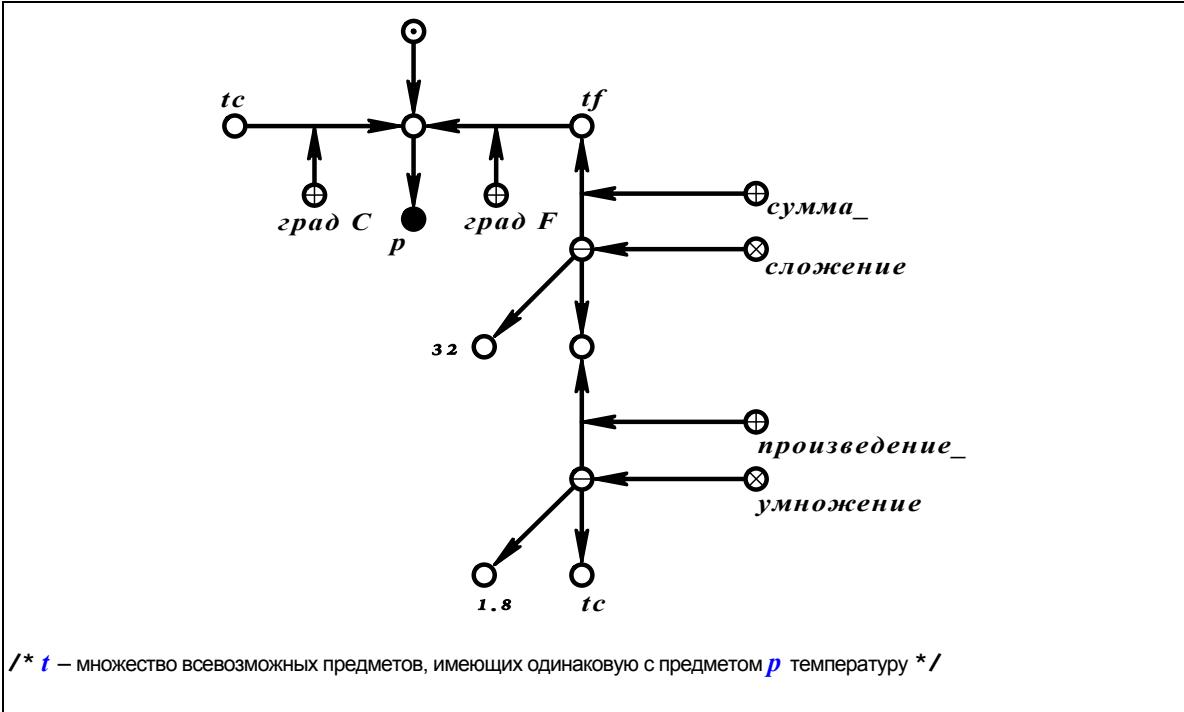


Температурная шкала Цельсия и температурная шкала Фаренгейта связаны между собой следующим соотношением: $tf = 1.8 * tc + 32$,
где

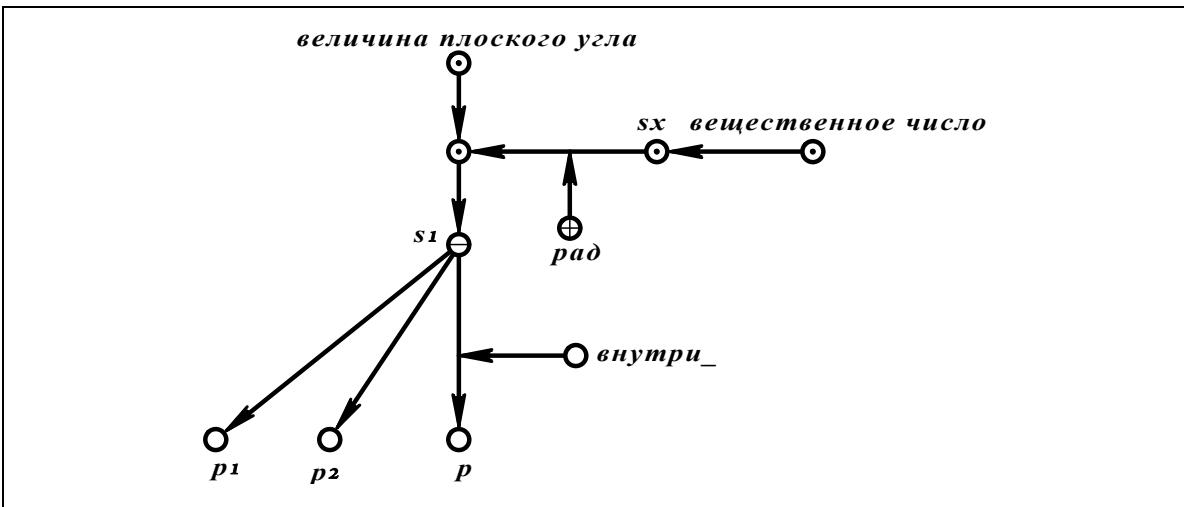
tc – отметка температуры по шкале Цельсия

tf – отметка температуры того же (!) предмета *p*, но по шкале Фаренгейта.

На языке SCB указанное соотношение выглядит следующим образом:



Приведём пример записи результата измерения величины плоского угла в радианах. Измеряемым объектом здесь можно считать тернарный кортеж, состоящий из трёх геометрических фигур, лежащих на одной плоскости. При этом две из этих фигур являются либо отрезками, либо прямыми, либо лучами, а третья фигура трактуется как фигура, лежащая внутри измеряемого угла. Собственно измеряемым углом здесь является один из четырех углов, образованных пересекающимися прямыми, на которых лежат указанные выше отрезки или лучи.



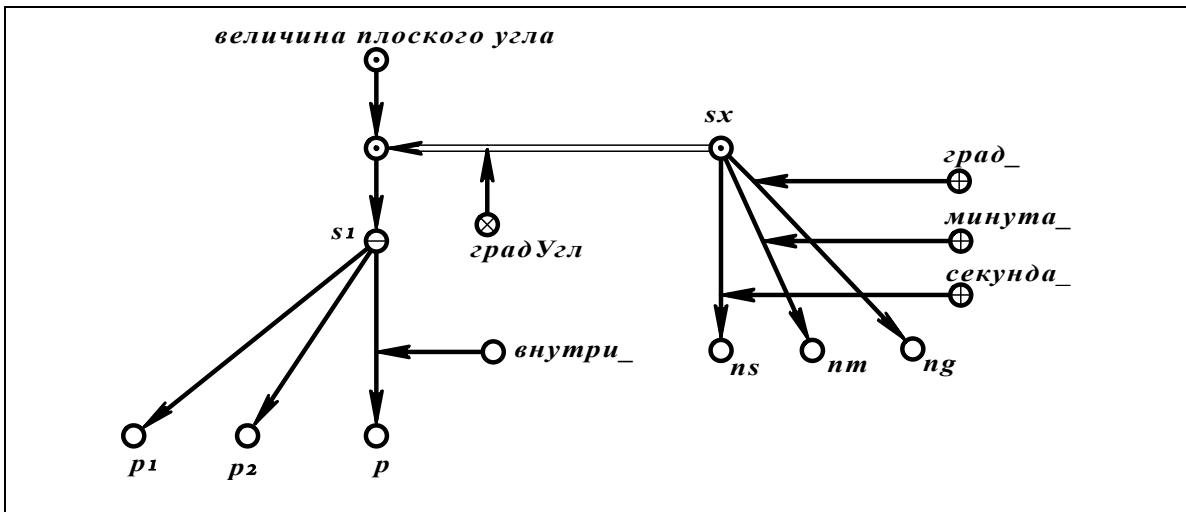
Эта конструкция означает, что число x есть результат измерения (в радианах) величины плоского угла, который составлен геометрическими фигурами p_1 и p_2 (каковыми могут быть прямые, лучи, отрезки, множества точек, лежащих на одной прямой точек) и внутри которого находится геометрическая фигура p .

Если измерение величины плоского угла осуществляется в угловых градусах, то результатом измерения будет уже не число, а тернарный кортеж чисел, компонентами которого являются:

- целое число в диапазоне от 0 до 360 , указывающее количество угловых градусов;

- целое число в диапазоне от **0** до **60**, указывающее количество угловых минут;
- целое число в диапазоне от **0** до **60**, указывающее количество угловых секунд.

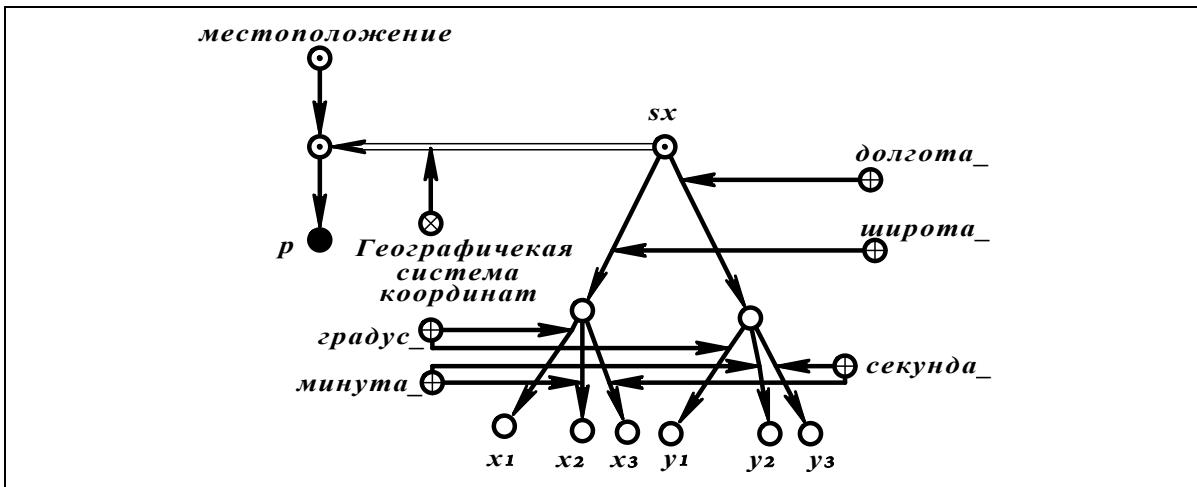
Приведем пример записи результата такого измерения.



Примечание. Особое внимание обратим на то, что результат измерения с парой “**измеряемый параметр – измеряемый объект**” связан здесь ориентированной парой, которая парой принадлежности не является. Такого рода результаты измерения будем называть векторными, противопоставляя их рассмотренным выше скалярным.

Приведём несколько примеров записи результатов измерения, когда этими результатами являются векторные величины.

Рассмотрим пример записи результата измерения местоположения некоторого объекта **p** на земной поверхности в географической системе координат.



Эта конструкция означает то, что местоположение объекта **p** в географической системе координат определяется широтой, которая задаётся кортежем

(**градус_** : **x1** , **минута_** : **x2** , **секунда_** : **x3**) ;
и долготой, которая задаётся кортежем

(**градус_** : **y1** , **минута_** : **y2** , **секунда_** : **y3**) ;
Здесь

x1 – целое число в диапазоне от -90 до +90;

y_1 – целое число в диапазоне от -180 до +180;

x_2, x_3, y_2, y_3 – целые числа в диапазоне от 0 до 60.

Кроме географической системы координат для измерения местоположения объектов существует большое количество других систем координат.

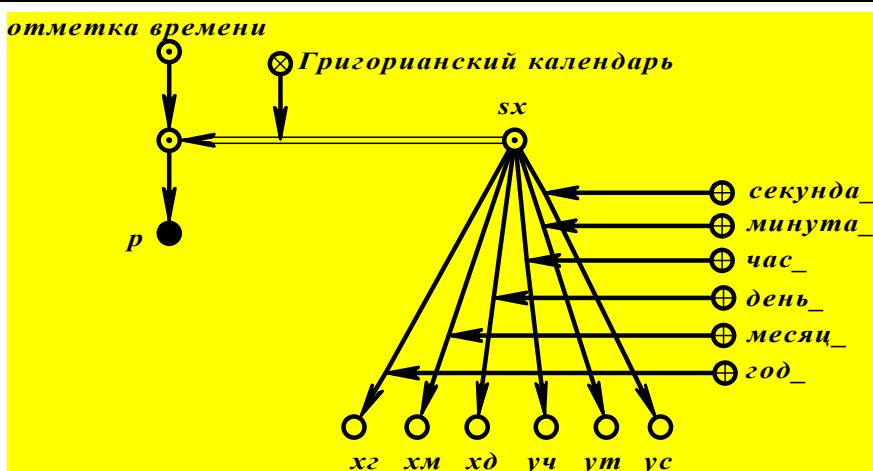
система координат → Географическая система координат,
 Геодезическая система координат,
 Горизонтальная система небесных координат,
 Первая экваториальная система небесных координат,
 Эклиптическая система небесных координат,
 Галактическая система небесных координат;

система координат ⊃ декартова система координат;

система координат ⊃ сферическая система координат;

система координат ⊃ цилиндрическая система координат;

Рассмотрим пример записи результата измерения отметки времени для некоторого события (процесса, ситуации).



Пусть $x_2 = 1999$, $x_m = 12$, $x_g = 15$, $x_ч = 14$, $x_т = 30$, $x_s = 30$.

Тогда приведённая конструкция означает то, что событие p произошло 15 декабря 1999 года в 14 часов 30 минут 30 секунд.

Для рассматриваемой конструкции:

x_2 представляет собой целое число в диапазоне от $-\infty$ до $+\infty$,

x_m – целое число в диапазоне от 1 (январь) до 12 (декабрь);

x_g – целое число в диапазоне от 1 до 31;

$x_ч$ – целое число в диапазоне от 0 до 23;

$x_т$, x_s – целые числа в диапазоне от 0 до 59.

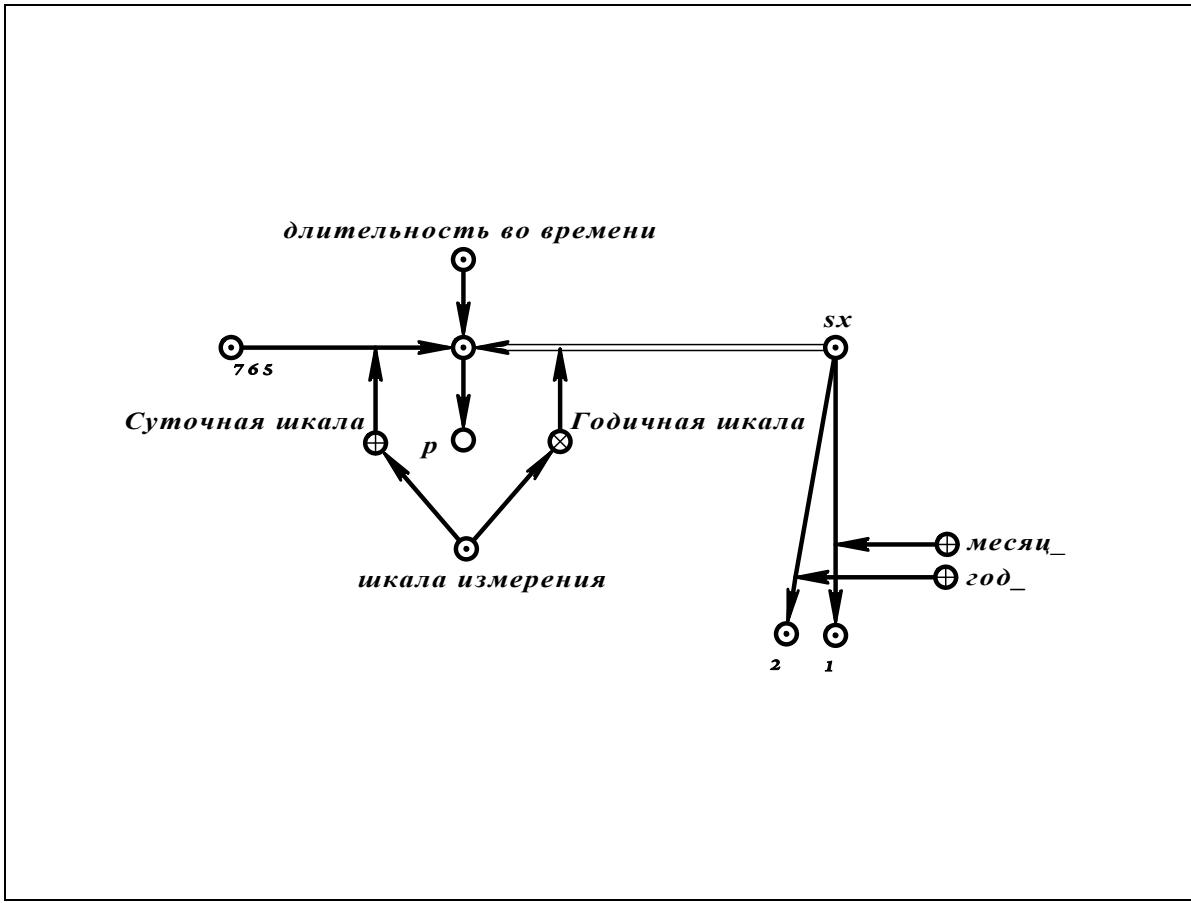
Кроме григорианского календаря известны и другие шкалы измерения отметки времени:

шкала измерения отметки времени →

Григорианский календарь,
 Юлианский календарь;

Параметр "отметка времени" на некоторой шкале времени не следует путать с параметром "длительность во времени", который характеризует отрезок времени, в течение которого происходит некоторое событие, или, образно говоря, характеризует период "существования" (время "жизни") указанного события.

Приведём примеры записи результата измерения длительности во времени (отрезка времени).



Приведённая запись означает, что некий процесс *p* длился *2* года *1* месяц или (по другой шкале) *765* суток (напомним, что года могут быть високосными, а месяцы могут иметь разное количество дней).

Всё, о чём говорилось выше, имеет отношение к изменению скалярных (!) параметров. Но кроме скалярных параметров, есть векторные параметры, результатом измерения которых являются не числа, а кортежи (!) чисел.

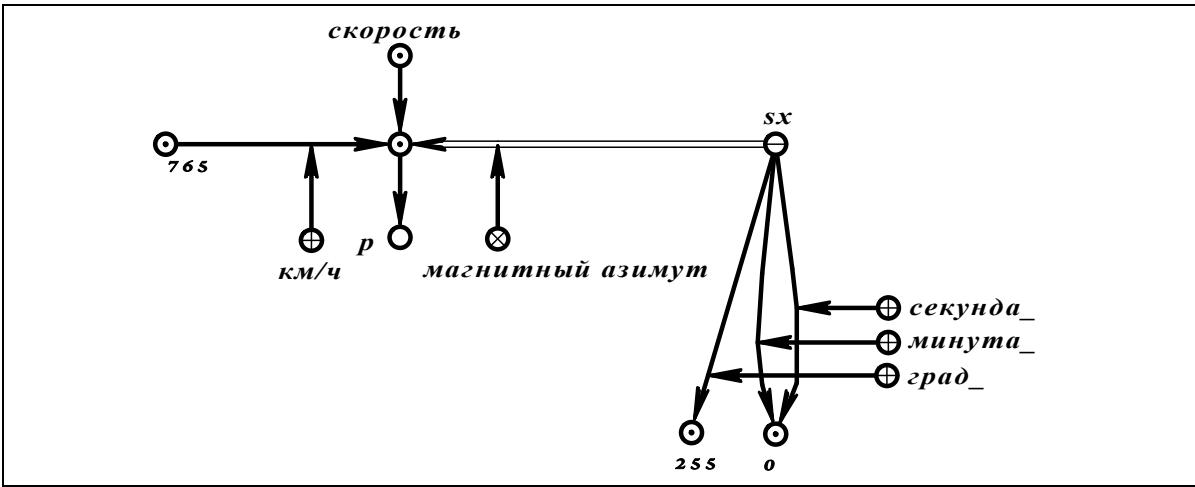
Заметим то, что результатом измерения одного и того же параметра в зависимости от шкалы измерения может быть как скалярная величина, так и векторная величина (кортеж чисел).

В приведённой конструкции введено понятие шкалы измерения времени (шкалы измерения длительности во времени).

<i>шкала измерения времени</i> →	<i>Годичная шкала</i> ,
	<i>Суточная шкала</i> ,
	<i>час /* измерение длительности в часах */</i> ,
	<i>минута</i> ,
	<i>секунда</i> , <i>миллисекунда</i> , <i>наносекунда</i> ;

Приведём пример записи результата измерения параметра, который характеризуется не только величиной, но и направлением. Примером такого параметра является **скорость**. Рассмотрим запись

утверждения о том, что некий объект p в некий (неуточняемый) момент времени движется со скоростью 72 километра в час (км/ч) в направлении на юго-запад. Направление движения на поверхности земли задаётся азимутом (например, магнитным азимутом) и измеряется величиной угла между направлением на точку севера и соответственно направлением движения. При этом отсчёт величины угла осуществляется от направления на север по часовой стрелке.



Здесь при измерении азимута градус задаётся натуральным числом в диапазоне от 0 до 360 (точнее, до 359), а минута и секунда – натуральным числом в диапазоне от 0 до 60 (точнее, до 59).

Введём понятие шкалы измерения величины скорости и понятие шкалы измерения направления (в частности направления движения).

шкала измерения величины скорости \rightarrow км/ч ,
м/с ,
узел /* морской */ ;

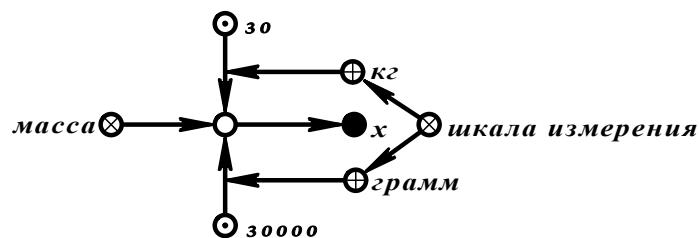
шкала измерения направления \rightarrow магнитный азимут ,
истинный азимут /* географический */ ;

Подведя итог вышесказанному, можно ввести обобщённое отношение “измерение”, которое представляет собой бинарное ориентированное отношение, каждая пара которого связывает:

- 1) знак пары принадлежности, связывающей знак измеряемого параметра (измеряемой характеристики) со знаком измеряемого объекта (каковым может быть всё что угодно);
- 2) результат измерения, каковым может быть как число (скалярная величина), так и числовой кортеж (векторная величина).

При этом, если результат измерения является скалярной величиной, то соответствующая пара отношения “измерение” является парой принадлежности (!).

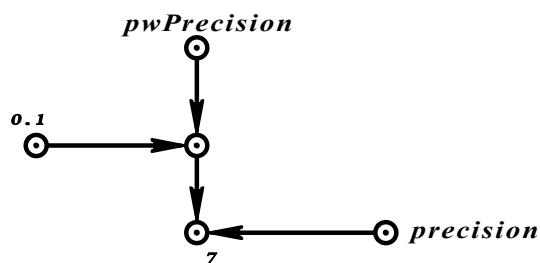
Отношение “измерение” разбивается на целый ряд подмножеств, каждому из которых соответствует та или иная **шкала измерения** (в частности, единица измерения). Приведём ещё несколько примеров.



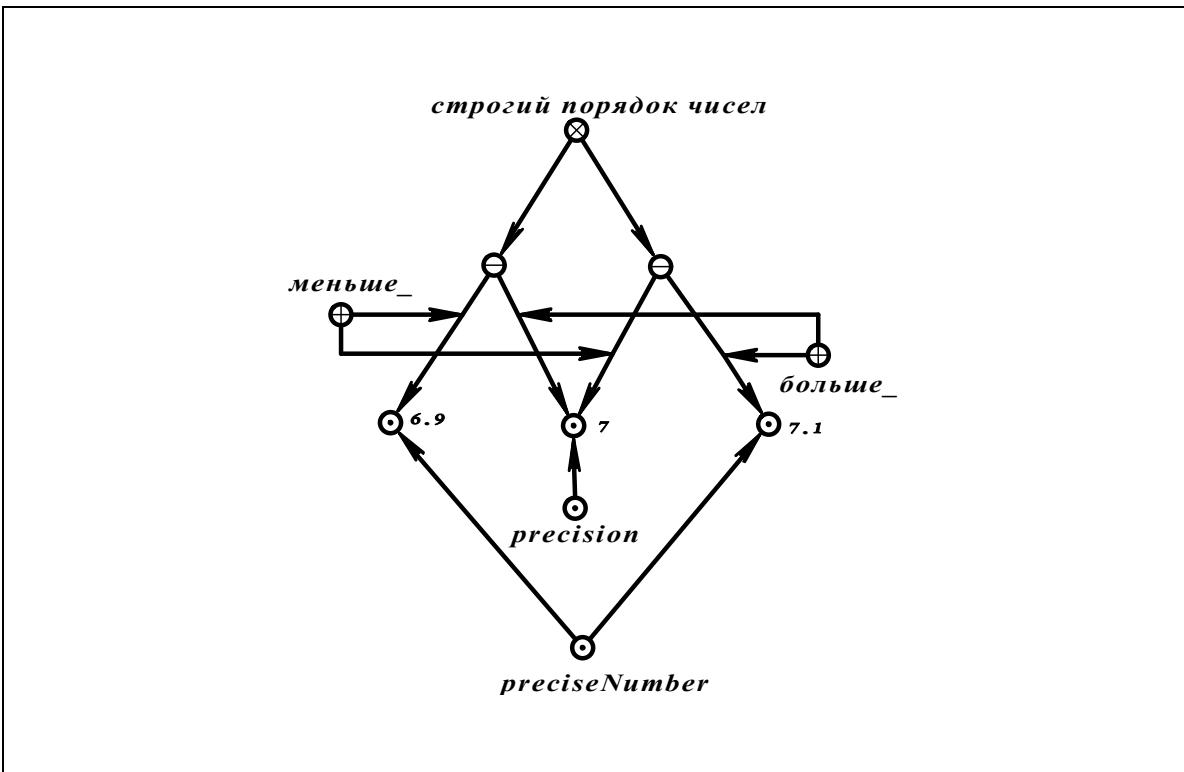
/* некоторый объект имеет массу, равную 30 кг */

Ключевой узел **fuzSet** является знаком унарного отношения "Быть нечетким, неопределенным множеством", т.е. множеством, строгое определение которого отсутствует или, другими словами, отсутствует четкий критерий, позволяющий установить принадлежность или непринадлежность произвольного sc-элемента этому множеству. Характерным признаком нечетких множеств является наличие большого количества нечетких константных sc-дуг, выходящих из sc-узла, обозначающего нечеткое множество. Хотя, конечно, некоторое количество нечетких константных sc-дуг может выходить из sc-узлов, обозначающих также и четкие множества. Но нечеткость таких дуг обусловлена не отсутствием критериев принадлежности таким множествам, а просто недостаточностью (для соответствующих критериев) сведений о конкретных потенциальных элементах этих множеств. Проведение в некоторый константный sc-узел негативной константной sc-дуги из ключевого узла **fuzSet** означает то, что указанный sc-узел обозначает четкое множество. Ключевой узел **fuzSet** относится к группе ключевых узлов, используемых для описания теоретико-множественных соотношений.

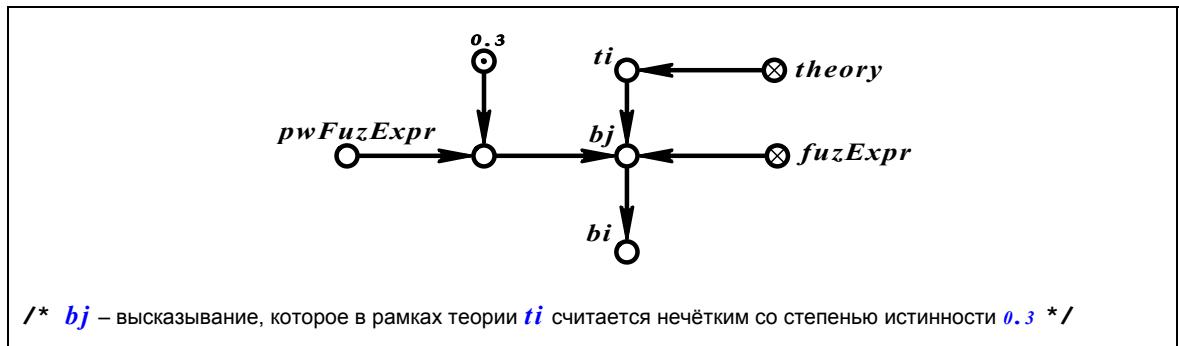
Приведём для сравнения два способа представления точности чисел (с использованием и без использования понятия измерения).



/* Здесь **precision** – знак множества всех неточных чисел, степень точности которых может быть дополнительно уточнена, **pwPrecision** – фактор-множество, множества неточных чисел по степени их точности */



Оценка степени истинности/ложности нечёткого высказывания осуществляется с помощью фактор-множества *pwFuzExpr* по шкале от *0* до *1*.



1.2. Описание динамических систем

Ключевые понятия и идентификаторы ключевых узлов: процесс, ситуация, действие, состояние, нестационарная информационная конструкция, нестационарная предметная область, стационарная константа, нестационарная константа.

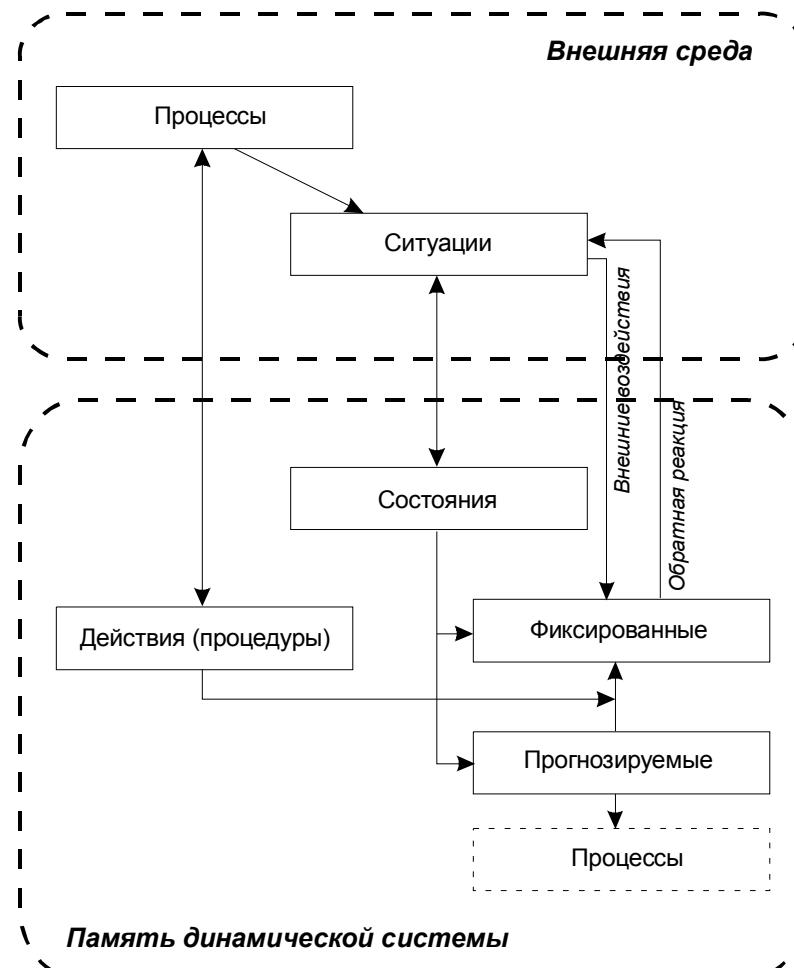
Как уже было отмечено, scl-теория является способом описания того или иного состояния немодифицируемой формальной модели переработки знаний. Немодифицируемые формальные модели переработки знаний всегда соответствуют стационарным предметным областям, а различные состояния таких моделей отражают различные состояния процесса решения задач в рамках этих моделей, т.е. задач, формулируемых по отношению к указанным стационарным предметным областям. В отличие от этого, в ходе выполнения семиотической (т.е. модифицируемой) модели переработки знаний осуществляется не только решение задач в рамках той или иной немодифицируемой формальной модели, но и порождение новых немодифицируемых формальных моделей путем того или иного преобразования имеющихся формальных моделей. Нетрудно заметить, что описать состояние семиотической модели на языке SCL - это значит построить некоторую scl-метатеорию,

описывающую некоторое множество scl-теорий, систему связей между этими scl-теориями, а также всевозможные правила их преобразования.

Одним из примеров такой scl-метатеории является описание нестационарной предметной области, в основе которого лежит трактовка нестационарной предметной области как иерархической системы квазистационарных предметных областей, называемых состояниями (или ситуациями) нестационарной предметной области. Нестационарные предметные области имеют как стационарные свойства, для представления которых, в частности, используются стационарные (неситуативные) связи и отношения, так и нестационарные свойства, зависящие от состояния.

Прежде чем перейти к рассмотрению средств языка SCL для создания динамических систем, приведем некоторые основные используемые понятия. Функционирование динамической системы будем рассматривать как процесс взаимодействия между внешней средой и рабочей памятью динамической системы (см. рис. 6.2.1). В рамках внешней среды будем различать **процессы** и **ситуации**. Таким образом, процессы, которые в памяти динамической системы будем трактовать как некоторые **действия** или процедуры, порождают соответствующие ситуации, которые в памяти динамической системы будем представлять в виде **состояний** этой системы. Таким образом, процесс может задаваться в виде последовательности ситуаций и элементарных действий, приводящих к этим ситуациям. Тогда процесс в рамках динамической системы будем трактовать как выполнение заданных действий между состояниями. Кроме того, описания в памяти динамической системы некоторых состояний будем также называть процессами в том случае, если это описание некоторых конкретных действий системы, направленных на преобразование ее состояний.

Рисунок 6.2.1. Схема функционирования динамической системы



Рассмотрим основные средства описания динамических систем (нестационарных предметных областей) средствами языка SCL. В качестве рабочей памяти динамической системы будем рассматривать sc-память, в которой хранятся и обрабатываются соответствующие scl-конструкции.

Знание о нестационарной предметной области на языке SCL представляется путем его расчленения на множество знаний о квазистационарных предметных областях, каждое из которых описывает некоторое состояние описываемой предметной области, трактуя нестационарную предметную область как стационарную на некотором промежутке времени по отношению к указываемым свойствам. Описание каждого такого состояния оформляется как scl-теория, являющаяся стационарным компонентом (state-компонентом) scl-метатеории, описывающей нестационарную предметную область в целом. Для задания таких scl-метатеорий вводится специальное отношение, обозначаемое ключевым узлом ***theoryDyn***.

Итак, sc-конструкция вида:

theoryDyn → ***ki*** → ***state_*** : ***bj*** ;

семантически означает, что ***bj*** есть высказывание, описывающее некоторое состояние (некоторую ситуацию) той нестационарной предметной области, которая описывается теорией ***ki***. Высказывание ***bj*** может быть либо атомарным, либо конъюнктивным. Элементы атомарного высказывания ***bj*** и элементы union-компонента конъюнктивного высказывания ***bj*** могут быть как константными, так и переменными. При этом, если такой константный sc-элемент является также константой scl-метатеории (элементом union-компонента высказывания ***ki***, если ***ki*** ← ***theoryDyn*** ;), то он называется **стационарной константой**. Если же указанный константный sc-элемент (как узел, так и дуга) не является константой scl-метатеории, то он называется **нестационарной** (ситуативной) **константой**.

Таким образом, state-компонент есть перечисление некоторых свойств, в частности, ситуативных связей нестационарной предметной области, которые сохраняются в течение некоторого отрезка времени. Следовательно, каждый state-компонент, не имеющий свободных переменных, являясь знаком соответствующего высказывания, однозначно соответствует квазистационарному процессу, описываемому этим высказыванием. Каждому такому процессу ставится в соответствие отрезок времени его существования с моментами начала и конца процесса.

Каждое высказывание, являющееся state-компонентом некоторого другого высказывания, в свою очередь, само может также иметь несколько state-компонентов, которые осуществляют разбиение некоторого состояния (процесса) на несколько более "мелких" процессов. Если некоторое высказывание является state-компонентом и имеет свободные переменные, то оно представляет собой высказывание о существовании соответствующего состояния.

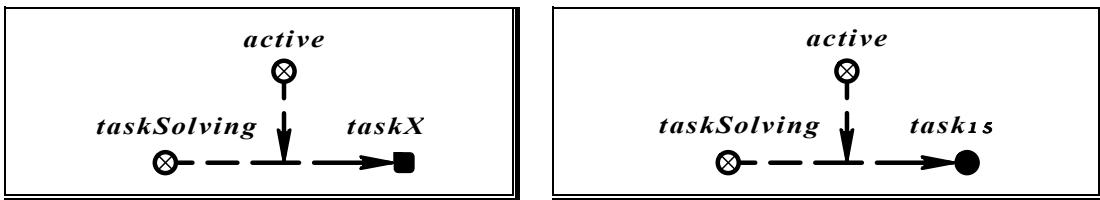
Поскольку моделирование и реализация динамических процессов сводится к анализу в каждый конкретный момент времени некоторой создавшейся к этому времени ситуации, которая определяет текущее состояние системы, необходимо рассмотреть типологию состояний динамической системы. В связи с этим, будем различать следующие типы состояний:

- 1) **фиксированное состояние** - представляет собой некоторое промежуточное состояние динамической системы, фиксируемое в некоторый момент времени и описываемое некоторой конкретной sc-конструкцией, состоящей из стационарных и ситуативных констант соответствующей scl-метатеории. При описании процессов (правил перехода из одного состояния в другое) совокупность **прогнозируемых** (предполагаемых) **состояний** описывается изоморфными sc-конструкциями, в которых ситуационные константы обозначаются переменными sc-элементами. Так, например, на sc-тексте 6.2.1 а) представлен пример описания прогнозируемого состояния, свидетельствующего о том, что в некоторый момент времени интеллектуальная обучающая система (ИОС), рассматриваемая как динамическая система, может перейти в состояние активизации стратегии решения задачи. На sc-тексте 6.2.1 б) представлено соответствующее фиксированное состояние, семантика которого заключается в том, что ИОС в данный момент находится в состоянии решения конкретной задачи;

С С - т е к с т 6 . 2 . 1 . Пример описания на языке SCL а) совокупности прогнозируемых состояний и
б) конкретного фиксированного состояния

а)

б)

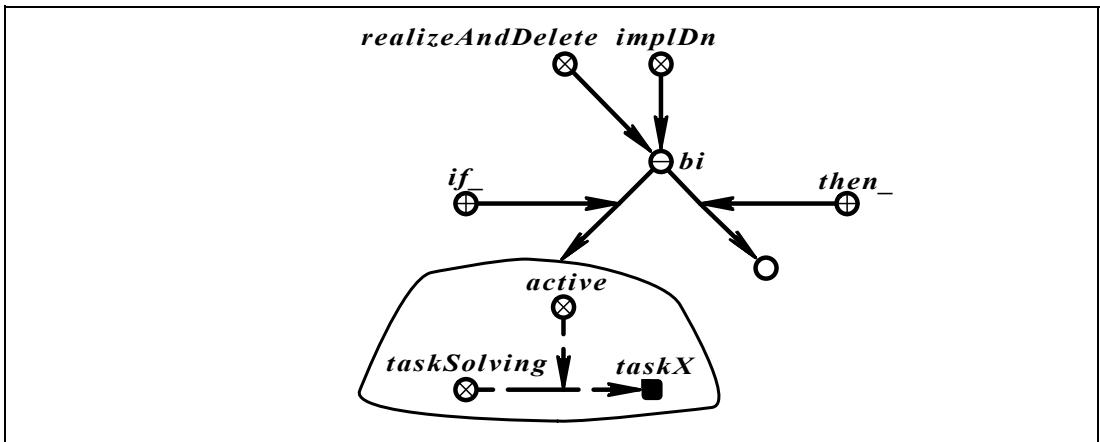


- 2) **состояние перехода** (переходное состояние). Описание переходного состояния представляет собой правило перехода из одного фиксированного состояния в другое. На языке SCL для описания переходных состояний используются ключевые узлы *implDn*, *transfDn*, *transfDnW*, *if_*, *then_*, *worker_*, описание семантики которых будет дано ниже. При формальном описании на языке SCL будем также различать **общее описание переходного состояния** и **частное описание переходного состояния**. Общее описание представляет собой scl-высказывание, которое постоянно хранится в sc-памяти и может быть применено (зафиксировано) многократно. Частное описание генерируется в sc-памяти в некоторый конкретный момент времени в результате каких-либо действий системы и после однократного применения удаляется. Для выделения множества частных описаний переходных состояний в рамках языка SCL используется соответствующее унарное отношение *realizeAndDelete*. Таким образом, если *bi* – знак некоторого состояния, то наличие sc-конструкции вида:

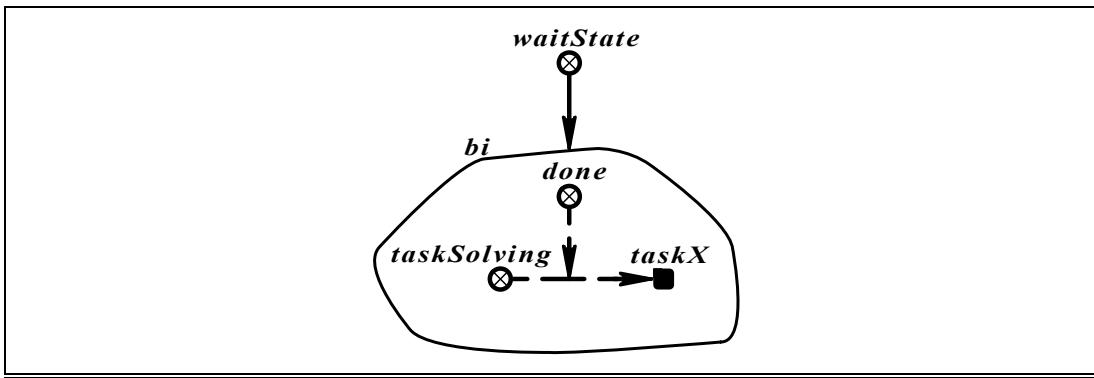
realizeAndDelete → *bi* ;

означает, что после фиксации ситуации, соответствующей описанию состояния *bi*, это описание будет удалено из sc-памяти. Например, как только оба компонента изображенного на scl-тексте 6.2.2 частного описания состояния перехода *bi* преобразуются в фиксированные (см. sc-текст 6.2.1 а)), sc-узелы *bi*, *be* и *bt* будут удалены из sc-памяти вместе с инцидентными им дугами. Состояния перехода будем также называть **процессами**;

SCL - текст 6.2.2. Пример описания на языке SCL частного состояния перехода

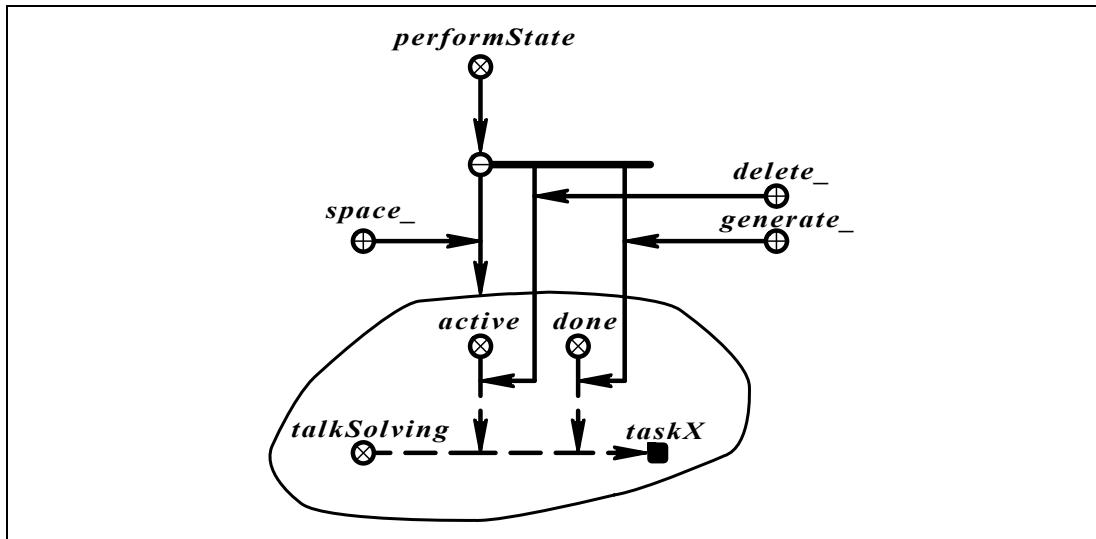


- 3) **состояние ожидания** - при описании на языке SCL помечается в составе нестационарной scl-матеории ключевым узлом *waitState*. SC-узел, обозначающий состояние ожидания, является знаком sc-конструкции, описывающей ожидаемое состояние фрагмента sc-памяти. Примером состояния ожидания в ИОС является ожидание завершения реализации некоторой стратегии обучения (см. scl-текст 6.2.3);

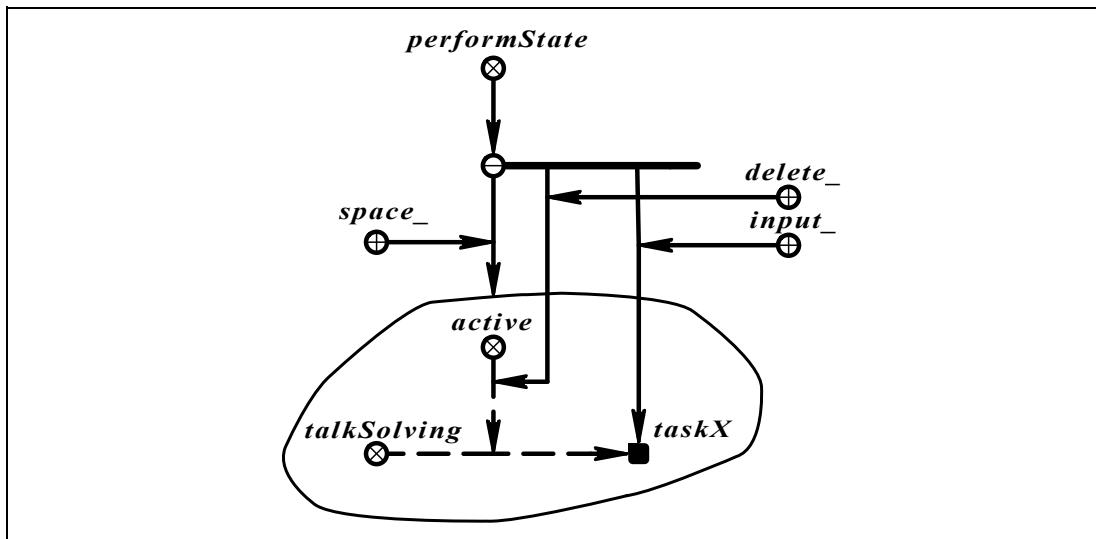
SCL-текст 6.2.3. Пример описания на языке SCL состояния ожидания

- 4) **состояние преобразования памяти** - описывается в составе нестационарной scl-метатеории с помощью отношения *performState*, которое является отношением нефиксированной арности и имеет атрибуты: *space_* - область памяти, которая подлежит преобразованию; *generate_* - указывает на элемент, который необходимо сгенерировать в sc-памяти, *delete_* - указывает на элемент, подлежащий удалению, *input_* - указывает на sc-узел, содержимое которого необходимо загрузить в sc-память. Обработка указанного отношения приведет к реализации процесса по преобразованию заданного фрагмента sc-памяти. На scl-текстах 6.2.4 и 6.2.5 приведены примеры описания состояния преобразования памяти. Смысл scl-текста 6.2.4 заключается в том, что в выделенном фрагменте sc-памяти необходимо удалить sc-дугу, выходящую из узла *active*, и сгенерировать sc-дугу, выходящую из узла *done*. Семантически описание данной операции в составе ИОС означает преобразование стратегии решения задач, обозначенной sc-узлом *taskSolving*, из активной (*active*) в выполненную (*done*). Обработка scl-текста 6.2.5 будет заключаться в том, чтобы сгенерировать sc-дугу, выходящую из узла *active* и входящую в соответствующую дугу (тем самым активизировав стратегию решения задач ИОС), а также загрузить в память содержимое sc-узла *taskX* (в котором в виде sc-текста может храниться условие подлежащей решению задачи).

SCL-текст 6.2.4. Пример описания на языке SCL состояния преобразования памяти

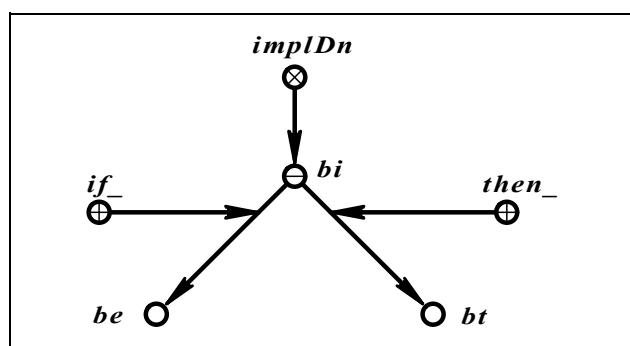


SCL-текст 6.2.5. Пример описания на языке SCL состояния преобразования памяти

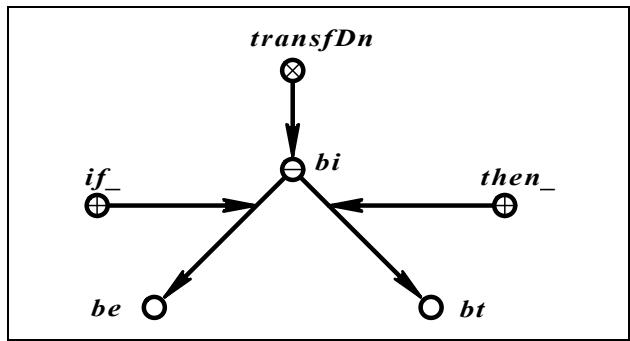


Ниже, на scl-текстах 6.2.6-6.2.8 приведены общие представления отношений для описания состояний перехода в динамической системе.

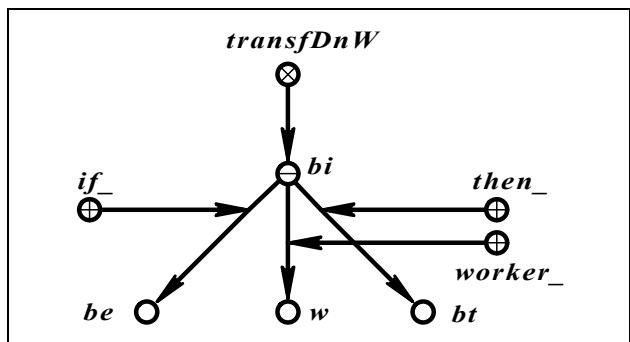
SCL-текст 6.2.6. Отношение *implDn*



Данная конструкция семантически означает, что для каждого состояния (state-компоненты) процесса *bi*, которое удовлетворяет условию *be*, имеет место также и *bt*, т.е. *implDn*-компонент описывает общее свойство некоторого класса состояний соответствующего процесса.

SCL-текст 6.2.7. Отношение *transfDn*

Данная конструкция семантически означает, что каждое состояние процесса *bi*, удовлетворяющее условию *be*, обязательно (независимо ни от чего) преобразуется в следующее за ним состояние, описываемое высказыванием *bt*. Нетрудно заметить, что *transfDn*-высказывания есть способ формального описания всевозможных причинно-следственных связей.

SCL-текст 6.2.8. Отношение *transfDnW*

Данная конструкция описывает преобразование состояния (ситуации) *be* в непосредственно следующее за ним состояние *bt*, к которому может привести некий исполнитель *w*. Описание условий, необходимых исполнителю *w* для выполнения указанного преобразования, также входит в состав высказывания *be*.

Перечислим также ряд отношений, заданных на множестве процессов и описывающих различные соотношения между процессами.

Пусть *bi*, *bj*, *be*, *bt*, *ti* - теории, описывающие нестационарные или квазистационарные процессы. Тогда:

1) конструкция

localBeginStateDn \rightarrow *bi* ;

означает, что процесс *bi* имеет ограничение по моменту своего начала, а конструкция

localBeginStateDn \rightarrowtail *bi* ;

означает, что для процесса *bi* не существует такой точки на оси времени, по отношению к которой момент начала процесса *bi* был бы позже, т.е. здесь момент начала процесса *bi* стремится к минус бесконечности;

2) конструкция

localEndStateDn \rightarrow *bi* ;

означает, что процесс *bi* имеет ограничение по моменту завершения, а соответственно этому конструкция

localEndStateDn \rightarrowtail *bi* ;

означает, что процесс *bi* длится без конца;

3) конструкция

localBeginStateDn , localEndStateDn → bi ;

означает, что процесс **bi** полностью локализован во времени;

4) конструкция

localSpaceStateDn → bi ;

означает, что процесс **bi** ограничен (локален) в пространстве, т.е. в каждый момент времени, в каждом состоянии процесс **bi** не выходит за рамки некоторой пространственной области. При этом в разные моменты времени процесс **bi** может находиться в разных местах пространства, т.е. **bi** может перемещаться в пространстве;

5) конструкция

bi → state_ : bj ;

означает, что процесс **bj** является частью (во времени) процесса **bi**, т.е. является этапом (стадией, подпроцессом) процесса **bi**;

6) конструкция

bi → beginStateDn_ : bj ;

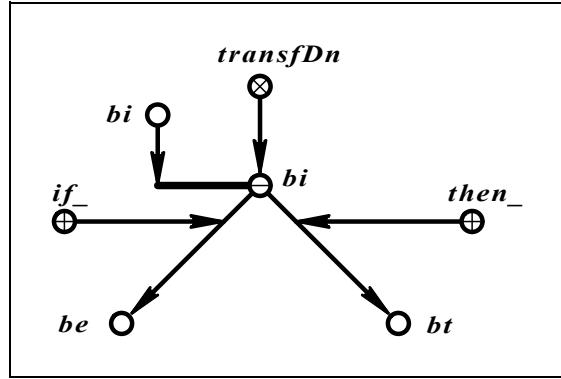
означает, что процесс **bj** является начальной стадией (начальным, стартовым этапом, этапом "рождения") процесса **bi**, т.е. таким подпроцессом процесса **bi**, который начался одновременно с ним, но закончился, естественно, раньше (поскольку это подпроцесс);

7) конструкция

bi → endStateDn_ : bj ;

означает, что процесс **bj** является конечной стадией (конечным, финишным этапом, этапом завершения) процесса **bi**, т.е. таким подпроцессом процесса **bi**, который одновременно с ним закончился;

8)



данная конструкция означает то, что процессы **ti**, **be** и **bt** являются подпроцессами процесса **bi**. При этом **ti** есть процесс преобразования (трансформации) процесса **be** в следующий за ним (во времени) процесс **bt**, т.е. момент завершения процесса **be** совпадает с моментом начала переходного процесса **ti**, а момент завершения процесса **ti** совпадает с моментом начала (появления, "рождения") процесса **bt**;

9) конструкция

eqBeginStateDn → (bi , bj) ;

означает, что процессы (события) **bi** и **bj** одновременно начались. Здесь **eqBeginStateDn** есть свойство "быть одновременно начавшимися процессами", заданное на множестве процессов;

10) конструкция

$$eqEndStateDn \rightarrow (bi , bj) ;$$

означает, что процессы *bi* и *bj* одновременно завершились;

11) конструкция

$$eqDurationStateDn \rightarrow (bi , bj) ;$$

означает, что процессы *bi* и *bj* имеют одинаковую длительность (одинаковое "время жизни"). При этом начаться они могут в разное время;

12) конструкция

$$comprBeginStateDn \rightarrow (bi , grt_ : bj) ;$$

означает, что процесс *bj* начался позже процесса *bi* ;

13) конструкция

$$comprEndStateDn \rightarrow (bi , grt_ : bj) ;$$

означает, что процесс *bj* кончился позже процесса *bi* ;

14) конструкция

$$comprDurationStateDn \rightarrow (bi , grt_ : bj) ;$$

означает, что процесс *bj* имеет большую длительность, чем процесс *bi* ;

15) конструкция

$$nextTimeStateDn \rightarrow (bi , next_ : bj) ;$$

означает, что момент завершения процесса *bi* совпадает с моментом начала процесса *bj*. При этом совсем не обязательно, чтобы *bi* и *bj* были стадиями (состояниями, подпроцессами) одного и того же процесса.

Кроме того, для процессов (событий) можно ввести такие измеряемые параметры, как отметка времени начала процесса, длительность процесса. Подробнее об этом см. в подразделе 6.1.

Помимо описанных изобразительных средств для обработки знаний, описывающих нестационарные (динамические) предметные области, имеются соответствующие операции. Перечислим основные из них:

- 1) операции обработки состояний, имеющих некоторое общее свойство. Иначе говоря, это операции реализации implDn-высказываний. Данный класс операций аналогичен операциям реализации продукции в прямом и обратном направлении;
- 2) операции реализации причинно-следственных связей между состояниями (transfDn-высказываний). С помощью данного класса операций осуществляется переход динамической системы из одного фиксированного состояния в другое;
- 3) операции реализации состояний, вызванных некоторым исполнителем, т.е. transfDnW-высказываний. Данный класс операций разбивается на три вида. К первому виду относится операция, которая инициируется в случае появления в БЗ некоторого активного исполнителя, который был ранее описан в составе transfDnW-высказывания. В процессе реализации данной операции производится поиск в БЗ всех высказываний, содержащих описание активного исполнителя, а затем из них выбирается то, в одном из компонентов которого описано состояние, в котором находится в текущий момент система. Далее производится переход из текущего состояния в следующее согласно найденному transfDnW-высказыванию. Ко второму виду операций данного класса относится операция, которая инициируется в случае наличия в sc-памяти состояния, описанного в if-компоненте обрабатываемого ею transfDnW-высказывания. В процессе реализации данной операции осуществляется поиск соответствующего активного исполнителя путем формирования запроса. После получения подтверждения от исполнителя осуществляется переход в состояние, описанное

в then-компоненте обрабатываемого высказывания. К третьему виду операций реализации состояний, вызванных некоторым исполнителем, относится операция, инициируемая в случае перехода системы в состояние, описанное в then-компоненте обрабатываемого transfDnW-высказывания. Результатом выполнения данной операции является формирование сообщения исполнителю о том, что система перешла в соответствующее состояние;

- 4) операция поддержки состояний ожидания, т.е. waitState-состояний. Целью данной операции является поиск в памяти sc-конструкций, представляющих собой описание фиксированного состояния, изоморфной конструкции, описанной в обрабатываемом прогнозируемом состоянии ожидания. Данная операция является выполненной успешно в случае успешного поиска указанной конструкции. В обратном случае поиск возобновляется;
- 5) операции реализации состояний преобразования памяти, т.е. performState-состояний. Данная группа операций на этапе выполнения разбивается на виды, соответствующие тому, какие действия нужно произвести над sc-памятью. Каждый из видов является аналогом scl-оператора в обобщенном виде. Первая из операций данного класса выполняет генерацию в sc-памяти sc-элементов, помеченных в описании performState-состояния атрибутом *generate*_. Вторая осуществляет удаление sc-элементов, помеченных атрибутом *delete*_. Результатом выполнения третьей операции данного класса является загрузка в sc-память содержимого sc-узла, помеченного атрибутом *input*_ в составе performState-состояния;
- 6) операции поддержки различных соотношений между состояниями. Данный класс scl-операций разбивается на виды, каждый из которых осуществляет обработку отношений, задаваемых с помощью следующих ключевых узлов: *localBeginStateDn*, *localEndStateDn*, *localSpaceStateDn*, *beginStateDn_*, *endStateDn_*, *eqBeginStateDn*, *eqEndStateDn*, *eqDurationStateDn*, *comprBeginStateDn*, *comprEndStateDn*, *comprDurationStateDn*, *nextTimeStateDn*.

Каждая операция, таким образом, в рамках динамической системы представляет собой некий процесс по обработке соответствующего типа знаний.

1.3. Описание целей в графодинамических ассоциативных машинах

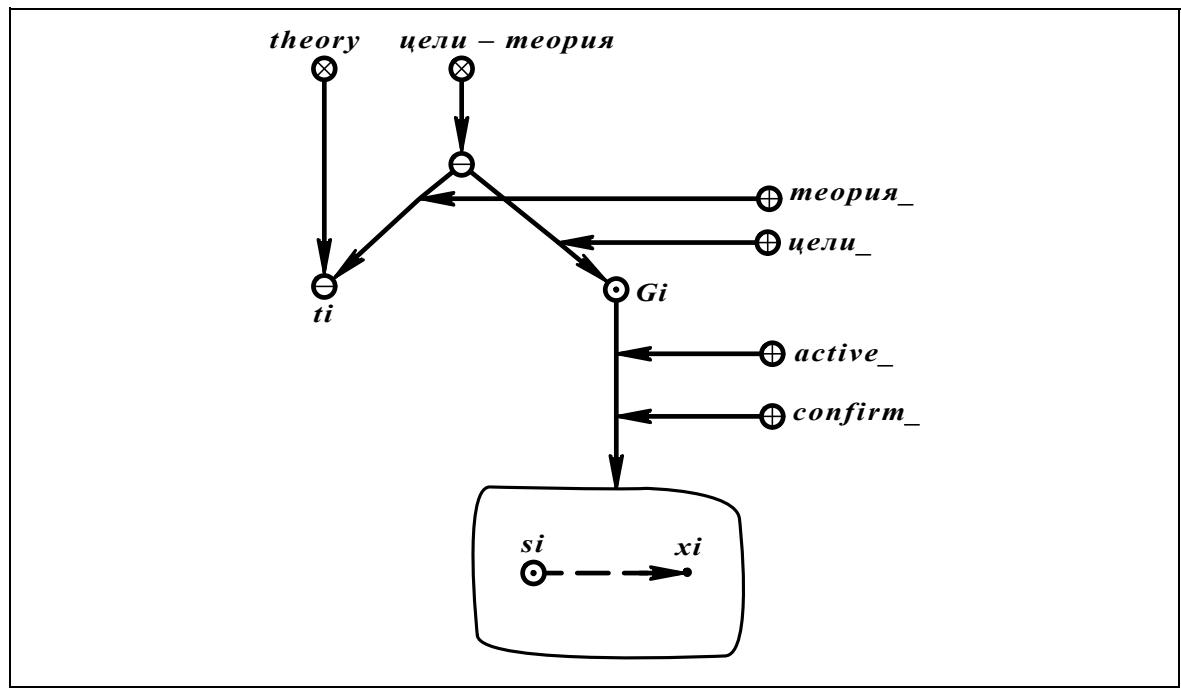
Ключевые понятия и идентификаторы ключевых узлов: цель, задание, команда, подцель, информационная цель, поведенческая цель, задача.

Важнейшей частью любого языка представления знаний являются средства явного описания целей, поскольку по принципу инициирования элементарных информационных процессов машины переработки знаний относятся к классу машин, управляемых потоком целей (см. подраздел 1.3.).

Цель (задание) – это либо желаемое и подлежащее достижению состояние информационной конструкции, хранимой в памяти машины переработки информации (такие цели будем называть информационными), либо желаемое и подлежащее достижению состояние внешней среды (такие цели будем называть поведенческими или внешними). Частным видом описания информационных целей можно считать команды – информационные операторы или программы, которые представляют собой описание подлежащих выполнению действий, направленных на переработку хранимых в памяти информационных конструкций. Действительно, в информационном операторе или информационной программе путь и не явно, но все же содержится указание на будущее (целевое) состояние хранимой информационной конструкции, к которому машина переработки информации стремится (т.е. указание на состояние, которое является результатом реализации оператора или программы). Напомним при этом, что реализация процедурных программ сводится к реализации операторов, входящих в эти программы. Совершенно аналогичным образом частным видом описания поведенческих (внешних) целей можно считать поведенческие операторы или программы, направленные на преобразование внешней среды. Принципиальная разница между оператором, программой и описанием цели, которое не является ни оператором, ни программой, заключается в том, что оператор и программа есть описание цели, совмещенное с явным указанием метода (способа) достижения этой цели. Тогда, как описание цели, не являющееся ни оператором, ни программой, не содержит в себе явного указания на метод (способ) достижения такой цели. Этот метод будет зависеть от того, какой контекст имеет указанная цель. Анализ этого контекста и выбор соответствующего метода достижения цели как раз и составляет основу всех моделей переработки знаний. Такие цели будем называть информационными заданиями (запросами, вопросами, описаниями информационных потребностей). В данном подразделе будут рассмотрены описания информационных целей, не являющиеся ни операторами, ни программами. Принципы представления операторов и программ средствами языка SC рассматриваются в [411] ([ПрогрВАМ-2001кн](#)). Таким образом, будем различать цели на задания и команды.

Для представления информационных заданий в язык SCL вводятся специальные ключевые узлы (*goal*, *confirm_*, *deny_*), которые непосредственно определяют целевые состояния самого общего вида. Все многообразие целей задается различными контекстами указанных целей общего вида. На scl-тексте 6.3.1 показан самый общий вид информационных заданий, представляемых с помощью ключевого узла *confirm_*. Выполнение такого вида задания предполагает довольно сложный анализ и изменение всей окрестности (всего контекста) соответствующей всем sc-переменным включённым в задание. При этом характер и метод (алгоритм) такого изменения заранее не известен и определяется структурой контекста sc-переменных.

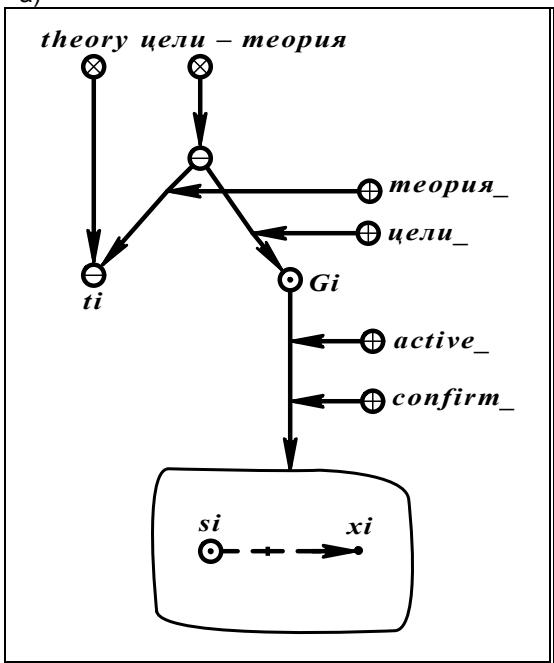
SCL-текст 6.3.1. Задание на преобразование текущего состояния обрабатываемой scl-теории *ti* таким образом, чтобы было подтверждено наличие константной позитивной sc-дуги, выходящей из узла *si* и входящей в элемент *xi* (в зависимости от контекста, это задание может подразумевать превращение соответствующей негативной или нечеткой sc-дуги в позитивную). *Gi* – множество целей формальной теории *ti*



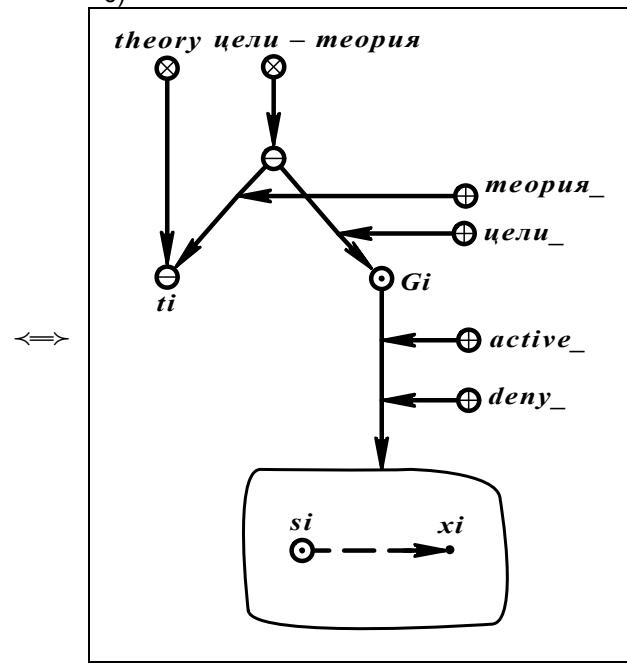
Используемые в абстрактной scl-машине частные виды информационных заданий, описываемых с помощью ключевого узла *confirm_*, показаны на scl-текстах 6.3.3 – 6.3.15.

SCL-текст 6.3.2. Задание на преобразование текущего состояния обрабатываемой scl-теории *ti* таким образом, чтобы было опровергнуто наличие константной позитивной sc-дуги, выходящей из узла *si* и входящей в элемент *xi* (в зависимости от контекста, это задание может подразумевать превращение соответствующей позитивной или нечеткой sc-дуги в негативную)

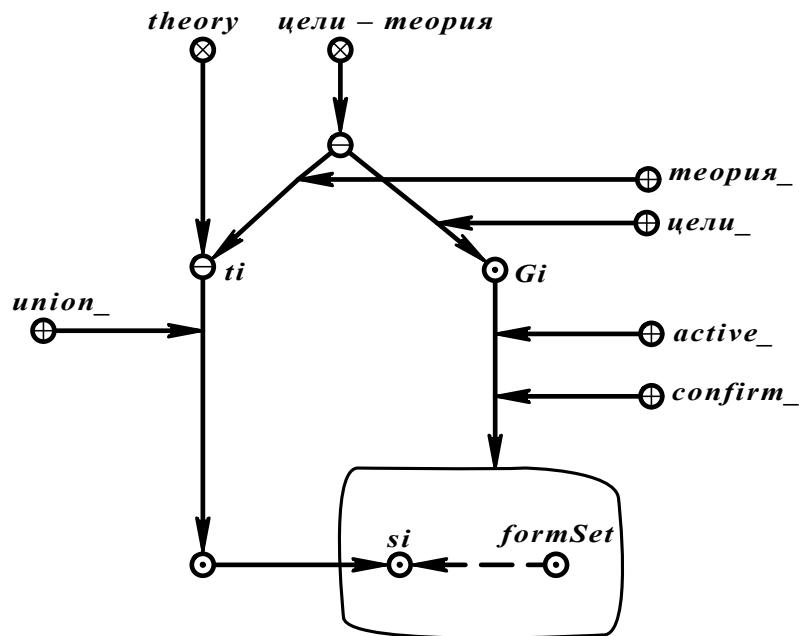
а)



б)



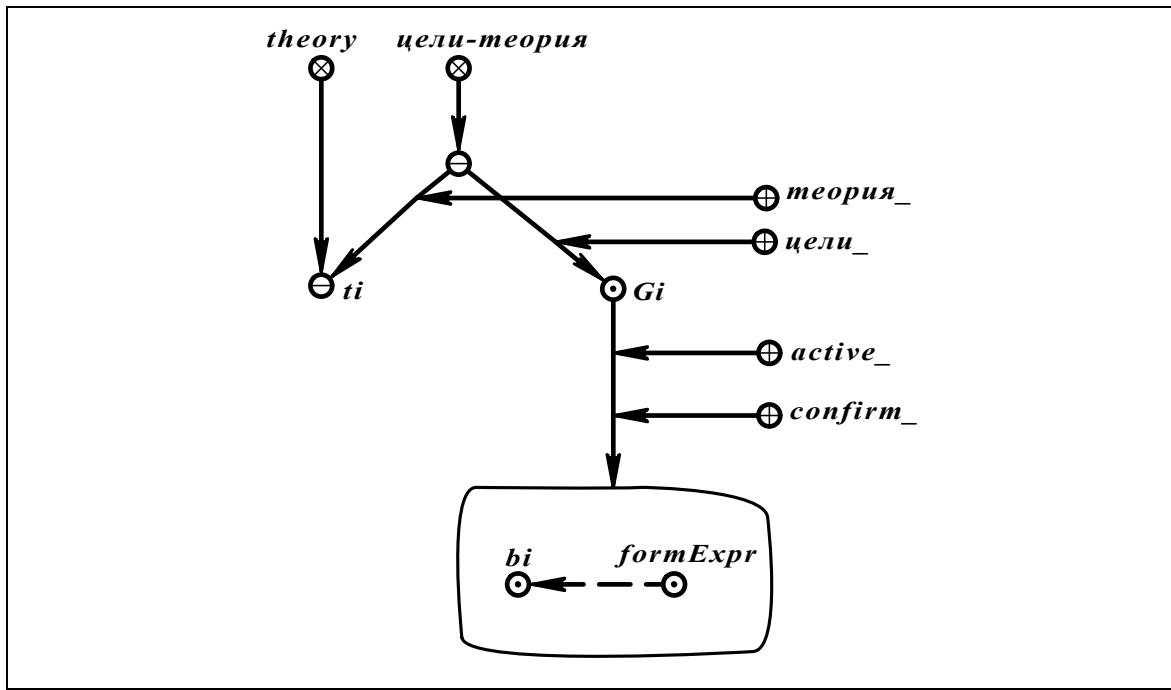
SCL-текст 6.3.3. Задание на построение полного перечня всех элементов множества *si*



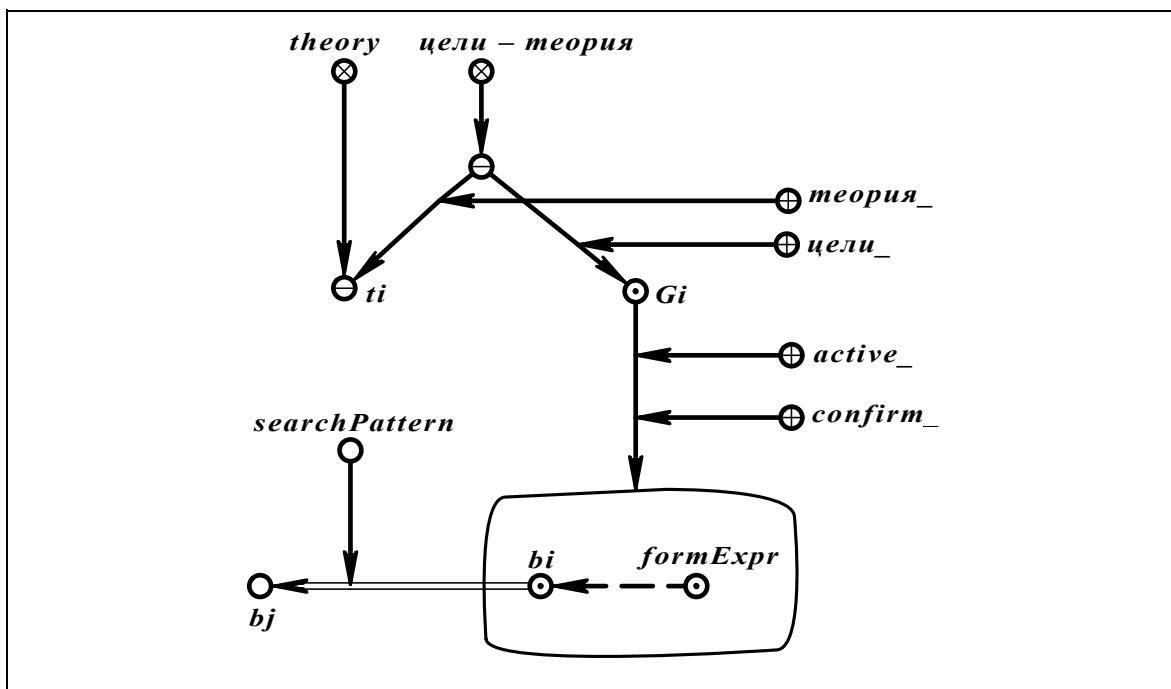
На ск-тексте 6.3.3 приведен пример задания на построение полного перечня всех элементов множества *si*. Результатом выполнения этого задания является построение всех константных sc-дуг, выходящих из sc-узла *si*. При этом входить указанные sc-дуги могут как в константные, так и в переменные sc-элементы. Заметим также, что изначально из узла *formSet* константная дуга может как выходить – нечёткая, либо негативная, так и вообще отсутствовать.

На scl-тексте 6.3.4 приведён пример задания на формирование логической формулы *bj*, входящей в состав (подчиненной) scl-теории *ti*. Напомним, что все константные sc-узлы, обозначающие scl-формулы, входящие в состав scl-теории, а также все константные sc-дуги, инцидентные этим sc-узлам, по умолчанию считаются константами указанной scl-теории. Сформировать scl-формулу – это значит сформировать не только компоненты этой scl-формулы, но и все остальные scl-формулы, входящие в её состав, вплоть до атомарных.

SCL-текст 6.3.4. Задание на формирование scl-формулы bj , входящего в состав scl-теории ti



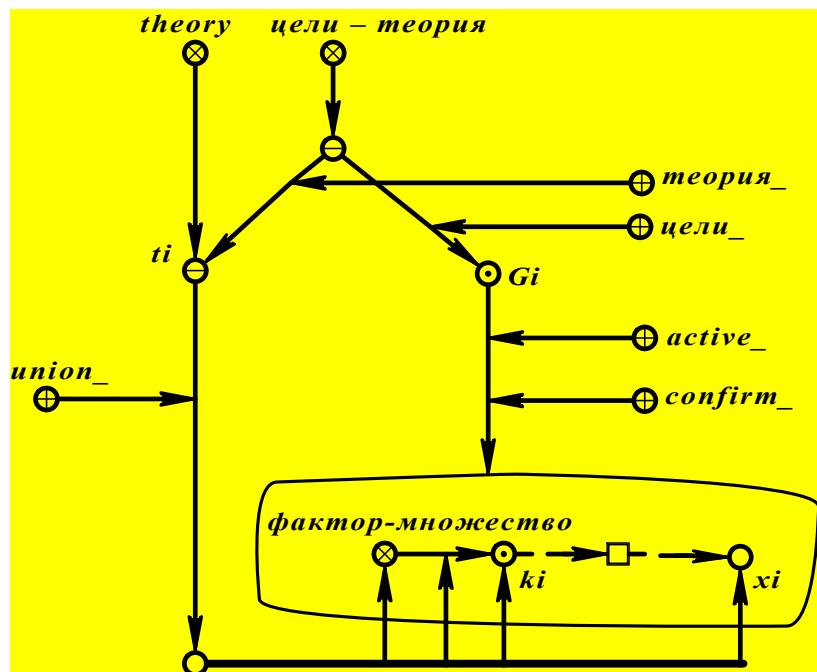
SCL-текст 6.3.5. Общий вид задания на формирование или информационный поиск



На scl-тексте 6.3.5 приведен общий вид задания на формирование или информационный поиск одного из высказываний (bi), удовлетворяющих образцу поиска, заданному высказыванием bj . На высказывание bj должен неявно навешиваться квантор существования. Формируемое высказывание bi отличается от высказывания bj только тем, что в нем все переменные, связываемые в высказывание bj квантором существования, заменяются на константы теории ti . Это задание является частным по отношению к заданию на формирование scl-формулы. Граф, информационная конструкция, описывающая частную цель всегда является подграфом графа, описывающего цель более общего вида, что можно видеть из рисунков 6.3.5 и 6.3.4.

На scl-тексте 6.3.6 приведен пример задания на классификацию (распознавание), т.е. на определение того, к какому классу из заданного семейства *ki* относится заданный объект *xi*. Как можно видеть это задание не является частным по отношению к заданию, показанному на scl-тексте 6.3.4.

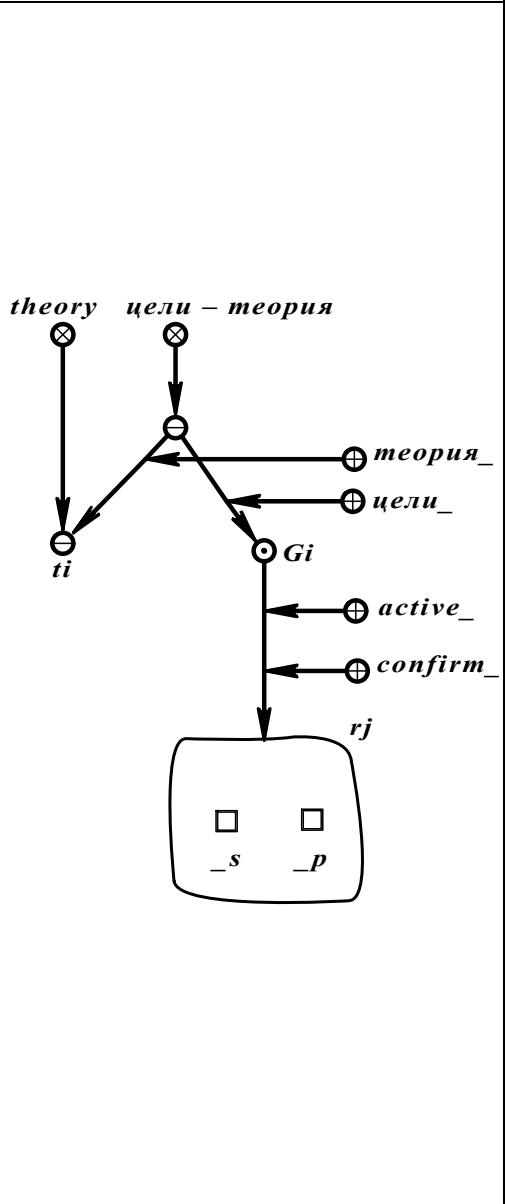
SCL-текст 6.3.6. Задание на классификацию



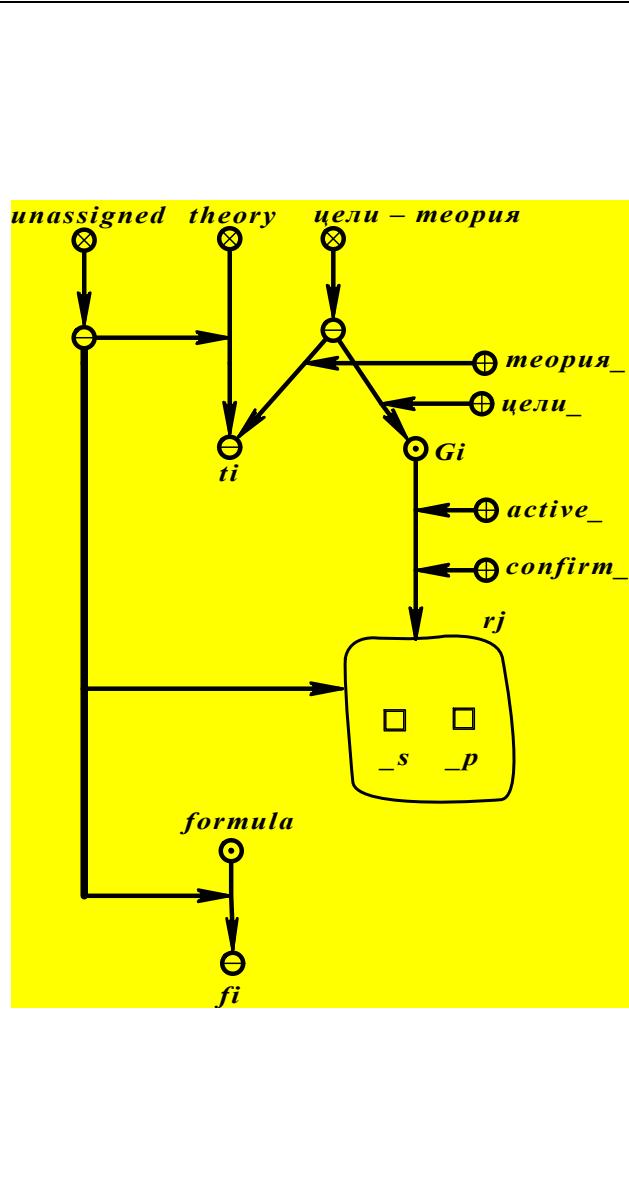
На scl-тексте 6.3.7 приведен пример задания на логический вывод или на информационный поиск всех фактографических высказываний, удовлетворяющих заданному образцу поиска. Вариант б) – это частный вид такого задания, по отношению к заданию варианта а).

SCL-текст 6.3.7. Задание на логический вывод формулы ri , удовлетворяющей ограничениям заданным формулой fi :

а)

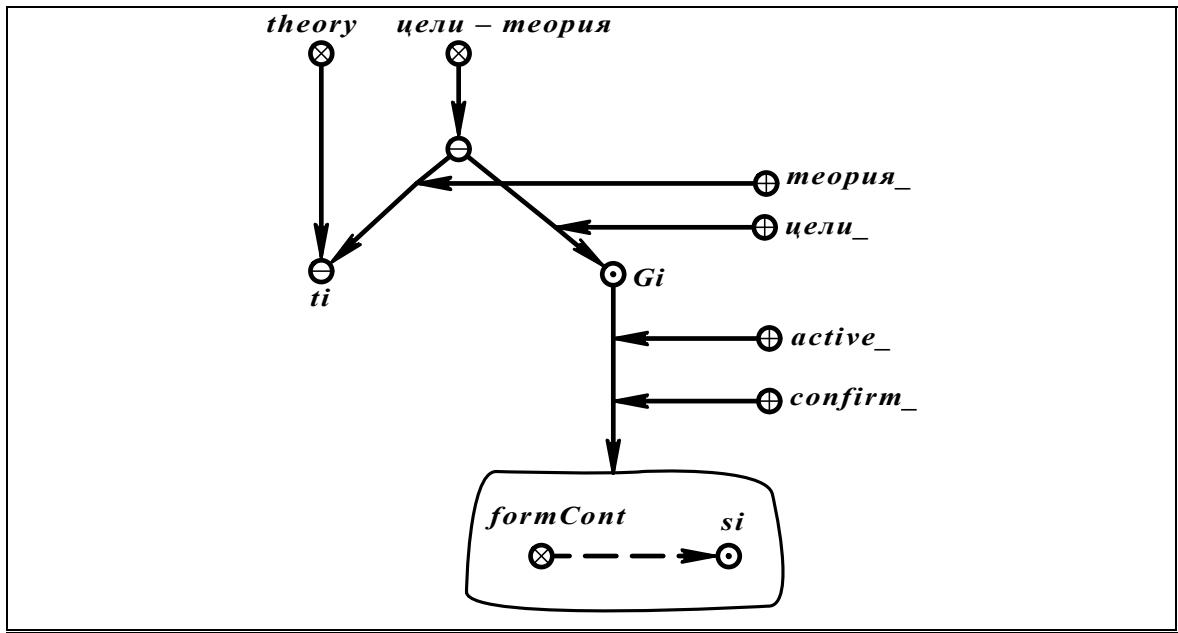


б)

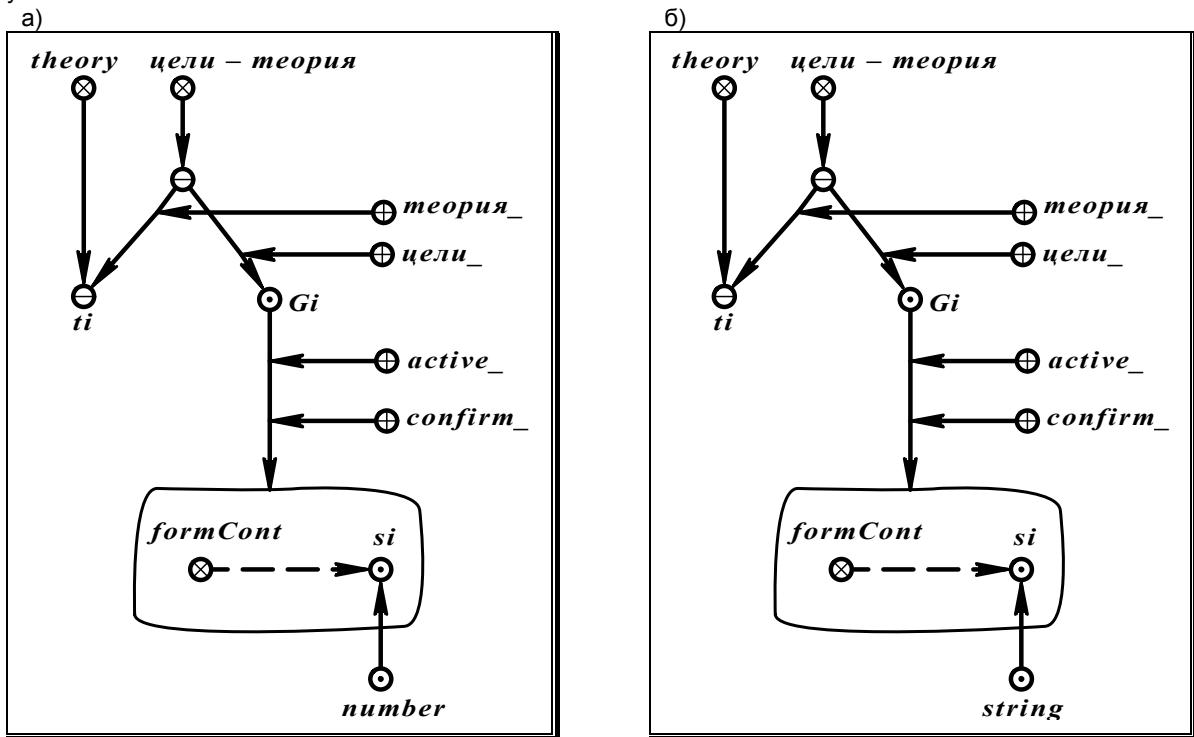


На scl-тексте 6.3.9 а) приведен пример задания на вычисление числа, обозначенного sc-узлом si . В результате выполнения этого задания sc-узел si может быть склеен с другим числовым sc-узлом, имеющим такое же фиксированное содержимое. Числовые sc-узлы, имеющие одинаковое и фиксированное содержимое, семантически эквивалентны. Данное задание является частным по отношению к заданию на формирование содержимого (см. scl-текст 6.3.8)

SCL-текст 6.3.8. Задание на формирование содержимого sc-узла *si*

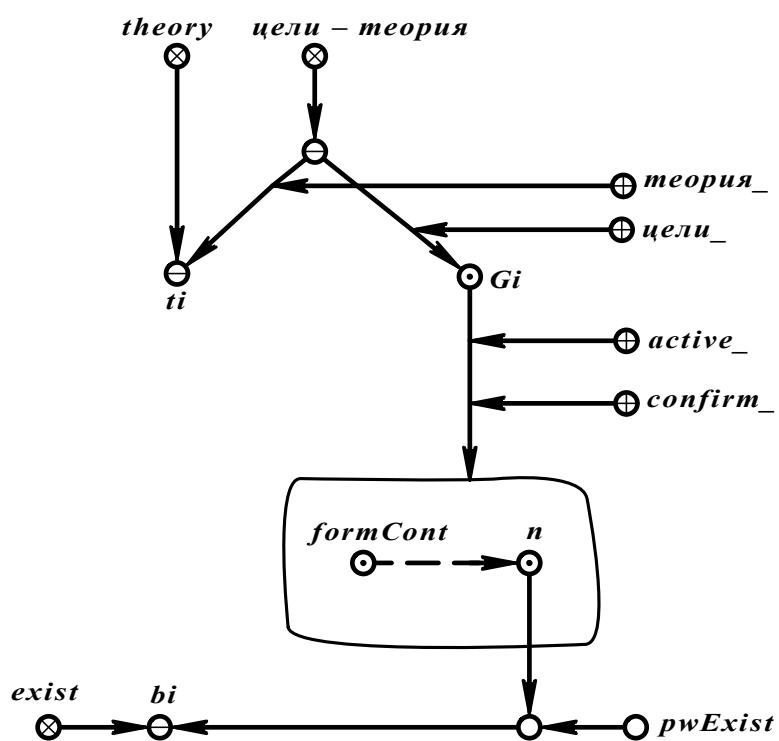


SCL-текст 6.3.9. Задание на вычисление содержимого определённого типа (числа, строки) узла *si*

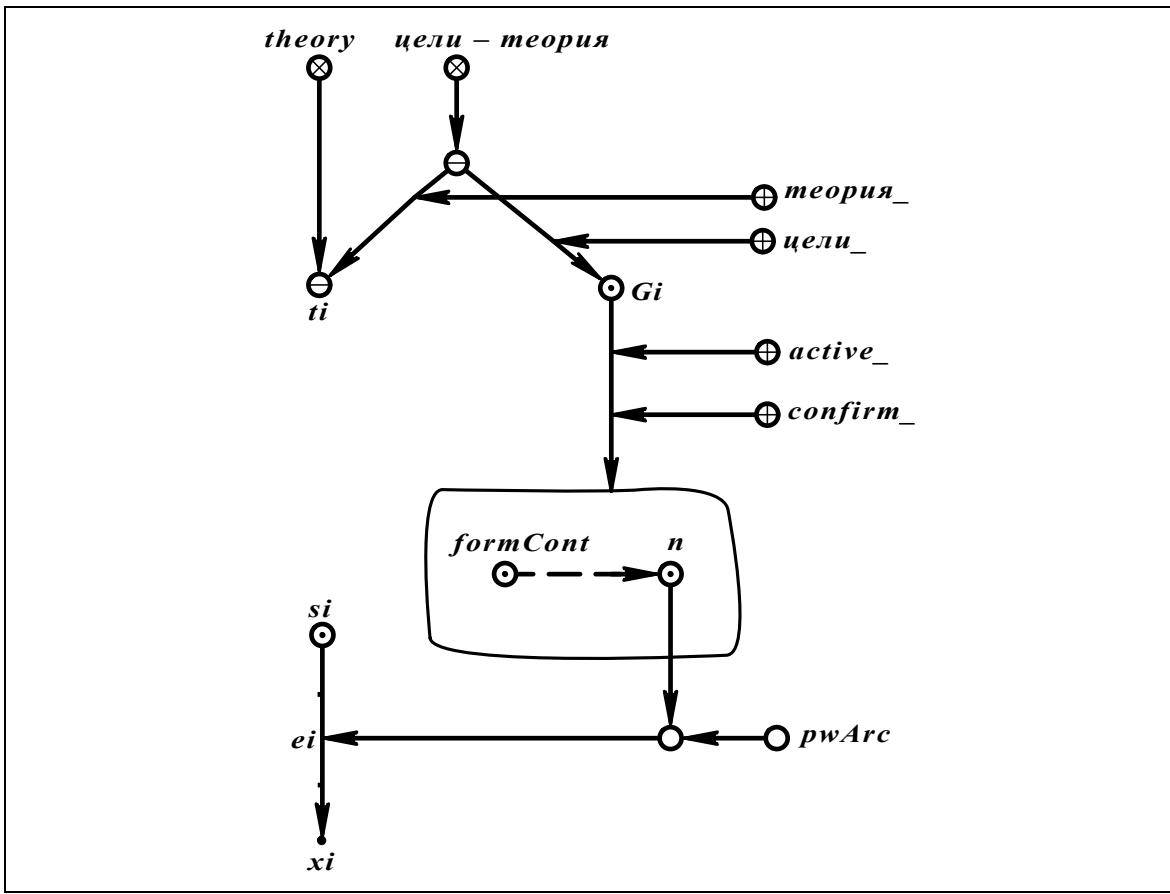


На scl-тексте 6.3.10 приведен пример задания на определение количества всех высказываний, удовлетворяющих формуле *bj* в рамках scl-теории *ti*. В частности, это может быть заданием на перечисление объектов, удовлетворяющих заданным свойствам. Это задание является частным по отношению к заданию на вычисление числа (см. scl-текст 6.3.9).

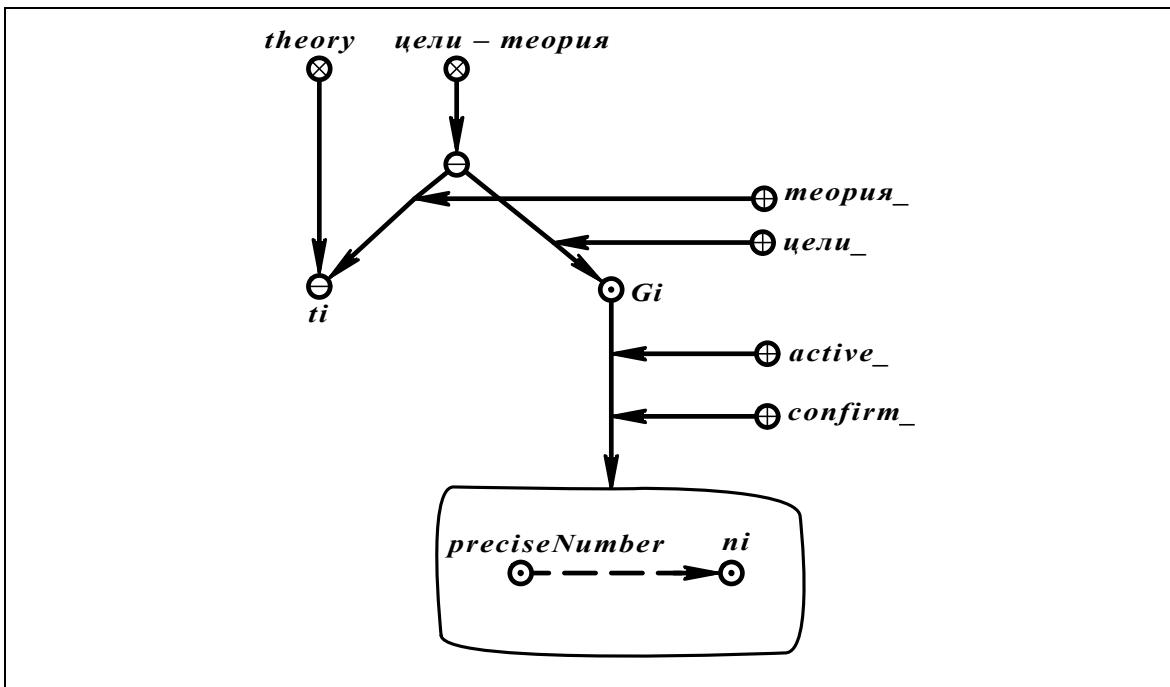
SCL-текст 6.3.10. Задание на определение количества всех (фактографических) высказываний, удовлетворяющих формуле *bj*



SCL-текст 6.3.11. Задание на определение степени четкости нечеткой sc-дуги *ei*

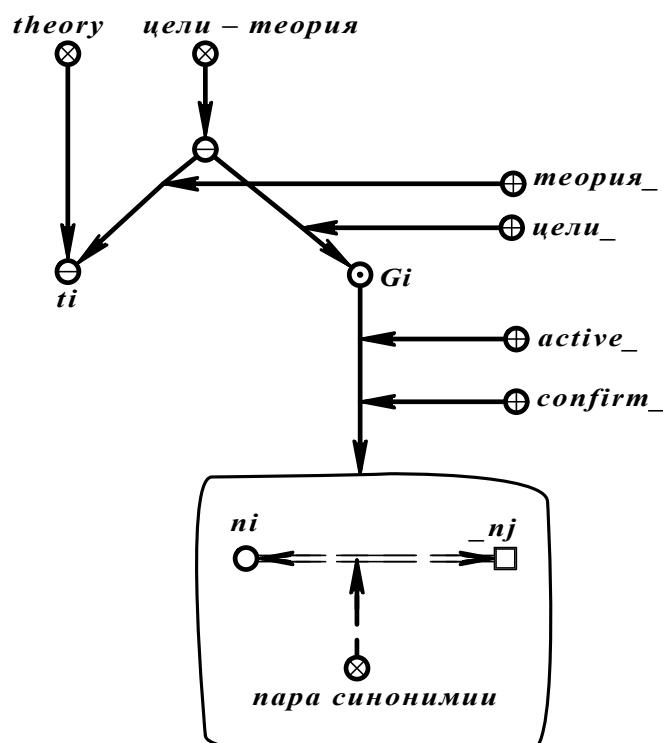


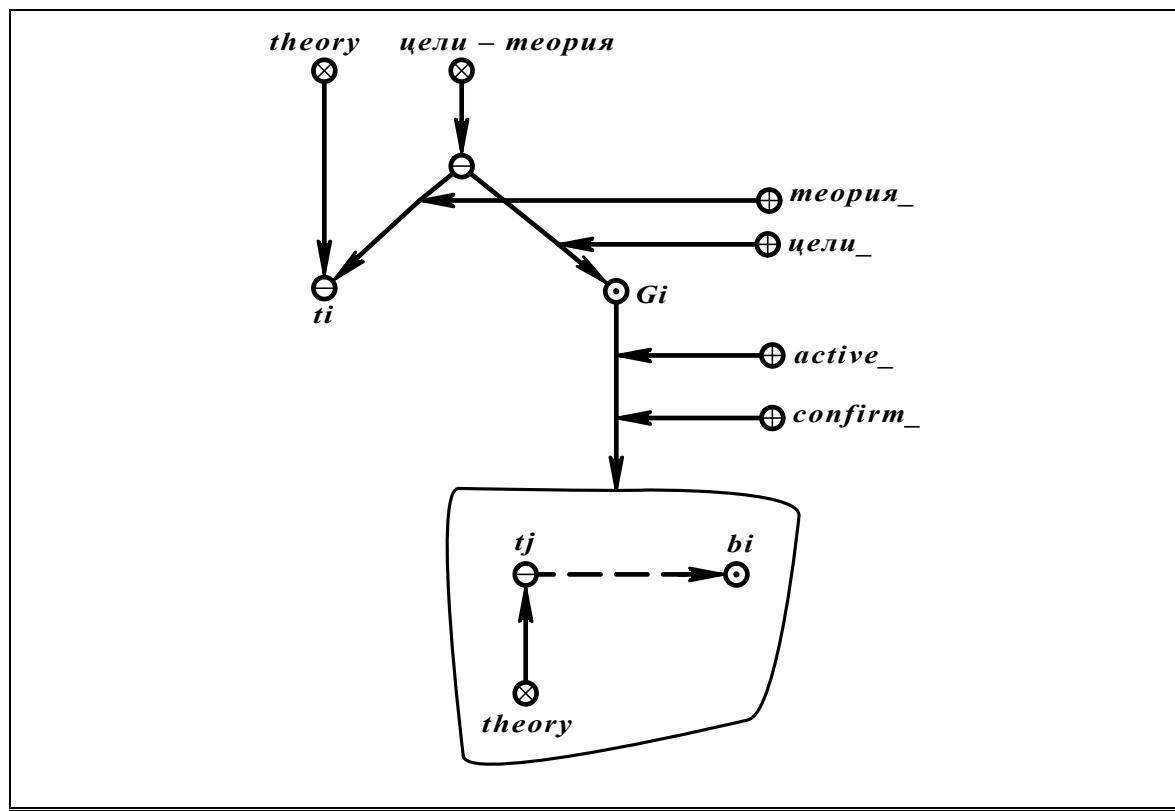
SCL-текст 6.3.12. Задание на повышение точности числа *ni*



На scl-тексте 6.3.13 приведен пример задания на поиск среди всех констант scl-теории *ti* (т.е. среди констант, явно перечисленных в текущем состоянии scl-теории) такого sc-узла, который семантически эквивалентен sc-узлу *ni*. Результатом выполнения этого задания является склеивание sc-узла *ni* с найденным sc-узлом.

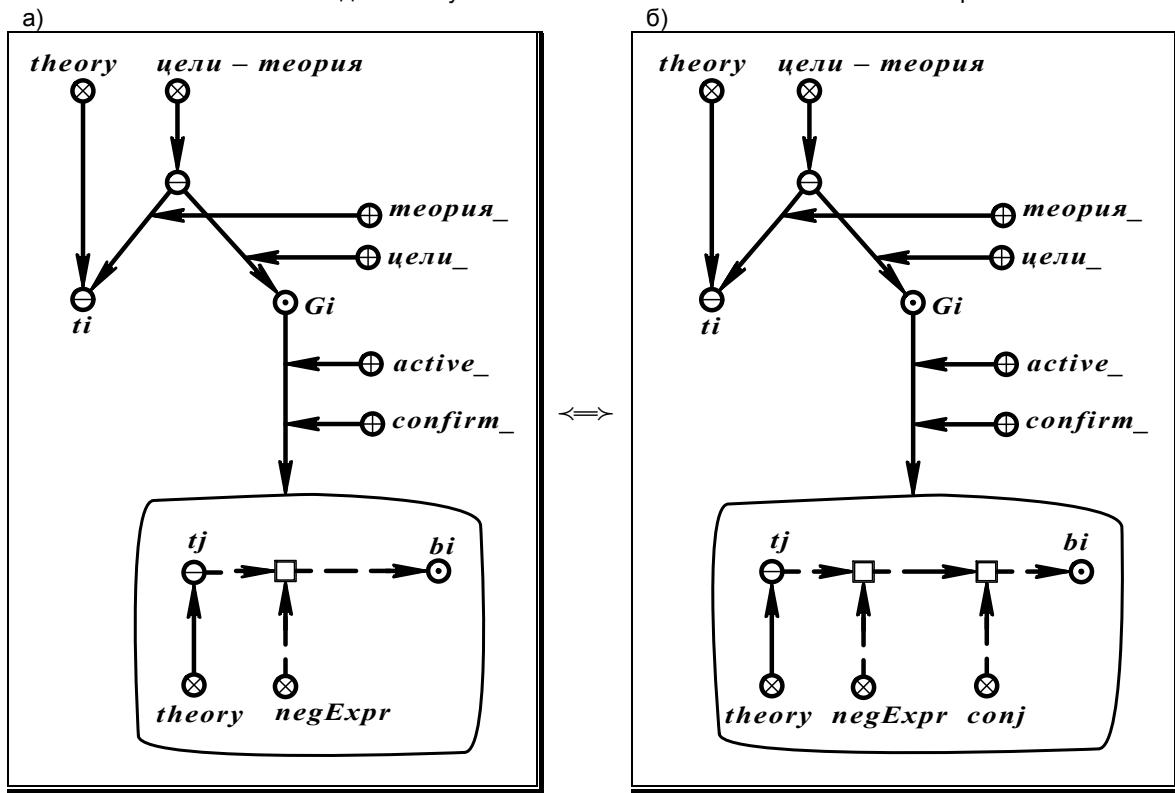
SCL-текст 6.3.13. Задание на поиск семантически эквивалентных sc-элементов



SCL-текст 6.3.14. Задание на установление истинности высказывания *bi* теории *ti*

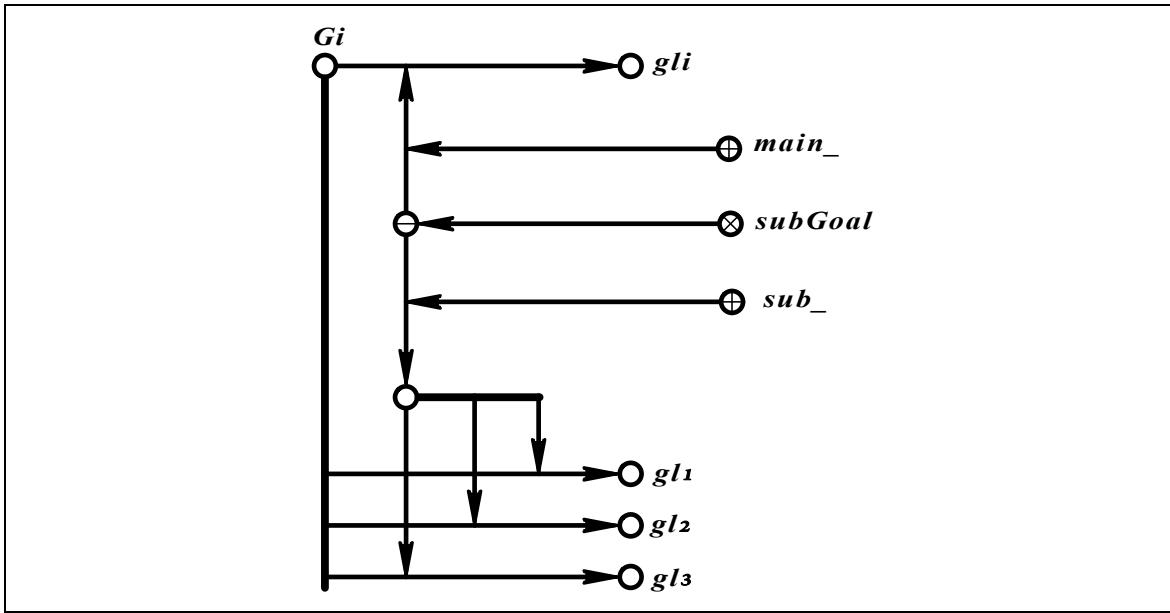
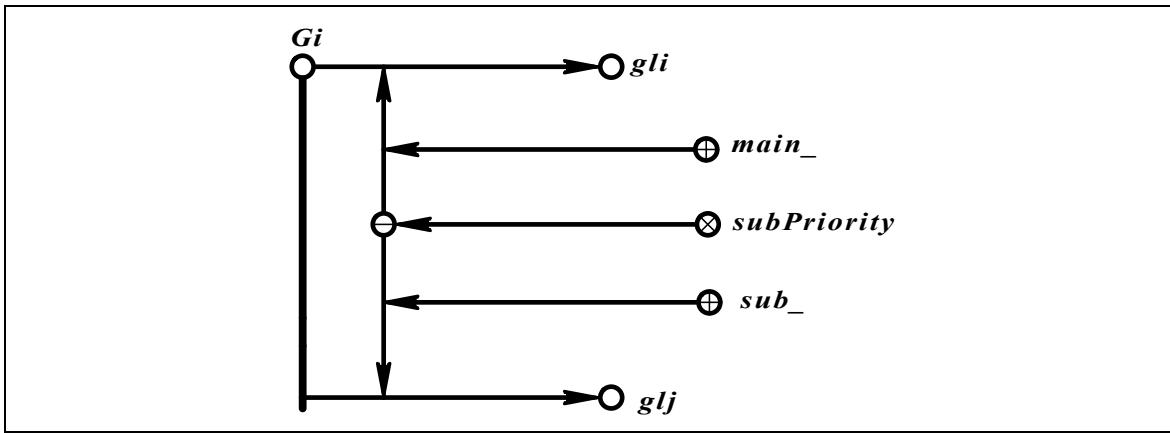
На scl-тексте 6.3.14 приведен пример задания на установление истинности scl-высказывания (утверждения) *bi* в рамках scl-теории *ti*. SCL-высказывание *bi* может быть высказыванием любого вида.

SCL-текст 6.3.15. Задание на установление ложности высказывания bi теории ti



На множестве целей задан целый ряд отношений, наиболее важными из которых являются отношение, связывающее (основные) цели с их И-подцелями (И-подцели – набор целей, достижение каждой из которых и только достижение каждой из которых гарантирует достижение основной цели), и отношение, упорядочивающее множество целей по их приоритету (важности). Формально областью определения этих отношений будем считать константные позитивные sc-дуги, принадлежащие множествам $confirm_{_}$, $deny_{_}$. Отношение, связывающее цели с их И-подцелями, обозначается ключевым узлом $subGoal$, является асимметричным отношением нефиксированной арности и использует два атрибута ($main_{_}$ и $sub_{_}$). Кортеж отношения $subGoal$ связывает некоторую цель, указанную под атрибутом $main_{_}$, со всеми ее И-подцелями, каждая из которых отмечается атрибутом $sub_{_}$. Смысль И-подцелей заключается в том, что после достижения всех И-подцелей достижение исходной цели гарантируется с помощью метода, известного scl-машине. Подчеркнем, что каждая цель может быть сведена к своим И-подцелям в общем случае несколькими способами. Стоит отличать понятие подцели от понятия частной цели.

Отношение, упорядочивающее множество целей по их приоритету, обозначается ключевым узлом $goalPriority$, является асимметричным бинарным отношением и использует два атрибута ($main_{_}$ и $sub_{_}$). Кортеж отношения $goalPriority$ сравнивает две цели, одна из которых (указываемая под атрибутом $main_{_}$) считается более приоритетной по отношению к другой цели.

SCL-текст 6.3.16. Пример связки отношения *subGoal***SCL-текст 6.3.17.** Пример связки отношения *goalPriority*

Каждой цели ставится в соответствие некий субъект, являющийся автором (постановщиком) этой цели. В частном случае автором цели может быть сама интеллектуальная система. Если цель поставлена другим (внешним) субъектом, то интеллектуальная система должна определить корректность (возможность достижения) этой цели и, в случае если цель поведенческая, соответствие ее принятым нормам (правилам) поведения интеллектуальной системы во внешней среде. После всего этого интеллектуальная система может принять или не принять к исполнению цель, поставленную внешним субъектом. Последнее означает, что указанная цель становится также собственной целью интеллектуальной системы.

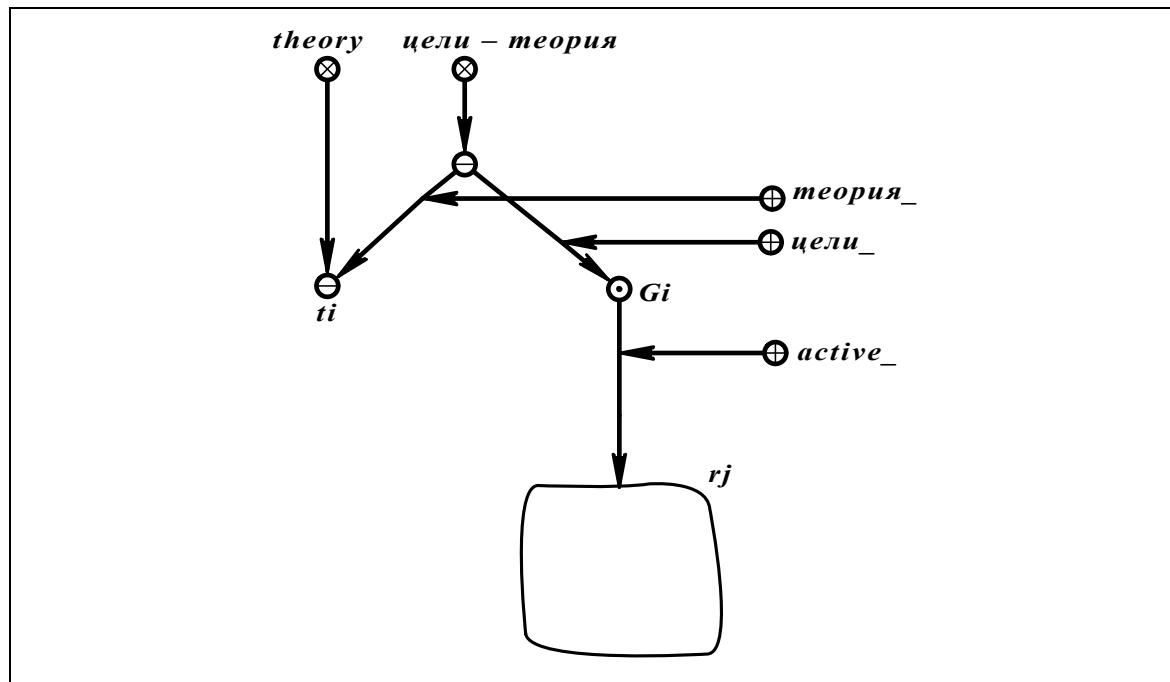
Цели могут быть как инициированными, т.е. подлежащими достижению (выполнению) в текущий период времени, так и неинициированными (в частности, достигнутыми). Инициированные цели в языке SCL дополнительно отмечаются специальным атрибутом *active_*, см. scl-текст 6.3.18. Достигнутые цели помечаются соответствующим атрибутом *denied_* или *confirmed_* соответственно тому: был ли запрос *rj* подтверждён, либо опровергнут.

Более подробное рассмотрение средств описания целей в языке SCL приведено в работе [151] ([Голенков В.В.. 1995пр-ПредсЗРВ](#)).

Задача, решаемая интеллектуальной системой (задачная ситуация) определяется 1) некоторым исходно заданным конструктивным объектом и 2) описанием некоторой цели – обобщенным

описанием свойств требуемого (результатирующего) конструктивного объекта. Задача может быть не решаема (не корректна) либо из-за неполноты исходных данных, либо из-за противоречия между исходными данными и целью. Исходными данными (исходным конструктивным объектом) для задачи, формулируемой в рамках базы знаний, может быть, в частности, все текущее состояние указанной базы знаний. Задачи являются неотъемлемыми компонентами баз знаний и процесса функционирования интеллектуальной системы в целом, так как функционирование интеллектуальной системы трактуется как совокупность взаимодействующих процессов, каждый из которых направлен на решение некоторой конкретной задачи. Каждая возникающая в базе знаний задача инициирует в общем случае несколько параллельных процессов, пытающихся решить эту задачу разными способами.

SCL-текст 6.3.18. Общий вид инициированных заданий



1.4. Гипертекстовые семантические сети

Ключевые понятия и идентификаторы ключевых узлов: гипертекстовая семантическая сеть, информационная конструкция, текст, изображение, видео-информация, аудио-информация, семантическая эквивалентность текстов.

Язык SC может быть использован для построения гипертекстовых семантических сетей. Такие семантические сети включают разнородные информационные конструкции, для связи которых используются метаязыковые и ассоциативные возможности языка SC. Для просмотра гипертекстовых семантических сетей необходима соответствующая навигационно-поисковая графодинамическая ассоциативная машина (см. раздел 7). Для управления методами вывода различных информационных конструкций в зависимости от их класса вводятся понятие стиля отображения (воспроизведения) и специальные отношения между стилем воспроизведения и воспроизводимыми информационными конструкциями.

Пусть дано:

- множество информационных конструкций самого различного вида (тексты, изображения, видеоинформация, аудиоинформация);
- множество знаков, обозначающих самые различные объекты (конкретные предметы некоторой предметной области, конкретные информационные конструкции, конкретные множества, связи, понятия, отношения). В языке SC такие знаки в основном представлены sc-узлами и реже sc-связками (дугами, ребрами);

- множество идентификаторов (имен), которые взаимно однозначно соответствуют множеству вводимых знаков и являются строковым (линейно-символьным) вариантом изображения знаков.

При этом будем считать, что:

- sc-узел, являющийся знаком некоторой информационной конструкции, **содержит** обозначаемую им информационную конструкцию (другими словами, информационная конструкция считается **содержимым** того sc-узла, который её обозначает);
- информационная конструкция (в частности, текстовая информационная конструкция) может включать в себя идентификаторы (имена) некоторых знаков, представленных sc-узлами. Это трактуется как ссылка на соответствующий знак (sc-узел);
- типология информационных конструкций, соотношения между ними, а также их синтаксическая структура и семантика, описываются в виде sc-конструкций с помощью целого ряда вводимых понятий и отношений.

Гипертекстовой семантической сетью будем называть некоторое заданное множество информационных конструкций вместе с sc-конструкцией, являющейся надстройкой над множеством sc-узлов, обозначающих указанные информационные конструкции, и описывающей типологию этих информационных конструкций, соотношения между ними, а также их синтаксическую структуру и семантику. Гипертекстовую семантическую сеть можно также называть:

- семантически структурированным гипертекстом;
- результатом интеграции гипертекстовых технологий и технологий, основанных на семантических сетях;
- семантически структурированной гипертекстовой мультимедийной базой знаний.

В языке SC для информационных конструкций, которые не является текстами языка SC, используется специальный способ их представления и хранения в виде содержимого предметных sc-узлов. Информационные конструкции образуют иерархию, так как всегда можно построить новую информационную конструкцию, которая является объединением других информационных конструкций.

Примечание. Тот sc-узел, который обозначает некоторую информационную конструкцию, вовсе не обязан явно “хранить” эту информационную конструкцию в качестве своего содержимого. Эта информационная конструкция может быть неизвестна (не сформирована). Эта информационная конструкция может быть разбита на фрагменты, каждый из которых представлен явно, и, следовательно нет никакой необходимости явно представлять и хранить всю исходную информационную конструкцию.

Рассмотрим некоторые понятия, определяющие типологию информационных конструкций. Для каждого такого понятия вводится sc-узел, обозначающий соответствующий тип информационных конструкций:

информационная конструкция

- **текст** (дискретная информационная конструкция)
 - **ея-текст** (текст естественного языка)
 - **официальный документ** (административный документ)
 - **приказ**
 - **служебная записка**
 - **заявление**
 - **закон**
 - **договор**
 - **протокол заседания**
 - **художественный текст**
 - **рассказ**
 - **повесть**
 - **роман**
 - **стихотворение**
 - **поэма**
 - **научно-технический документ**
 - **научно-технический отчет** (отчет о научно-исследовательской работе)
 - **техническое задание**
 - **научная монография**
 - **научная статья**
 - **учебно-методический документ**

- *учебный план специальности*
- *квалификационная характеристика специальности*
- *программа учебной дисциплины*
- *учебное пособие*
- *учебник*
- *методическое пособие*
- *текст формального языка*
 - *нотный текст*
 - *scg-текст* (текст языка SCg)
 - *scs-текст* (текст языка SCs)
 - *текст логического языка*
 - *sclg-текст* (текст языка SCLg)
 - *scls-текст* (текст языка SCLs)
 - *текст языка программирования*
 - *текст языка представления знаний*
 - *текст языка запросов*
 - *текст фактографических языков*
 - *scbg-текст* (текст языка SCBg)
 - *scbs-текст* (текст языка SCBs)
- *изображение*
 - *чертеж*
 - *карта*
 - *фотография*
 - *рисунок*
- *аудиоинформация*
 - *запись музыкального произведения*
 - *запись речевого сообщения*
 - *аудиозапись интервью*
 - *аудиозапись лекции*
 - *аудиозапись выступления*
- *видеоинформация*
- *видеоаудиоинформация*
 - *художественный фильм*
 - *документальный фильм*
 - *видеозапись интервью*
 - *видеозапись лекции*
 - *видеозапись выступления*
 - *мульфильм*

текст

- *линейный текст* (текст линейного языка)
- *нелинейный текст* (текст графового языка)

ея-текст

- *русский текст* (текст русского языка)
- *английский текст* (текст английского языка)
- *немецкий текст* (текст немецкого языка)

ея-текст

- **ея-определение**
- **ея-пояснение** (нестрогое определение)
- **ея-утверждение** (ея-описание некоторой закономерности)
- **фактографическийeya-текст**
- **ея-комментарий**
- **ея-вопрос** (ея-формулировка информационной цели)
- **ея-формулировка поведенческой цели**
- **ея-формулировка задачи**
- **ея-запись информационной программы** (программы обработки информации)
- **ея-запись поведенческой программы**
- **ея-протокол решения задачи** (запись протокола решения задачи на естественном языке)
 - **ея-протокол доказательства**

Перечислим некоторые отношения, заданные на множестве информационных конструкций общего вида:

- **семантическая эквивалентность информационных конструкций**
- **информационная конструкция – ключевое понятие**
 - **портрет – объект**
 - **определение – определяемое понятие**
 - **пояснение – поясняемое понятие**
- **последовательность информационных конструкций** (порядок просмотра информационных конструкций в рамках более крупной информационной конструкции)
- **разбиение информационных конструкций** (разбиение на информационные конструкции, являющиеся фрагментами)
- **информационная конструкция – автор**
- **информационная конструкция – рецензент**
- **информационная конструкция – издание**

Перечислим некоторые отношения, заданные на множестве текстов:

- **семантическое включение** (когда информация, содержащаяся в одном тексте, содержится также и в другом);
- **семантическая эквивалентность текстов** (перевод с одного языка на другой, с одной формы на другую);
- **текст-ключевое понятие;**
- **определение – определяемое понятие;**
- **пояснение-поясняемое понятие;**
- **комментарий – комментируемое понятие;**
- **понятие – пример** (описание какого-либо примера);
- **включение текста** (в частности, это может быть связь между некоторым библиографическим источником и взятой из него цитатой);
- **последовательность текстов** (не обязательно линейная);
- **разбиение текста** (книги – на разделы, разделы (главы) – на подразделы, подразделы – на пункты);
- **синтаксическая эквивалентность текстов**.

Перечислим некоторые отношения, заданные на множестве документов:

- **документ – автор**
- **документ – издательство**
- **документ – рецензент**

Отношение “**синтаксическая эквивалентность текстов**” – бинарное неориентированное отношение, каждая связка которого связывает два scb-узла, содержимым которых являются информационные конструкции, имеющие абсолютно одинаковый вид. Эти конструкции могут принадлежать одному и тому же языку или разным языкам. Особое значение имеют синтаксически одинаковые конструкции, имеющие разный смысл.

Каждая связка бинарного неориентированного отношения “**семантическая эквивалентность текстов**” связывает либо scb-узел, обозначающий атомарное или неатомарное высказывание , представленное на языке SCL, с scb-узлом, содержимым которого является записанный на каком-либо языке текст, семантически эквивалентный указанному выше высказыванию, либо два scb-узла, содержимым которых являются семантически эквивалентные тексты, записанные на одном и том же языке или на разных языках.

Кроме содержимого sc-узлов важнейшей составляющей текстов языка SC также являются идентификаторы sc-элементов, представляющие собой обычные строки символов. В ходе решения задач в sc-машине эти идентификаторы никак не используются. Они необходимы для организации ввода / вывода информации и для организации интеграции различных баз знаний. Для выполнения этих действий необходимо учитывать то, что называется **синонимией и омонимией**.

Определение 6.4.1. Два разных sc-элемента считаются **синонимичными** (семантически эквивалентными) в том и только в том случае, если они представляют собой либо один и тот же знак, либо одну и ту же переменную, либо знак одного и того же множества.

Из двух синонимичных sc-элементов либо один, либо оба могут не иметь идентификаторов. Если оба синонимичных sc-элемента имеют идентификаторы, то возможны два случая:

- **тривиальная синонимия**, когда идентификаторы синонимичных sc-элементов совпадают,
- **нетривиальная синонимия**, когда синонимичные sc-элементы имеют разные идентификаторы.

Следует четко отличать синонимию sc-элементов от семантической эквивалентности различных текстов.

В языке SC синонимичные sc-элементы связываются парами бинарного отношения “**пара синонимии**” (см. раздел 2). В случае тривиальной синонимии пары синонимии проводятся по умолчанию. То есть все sc-элементы с одинаковыми идентификаторами по умолчанию считаются синонимичными.

Бинарное отношение “**пара синонимии**” является отношением эквивалентности. Для любого отношения эквивалентности можно построить отношение, осуществляющее разбиение области определения отношения эквивалентности на классы эквивалентности. Сделаем это для отношения “**пара синонимии**” и построим отношение “**синонимичные sc-элементы**” (= **синонимы**). Каждая связка этого отношения связывает все sc-элементы, являющиеся синонимичными некоторому условно выделенному sc-элементу.

Примечание 1. В область определения отношения “**пара синонимии**” и отношения “**синонимичные sc-элементы**” входят не только константы (знаки множеств), но и переменные, т.е. синонимичными могут быть не только sc-константы, но и sc-переменные.

Примечание 2. Из синонимии двух знаков множеств следует равенство этих множеств. Но обратное, вообще говоря, не верно. То есть могут существовать равные (то есть состоящие из одних и тех же элементов), но разные множества. И, следовательно, знаки, обозначающие эти разные множества не могут считаться синонимичными.

Примечание 3. В языке SC синонимия sc-элементов нужна:

- 1) для того, чтобы в логических формулах можно было указывать факт совпадения значения некоторой переменной с некоторой константой или со значением другой переменной. При этом такую переменную пару синонимии следует отличать от константной пары синонимии, связывающей два синонимичных переменных sc-элементов;
- 2) для того, чтобы рассмотреть множество различных вариантов именования какого-либо объекта (идентификаторы синонимичных sc-элементов и есть синонимичные имена одного и того же объекта).

Введем ключевой узел “**главный синоним**”, обозначающий множество sc-элементов, каждый из которых является главным по отношению к своим синонимам. Поскольку отношение “**синонимичные sc-элементы**” является отношением эквивалентности, указание главного узла (среди синонимов) совсем не обязательно осуществлять с помощью атрибута.

Определение 6.4.2. Два разных sc-элемента считаются **омонимичными** в том и только в том случае, если они представляют собой либо разные знаки, либо разные переменные.

Как и синонимия, омонимия sc-элементов бывает тривиальной и нетривиальной. Если два омонимичных sc-элемента имеют разные идентификаторы, то будем их называть **тривиально**

омонимичными. Если два омонимичных sc-элемента имеют одинаковые идентификаторы, то будем их называть **нетривиально омонимичными**.

Примечание. Если для двух sc-элементов либо один из них не имеет идентификатора, либо оба не имеют идентификаторов, либо они имеют разные идентификаторы, то по умолчанию (если не оговорено обратное) считается, что указанные sc-элементы являются омонимичными.

Итак, в языке SC допускается использование синонимичных и омонимичных sc-элементов, но при этом должны выполняться следующие правила.

Правило 6.4.1. Если у какого-либо sc-элемента имеются синонимичные ему sc-элементы, то из всей этой группы синонимов должен быть выделен главный синоним с помощью ключевого sc-узла **“главный синоним”**.

Правило 6.4.2. Нетривиально омонимичные sc-элементы должны быть явно указаны с помощью бинарного отношения **“пара омоними”**. Как и для синонимичных sc-элементов, построим отношение **“омонимичные sc-элементы”** (= **омоними**), осуществляющее разбиение области определения отношения омонимии на классы омонимии. Каждая связка этого отношения связывает все sc-элементы, являющиеся омонимичными некоторому условно выделенному sc-элементу.

Правило 6.4.3. Если построить множество из sc-элементов, являющихся главными синонимами, и из sc-элементов, которые не имеют синонимов (в текущий момент времени), то в этом множестве не должно оказаться ни синонимичных, ни омонимичных sc-элементов.

Для выполнения последнего правила необходимо по крайней мере для одного из двух омонимичных sc-элементов строить синонимичный ему sc-элемент, объявляемый как главный синоним.

Таким образом, борьба с синонимией и омонимией в языке SC осуществляется не путем их искоренения, а путем четкой фиксации синонимичных и омонимичных sc-элементов (если такие sc-элементы возникает необходимость вводить) и путем организации корректного и безопасного использования таких sc-элементов.

Наиболее актуальным применением гипертекстовых семантических сетей являются электронные учебники нового поколения, в которых используются не только гипертекстовые и мультимедийные технологии, но и глубокая семантическая структуризация учебного материала. Такие электронные учебники будут называть ассоциативными электронными учебниками. Подробно они рассмотрены в [236] (**ИнтелОСиВУО-2001кн**)

1.5. Принципы представления нейросетевых моделей

Ключевые понятия и идентификаторы ключевых узлов: нейросетевая модель, нейрон.

Графодинамический подход к представлению и переработке информации можно использовать для интерпретации и интеграции различных моделей представления и переработки информации, в частности, для интеграции нейросетевых моделей переработки информации. Ниже будет рассмотрен пример реализации нейросетевых моделей особого класса **псевдооптических нейронных сетей** (ПНС).

Следует обратить внимание, что модели ПНС ориентированы на моделирование быстрых интеллектуальных процессов мозга. Хорошо известно, что существует обширное множество интеллектуальных процессов, которые в медленном мозгу протекают гораздо быстрее, чем в быстром компьютере. К таким процессам относятся как обработка внешних данных (обработка образных представлений, узнавание, установление сходства), так и сложные внутренние интеллектуальные процессы – быстрое схватывание сути дела, выделение существенного, оперирование сложной ситуацией как целостным представлением. Многие из них не имеют адекватных аналогов в современных методах искусственного интеллекта. Анализ этих процессов приводит к мысли, что в их основе лежат принципиально другие механизмы – несимвольные представления данных и методы их обработки, высокая скорость которых обеспечивается их малой глубиной и сверхвысоким уровнем

параллелизма, природа которого мало похожа на параллелизм вычислительных систем [277] ([Кузнецов О.П. 1995ст-НеклаПвИИ](#)).

Одним из перспективных подходов к исследованию таких механизмов является реализация неоднократно высказывавшейся различными специалистами [630, 609, 410, 20, 188, 678] ([Heerden P.J. 1968bk-FoundOEK](#), [Gabor D.1969art-AssocHM](#), [Прибрам К.1975кн-ЯзыкиМ](#), [Арбид М.А.1976кн-МетафМ](#), [Денисюк Ю.Н.1982ст-НекотПиПГ](#), [Sowa J.F.1984bk-ConceSIP](#)) гипотезы о сходстве многих информационных процессов мозга с процессами обработки изображений в оптической голограммы. В работе [281] ([Кузнецов О.П.1992ст-ГологМОИвНС](#)) впервые предложен класс нейронных сетей (впоследствии названных псевдооптическими нейронными сетями - ПНС), в которых, как было показано, возможны голограммические эффекты. ПНС используют новую модель интерферирующего нейрона, который воспринимает непрерывные периодические сигналы и характеризуется не только порогом, но и дополнительным непрерывным параметром - потенциалом. Значение потенциала может изменяться от нуля до порога под действием входных сигналов, которые суммируются по закону интерференции. При достижении потенциалом порога нейрон генерирует собственный периодический сигнал.

Модели ПНС, рассмотренные в [277; 281; 284] ([Кузнецов О.П.1995ст-НеклаПвИИ](#); [Кузнецов О.П.1992ст-ГологМОИвНС](#); [Кузнецов О.П..2000ст-ПсевдНС](#)), являются геометрическими: в них существенны их геометрические характеристики - расстояния между нейронными слоями сети и нейронами внутри слоя, а также геометрические свойства поверхностей, на которых расположены слои. Правда, в работе [277] ([Кузнецов О.П.1995ст-НеклаПИИ](#)) отмечается, что геометрические характеристики можно заменить задержками на синапсах.

1.5.1. Краткое описание полной прямолинейной модели псевдооптической нейронной сети и методов расчета её поведения

Ключевые понятия и идентификаторы ключевых узлов:
псевдооптическая нейронная сеть, полная прямолинейная модель.

Здесь мы рассмотрим одну из наиболее изученных моделей ПНС - полную прямолинейную модель, описанную в [280] ([Кузнецов О.П..2000ст-ПсевдНС](#)). Начнем с описания основной единицы ПНС - интерферирующего нейрона. Оно содержится в [277; 281; 282; 284] ([Кузнецов О.П.1995ст-НеклаПвИИ](#); [Кузнецов О.П.1992ст-ГологМОИвНС](#); [Кузнецов О.П.1993ст-МоделГПОИвНС](#); [Кузнецов О.П.1996ст-ПсевдНСПМ](#)).

Интерферирующий нейрон N имеет m_N входов, q_N выходов и характеризуется тремя положительными действительными числами: порогом P_N , выходной интенсивностью I_N и потенциалом $U_N(t)$, зависящим от времени и не превышающим порога. Нейрон может находиться в пассивном состоянии, в котором он воспринимает входные сигналы, или в активном состоянии, в котором он генерирует выходной сигнал. Нейроны соединены между собой односторонними волокнами, имеющими две характеристики: длину d и скорость v прохождения сигналов. Сигнал S_i длительности τ_i - это функция $S_i(t) = I_i s_i(t)$, определенная на интервале длины τ_i , где $s_i(t)$ - периодическая функция с частотой v_i , а I_i - константа, называемая интенсивностью сигнала. Пример функции s_i - функция $s_i(t) = \sin 2\pi v_i t$. В дальнейшем считаем, что конкретный вид функции $s_i(t)$ несущественен, сигнал S_i полностью определяется тройкой параметров (I_i, v_i, τ_i) , причем все сигналы, поступающие на вход одного нейрона, имеют одинаковую частоту v . Сигнал S_i возникает в точке волокна в момент t_{1i} и оканчивается в момент $t_{0i} = t_{1i} + \tau_i$, если в этой точке функция S_i определена на интервале $[t_{1i}, t_{0i}]$. Распространение сигнала по волокну с параметрами d и v означает, что, если в начальной точке волокна сигнал S_i возник в момент t_1 , то в конечной точке он возникнет в момент $t_1 + d/v$. Для сигнала S_i , распространяющегося со скоростью v , введем понятие

длины волны $\lambda_i = \frac{\nu}{V_i}$. Если на входе N в момент t_{1i} возник сигнал S_i , а на другом входе в момент t_{1j} возник сигнал S_j , то величину

$$\varphi_{ij} = 2\pi\nu(t_{1j} - t_{1i}) \quad (6.5.1)$$

назовем разностью фаз между S_i и S_j на входе N . Состоянием входов нейрона в момент t называется вектор $\sigma(t) = (I_1(t), \dots, I_m(t))$, где $I_j(t) = 0$, если на j -м входе нет сигнала в момент t , и $I_j(t) = I_j$, если на нем есть сигнал с интенсивностью I_j . Величину $I(t)$, вычисляемую по формуле

$$I(t) = \sum_{i \leq m} \sum_{j \leq m} \sqrt{I_i(t)I_j(t)} \cos \varphi_{ij}, \quad (6.5.2)$$

назовем суммарной входной интенсивностью в момент t .

Нейрон функционирует следующим образом. Пусть в момент t нейрон N пассивен и имеет потенциал $U_N(t)$, состояние входов (I_1, \dots, I_m) на отрезке $[t, t']$ постоянно, I - суммарная интенсивность. Тогда

1) если $U_N(t) + I\nu(t' - t) < P_N$, то

$$U_N(t') = U_N(t) + I\nu(t' - t); \quad (6.5.3)$$

2) в противном случае существует момент t^* , $t < t^* < t'$, такой, что $U_N(t) + I\nu(t^* - t) = P_N$; в момент t^* нейрон становится активным, и на каждом из его q_N выходов возникает сигнал $S_N = (\frac{I_N}{q_N}, \nu, \tau_N)$, где

$$\tau_N = \frac{P_N}{I_N \nu} \quad (6.5.4)$$

(время разряда равно времени заряда от сигнала с теми же параметрами). В момент $t^* + \tau_N$ нейрон снова переходит в пассивное состояние; $U_N(t^* + \tau_N) = 0$.

Формула (6.5.3) предполагает, что на данном отрезке времени существует фиксированное число входных сигналов. Однако на входах нейрона в разные моменты времени существуют разные сигналы; каждый сигнал S_i возникает в некоторый момент t_{1i} , заканчивается в момент t_{0i} и имеет длительность $\tau_i = t_{0i} - t_{1i}$. Для произвольного временного интервала справедливо следующее утверждение (его доказательство содержится в [282] ([Кузнецов О.П. 1993](#) [ст-МоделГПОИвНС](#))).

Теорема интерференции. Если на интервале $[t, t']$ на вход нейрона N поступило m сигналов и потенциал U_N не превысил порога, то

$$U_N(t+t') = U_N(t) + v \left(2 \sum_{i,j \leq m} \sqrt{I_i I_j} \cos \varphi_{ij} \tau_{ij} + \sum_{i=1}^m I_i \tau_i \right), \quad (6.5.5)$$

где τ_i - длительность сигнала на i -м входе, τ_{ij} - длительность одновременного существования сигналов на i -м и j -м входах., а суммирование ведется по всем неупорядоченным парам (i, j) .

В первой сумме формулы (6.5.5) из двух пар (i, j) и (j, i) берется только одна, причем от выбора той или иной пары зависит знак разности фаз φ_{ij} . Но поскольку $\cos \alpha = \cos(-\alpha)$, то для вычислений либо пара (i, j) всегда выбирается так, что $\varphi_{ij} \geq 0$, т.е. $t_{1j} \geq t_{1i}$, либо используется более компактный вариант формулы (6.5.5):

$$U_N(t+t') = U_N(t) + v \left(\sum_{i \leq m} \sum_{j \leq m} \sqrt{I_i I_j} \cos \varphi_{ij} \tau_{ij} \right), \quad (6.5.6)$$

где i и j независимо принимают все значения от 1 до m и, следовательно, сумма содержит пару (i, j) , и пару (j, i) . Второй сумме формулы (6.5.5) в (6.5.6) соответствует сумма пар (i, i) , учитывая, что $\cos \varphi_{ii} = 1$, а $\tau_{ii} = \tau_i$. Нетрудно видеть, что, если на интервале $[t, t']$ сигналы не меняются, то (6.5.6) переходит в (6.5.3).

Следствие 1. В случае, когда все интенсивности одинаковы и равны I , формула (6.5.6) приобретает вид

$$U_N(t+t') = U_N(t) + Iv \left(\sum_{i \leq m} \sum_{j \leq m} \cos \varphi_{ij} \tau_{ij} \right)$$

Величина τ_{ij} выражается через разность фаз следующим образом (доказательство дано в [9]): если $0 \leq \varphi_{ij} \leq 2\pi v \tau$, то

$$\tau_{ij} = \tau_{ij} - \frac{\varphi_{ij}}{2\pi v}, \text{ если } t_{0j} \geq t_{0i}, \quad (6.5.7)$$

$$\tau_{ij} = \tau_j, \text{ если } t_{0j} < t_{0i}.$$

Если все длительности сигналов одинаковы и равны τ , из $t_{1j} > t_{1i}$ следует $t_{0j} > t_{0i}$ и второй случай в (6.5.7) невозможен, т.е. $\tau_{ij} = \tau - \frac{\varphi_{ij}}{2\pi v}$. Это приводит к следующему утверждению.

Следствие 2. Если все интенсивности входных сигналов равны I , а все их длительности равны τ , то в условиях теоремы

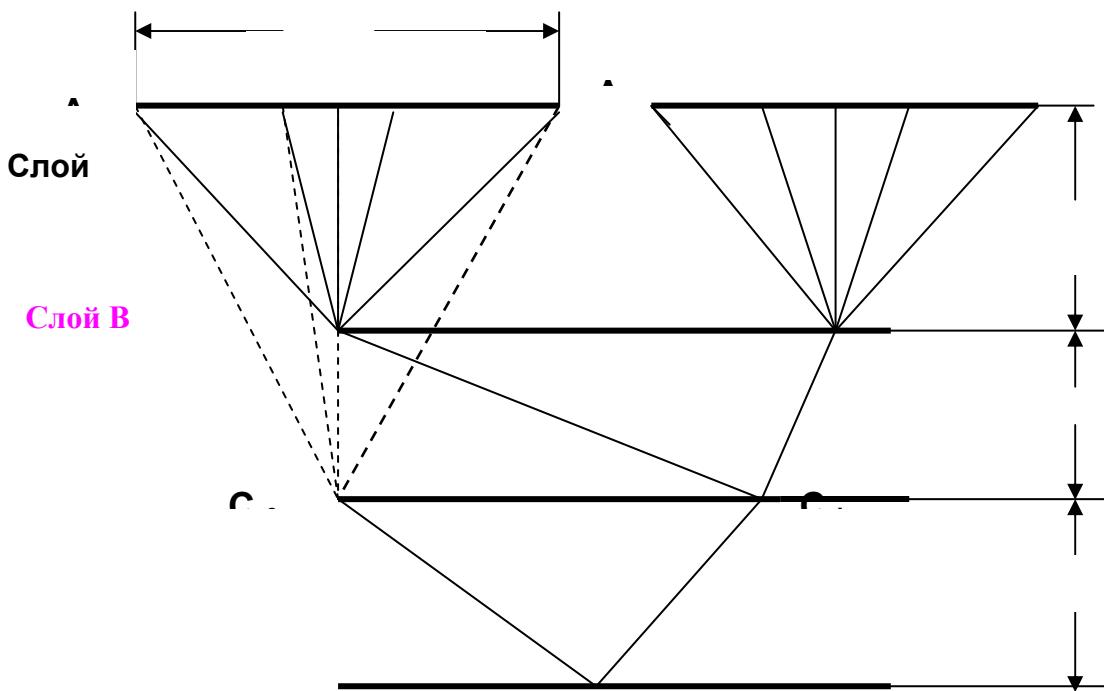
$$U_N(t+t') = U_N(t) + Iv \left(\sum_{i \leq m} \sum_{j \leq m} \left(\tau - \frac{\varphi_{ij}}{2\pi v} \right) \cos \varphi_{ij} \right), \quad (6.5.8)$$

где сумма берется по всем i, j , таким, что $|\varphi_{ij}| \leq 2\pi v \tau$.

Поскольку качество голограммических эффектов улучшается с увеличением числа нейронов, понижение сложности вычислений при программном моделировании ПНС является важной задачей.

Все геометрические модели ПНС основаны на общей схеме, повторяющей схемы оптической голограмии. Эта схема описана в [277; 282] ([Кузнецов О.П. 1995 ст-НеклаПвИИ](#); [Кузнецов О.П. 1993 ст-МоделГПОИвНС](#)) и содержит четыре нейронных слоя: слой-источник A , слой B , в котором размещается образ-объект, изображаемый распределением потенциалов его нейронов (чем ближе к порогу потенциал нейрона, тем "ярче" соответствующая точка образа), слой C (голограмма), в котором в результате интерференции сигналов от A и B возникает распределение потенциалов, являющееся голографической записью информации об образе B , и слой D , в котором после "освещения" голограммы C источником A восстанавливается образ B . Каждый слой – это множество нейронов с одинаковыми параметрами, расположенных на некоторой поверхности на равном расстоянии друг от друга и не связанных между собой.

Рисунок 6.5.1



Из описания этой схемы видно, что выходы A должны быть связаны со входами B , выходы A и B – со входами C , а выходы C – со входами D . Скорости и частоты сигналов во всей сети одинаковы. Главная задача при выборе параметров модели – получение в D образа, "похожего" на образ в B .

Полная прямолинейная модель – это геометрическая модель ПНС со следующими параметрами и свойствами:

- 1) Все четыре слоя – это прямолинейные параллельные отрезки, лежащие на одной плоскости. Расстояния между ними обозначим через r_{AB} (от A до B), r_{AC} , r_{BC} , r_{CD} , соответственно, причем $r_{AB} + r_{BC} = r_{AC}$. Следуя принципам оптической голограмии (восстановленное изображение объекта находится по другую сторону голограммы на том же расстоянии от нее, на котором находился сам объект), полагаем $r_{BC} = r_{CD}$. Волокна, соединяющие нейроны разных слоев, также прямолинейны. В дальнейшем будем называть их лучами. Все нейроны слоя A имеют одинаковые порог P_A и выходную интенсивность I_A . Аналогичные параметры слоев B, C обозначаются через P_B, P_C, I_B, I_C соответственно.
- 2) Число нейронов n_C и n_D в C и D одинаково: $n_C = n_D = n$, нейроны пронумерованы от 0 до $n - 1$; расстояния между нейронами одинаковы и равны e . Таким образом, отрезок C разбит точками C_0, \dots, C_{n-1} , в которых находятся нейроны, на $n - 1$ отрезков длины e ; длина C равна $e(n - 1)$. То же относится и к D . Все расстояния измеряются числом волн; или, что то же самое: расстояния измеряются в обычных единицах длины, но $\lambda=1$.

- 3) Отрезок B также имеет длину $e(n - 1)$ и разбит n точками B_0, \dots, B_{n-1} на $n - 1$ отрезков длины e . Однако, в отличие от C и D , нейроны слоя B могут находиться не во всех точках. Номер нейрона слоя B – это номер точки, в которой он находится.
- 4) Коэффициенты ветвления (числа выходов) нейронов слоев B, C равны: $q_B = q_C = n$.
- 5) Параметры слоя A выбираются с учетом того, что его излучение должно моделировать два классических случая оптической голографии: точечный источник и плоскую волну. В [284] ([Кузнецов О.П. 1996 ст-ПсевдНСПМ](#)) плоская волна моделировалась упрощенным образом: нейрон A_i был соединен только с нейроном C_i (т.е. $q_A = 1$, и интерференция в плоской волне не учитывалась). Это сильно упрощало вычисления, но снижало общность модели. В полной модели это ограничение снимается. Связи A с B строятся на основе следующих соображений: 1) каждая точка B_i соединена с n точками слоя A ; совокупность этих n лучей будем называть входным пучком B_i ; 2) геометрия входного пучка B_i не зависит от его номера; этот пучок всегда симметричен относительно перпендикуляра A_iB_i . Это приводит к структуре, показанной на [рис.6.5.1](#). Из нее видно, что в случае плоской волны число нейронов в A должно быть больше n , точнее, $n_A \geq 2n$. Аналогичная структура связей имеет место между A и C . Поэтому для произвольного A_i общее число выходных лучей $q_{Ai} \leq 2n$, но для простоты полагаем всегда $q_{Ai} = q_A = 2n$. Для случая точечного источника $n_A = 1$, причем номер единственного нейрона A_i произволен, но по-прежнему $q_A = 2n$.
- 6) В начальный момент потенциалы $U_{C_i} = U_{D_i} = 0$ для всех нейронов C_i и D_i . В точке B_i нейрон может либо отсутствовать, либо присутствовать. В последнем случае его потенциал U_{B_i} произволен. Распределение потенциалов в слое B представляет одномерный образ объекта: нейрон с высоким потенциалом соответствует яркой точке объекта, нейрон с низким потенциалом – темной точке объекта, отсутствие нейрона – отсутствию объекта в данной точке.
- 7) Другие параметры сети и их обозначения: $U_{Ai}, U_{Bj}, U_{Ck}, U_{Di}$ – потенциалы нейронов A_i, B_j, C_k, D_l соответственно; $S_{Ai}, S_{Bj}, S_{Ck}, S_{Di}$ – их выходные сигналы, a_{ij} – расстояние между A_i и B_j (т.е. длина волокна A_i и B_j), b_{jk} – расстояние между B_j и C_k , c_{ik} – расстояние между A_i и C_k , d_{kl} – расстояние между C_k и D_l .
- 8) Общая схема сети приведена на [рис.6.5.1](#). Из нее видно, что расстояния $a_{ij}, b_{jk}, c_{ik}, d_{kl}$ вычисляются по теореме Пифагора, например, $b_{0k} = \sqrt{r_{BC}^2 + k^2 e^2}$. Число различных лучей B_jC_k равно n^2 , однако число различных расстояний b_{ik} равно n : поскольку $b_{ik} = b_{0,|i-k|}$, то массив $\{b_{00}, \dots, b_{0,n-1}\}$ содержит все расстояния b_{ik} . Для краткости вместо b_{0k} будем писать b_k . Тогда $b_{ik} = b_{|i-k|}$, в частности, $b_{ii} = b_0 = r_{BC}$. Аналогичные соотношения и обозначения сохраняются и для расстояний между другими парами слоев. Кроме того, $b_{ij} = d_{ij}$ для всех i, j .

Подробное описание методов расчёта поведения вышеописанной модели псевдооптической нейросети приведено в источнике [280] ([Кузнецов О.П.. 2000 ст-ПсевдНС](#)).

1.5.2. Принципы представления псевдооптических нейросетей в памяти графодинамических машин

Ключевые понятия и идентификаторы ключевых узлов: нейрон, [слой](#), [нейрон](#), [нейросеть](#).

Представление псевдооптических нейронных сетей в графодинамической памяти можно организовать следующим образом. В рассматриваемой нейросетевой модели, как и в большинстве других нейросетевых моделей можно выделить такие понятия, как слой, нейрон, связь. Рассмотрим возможное представление этих понятий в графодинамической памяти.

Каждый нейрон в графодинамической памяти будем представлять связкой:

нейрон → (процедура_обратного_распространения_ : backward_proc ,
	процедура_прямого_распространения_ : forward_proc ,
	множество_параметров_ : par ,
	прямые_входные_параметры_ : f_In ,
	прямые_выходные_параметры_ : f_Out ,
	обратные_входные_параметры_ : b_In ,

<i>обратные_выходные_параметры_</i>	: <i>b_Out</i>) ;
-------------------------------------	--------------------

для ПНС множество параметров “*par*” содержит порог, потенциал, например:

<i>par</i> → <i>порог_</i> : 10 , <i>потенциал_</i> : 8 , <i>количество_выходных_связей_</i> : 10 ;

процедуры “*forward_proc*” и “*backward_proc*” могут представлять собой программы, реализующие функцию активации и функцию обучения, соответственно, написанные на процедурном языке, который, возможно, является sc-подъязыком, например, на языке SCP.

Каждую связь, как и нейрон, в графодинамической памяти будем представлять связкой:

<i>волокно</i> → (<i>процедура_обратного_распространения_</i> : <i>b_fibre_proc</i> , <i>процедура_прямого_распространения_</i> : <i>f_fibre_proc</i> , <i>множество_параметров_</i> : <i>prm</i> , <i>прямые_входные_параметры_</i> : <i>f_IP</i> , <i>прямые_выходные_параметры_</i> : <i>f_OP</i> , <i>обратные_входные_параметры_</i> : <i>b_IP</i> , <i>обратные_выходные_параметры_</i> : <i>b_OP</i>) ;
--

для ПНС множество параметров “*prm*” содержит как неизменные характеристики связи – длину волокна, плотность среды

<i>prm</i> → <i>длина_</i> : 100 , <i>плотность_</i> : 1 ;

так и характеристики проходящего сигнала: интенсивность, длительность сигнала, пространственно-временное положение:

<i>prm</i> → <i>интенсивность_</i> : 1 , <i>длительность_</i> : 10 , <i>время_</i> : 2 , <i>положение_</i> : 70 ;
--

“*f_IP*” и “*b_IP*” – прямой выход нейрона-источника и обратный выход нейрона-приёмника соответственно, а “*f_OP*” и “*b_OP*” – соответственно включаются во множество прямых входных параметров нейрона-приёмника и во множество обратных входных параметров нейрона-источника; процедуры “*f_fibre_proc*” и “*b_fibre_proc*” строятся аналогичным методом, как и процедуры “*forward_proc*” и “*backward_proc*” для нейрона. Разумеется, такие процедуры для нейронов источников сигнала, нейронов промежуточных слоёв и нейронов, задающих восстановленный образ, могут быть различны.

Слой нейронной сети задаётся как множество нейронов:

<i>слой</i> → <i>layer1</i> ; <i>layer1</i> → <i>neuron1</i> , <i>neuron2</i> , <i>neuron3</i> ;

т. е. слой “*layer1*” состоит из трёх нейронов “*neuron1*”, “*neuron2*”, “*neuron3*”.

Сеть состоит из слоёв и процедуры обработки поведения сети:

```
нейросеть → ( 1_ : layer1 ,
  2_ : layer2 ,
  3_ : layer3 ,
  4_ : layer4 ,
  процедура_ : net_proc );
```

процедура обработки сети запускается по запросу пользователя и создаёт условие запуска процедур нейронов на первом слое. Те в свою очередь создают условия запуска процедур волокон и нейронов на последующих слоях. Кроме этого, существует набор вспомогательных процедур, которые позволяют автоматическим образом создавать сети определённой конфигурации, менять сохраняемый образ, управлять процессами записи и восстановления образа, сохранять текущее состояние сети. Все эти процедуры объединяются в CASE-систему средств для работы с нейросетями.

Ниже приведём примеры запросов пользователя на построение сетей определённого вида на языке SC:

```
множество связей между слоями_ →
[множество связей между слоями_ : "1"/ ,
слой, в который идут связи_ : "2"/ ] ,
[слой, из которого идут связи_ : "1"/ ,
слой, в который идут связи_ : "3"/ ] ,
[слой, из которого идут связи_ : "2"/ ,
слой, в который идут связи_ : "3"/ ] ;
множество задаваемых начальных потенциалов_ →
[для какого слоя задаются потенциалы_ : "2"/ ,
[порядковый номер нейрона_ : "1"/ ,
начальный потенциал нейрона_ : "2"/ ] ,
[порядковый номер нейрона_ : "2"/ ,
начальный потенциал нейрона_ : "0"/ ]];
```

Выводы к разделу 6

В разделе 6 были рассмотрены способы представления знаний различных видов. Рассмотрены способы введения различных шкал измерения. Сформулированы принципы описания динамических систем и нейросетевых моделей. Раскрыто понятие гипертекстовой семантической сети, являющейся моделью представления информации в навигационно-поисковой машине (см. раздел 7) и использующейся для систематизации и структуризации знаний о предметной области. Описаны различные виды информационных целей (в виде заданий), которые возможны для графодинамических ассоциативных машин. Отметим, что представление целей в виде заданий позволяет перейти на декларативный формат представления информации в графодинамических машинах, приближая внутренний язык машины к естественному языку. Выстроена иерархия целей, на основе введённых ключевых узлов, когда цель частного вида формируется на основе контекста цели общего вида. Все вышеперечисленное позволяет перейти к рассмотрению различных абстрактных графодинамических ассоциативных машин, в которых будут использованы описанные принципы и способы представления информации, в том числе и служебной (системной) информации, описывающей состояние таких машин.

1. Навигационно-поисковая ассоциативная машина

графодинамическая

В данном разделе описывается абстрактная навигационно-поисковая графодинамическая ассоциативная машина, обеспечивающая навигацию и поиск в рамках текущего состояния хранимой в памяти машины базы знаний. Реализация интерфейса этой машины описана в разделе 5 книги [411] (*ПрогрВАМ-2001кн*).

Навигационно-поисковая машина обычно интегрируется в другие графодинамические ассоциативные машины.

Данный раздел может быть использован в качестве учебного пособия по дисциплине «Модели представления знаний, базы данных и СУБД» специальности «Искусственный интеллект».

1.1. Операции навигационно-поисковой графодинамической ассоциативной машины

Ключевые понятия: навигационно-поисковая графодинамическая ассоциативная машина; цель; семантическая окрестность; изоморфный поиск; гипертекстовая семантическая сеть.

При реализации навигационно-поисковой графодинамической ассоциативной машины, так же как и при реализации графодинамической ассоциативной машины вывода (см. раздел 8) в качестве языка микропрограммирования выбирается язык SCP, который описан в разделе 4 [411] (*ПрогрВАМ-2001кн*). Набор операций навигационно-поисковой графодинамической ассоциативной машины строго не фиксирован. Ниже рассмотрим некоторые операции навигационно-поисковой графодинамической ассоциативной машины. Всё множество рассматриваемых операций разбивается на следующие семейства операций:

- семейство операций поиска теоретико-графовой окрестности указываемого sc-элемента;
- семейство операций поиска в рамках указываемой формальной теории всех истинных высказываний, релевантных указываемой высказывательной форме (заданному образцу);
- семейство операций поиска семантических окрестностей указываемого sc-элемента;
- семейство операций поиска семантической связи между (двумя или более) указываемыми sc-элементами;
- семейство навигационно-поисковых операций в гипертекстовой семантической сети.

1.1.1. Информационные конструкции, описывающие состояние навигационно-поисковой графодинамической ассоциативной машины

Ключевые понятия: цель; адресат; результат.

Для представления информационных конструкций, описывающих состояние абстрактной навигационно-поисковой графодинамической ассоциативной машины, вводятся следующие ключевые узлы:

SCs-текст 7.1.1.1 Ключевые узлы навигационно-поисковой машины

```
result ;      /* Описание ключевого узла result */  
пояснение → (result , текст пояснения_ : /* Ключевой узел result является знаком  
бинарного отношения, связывающего множество, обозначающее цель, и sc-узел, который  
обозначает результат цели "/ );  
главный синоним → result ;  
  
sender ;      /* Описание ключевого узла sender */
```

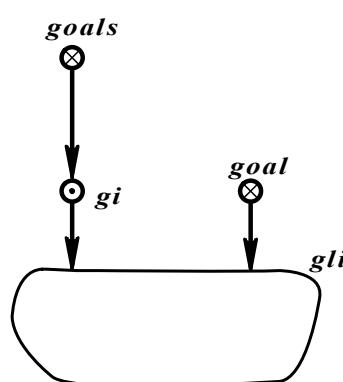
пояснение → (*sender*, *текст пояснения_* : /" Ключевой узел *sender* является знаком бинарного отношения, связывающего множество, обозначающие цель, и sc-узел, который обозначает того, кто инициировал эту цель "/);

главный синоним → *sender* ;

Опишем типичные конструкции, которые будут необходимы для функционирования операций навигационно-поисковой графодинамической ассоциативной машины.

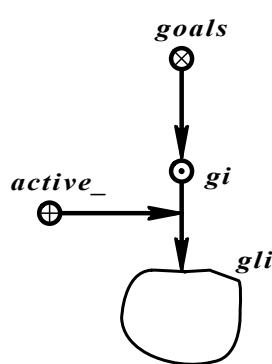
Рассмотрим описание **целей** навигационно-поисковой графодинамической ассоциативной машины.

С С г -т е к с т 7.1.1.1. Общий вид описания цели навигационно-поисковой графодинамической ассоциативной машины



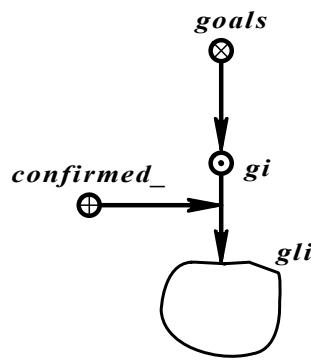
Здесь sc-узел *gi* обозначает множество целей, sc-узел *gli* обозначает конкретную цель навигационно-поисковой графодинамической ассоциативной машины.

С С г -т е к с т 7.1.1.2. Общий вид описания активной цели навигационно-поисковой графодинамической ассоциативной машины



Здесь sc-узел *gli* обозначает конкретную активную цель навигационно-поисковой графодинамической ассоциативной машины.

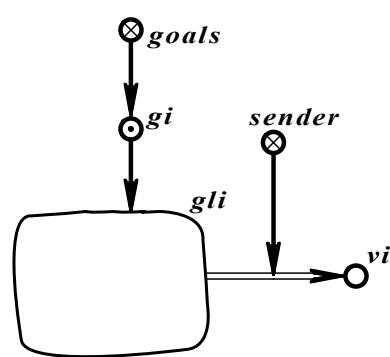
S Cg-текст 7.1.1.3. Общий вид описания завершенной цели навигационно-поисковой графодинамической ассоциативной машины



Здесь sc-узел **gli** обозначает конкретную завершенную цель навигационно-поисковой графодинамической ассоциативной машины.

Рассмотрим описание **адресата цели**, т.е. кто прислал запрос и кому соответственно необходимо передать результат работы конкретной цели.

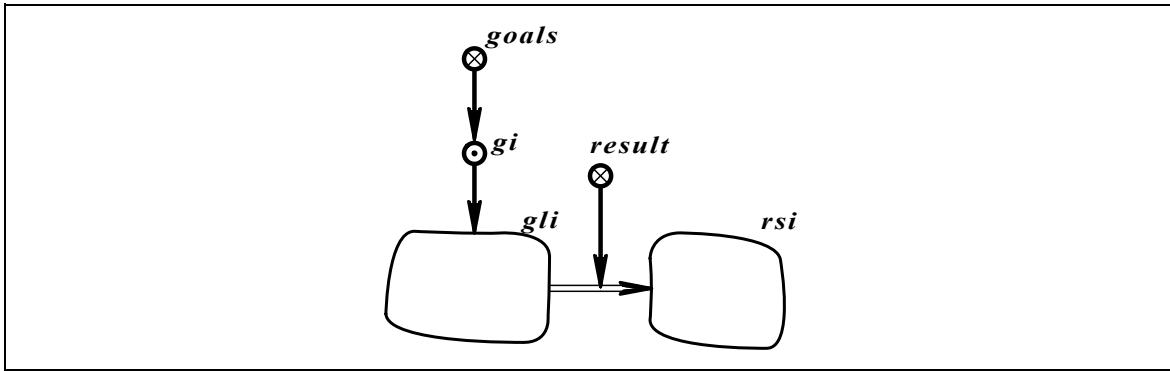
S Cg-текст 7.1.1.4. Общий вид описания адресата цели навигационно-поисковой графодинамической ассоциативной машины



Здесь sc-узел **gi** обозначает множество целей навигационно-поисковой графодинамической ассоциативной машины, sc-узел **gli** обозначает конкретную цель навигационно-поисковой графодинамической ассоциативной машины, sc-узел **vi** обозначает адресат цели и связан с sc-узлом **gli** бинарным ориентированным отношением **sender**, которое связывает цель с адресатом.

Рассмотрим описание **результата выполнения цели**.

SCg-текст 7.1.1.5. Общий вид описания результата выполнения цели навигационно-поисковой графодинамической ассоциативной машины



Здесь sc-узел *gi* обозначает множество целей навигационно-поисковой графодинамической ассоциативной машины, sc-узел *gli* обозначает конкретную цель навигационно-поисковой графодинамической ассоциативной машины, sc-узел *rsi* обозначает конкретный результат цели *gli*. SC-узел цели и sc-узел результата связаны бинарным ориентированным отношением *result*, которое связывает цель с результатом.

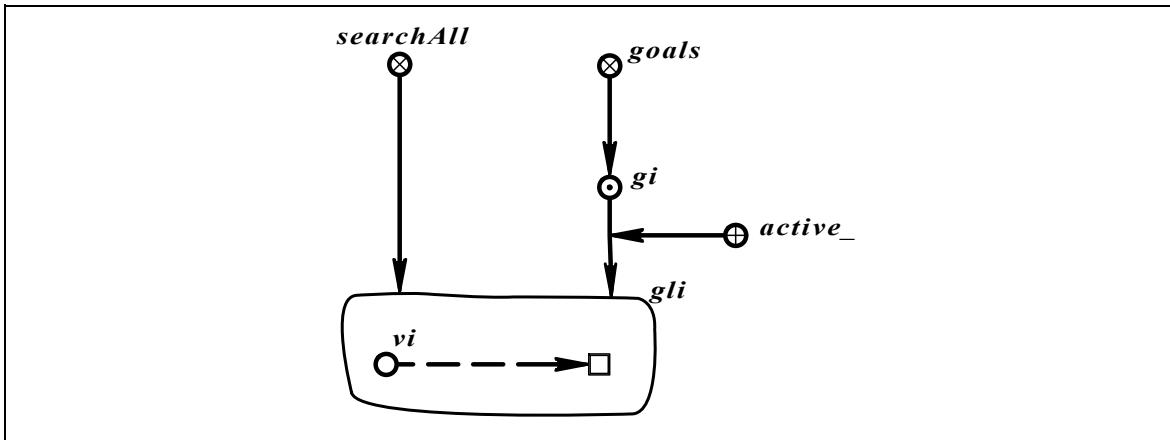
1.1.2. Семейство операций поиска теоретико-графовой окрестности указываемого sc-элемента

В рассматриваемое семейство операций навигационно-поисковой графодинамической ассоциативной машины входят следующие операции:

- операция поиска теоретико-графовой окрестности указываемого sc-элемента по выходящим или входящим парам принадлежности указанного типа. Указываемый тип может быть составлен из комбинации “константный – переменный – метапеременный” и “позитивная – негативная – нечеткая”;
- операция поиска теоретико-графовой окрестности указываемого sc-элемента по связкам указанного отношения с указанием атрибута, который должен быть помечен в искомых связках;
- операция поиска теоретико-графовой окрестности указываемого sc-элемента в рамках всей памяти или в рамках указанной формальной теории и семейства указываемых формальных теорий.

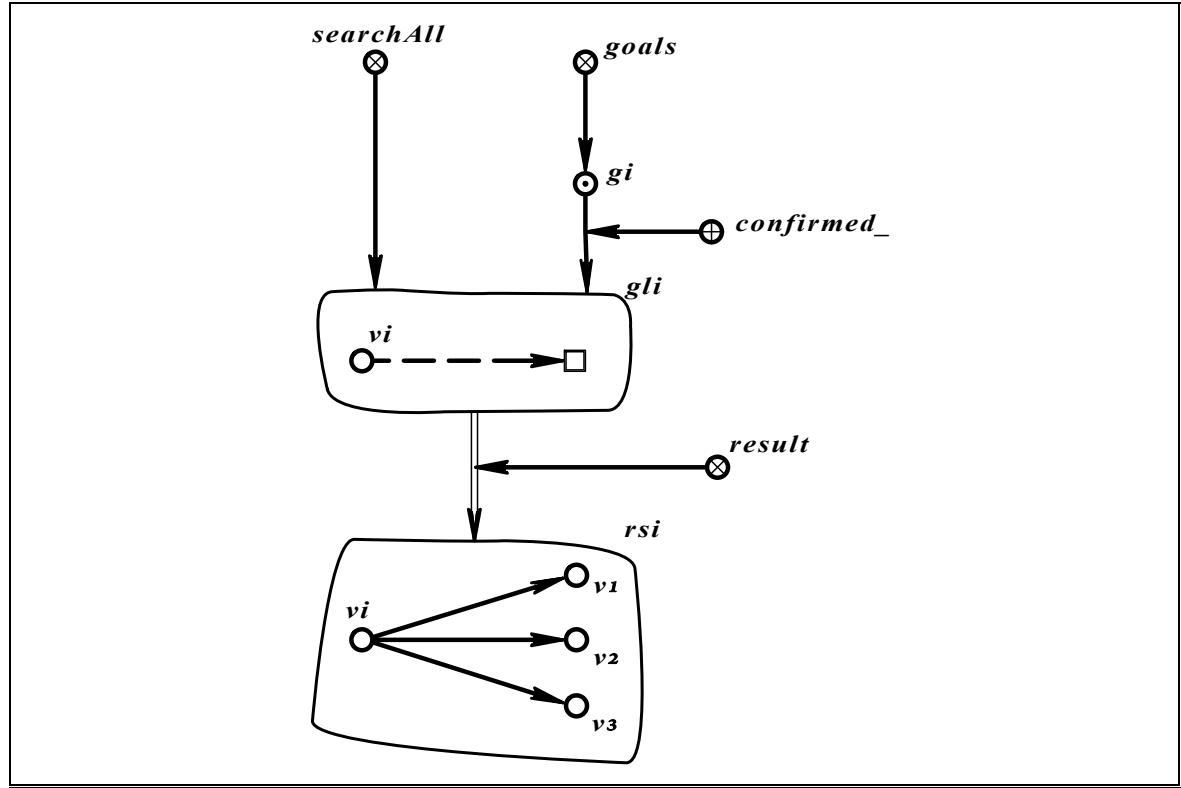
Рассмотрим в качестве примера **операцию поиска теоретико-графовой окрестности указываемого sc-элемента по выходящим парам принадлежности константного позитивного типа**.

Условием выполнения операции поиска теоретико-графовой окрестности указываемого sc-элемента по выходящим парам принадлежности константного позитивного типа является наличие конструкции следующего вида:



Здесь sc-узел *vi* является sc-узлом, относительно которого осуществляется поиск теоретико-графовой окрестности указываемого sc-элемента по выходящим парам принадлежности константного позитивного типа. Ключевой sc-узел *searchAll* указывает, что необходимо найти все элементы.

Результатом выполнения операции поиска теоретико-графовой окрестности указываемого sc-элемента по выходящим парам принадлежности константного позитивного типа является сформированное следующие множество:



Здесь результат *rsi* цели *gli* включает sc-узел *vi*, который является аргументом поиска, и sc-узлы *v1*, *v2*, *v3*, которые связаны с sc-узлом *vi* константными позитивными парами принадлежности.

1.1.3. Семейство операций поиска в рамках указываемой формальной теории всех истинных высказываний, релевантных указанной высказывательной форме (заданному образцу)

Ключевые понятия: высказывание; изоморфный поиск.

Напомним, что в языке SCL высказывание, релевантное заданной высказывательной форме, и сама эта высказывательная форма являются изоморфными логическими формулами. При этом в рамках указанного изоморфизма:

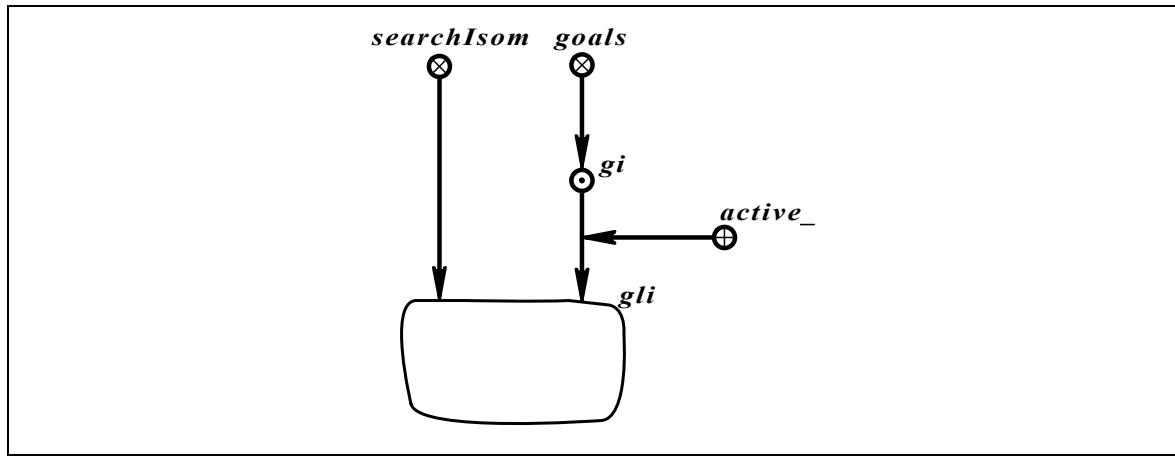
- свободным переменным высказывательной формы ставятся в соответствие константы релевантного высказывания;
- константы высказывательной формы ставятся в соответствие самим себе;
- знаки логических формул, входящих в состав высказывательной формы, ставятся в соответствие знакам логических формул, входящих в состав релевантного высказывания.

Очевидно, что если высказывательная форма является атомарной логической формулой, то релевантное высказывание представляет собой подграф (фрагмент структуры) sc-конструкции, являющейся представлением той предметной области, которая описывается указанной выше формальной теорией.

В рассматриваемое семейство операций входит операция изоморфного поиска по заданному образцу произвольного размера и произвольной конфигурации.

Рассмотрим операцию изоморфного поиска по заданному образцу.

Условием выполнения операции изоморфного поиска является наличие конструкции следующего вида:

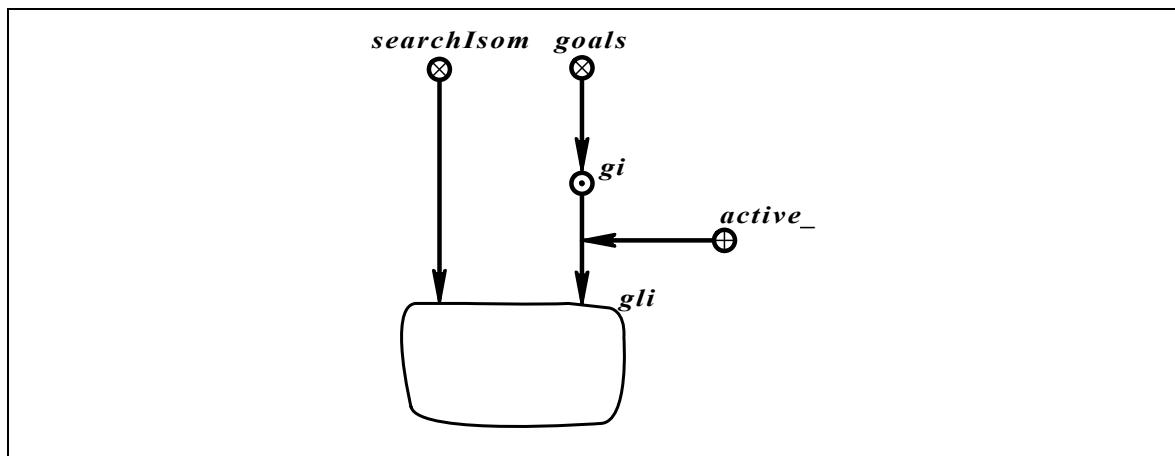


Здесь sc-узел *gli* является множеством, которое содержит образец для изоморфного поиска.

Результатом выполнения операции изоморфного поиска является сформированное множество результатов. Связь цели с результатом осуществляется с помощью отношения *result* (см scg-текст 7.1.1.5)

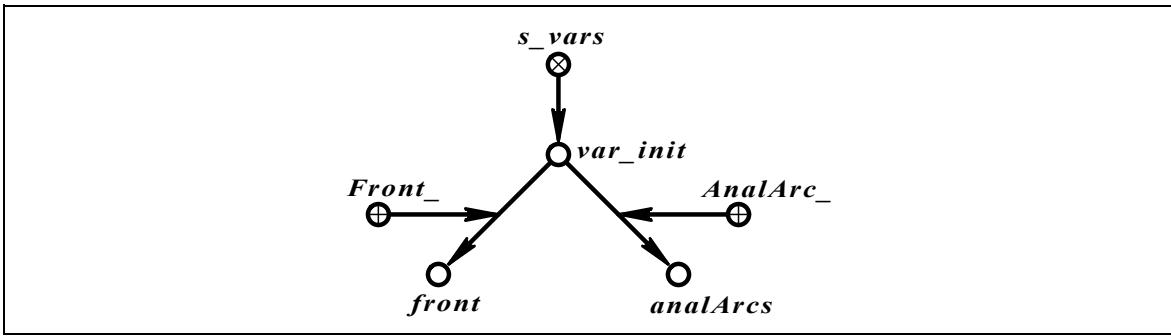
Микрограмма операции изоморфного поиска имеет следующий вид.

Шаг 1. Проверить условие выполнения операции, т.е. найти конструкцию вида:

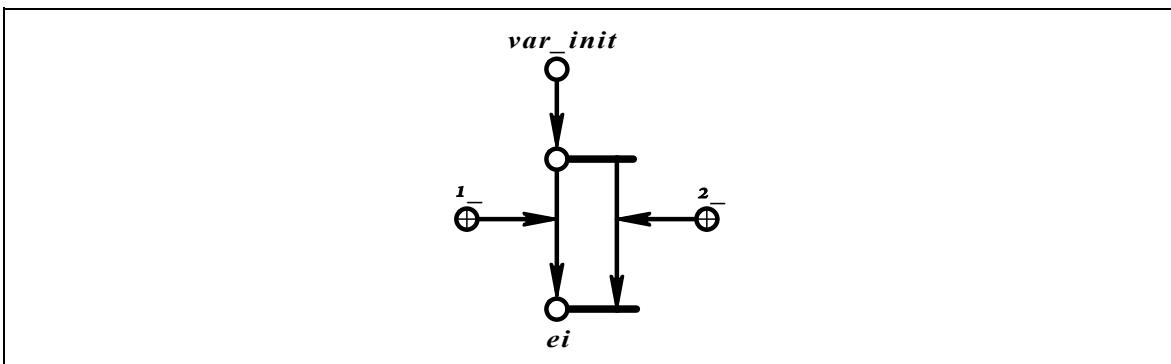


Шаг 2. Если такой конструкции не найдено, то перейти к шагу 1. Это означает, что в текущем состоянии навигационно-поисковой графодинамической ассоциативной машины нет активного запроса на поиск изоморфного подграфа.

Шаг 3. Создать множество выполняемых вариантов (обозначим его *s_vars*). Под вариантом будем понимать кортеж, в состав которого входят множества: помеченное атрибутом *Front_*, которое содержит такие элементы, от которых можно осуществлять поиск соседних элементов, помеченное атрибутом *AnalArc_*, которое содержит лишь те дуги, которые еще не имеют соответствий в рамках текущего варианта, а все остальные, – знаки соответствий. Включить в него начальный выполняемый вариант *var_init*:



Шаг 4. Просмотреть все элементы шаблона. Все константные элементы поместить в множество *front*, установить соответствие константного элемента самому себе - сгенерировать конструкцию следующего вида:



Шаг 5. Все переменные дуги поместить в множество *analArcs*.

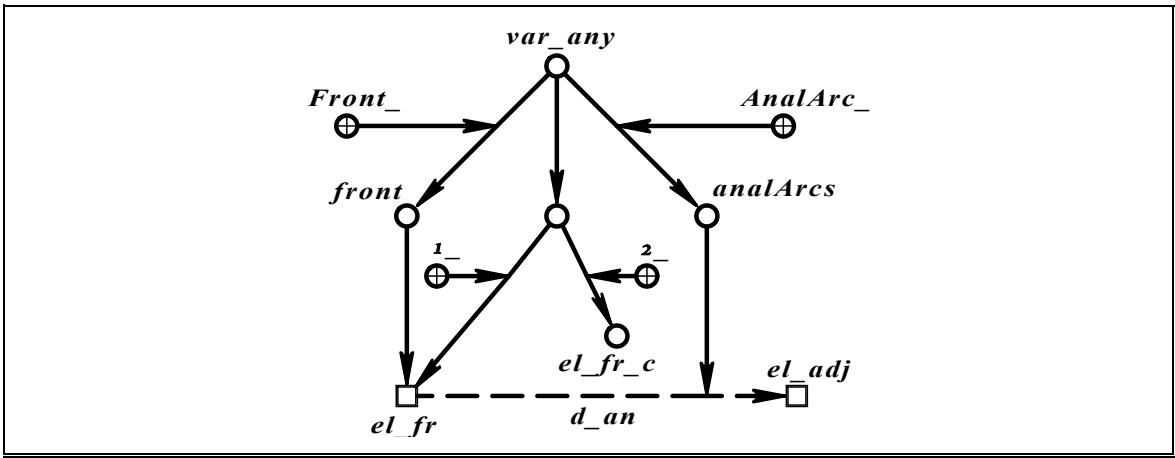
Шаг 6. Просмотреть все элементы множества *front*. Исключить те элементы, в которые не входят дуги принадлежащие множеству *analArcs*. Т.е. удалить те элементы, от которых невозможно вести поиск, т.к. их окрестность уже найдена.

Шаг 7. Если множество *front* пусто, то перейти к шагу 10.

Шаг 8. Начало цикла по элементам *var_any* множества *s_vars*.

Шаг 9. Если множество *analArcs* варианта *var_any* пусто, то удалить множество *front* и *analArcs*, исключить *var_any* из *s_vars* и занести его во множество результатов *s_result*. Перейти к шагу 8.

Шаг 10. Иначе выбрать элемент множества *front* текущего варианта *var_any* - sc-элемент *el_fr*. Среди sc-дуг, входящих в этот элемент или выходящих из него и принадлежащих множеству *analArcs* варианта *var_any* выбрать одну из них - sc-дугу *d_an*.



Шаг 11. Если *el_adj* принадлежит множеству *front* варианта *var_any*, т.е. ему уже присвоено соответствие в составе варианта *var_any*, то

Шаг 12. Найти элемент *el_adj_c*, являющийся соответствием элемента *el_adj* в рамках варианта *var_any*.

Шаг 13. Если между элементами *el_adj_c* и *el_fr_c* существует sc-дуга *d_an_c*, "ориентированная" относительно *el_fr_c* точно таким же образом, как *d_an* - относительно *el_fr* (т.е. обе эти дуги - либо входящие, либо выходящие), и не проведенная в варианте *var_any* никакой sc-дуге, то

Шаг 14. Фиксируем, что дуге *d_an* в рамках варианта *var_any* соответствует дуга *d_an_c* и исключаем её из множества *analArcs*.

Шаг 15. Если множество *analArcs* варианта *var_any* становится пустым, то следует перейти к шагу 9 (вариант завершается успешно).

Шаг 16. Если элемент *el_fr* больше не инцидентен ни одной sc-дуге, принадлежащей множеству *analArcs* варианта *var_any*, то *el_fr* исключается из множества *front* варианта *var_any*, как элемент, от которого невозможно осуществить поиск соседей, т.к. всем инцидентным ему дугам уже найдены соответствия.

Шаг 17. Если элемент *el_adj* больше не инцидентен ни одной sc-дуге, принадлежащей множеству *analArcs* варианта *var_any*, то *el_fr* исключается из множества *front* варианта *var_any*.

Шаг 18. Если в sc-дуге *d_an* входят sc-дуги, являющиеся элементами множества *analArcs* варианта *var_any*, то занести *d_an* в множество *front* варианта *var_any*.

Шаг 19. Перейти к шагу 10.

Шаг 20. Иначе sc-дуге *d_an* невозможно присвоить соответствие.

Шаг 21. Удалить множества *front* и *analArcs*, удалить все найденные соответствия, вариант *var_any* исключить из множества *s_vars*, (фиксируется его безуспешное завершение). Перейти к шагу 8.

Шаг 22. Сформировать множество *m_el_adj_c* элементов, которые могут быть поставлены в соответствие элементу *el_adj* и которые не были поставлены другим элементам шаблона в рамках варианта *var_any*. В это множество не могут быть включены элементы, которые уже были поставлены в соответствие некоторому элементу шаблона в рамках текущего варианта.

Шаг 23. Если множество $m_el_adj_c$ пусто, то перейти к шагу 22 (вариант var_any является неудачным и его следует исключить из множества вариантов).

Шаг 24. Иначе sc-дуга d_an исключается из множества $front$ варианта var_any .

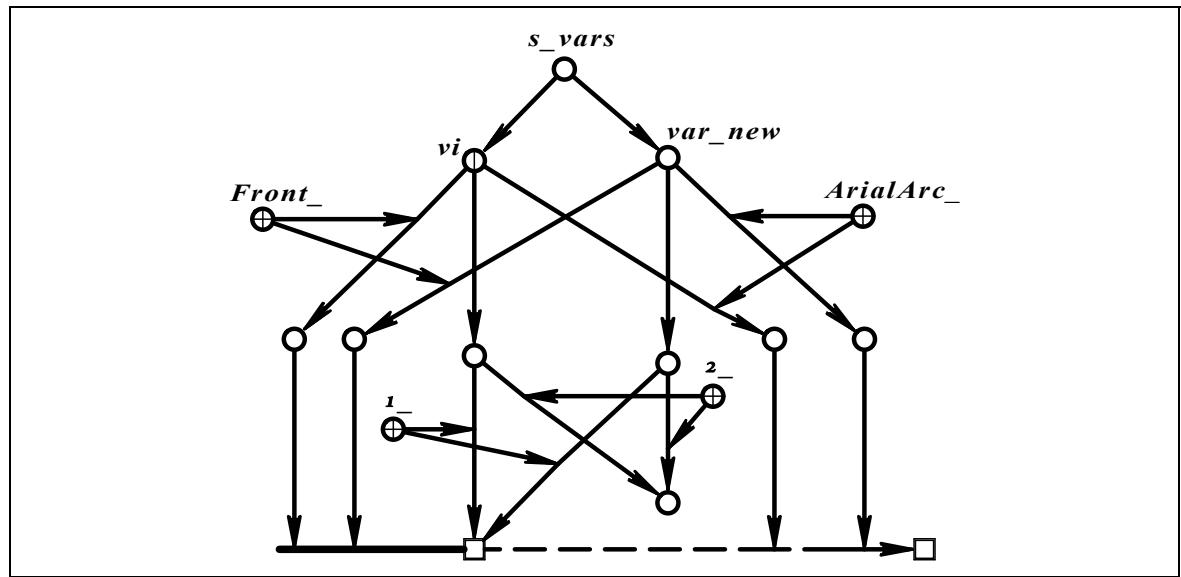
Шаг 25. Если в sc-дугу d_an входят sc-дуги, являющиеся элементами множества $analArcs$ варианта var_any , то занести d_an во множество $front$ варианта var_any .

Шаг 26. Если в sc-элемент el_adj входят sc-дуги, являющиеся элементами множества $analArcs$ варианта var_any , то занести el_adj во множество $front$ варианта var_any .

Шаг 27. Если элемент el_fr больше не инцидентен ни одной sc-дуге, принадлежащей множеству $analArcs$ варианта var_any , то el_fr исключить из $front$ варианта var_any .

Шаг 28. Начало цикла по элементам el_adj_c множества $m_el_adj_c$.

Шаг 29. Завести новый вариант var_new , множества $front$ и $analArcs$ варианта var_any копировать в соответствующие им множества в варианте var_new , также в новый вариант копировать все найденные соответствия.



Шаг 30. Еще раньше, при занесении элемента el_adj_c во множество $m_el_adj_c$ должна была быть зафиксирована sc-дуга d_an_a , связывающая sc-элементы el_fr_c и el_adj_c связью соответствующего направления. Теперь устанавливаем соответствие между элементами d_an и d_an_a , а также между el_adj и el_adj_c в рамках варианта var_new , т.е. присваиваем соответствия элементам d_an и el_adj в рамках варианта var_new .

Шаг 31. Если множество $analArcs$ варианта var_new пусто, то удаляется множества $front$ и $analArcs$ в рамках варианта var_new , вариант var_new исключается из множества s_vars и заносится в результирующее множество s_result . Перейти к шагу 39. (Зафиксировать успешное завершение варианта var_new).

Шаг 32. Если el_adj - sc-узел, то перейти к шагу 43.

Шаг 33. Считаем, что значением da является el_adj .

Шаг 34. Зафиксировать da_beg - начало sc-дуги da ; da_end - конец sc-дуги da .

Шаг 35. Находим sc-дугу da_c , соответствующую sc-дуге da . Зафиксировать: da_c_beg - начало sc-дуги da_c ; da_c_end - конец sc-дуги da_c .

Шаг 36. Если элемент, являющийся соответствием элемента *da_beg* в рамках варианта *var_new* определен и не равен *da_c_beg* или если элемент, являющийся соответствием элемента *da_end* в рамках варианта *var_new* определен и не равен *da_c_end*, то зафиксировать безуспешное завершение варианта *var_new* (уничтожаются множества *front* и *analArcs* варианта *var_new*, уничтожаются все найденные соответствия, вариант *var_new* исключается из множества *s_vars*). Перейти к шагу 43.

Шаг 37. Если элемент, являющийся соответствием элемента *da_beg* в рамках варианта *var_new* не определен, то назначаем соответствующим ему в рамках варианта *var_new* элементом *da_c_beg*.

Шаг 38. Если элемент, являющийся соответствием элемента *da_end* в рамках варианта *var_new* не определен, то назначаем соответствующим ему в рамках варианта *var_new* элементом *da_c_end*.

Шаг 39. Исключить sc-дугу *da* из множества *analArcs* варианта *var_new*.

Шаг 40. Если множество *analArcs* варианта *var_new* пусто, то перейти к шагу 32. (зафиксировать успешное завершение варианта *var_new*).

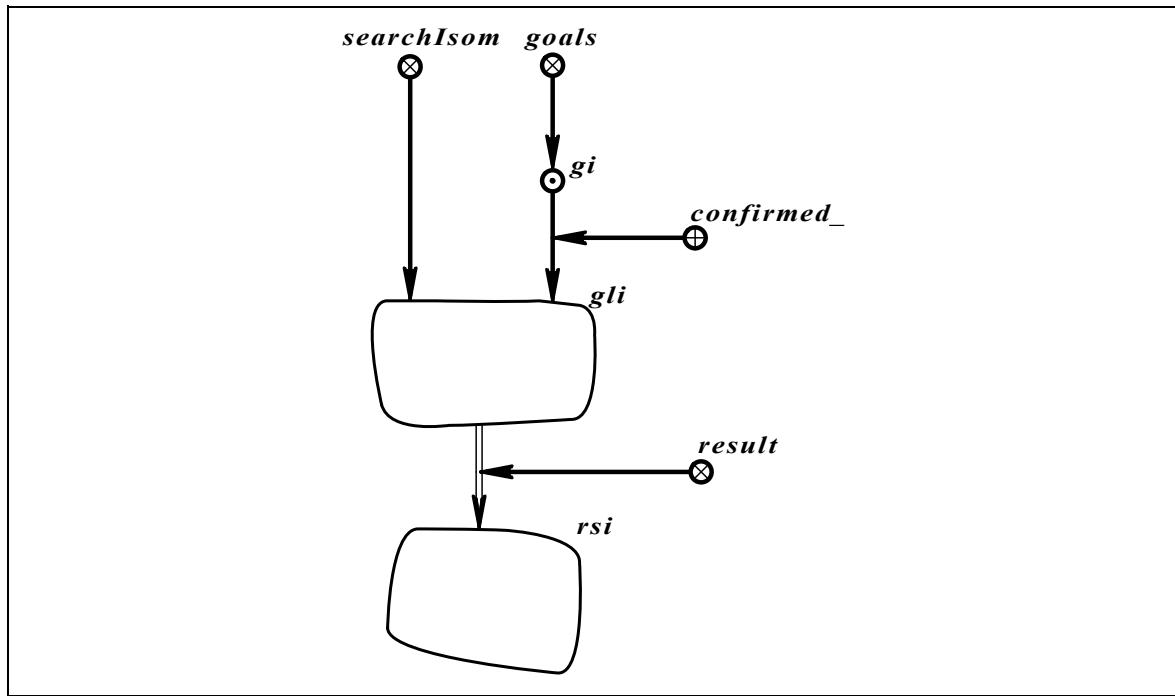
Шаг 41. Если *da_end* - sc-дуга, принадлежащая множеству *analArcs* варианта *var_new*, то Считаем, что значением *da* является *da_end*; Перейти к шагу 37.

Шаг 42. Конец цикла по элементам *el_adj_c* множества *m_el_adj_c*.

Шаг 43. Исключить вариант *var_any* из множества выполняемых вариантов *s_vars*.

Шаг 44. Конец цикла по элементам *var_any* множества *s_vars*.

Шаг 45. Пометить исходную цель как достигнутую, что сводится к формированию следующей sc-конструкции:



Конец микропрограммы.

Пример выполнения операции изоморфного поиска приведен в подразделе 7.2.

Реализация операции изоморфного поиска на языке SCP приведена в [411] ([ПрогрВАМ-2001кн](#))

1.1.4. Семейство операций поиска семантических окрестностей указываемого sc-элемента

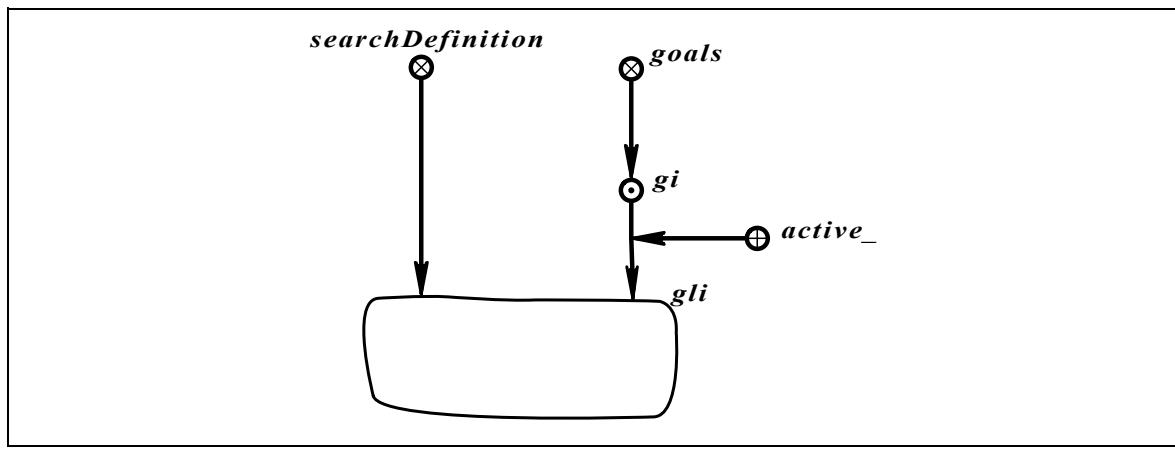
Ключевые понятия: формальная теория; высказывание; определение; семантическая окрестность.

В рассматриваемое семейство операций навигационно-поисковой графодинамической ассоциативной машины входят следующие операции:

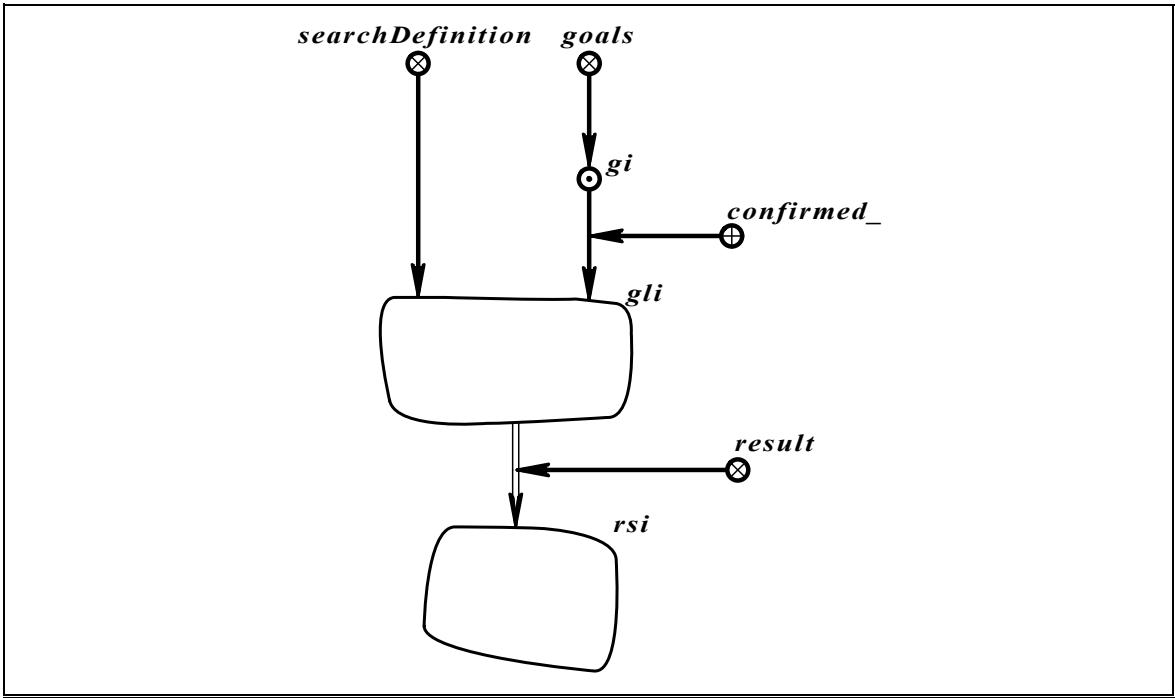
- операция поиска всех высказываний всевозможных формальных теорий, в состав которых указываемый sc-элемент входит в качестве элемента какой-либо атомарной логической формулы;
- операция поиска в рамках указываемой формальной теории всех высказываний, в состав которых указываемый sc-элемент входит в качестве элемента какой-либо атомарной логической формулы;
- операция поиска всех определений указываемого понятия в рамках всевозможных формальных теорий или в рамках указываемой формальной теории;
- операция поиска в рамках указываемой формальной теории всех истинных высказываний, которые являются утверждениями, описывающими свойства (закономерности) понятий, обозначаемого указываемым sc-элементом;
- операция вывода классификационной схемы указываемого понятия.

Рассмотрим **операцию поиска определения понятия** указываемого sc-элемента

Условием выполнения операции поиска определения понятия указываемого sc-элемента является наличие в памяти конструкции вида:



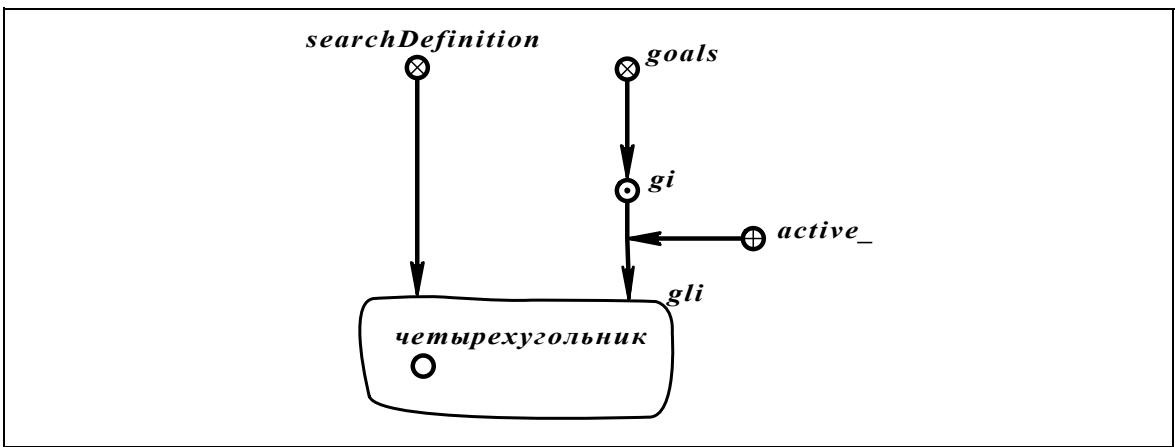
Результатом выполнения операции поиска определения понятия указываемого sc-элемента является генерация конструкции свидетельствующее о том, что запрос успешно обработан. Результаты операции находятся в сформированном множестве *rsi*.



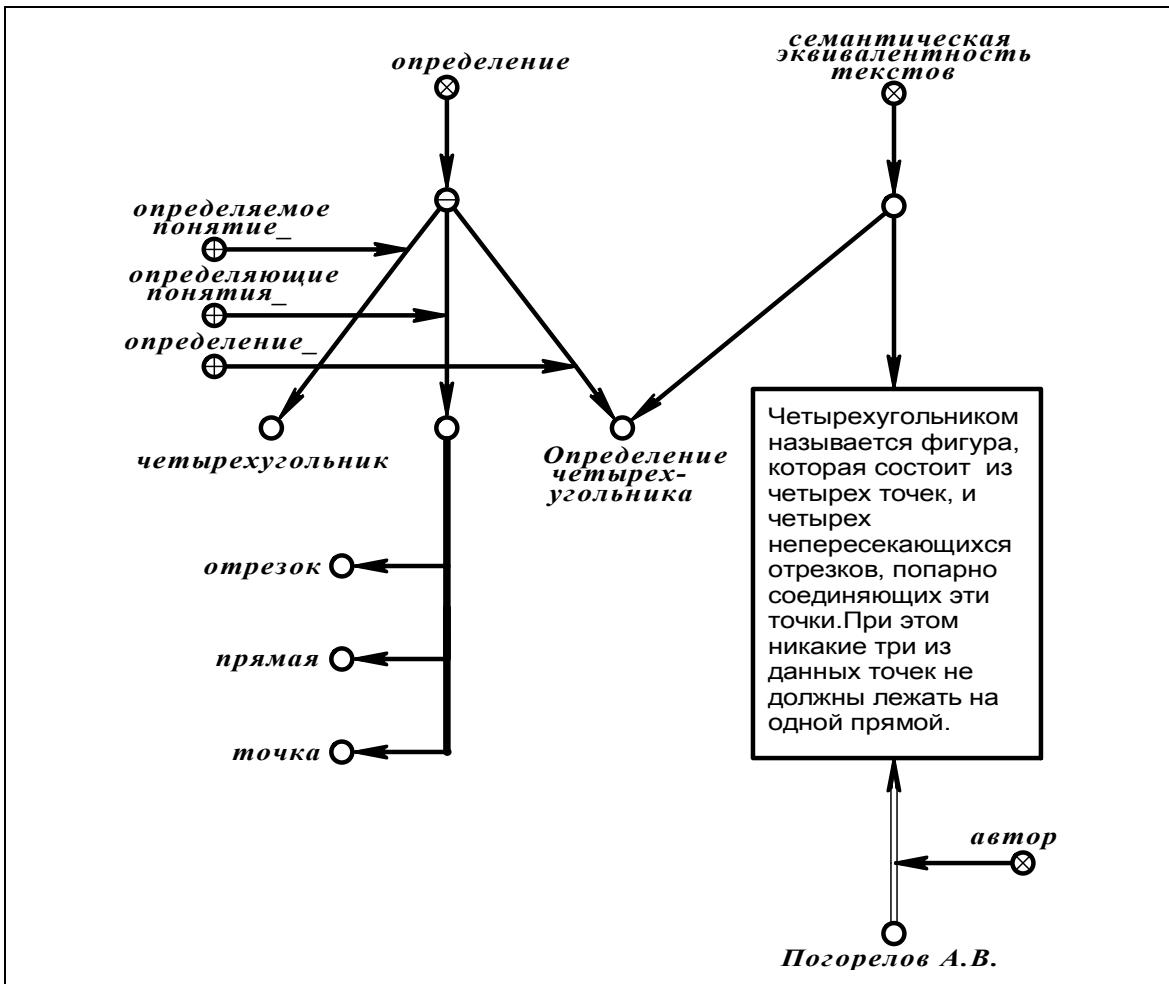
При выполнении операции осуществляется поиск отношения *определение*, в которое входит sc-элемент, определение которого мы ищем, с атрибутом *определяемое понятие*. Далее ищется sc-конструкция, содержащая запись определения на естественном языке, а также конструкция, описывающая библиографическую ссылку. Все эти найденные конструкции включаются в результатирующее множество.

Приведем пример работы операции поиска определения понятия.

Пусть требуется найти определение для понятия “четырехугольник”. Целевое множество выглядит следующим образом:



В результате выполнения операции в результирующем множестве будет следующая конструкция:



Здесь sc-узел с идентификатором “*Определение четырехугольника*” обозначает формальное определение понятия *четырехугольник*. Так же этот sc-узел связан с определением понятия *четырехугольник*, которое записано на русском языке.

1.1.5. Семейство операций поиска семантической связи между (двумя или более) указываемыми sc-элементами

В рассматриваемое семейство операций навигационно-поисковой графодинамической ассоциативной машины входят следующие операции:

- операция поиска теоретико-множественных связей между указываемыми sc-элементами;
- операция поиска семантической связи “*определенное-определяющее понятие*” между двумя указываемыми sc-элементами;
- операция поиска вхождения указываемых sc-элементов в одни и те же утверждения;
- операция поиска минимальных теоретико-графовых маршрутов указываемых sc-элементов по связкам различных отношений.

1.1.6. Семейство навигационно-поисковых операций в гипертекстовой семантической сети

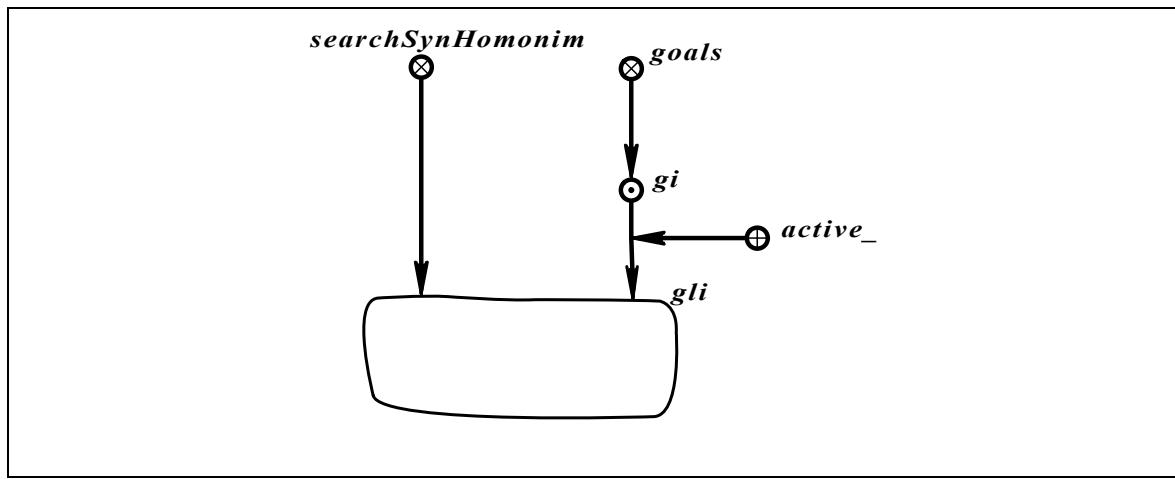
Ключевые понятия: гипертекстовая семантическая сеть, синонимы, омонимы.

В рассматриваемое семейство операций навигационно-поисковой графодинамической ассоциативной машины входят следующие операции:

- операция поиска всех синонимов и омонимов указываемого sc-элемента (синонимичные sc-элементы – это семантические эквивалентные sc-элементы, имеющие разные идентификаторы; омонимичные sc-элементы – это семантически неэквивалентные sc-элементы, имеющие одинаковые идентификаторы);
- операция поиска всех константных sc-узлов, содержимое каждого из которых представляет собой информационную конструкцию, являющуюся комментарием для указываемого sc-элемента.

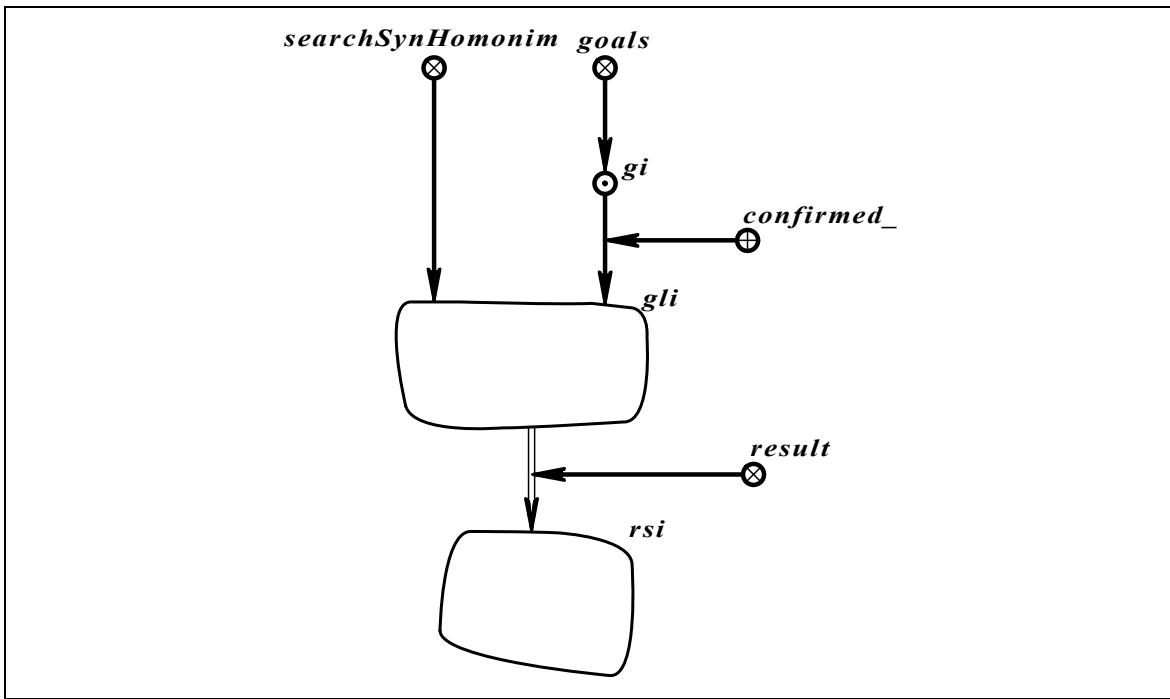
Рассмотрим операцию поиска синонимов и омонимов указываемого sc-элемента.

Условием выполнения операции поиска всех синонимов и омонимов указываемого sc-элемента является наличие в памяти конструкции вида:



Здесь в множество *gli* включаются те sc-элементы, синонимы и омонимы которых необходимо найти.

Результатом выполнения операции поиска всех синонимов и омонимов указываемого sc-элемента является формирование результирующего множества *rsi*, которое содержит синонимы и омонимы указываемого sc-элемента, а также с указанием отношений “**синонимичные sc-элементы**”, “**главный синоним**”, “**омонимичные sc-элементы**”

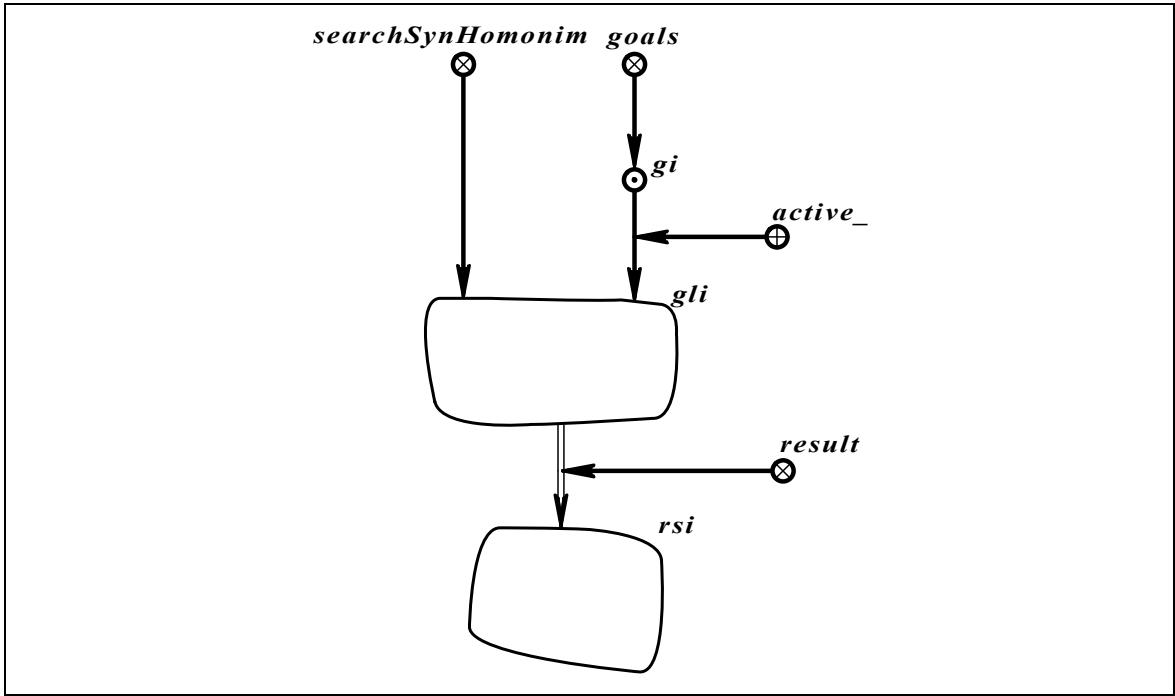


Микропрограмма операции поиска всех синонимов и омонимов указываемого sc-элемента:

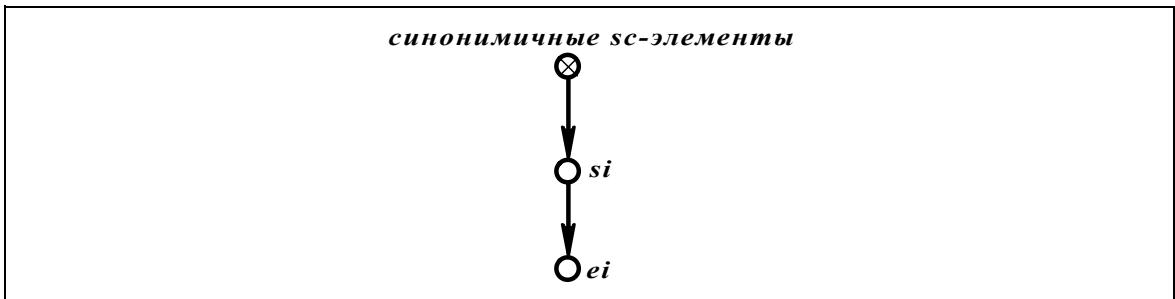
Шаг 1. Проверить условие выполнения операции, если конструкция найдена, то перейти к шагу 2, иначе шаг 1.

Шаг 2. Найти множество sc-элементов (обозначим его *gli*), которое является описанием задачи. Элементами этого множества являются sc-элементы, для которых надо найти синонимичные sc-элементы и омонимичные sc-элементы

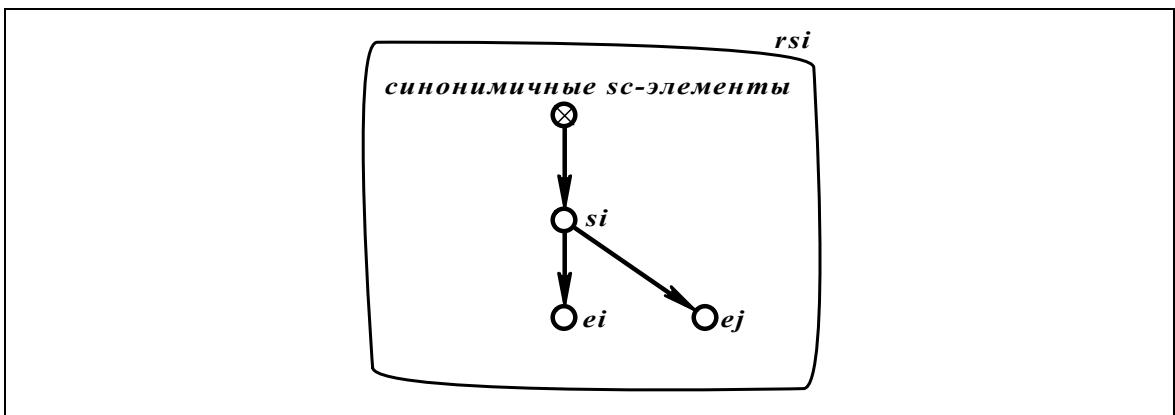
Шаг 3. Сформировать множество (обозначим его rsi), описывающее результат обработки запроса, т.е. надо сформировать следующую sc-конструкцию:



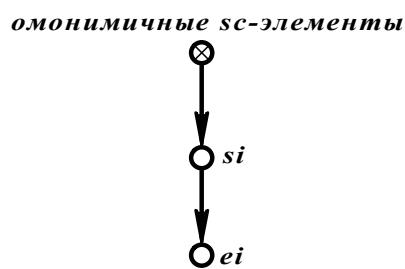
Шаг 4. Для каждого sc-элемента (обозначим его ei) из множества gli найти знак множества синонимичных sc-элементов (обозначим его si), т.е. надо найти sc-конструкцию вида:



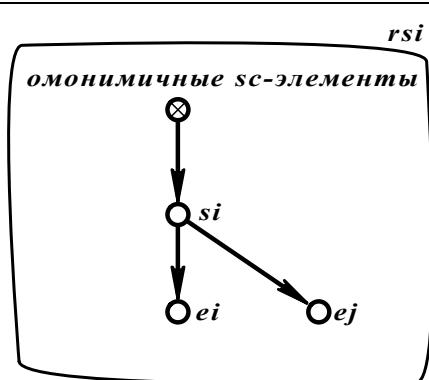
Шаг 5. Включить все элементы множества si в результирующее множество (rsi), т.е. надо сформировать sc-конструкцию вида:



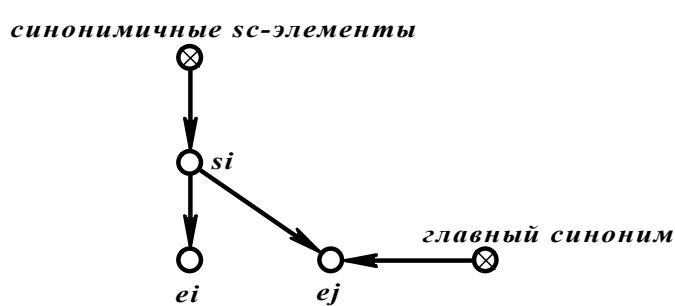
Шаг 6. Для каждого sc-элемента (обозначим его *ei*) из множества *gli* найти знак множества омонимичных sc-элементов (обозначим его *si*), т.е. надо найти sc-конструкцию вида:



Шаг 7. Включить все элементы множества *si* в результирующее множество (*rsi*), т.е. надо сформировать sc-конструкцию вида:

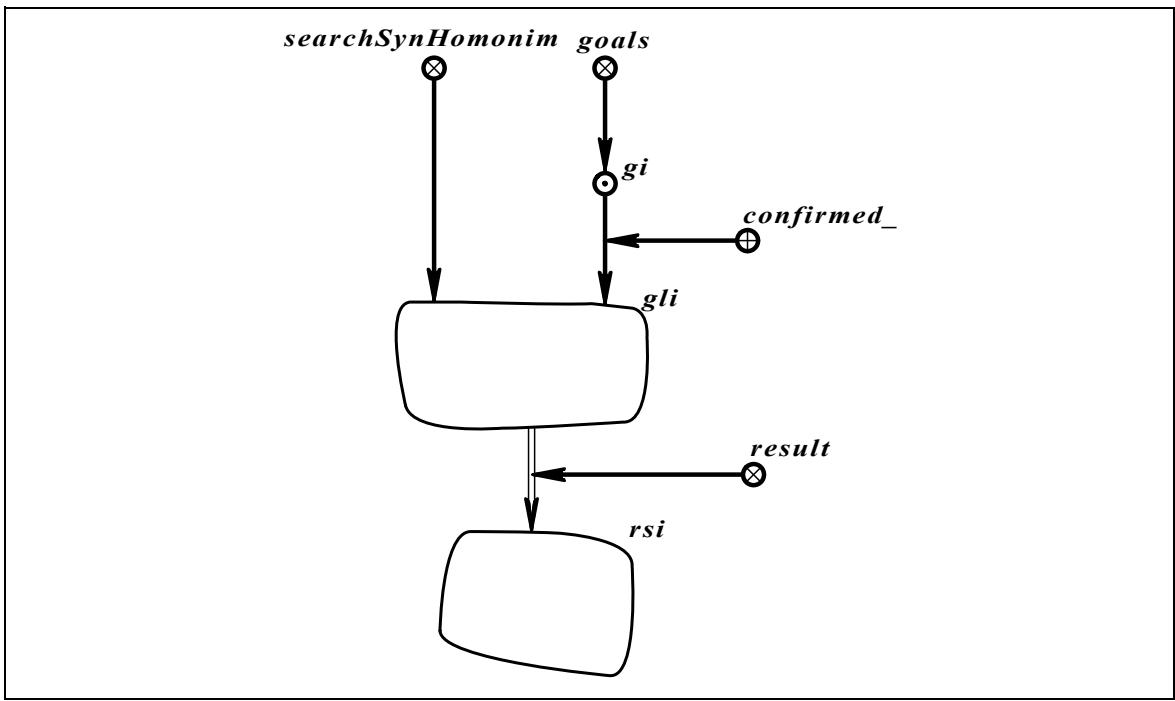


Шаг 8. Для каждого sc-элемента, омонима, (обозначим его *vi*) из множества *si* найти **главный синоним**, т.е. надо найти sc-конструкцию вида:



Шаг 9. Включить найденную sc-конструкцию в результирующее множество (*rsi*).

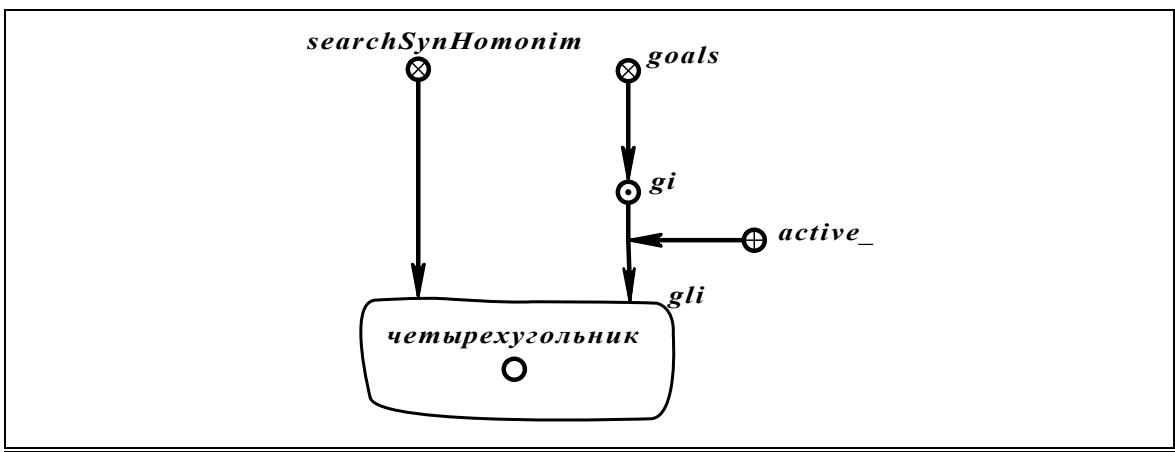
Шаг 10. Сгенерировать sc-конструкцию, свидетельствующую о том, что цель обработана, т.е удалить sc-дугу между ключевым узлом *active* и sc-узлом цели (*gli*) и сгенерировать sc-дугу между ключевым узлом *confirmed* и sc-узлом цели (*gli*):



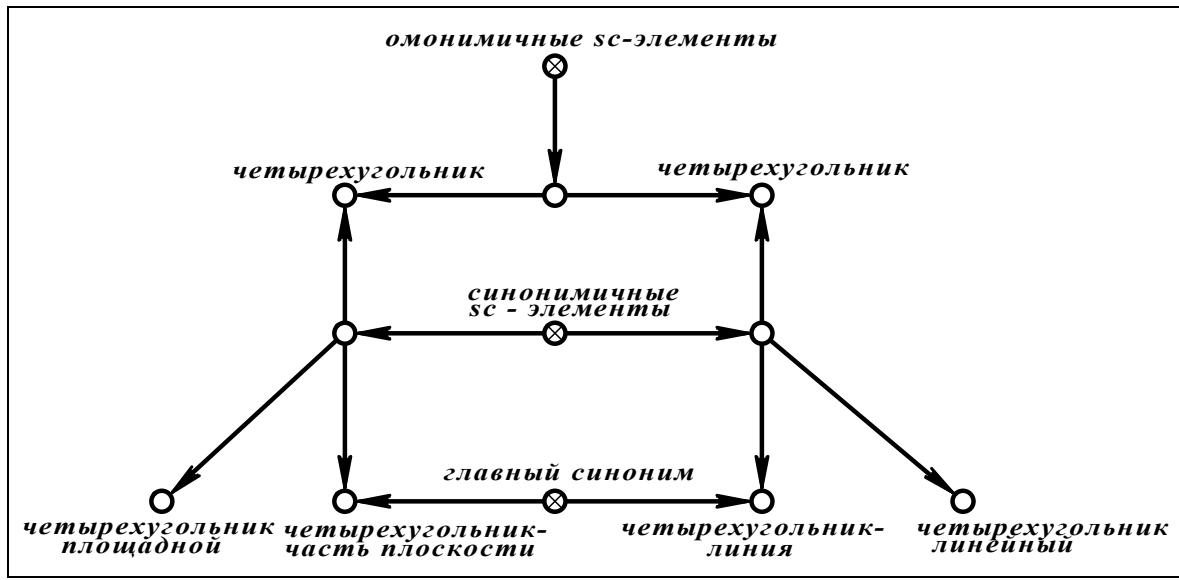
Конец микропрограммы.

Приведем пример работы операции поиска всех синонимов и омонимов указанного sc-элемента.

Пусть требуется найти все синонимы и омонимы для понятия “четырехугольник”. Целевое множество выглядит следующим образом:



В результате выполнения операции в результирующем множестве будет следующая конструкция:



Здесь sc-узлы с идентификаторами “четырехугольник площадной” “четырехугольник – часть плоскости” являются синонимами понятия четырехугольник. А sc-узел с идентификатором “четырехугольник” является нетривиальным омонимом (см. подраздел 6.4) понятию четырехугольник. Эти два понятия различаются *главными синонимами*.

Резюме к подразделу 7.1

Приведенная классификация операций навигационно-поисковой графодинамической ассоциативной машины является открытой, т.е. набор операций может быть легко расширен.

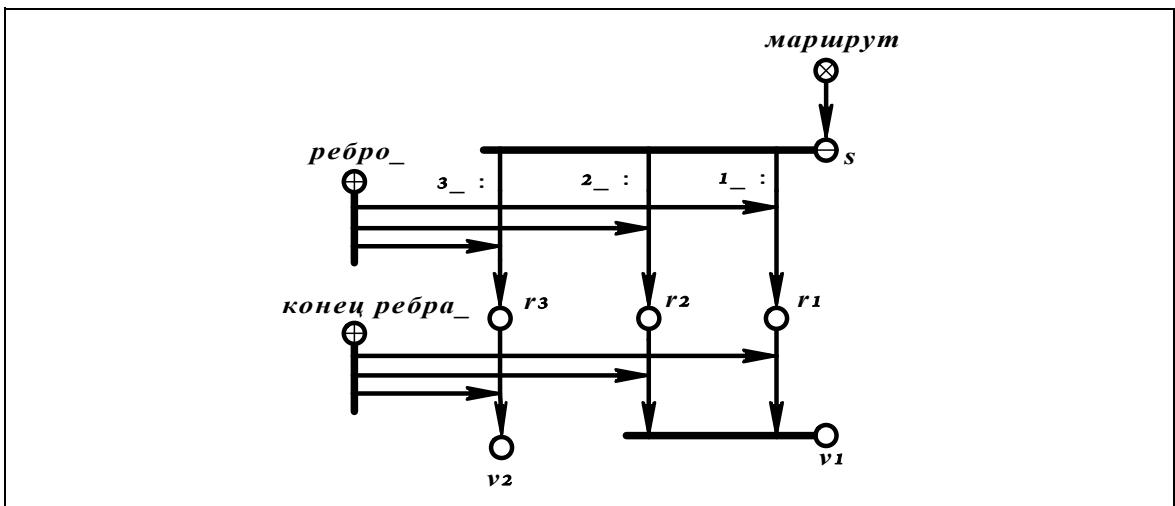
Если в структуре цели указан отправитель, т.е. цель инициирована пользователем, то результат выполнения цели выводится пользователю на виртуальный экран графодинамической ассоциативной машины (см. раздел 5 в [411] ([ПрогрВАМ-2001кн](#)))

1.2. Пример работы навигационно-поисковой графодинамической ассоциативной машины

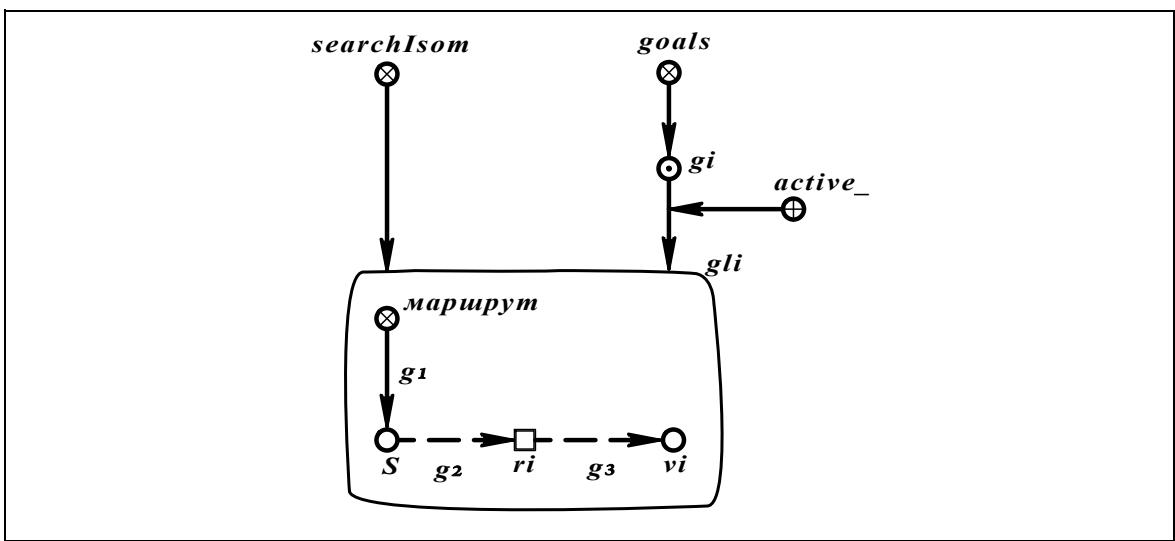
Ключевые понятия: цель; изоморфный поиск; результат.

Рассмотрим работу навигационно-поисковой графодинамической ассоциативной машины на примере операции изоморфного поиска.

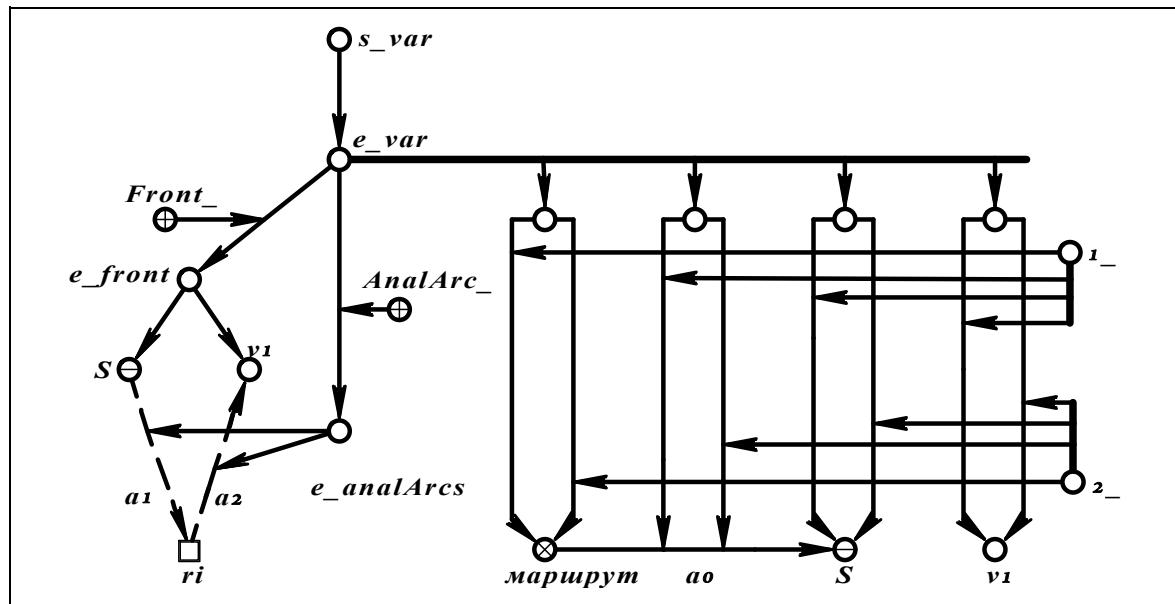
Пусть в памяти имеется конструкция, описывающая маршрут **S** в некотором графе, и нам необходимо найти участки этого маршрута, которые проходят через вершину **v1**.



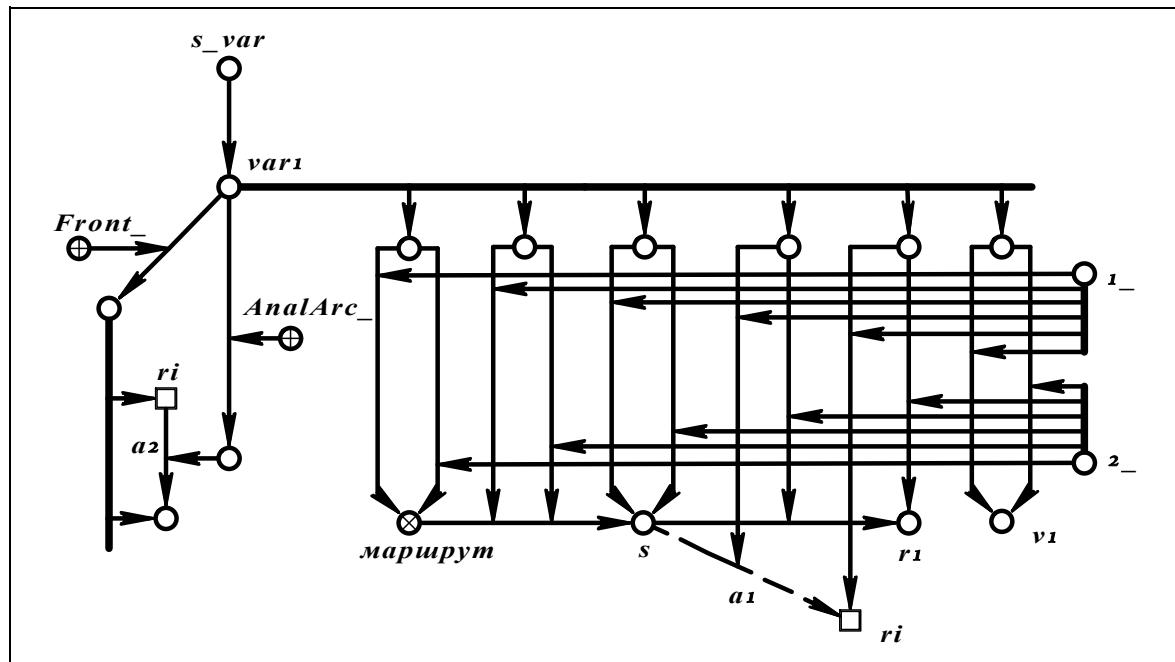
Конструкция цели выглядит следующим образом.

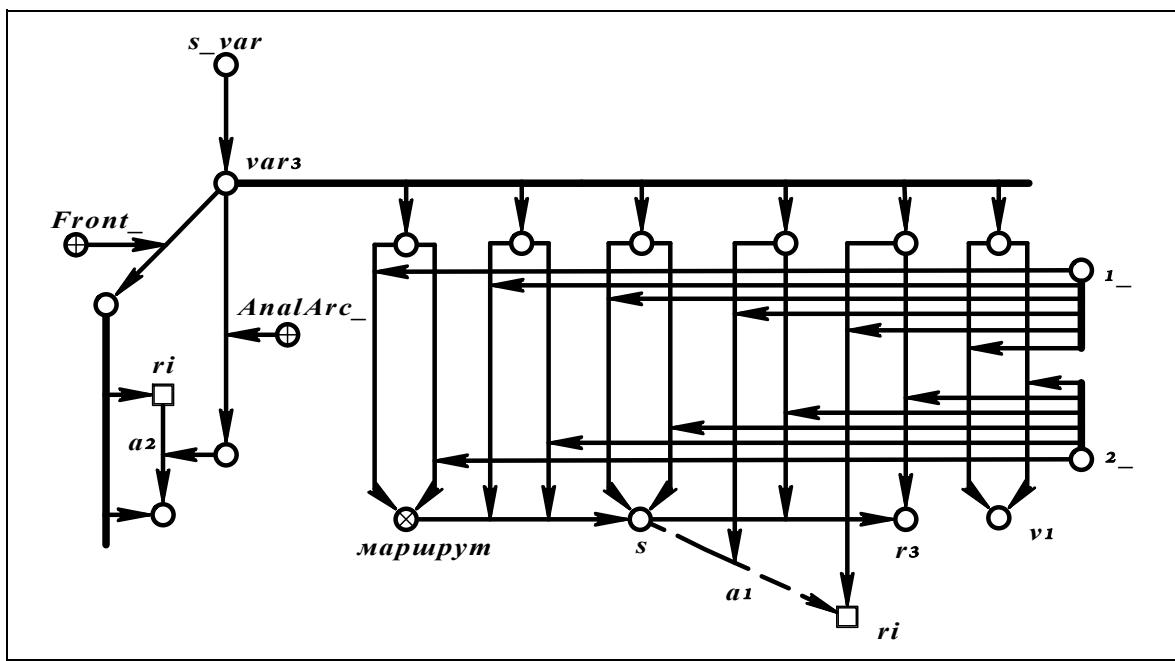
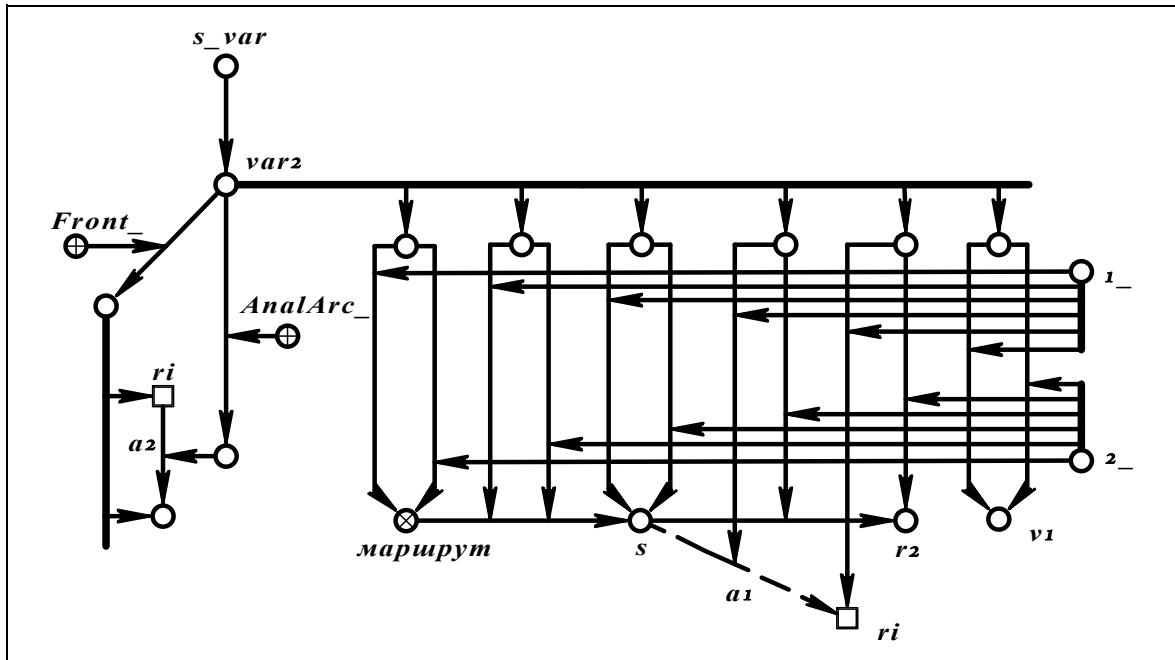


На первом этапе (шаги 1-7) происходит поиск запроса, разбор шаблона. Так константные элементы шаблона заносятся во множество **front**, ставятся в соответствие сами себе в рамках начального варианта, переменные дуги заносятся во множество **analArcs**:

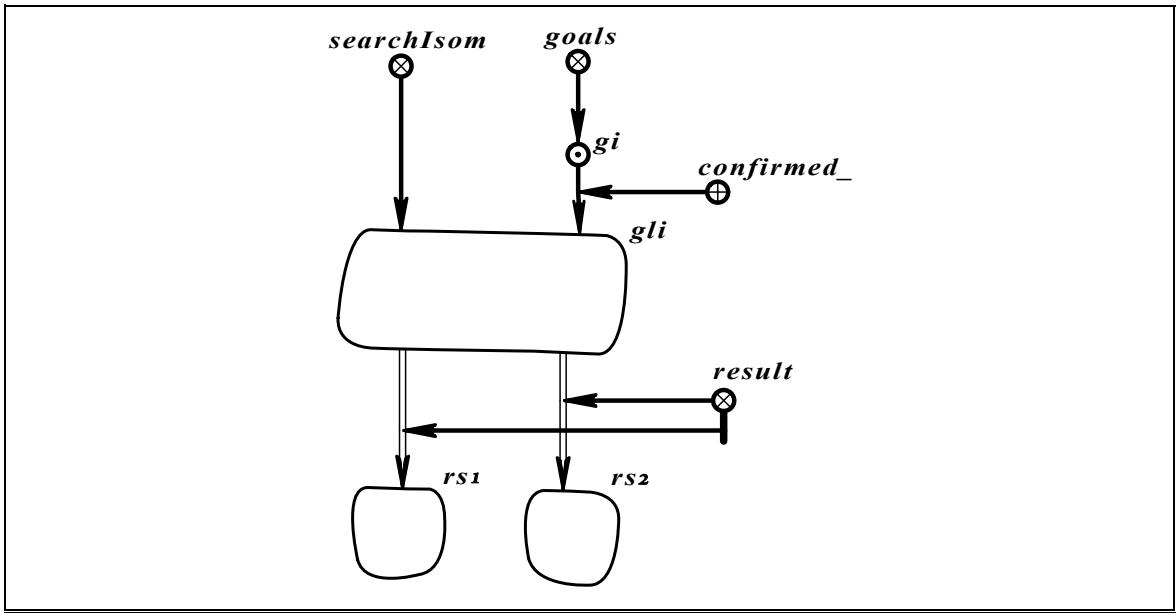


Далее начинается разбор вариантов, находящихся во множестве **s_var**. Берем любой элемент множества **e_front**, например, возьмем узел **S**. Берем любую дугу инцидентную узлу **S** и принадлежащую множеству **e_analArcs**. В нашем случае это будет дуга **a1**, как единственно возможная. Затем формируется множество дуг, которые могут быть поставлены в соответствие этой дуге. То есть ищем все константные дуги, выходящие из узла **S** (т.к. этот узел константный, то он соответствует сам себе) и которые входят в константный узел (дуга **a1** входит в переменный узел), при этом найденные дуги и узлы, в которые они входят, должны не иметь соответствий в рамках данного варианта. В нашем случае это будут три дуги: (**S** → **r1**), (**S** → **r2**), (**S** → **r3**). Формируем новые варианты и удаляем старый. Так, в результате обработки начального варианта получится три новых варианта:

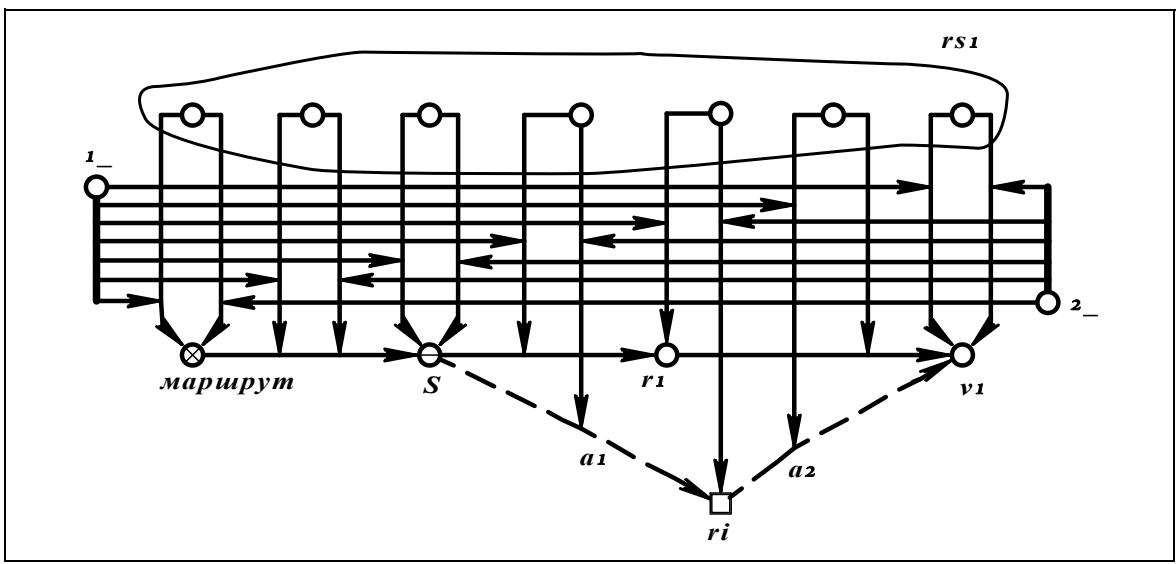




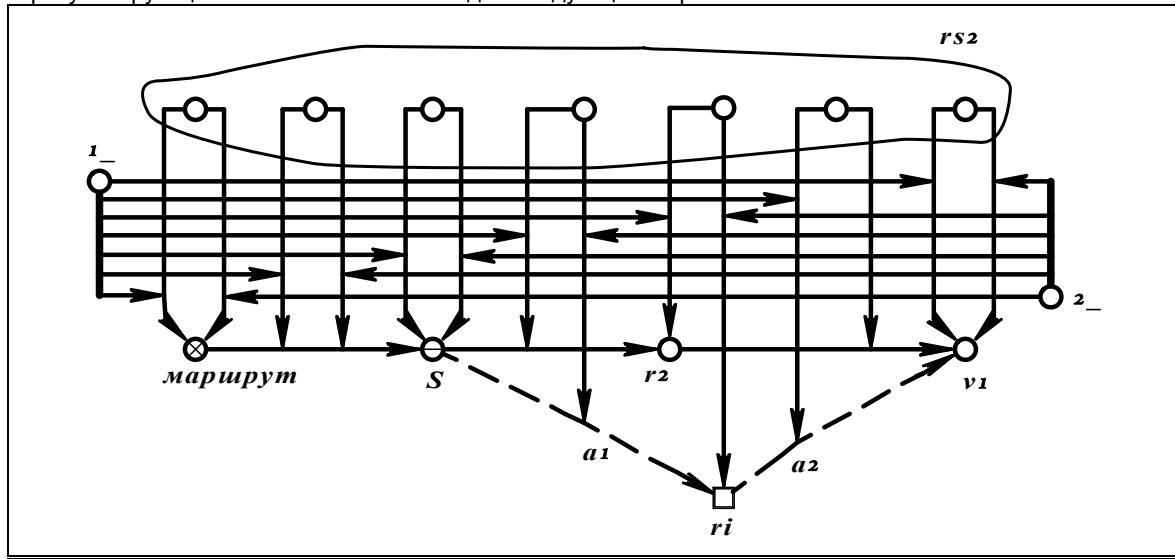
Далее процесс повторяется: берем вариант из множества вариантов, проверяем, закончен ли он (если закончен, то множество *analArcs* будет пустым). Если вариант не закончен, то пытаемся развить его дальше. То есть берем элемент фронта, инцидентную ему дугу из множества *analArcs* и ищем те элементы, которые могут быть сопоставлены им. Если таких элементов не оказывается, то данный вариант уничтожается и процесс начинается сначала. В противном случае для каждого возможного соответствия генерируется новый вариант. Если множество вариантов оказывается пустым, то уничтожаем знак этого множества и указываем, что операция завершилась. Так выглядит результат выполнения операции в нашем примере:



Здесь результирующее множество *rs1* выглядит следующим образом:



А результирующее множество rs_2 выглядит следующим образом:



Выводы к разделу 7

Рассмотренная в данном разделе навигационно-поисковая графодинамическая ассоциативная машина является частью любой другой графодинамической ассоциативной машины.

Одним из актуальных направлений использования навигационно-поисковой графодинамической ассоциативной машины являются ассоциативные электронные учебники, в которых учебный материал представлен в виде гипертекстовой семантической сети (см. подраздел 6.4). Подробно ассоциативные электронные учебники рассмотрены в книге [236] (*ИнтелОСиВУО-2001кн*).

Реализация навигационно-поисковой графодинамической ассоциативной машины и пользовательский интерфейс рассмотрены в книге [411] (*ПрогрВАМ-2001кн*).

1. Графодинамические ассоциативные машины вывода

В данном разделе описывается абстрактная графодинамическая ассоциативная машина вывода, обеспечивающая решение задач в рамках формальных теорий, описанных на языке SCL. Отметим, что графодинамическая ассоциативная машина вывода может быть легко интегрирована с навигационно-поисковой машиной описанной в разделе 7.

Данный раздел может быть использован в качестве учебного пособия по дисциплине «Логические основы интеллектуальных систем» для студентов специальности «Искусственный интеллект».

1.1. Семейство легко интегрируемых абстрактных графодинамических ассоциативных машин вывода

Ключевые понятия: графодинамическая ассоциативная машина вывода, scl-операция, цель, и-подцель, или-подцель, микропрограмма.

Открытые языки представления знаний (к числу которых относится и язык SCL) имеют в общем случае открытую нефиксированную операционную семантику. Это значит, что каждому языку представления знаний может соответствовать целое семейство абстрактных машин, поддерживающих разные стратегии решения задач на основе этого языка. Важнейшим средством повышения "интеллектуального" уровня машины переработки знаний (т.е. средством расширения множества решаемых ею задач) является интеграция этой машины с другой машиной переработки знаний, использующей в качестве внутреннего языка тот же язык представления знаний. Поэтому основным требованием, предъявляемым к современным машинам переработки знаний, наряду с открытым характером используемого языка представления знаний (что необходимо для расширения его изобразительных возможностей) является открытый характер операционной семантики указанного языка (что необходимо для неограниченного расширения логических возможностей этой машины).

Будем называть такие **scl-машинами** машины переработки знаний, которые: 1) в качестве внутреннего языка используют язык SCL, 2) в качестве вспомогательных информационных конструкций, необходимых для реализации микропрограмм, используют только sc-конструкции, хранимые в их памяти. Из вышесказанного, из определения sc-машин ([см. подраздел 4.7](#)) и из того, что язык SCL является подъязыком языка SC, следует, что все scl-машины относятся к семейству sc-машин. Как было отмечено в [подразделе 4.7](#), все sc-машины имеют открытый характер и легко интегрируются друг с другом. Для интеграции абстрактных sc-машин необходимо интегрировать хранимые в их памяти sc-конструкции, склеив синонимичные sc-узлы, и объединить наборы их операций. Таким образом, абстрактные scl-машины, являющиеся частным видом абстрактных sc-машин, также имеют открытый характер и легко интегрируются друг с другом.

Технология реализации scl-машин должна обеспечивать и поддерживать их открытый, легко расширяемый характер. Поэтому для реализации scl-машин важное значение имеет не только уточнение набора операций scl-машины (такие операции в дальнейшем будем называть **scl-операциями**), которым ставятся в соответствие демонические **микропрограммы scl-машины**, описывающие реализацию этих операций, но и уточнение системы базовых микропрограмм scl-машин, а также уточнение самого языка микропрограммирования, ориентированного на реализацию всего семейства scl-машин. Хорошо продуманная система базовых микропрограмм и язык микропрограммирования, ориентированный на все семейство scl-машин, образуют эффективную инструментальную среду, позволяющую достаточно легко расширять и модифицировать систему операций scl-машины.

В качестве языка микропрограммирования, ориентированного на реализацию scl-машин, предлагается язык SCP (Semantic Code Programming), являющийся графовым языком параллельного процедурного программирования. Язык SCP и абстрактная sc-машина, определяющая операционную семантику этого языка, описаны в разделе 4 книги [411] ([ПрогрВАМ-2001кн](#)). Интерпретируя scl-машину на sc-машине, осуществляется переход от sc-машины переработки знаний к sc-машине более низкого уровня.

Подробное рассмотрение базовых микропрограмм абстрактной scl-машины приведено в работе [153] ([Голенков В.В..1996мо-БазовПТЯSCL](#)). В этом подразделе ниже приведено естественноязыковое описание микропрограмм некоторых операций.

Во множестве scl-операций, в частности, можно выделить следующие классы:

- 1) операции, обеспечивающие выполнение элементарных заданий по преобразованию текущего состояния соответствующего знания, в состав которого указанные задания входят. К числу таких операций, обеспечивающих решение элементарных задач, относятся информационно-поисковые, арифметические, строковые, теоретико-множественные операции, например, **операция эпизодического информационного поиска**;
- 2) операции, обеспечивающие реализацию процедурных (в том числе продукционных) программ различного вида, а точнее, реализацию различных операторов, составляющих эти программы. Процедурным программам каждого вида соответствует определенный набор операторов;
- 3) операции, обеспечивающие решение задач в базах знаний путем ассоциативного поиска подходящих программ, интегрированных в базы знаний;
- 4) набор операций, обеспечивающих сведение неэлементарных заданий к подзаданиям (в конечном счете – к элементарным заданиям), например, **операция трассировки запроса сверху-вниз**, **операция трассировки запроса снизу-вверх**, **операция декомпозиции запроса**;
- 5) операции, обеспечивающие выполнение неэлементарного задания после выполнения всех заданий, составляющих один из наборов И-подзаданий указанного неэлементарного задания, например, **операция выполнения логических рассуждений**;
- 6) операции стирания (снятия) всех явных и неявных подзаданий для уже выполненных заданий. При этом удаляются все сопутствующие вспомогательные структуры, в том числе и связки отношения **subGoal** ;
- 7) операции дедуктивного логического вывода, например, **операция реализации продукции**;
- 8) операции поиска и устранения противоречий в знаниях (например, на основе высказываний об однозначности) ;
- 9) операции, обеспечивающие поиск синонимичных sc-узлов с последующим их склеиванием, например, **операция склейки идентичных формул**;
- 10) операции ввода, например, **операция добавления факта** и **операция добавления высказывания**;
- 11) операции, обеспечивающие вывод сформированных ответов на пользовательские запросы;
- 12) операции "сборки мусора", обеспечивающие удаление из базы знаний ненужной информации, в частности, информации, которая редко используется и может быть при необходимости логически восстановлена (**операция уничтожения промежуточных конструкций**) .

Операция (логический механизм) , обеспечивающая решение некоторого класса элементарных задач, осуществляет окончательное разрешение некоторой задачной ситуации, т.е. получает окончательный ответ на некоторое задание. Примерами таких операций для scl-машины являются: операции информационного поиска, арифметические операции (операция сложения, операция вычитания, операция умножения и т.д.) , операции стирания.

Операция информационного ассоциативного поиска осуществляет поиск ответа на некоторый вопрос, предполагая, что ответ на это задание непосредственно содержится в текущем состоянии scl-теории.

Операция арифметического сложения инициируется тогда, когда возникает задачная ситуация, содержащая 1) задание на вычисление пока неизвестного значения некоторой числовой константы и 2) связку, обозначающую элемент отношения сложения и указывающую, что числовая константа, имеющая искомое значение, является суммой двух или нескольких других числовых констант, имеющих известные (вычисленные) значения. Результатом выполнения операции сложения является ситуация, в которой запрашиваемое (искомое) значение числовой константы оказывается вычисленным и занесенным в содержимое sc-узла, обозначающего эту числовую константу. Аналогичным образом выполняются и другие арифметические операции.

К числу операций, обеспечивающих разрешение задачных ситуаций путем использования программ, относится операция поиска и инициирования имеющейся в памяти программы (алгоритма) , реализация которой обеспечивает получение ответа на некоторое задание, а также целый ряд операций, обеспечивающих реализацию инициированных программ. Для выполнения операции поиска нужной программы из числа имеющихся необходимо, чтобы каждая из программ, входящих в состав базы знаний, снабжалась обобщенным описанием того класса задачных ситуаций, с которым она справляется. В состав такого обобщенного описания класса задач может входить описание характера (вида) задания и описание исходной дополнительной информации, необходимой для выполнения программы.

Операции сведения задач к подзадачам реализуют различные стратегии целенаправленного логического вывода (различные стратегии решения задач), предполагающие использование той или иной информации, содержащейся в базе знаний. В результате выполнения этих операций для исходного задания, определяющего исходную задачную ситуацию, генерируется иерархическая система подзаданий (наводящих вопросов). Каждое задание в общем случае сводится к нескольким подзаданиям, на которые нужно получить ответы для того, чтобы мог быть сформирован ответ на исходное задание. В этом суть разбиения задач на И-подзадачи. Однако каждое задание может быть разбито на систему И-подзаданий в общем случае несколькими альтернативными способами. В этом суть разбиения задач на ИЛИ-подзадачи.

В качестве примера рассмотрим одну из операций сведения арифметической задачи к подзадачам. Пусть получен запрос на вычисление некоторой числовой константы и пусть известно, что эта числовая константа является суммой двух других констант, одна из которых определена (вычислена), а другая – нет. Тогда задание на вычисление второй константы будет оформлено как подзадание исходного задания (в результате выполнения операции сведения задачи к подзадачам).

Рассмотрим также логический механизм сведения задачи к подзадачам, предполагающий использование программ, хранимых в базе знаний. Суть этого механизма заключается в следующем. Если в базе знаний отсутствует программа, обеспечивающая построение ответа на некоторое задание, то ищется такая программа, которая могла бы построить этот ответ, но для выполнения которой недостаточно некоторых исходных данных. Задание на построение этих недостающих исходных данных оформляется как подзадание по отношению к исходному заданию. Механизм такой генерации подзаданий можно трактовать как один из механизмов целенаправленного синтеза требуемых программ из набора имеющихся.

В заключение еще раз подчеркнем, что рассмотренный набор операций абстрактной scl-машины легко расширяется без изменения денотационной семантики языка SCL (т.е. без изменения набора ключевых узлов языка SCL). Расширение набора операций scl-машины, в частности, может быть направлено на поддержку (реализацию) новых стратегий решения задач.

Условием применения каждой scl-операции является наличие в памяти scl-машины соответствующей sc-конструкции. Разным scl-операциям соответствуют разные инициирующие их sc-конструкции.

1.1.1. Информационные конструкции, описывающие состояние абстрактной графодинамической ассоциативной машины вывода

Ключевые понятия и идентификаторы ключевых узлов: формула, цель, *goals*, семантически близкие высказывания, фактографическое высказывание, релевантные sc-конструкции.

Построение хорошо продуманной системы базовых микропрограмм абстрактной scl-машины требует уточнения и унификации специальных вспомогательных sc-конструкций, используемых этими микропрограммами. Для организации таких sc-конструкций требуется введение sc-узлов, не являющихся ключевыми узлами языка SCL, но являющихся ключевыми узлами самой scl-машины.

Для представления информационных конструкций, описывающих состояние абстрактной графодинамической ассоциативной машины вывода, дополнительно к ключевым узлам языка SCL вводятся следующие ключевые узлы:

- *goals* – знак множества множеств целей, факторизованных по принадлежности к различным формальным теориям;
- *formula* – знак множества формул;
- *active_* – знак множества дуг принадлежности активной цели множеству целей формальной теории;
- *search*, *traceDown*, *traceUp*, *interprete*, *produce*, *deduce*, *associateSimply*, *associate*, *associateExtensively*, *findUnassigned*, *decompose*, *analyseStructure* – знаки множеств дуг принадлежности цели множеству целей формальной теории. Такая цель подлежит первоочередной обработке соответствующей операцией;

- *searched, tracedDown, tracedUp, interpreted, produced, deduced, simplyAssociated, associated, extensivlyAssociated, decomposed* – знаки множеств дуг принадлежности цели множеству целей формальной теории. Такая цель считается обработанной соответствующей операцией.

Опишем типичные конструкции, которые будут необходимы для функционирования операций scl-машины. Отношение *associative* – отношение, соединяющее формулу *fi* с ассоциированными в рамках теории *ti* с ней формулами в результате действия операции поиска семантически близких формул:

$$\text{associative} \rightarrow (\text{(formula} \rightarrow \text{fi}), \text{(theory} \rightarrow \text{ti}), \text{s}) ;$$

Быть атомарным формулой *fi*, имеющим фактографическую интерпретацию:

$$\text{factual} \rightarrow \text{fi} ;$$

Отношение *interpretation* – отношение, между двумя релевантными формулами и их интерпретацией (отношением релевантности) :

$$\begin{aligned} \text{interpretation} &= \text{релевантность sc-конструкций} = \text{релевантность} \rightarrow \\ &(\text{(formula} \rightarrow \text{fi}), \text{(formula} \rightarrow \text{f2}), \text{interpretation_ : Interpretation}) ; \\ \text{interpretation_} &= \text{отношение_} ; \end{aligned}$$

Отношение *selected* – отношение, связывающее формулу *fi* со множеством уже найденных для него релевантных формул (фактографических высказываний) :

$$\text{selected} \rightarrow \text{h} (\text{formula} \rightarrow \text{fi}), \text{ImagesSet}) ;$$

Отношение, связывающее формулу *fi* со множеством *Set*, множеством свободных для этой формулы переменных:

$$\text{unassigned} \rightarrow (\text{(formula} \rightarrow \text{fi}), \text{Set}) ;$$

Отношение, связывающее формулу *f1*, которая включает по структуре формулу *f2*:

$$\text{inclusion} \rightarrow \text{h} (\text{formula} \rightarrow \text{f1}), \text{in_ : (formula} \rightarrow \text{f2})) ;$$

Отношение между основной целью и множеством И-подцелей, используется для формирования ответа для основной цели и для объяснения процесса решения:

$$\text{reasoning} \rightarrow \text{h effect_ : ge , expectations_ : Expectations , causes_ : Causes) ;}$$

где *Causes* – знак множества дуг принадлежности достигнутых И-подцелей множеству целей *Goals* определённой формальной теории, а *Expectations* – знак множества дуг принадлежности не достигнутых И-подцелей, *ge* – дуга принадлежности основной цели множеству *Goals*.

Ключевой узел *explanation* обозначает отношение, связывающее выполненные (достигнутые) задания (цели) или построенные системы И-подзаданий с объясняющими scl-высказываниями. После того, как решение задачи закончено и пользователя перестают интересовать объяснения результатов решения, автоматически формируется задание на "сборку мусора" для решенной задачи, в результате выполнения которого соответствующие *explanation*-структуры будут удалены.

Приведём конструкции, описывающие типичные запросы. Здесь и далее *Goals* – произвольный элемент множества *goals*. Запрос на истинность высказывания *fi*:

```
active_ → (Goals → qi) ;
qi = [theory → ti →> fi] ;
```

Запрос на ложность высказывания **fi**:

```
active_ → (Goals → qi) ;
qi = [theory → ti →> fn →> fc →> fi ; conj →> fc ; negExpr →> fn ; ] ;
```

Запрос на поиск интерпретации высказывательной формы **fi**:

```
active_ → (Goals → Set) ;
unassigned → 1 ( formula → fi ) , Set ) ;
```

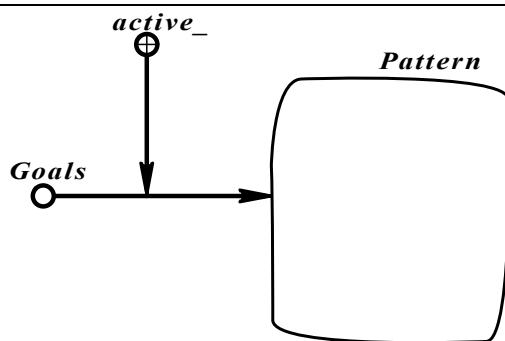
1.1.2. Операции абстрактной графодинамической ассоциативной машины вывода

Ключевые понятия: scl-операции, микропрограмма scl-машины.

Рассмотрим некоторые операции абстрактной графодинамической ассоциативной машины логического вывода. Отметим, что каждая приведённая ниже операция имеет свой абсолютный приоритет выполнения, оцениваемый по шкале от 0 до 1 (см. подраздел 6.1).

Операция классификации и управления запросами (SCL query classification & control operation) в зависимости от структуры запроса включает его в соответствующее множество приоритетных запросов, обрабатываемых какой-либо из перечисленных ниже специализированных операций.

Условием выполнения операции классификации и управления запросами является наличие конструкции вида:



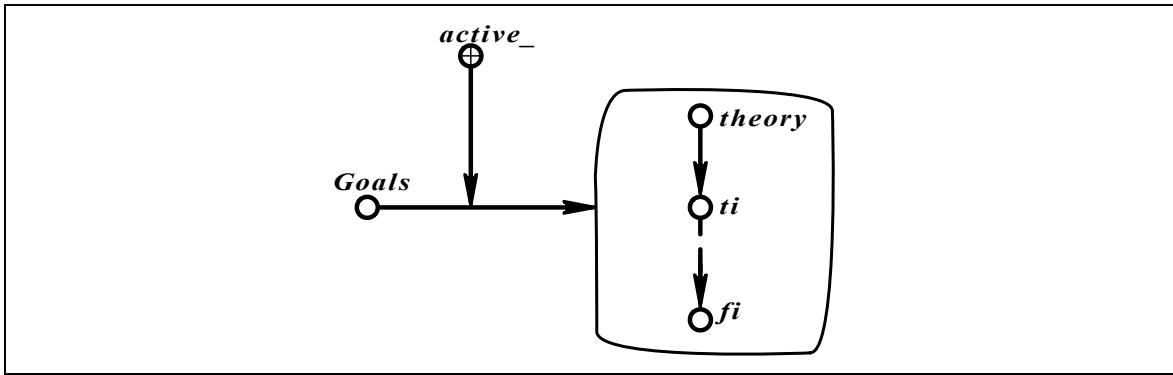
/* необходимое условие запуска операции – наличие дуги из **active_** */

Здесь “**Pattern**” – образец конструкции запроса, а “**Goals**” – множество целей данной формальной теории.

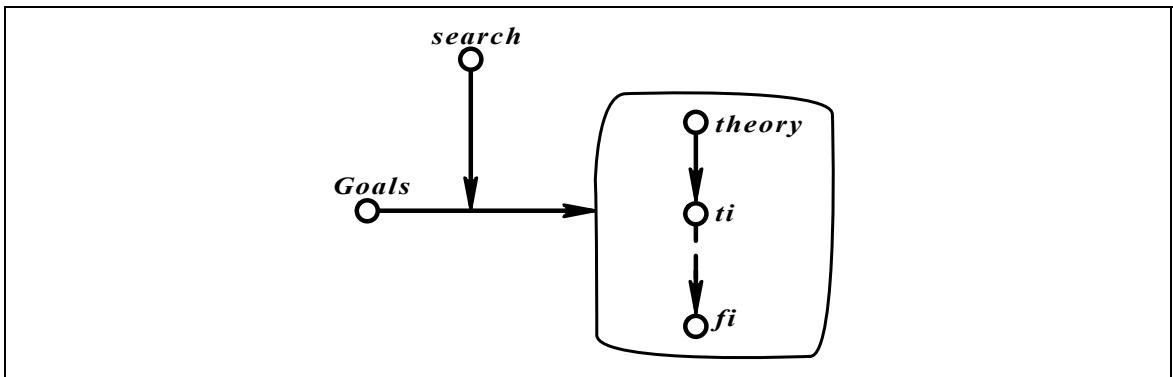
Результатом выполнения операции классификации и управления запросами могут являться конструкции, описывающие исходный запрос как запрос определённого вида, структура которых включает ключевой узел, являющийся знаком множества sc-элементов, предназначенных для первоочередной обработки соответствующей операцией (такими элементами множества всегда являются дуги вида (**Goals** → **Pattern**)), а также позитивную константную дугу из ключевого

узла в дугу (*Goals* → *Pattern*). При завершении операции классификации и управления запросами соответствующая дуга из ключевого узла *active_*, как правило, удаляется.

Приведём пример выполнения данной операции. При наличии в базе знаний конструкции:

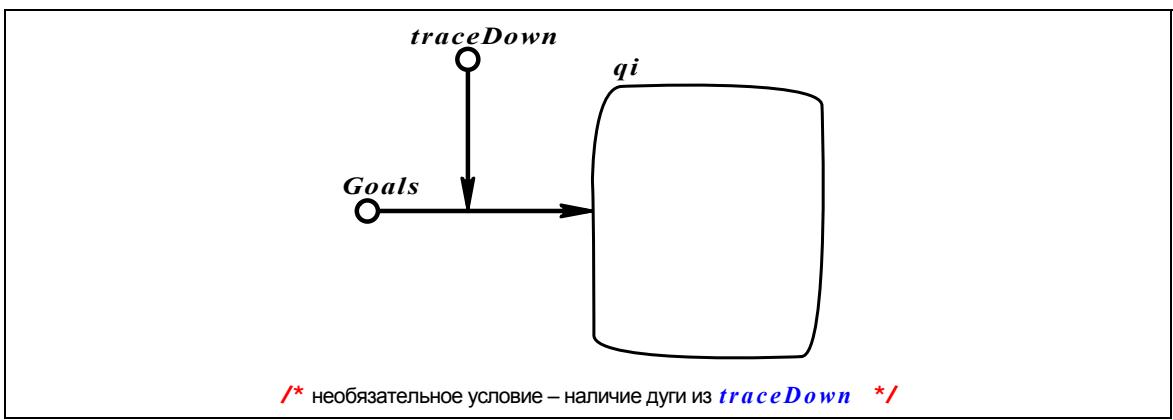


после выполнения операции мы получим, например, следующую конструкцию:



Операция трассировки запроса сверху вниз (SCL query trace-down operation) в зависимости от структуры запроса и типа формулы, к которой относится запрос, формирует запросы для всех формул, входящих в неё.

Условием выполнения операции трассировки запроса сверху-вниз является наличие конструкции вида:

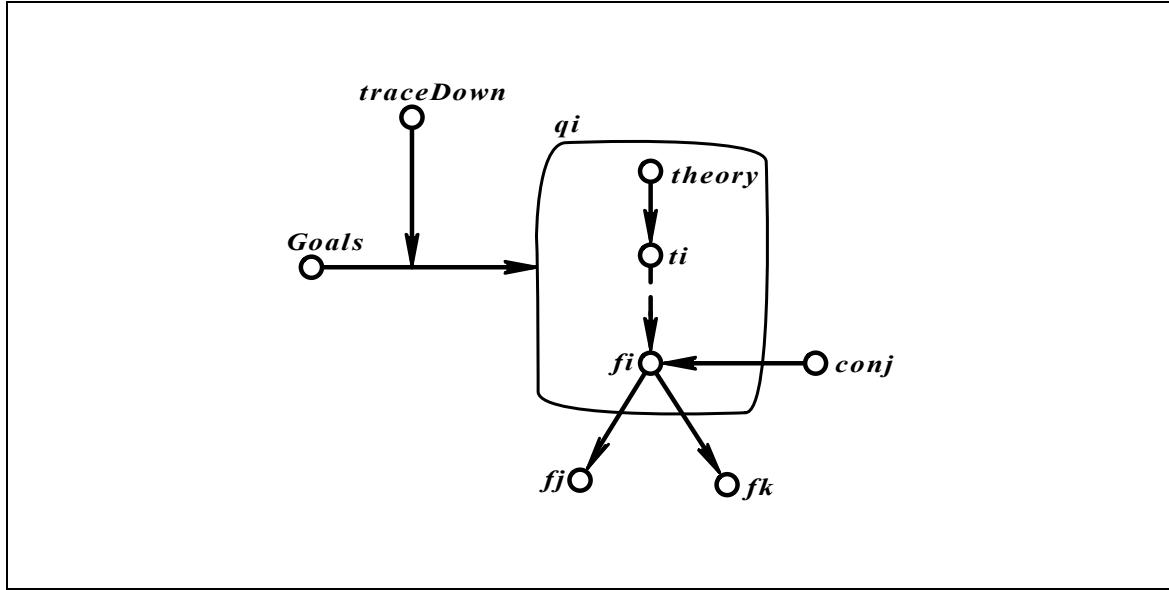


Операция, в зависимости от типа запроса *qi* – запрос ли это на выяснение истинности, либо ложности исходного высказывания, либо запрос на поиск интерпретации исходной формулы – и типа исходной

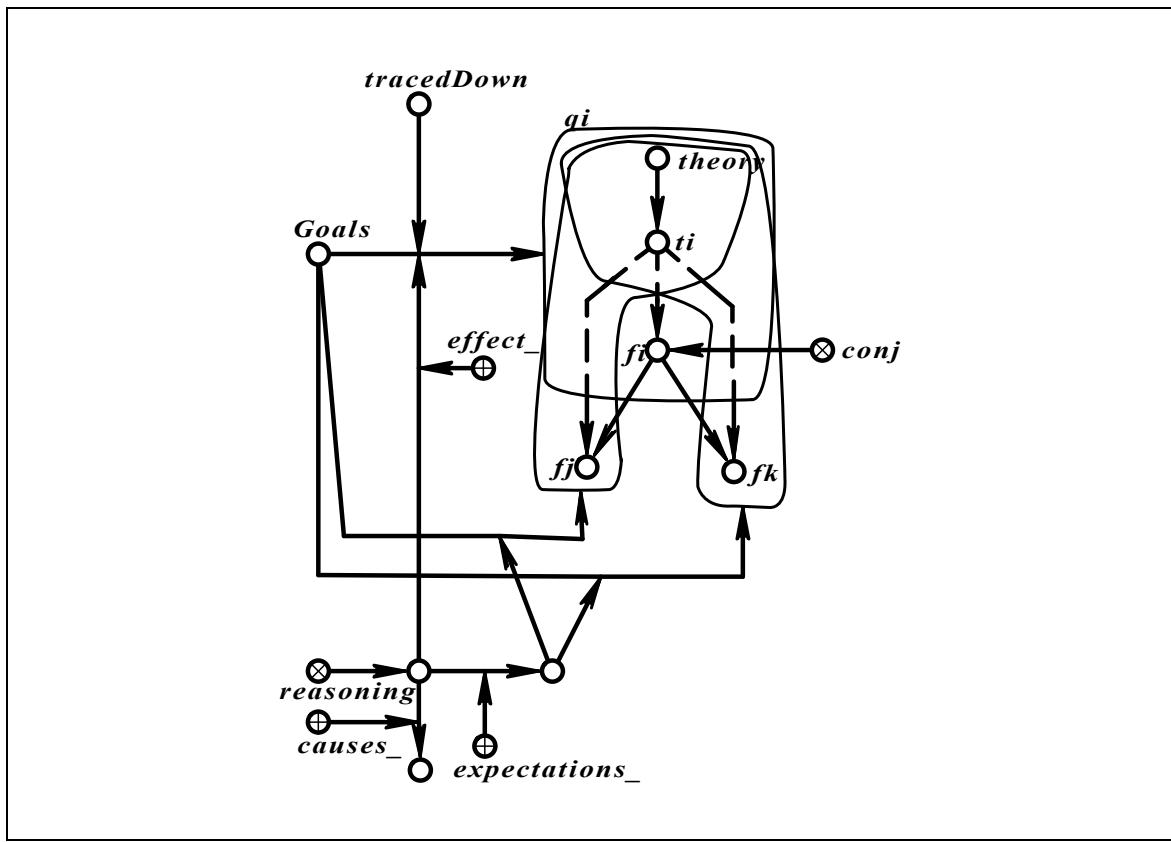
формулы, к которой относится запрос, формирует все возможные подзапросы к формулам, входящим в исходную, группируя по И-критерию сформированные подзапросы связками отношения ***reasoning***.

Результатом выполнения операции трассировки запроса сверху вниз в случае успеха является некоторое количество подзапросов сгруппированных связками отношения ***reasoning***. При завершении операции трассировки запроса сверху вниз соответствующая дуга из ключевого узла ***traceDown*** удаляется.

Приведём пример выполнения данной операции. При наличии в базе знаний конструкции:

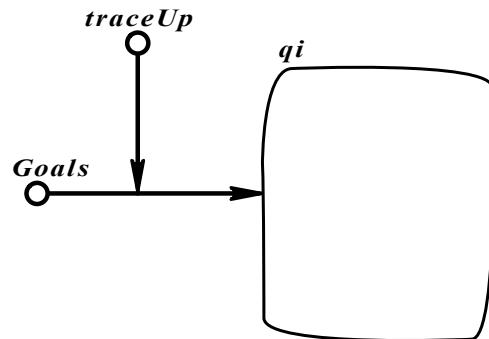


после выполнения операции мы получим следующую конструкцию:



Операция трассировки запроса снизу вверх (SCL query trace-up operation) в зависимости от структуры запроса и типа формулы, к которой относится запрос, формирует запросы для всех формул, включающих её.

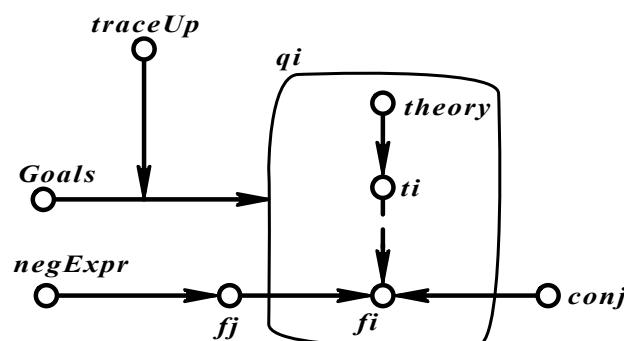
Условием выполнения операции трассировки запроса снизу вверх является наличие конструкции вида:



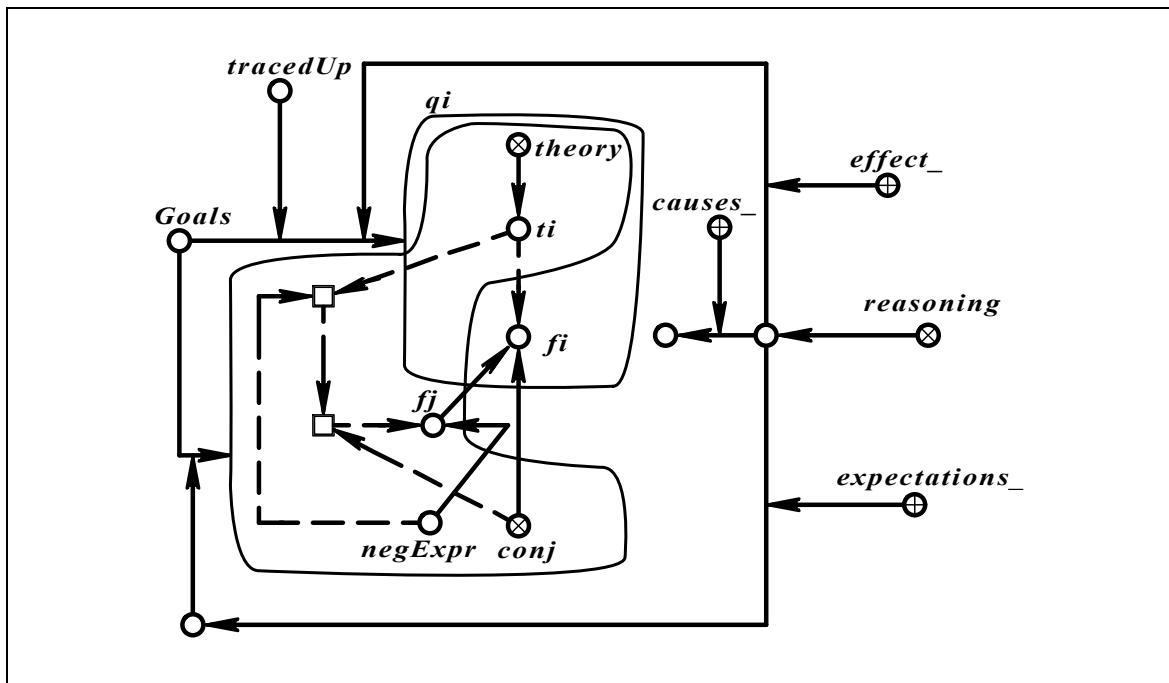
/* необязательное условие – наличие дуги из *traceUp* */

Операция в зависимости от типа запроса – запрос ли это на выяснение истинности, либо ложности исходного высказывания, либо запрос на поиск интерпретации исходной формулы – и типа исходной формулы, к которой относится запрос, формирует все возможные подзапросы к формулам, включающим исходную, и их остальным подформулам, группируя по И-критерию сформированные подзапросы связками отношения *reasoning*.

Приведём пример выполнения данной операции. При наличии в базе знаний конструкции:

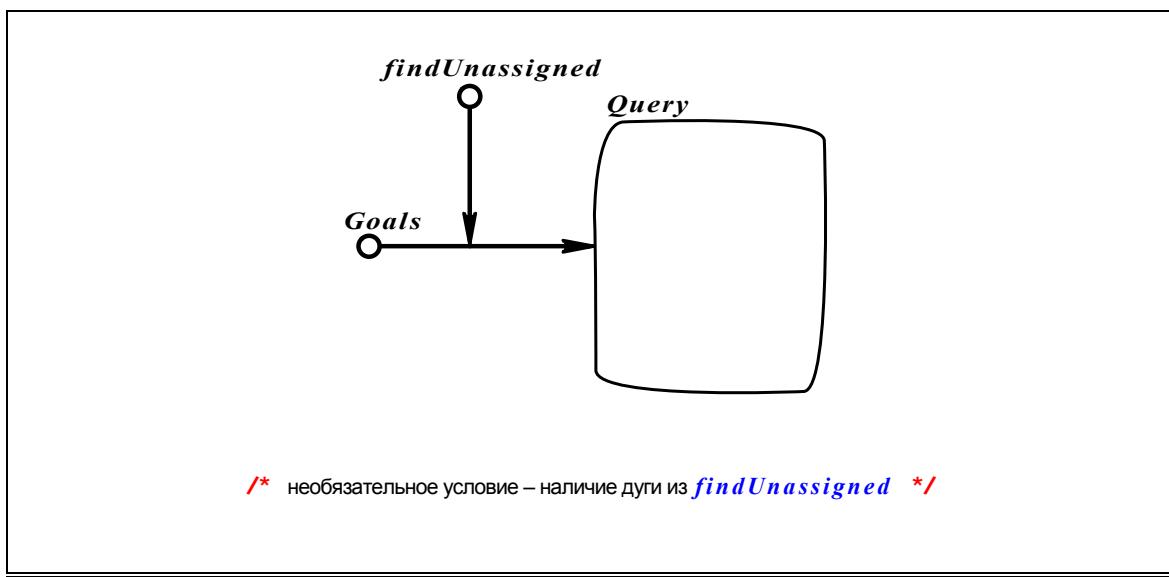


после выполнения операции мы получим следующую конструкцию:



Операция выявления множества свободных или связываемых переменных (SCL free variable locating operation) .

Условием выполнения операции формирования множества свободных или связываемых переменных является наличие конструкции вида:



Операция ищет неатомарную формулу связанную с запросом. Ищет множество свободных переменных: если такое множество есть, то операция завершается безуспешно, иначе она осуществляет поиск свободных переменных (которые имеют вхождения в каждую подформулу, включаемую текущей формулой) и включает эти переменные в соответствующее множество, которое привязывается к текущему высказыванию. При необходимости операция применяется к нахождению свободных переменных в подформулах на нижних уровнях.

Приведём пример выполнения данной операции. При наличии в базе знаний конструкции:

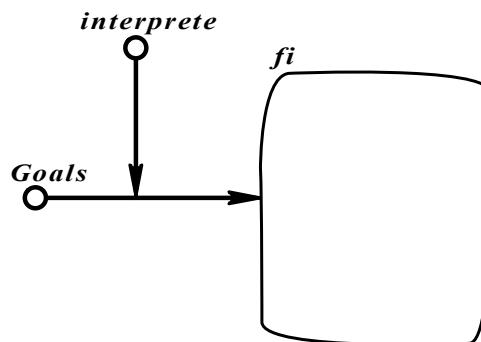
```
findUnassigned → (Goals → qi) ;
qi = [unassigned →>  $\exists$  (formula → fi), Set)] ;
allEqExpr → qi ;
qi → fk, fj ;
fi = [рефлексивное множество → _s] ;
fk = [_s →> _s] ;
existAtExpr → fj, fk ;
```

после выполнения операции мы получим следующую конструкцию:

```
unassigned → ( (formula → fi), Set) ;
Set = [_s];
```

Операция формирования интерпретации формулы (SCL interpretation forming operation).

Условием выполнения операции формирования интерпретации формулы является наличие конструкции вида:

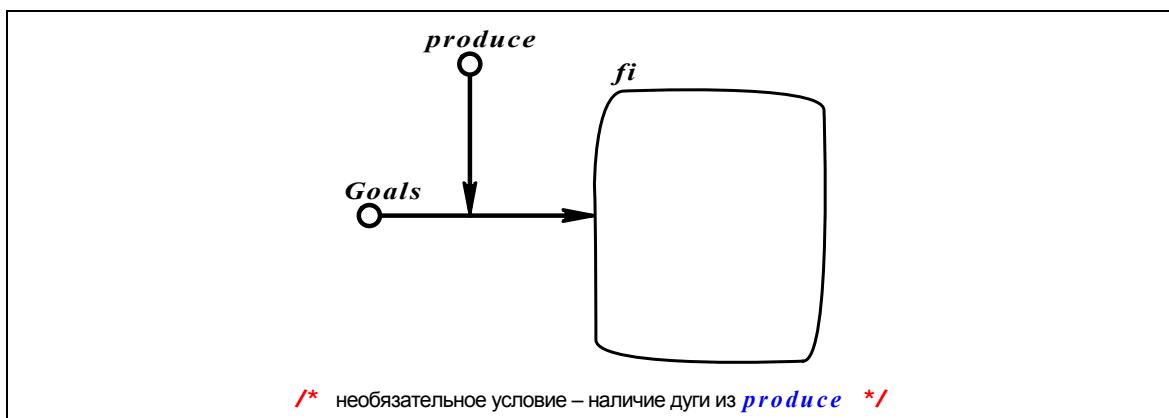


/* необязательное условие – наличие дуги из *interpret* */

Формируется новая, отличная от уже сформированных, интерпретация атомарной формулы *fi*, используя все подтверждённые связки (связки, в которых компонент с атрибутом *expectations* представляет собой пустое множество) отношения *reasoning*, на базе существующих интерпретаций И-подцелей исходной цели, выражаемой атомарной формулой *fi*, подцелей связанных между собой используемыми связками отношения *reasoning*.

Операция порождения интерпретации атомарной формулы (SCL interpretation producing operation) .

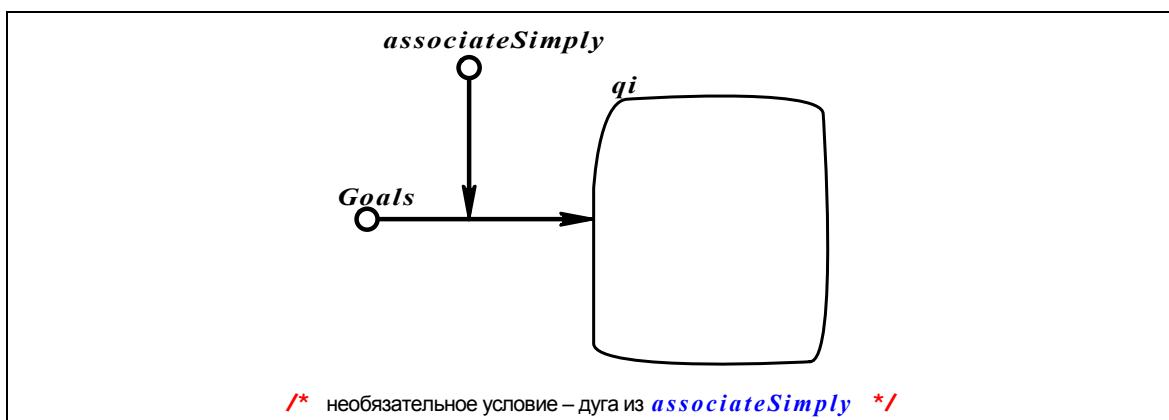
Условием выполнения операции порождения интерпретации атомарной формулы является наличие конструкции вида:



Если атомарная формула *fi* является высказыванием, которое обладает истинностным значением, то порождается релевантная ей формула, которая включается в также формируемую связку отношения интерпретации.

Операция поиска семантически близких атомарным формулам формул без учёта значений переменных (SCL simple atom associating operation) .

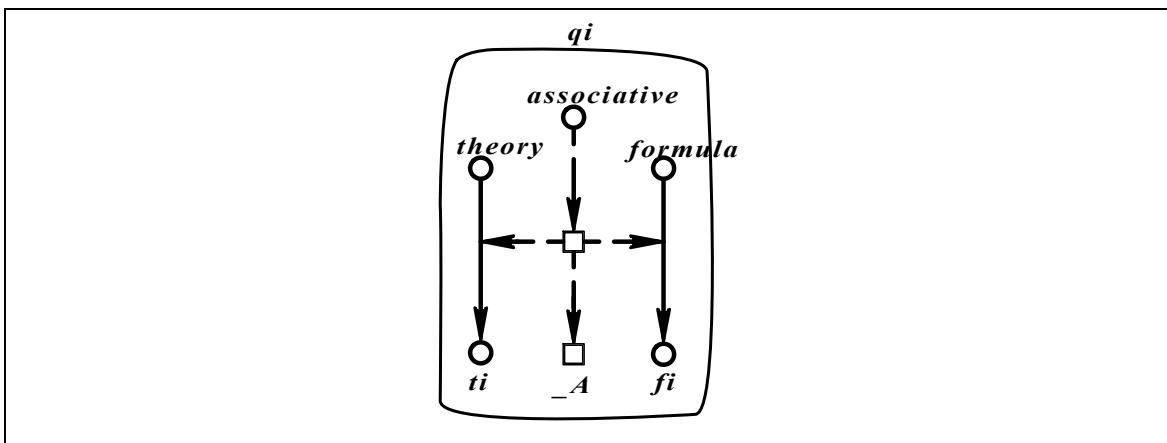
Условием выполнения операции поиска семантически близких атомарным формулам формул без учёта значений переменных является наличие конструкции вида:



Операция осуществляет поиск всех атомарных формул, включающих те же константы, что и исходная. Формируется соответствующая связка отношения *associative*.

Микропрограмма операции поиска семантически близких атомарным формулам формул без учёта значений переменных имеет следующий вид:

Шаг 1. Вначале операция проверяет структуру запроса: в запросе отыскиваются соответствующая атомарная формула (*fi*) и теория (*ti*), а в качестве допустимой разрешается структура запроса только следующего вида:



Шаг 2. В ходе проверки типа запроса находится дуга ($Goals \succ gq \succ qi$) и удаляется входящая в неё дуга из узла *associateSimply*. Если анализ структуры запроса прошёл успешно, то генерируется узел *s*, иначе – операция завершается с возвратом ошибки.

Шаг 3. Формируется множество всех констант атомарной формулы *fi*.

Шаг 4. Осуществляется формирование множества атомарных формул, включающих хотя бы одну константу из множества констант, сформированного на **третьем шаге**.

Шаг 5. Ищется формула *fj*, входящая во множество атомарных формул, сформированное на **четвёртом шаге**: если такой формулы нет, то осуществляется переход на **шаг 15**.

Шаг 6. Найденная формула *fj* исключается из множества формул, которое было сформировано на **четвёртом шаге** и формируется следующая конструкция:

sj , sk → fj ;

Шаг 7. Формируется множество *si* – множество неатомарных формул, включающих хотя бы одну формулу из сформированного множества *sj*, и не включённых во множество *sk*.

Шаг 8. Если во множестве *si* находится формула *ti*, то осуществляется переход на **шаг 13**.

Шаг 9. Если множество *si* – пустое, то осуществляется переход на **шаг 5**.

Шаг 10. Элементы множества *sj* добавляются ко множеству *sk*. После чего множество *sj* очищается.

Шаг 11. Элементы множества *si* добавляются ко множеству *sj*. После чего множество *si* очищается.

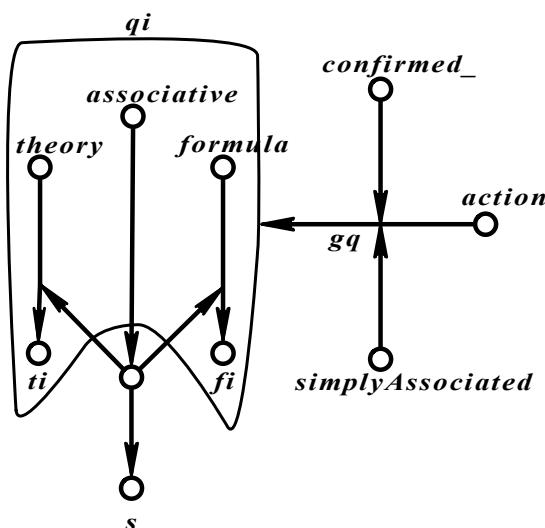
Шаг 12. Осуществляется переход на **шаг 7**.

Шаг 13. Формула *fj* добавляется во множество *s* – формируется конструкция вида:

s → fj ;

Шаг 14. Осуществляется переход на **шаг 5**.

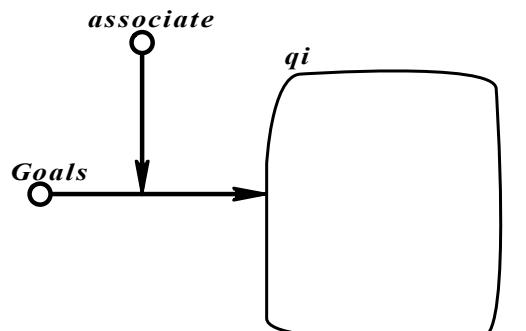
Шаг 15. Операция успешно завершается с формированием конструкции вида:



Конец микропрограммы.

Операция поиска семантически близких неатомарным формулам формул (SCL molecular associating operation).

Условием выполнения операции поиска семантически близких неатомарным формулам формул является наличие конструкции вида:

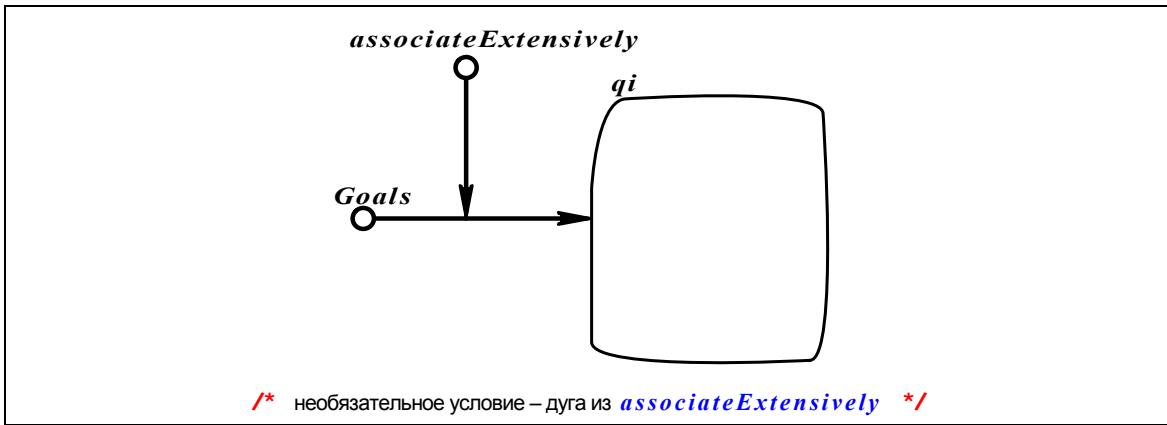


/ необязательное условие – дуга из associate */*

Операция осуществляет поиск всех логически тождественных формул, включающих все те же (идентичные) формулы, что и исходная формула. Все найденные формулы включаются в формируемую связку отношения ***associative***.

Операция поиска семантически близких атомарным формулам формул с учётом значений переменных (SCL extensive atom associating operation) .

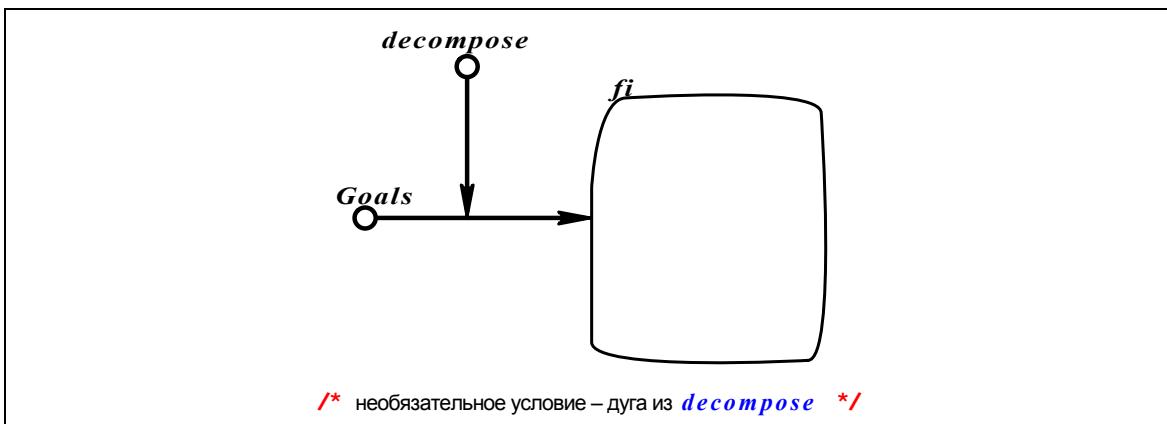
Условием выполнения операции поиска семантически близких атомарным формулам формул с учётом значений переменных является наличие конструкции вида:



Операция осуществляет поиск всех атомарных формул, включающих те же константы и те же значения переменных, что и исходная формула *fi*, включённая в запрос *qi*. Формируется соответствующая связка отношения *associative*.

Операция декомпозиции запроса на вывод атомарной формулы, на запросы к семантически близким атомарным формулам (SCL atom query distributing operation) .

Условием выполнения операции декомпозиции запроса на вывод атомарной формулы, на запросы к семантически близким атомарным формулам является наличие конструкции вида:

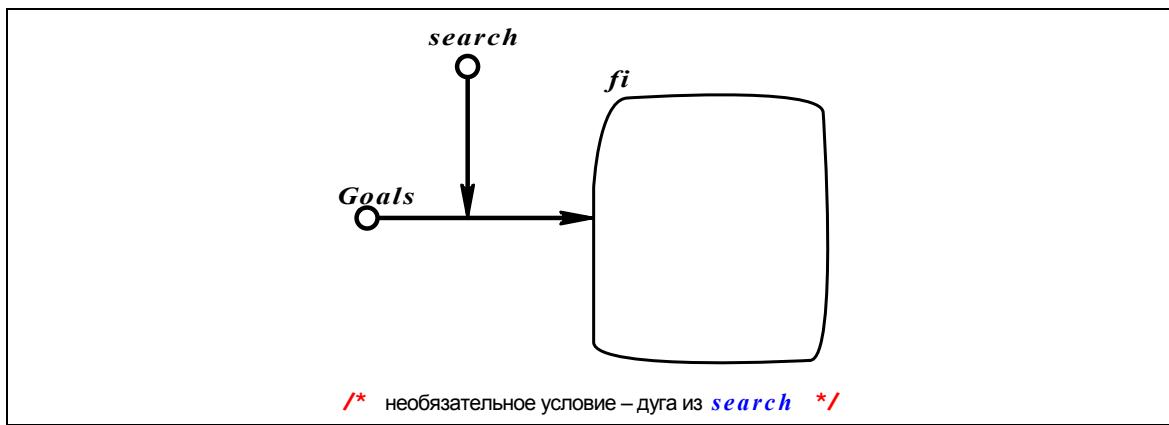


Операция ставит соответствующий запрос к подходящим семантически близким формулам в зависимости от типа запроса к исходной атомарной формуле *fi*. Между исходным и производными запросами формируются связки отношения *reasoning*. Результатом будет генерация новых запросов и конструкций вида:

<i>reasoning</i> → (<i>effect_</i> : <i>gqi</i> , <i>expectations_</i> : (<i>gx1</i>) , <i>causes_</i> : <i>C1</i>) ;
<i>Goals</i> → <i>gx1</i> → <i>qi</i> ;
<i>active_</i> → <i>gx1</i> ;

Операция эпизодического информационного поиска (SCL atom episodic confirm_by-facts operation) .

Условием выполнения операции эпизодического информационного поиска является наличие конструкции вида:



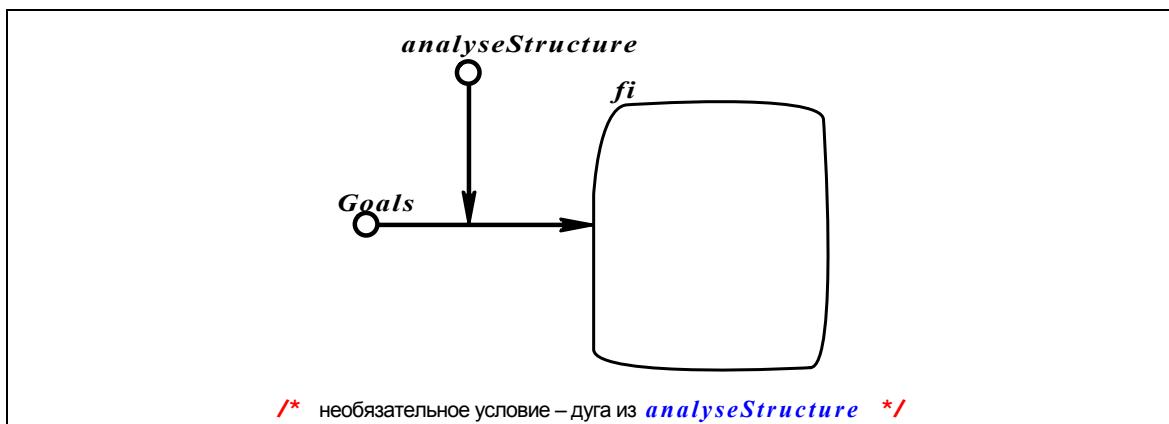
Операция осуществляет информационный поиск конструкции, соответствующей включённой в запрос **gq** формуле **fi** в указанной области поиска. Поиск прерывается в случае достижения свободной переменной, которая не инцидентна ни одной связанной переменной. При поиске учитываются уже присвоенные значения переменных формулы **fi** и найденные интерпретации (отобранные образы) .

Новый образ добавляется к отобранным и формируется запрос на формирование новой интерпретации. Результатом успешного завершения операции является конструкция вида:

factual → **fi** ;

Операция выявления идентичных, входящих и конфликтующих атомарных формул (SCL atom correlating operation) .

Условием выполнения операции выявления идентичных, входящих и конфликтующих атомарных формул является наличие конструкции вида:



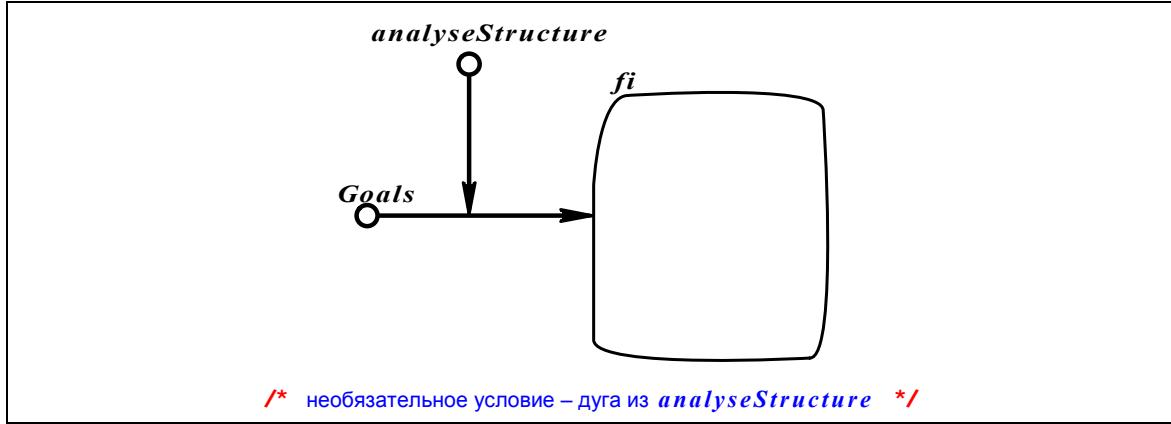
Операция анализирует структуру атомарных формул входящих в связку **qi** отношения **associative** и формирует связи отношения бинарного структурного включения **inclusion** между атомарными формулами. Если две формулы взаимно включают друг друга по структуре, и кванторы, которые навешены на переменные формул совпадают, то между такими формулами формируется связка отношения синонимии.

Результатом применения операции могут быть конструкции вида:

```
inclusion → ((formula → Formula1) , in_ : (formula → Formula2)) ;
Formula1=Formula2 ;
```

Операция выявления идентичных, входящих и конфликтующих неатомарных формул (SCL molecular correlating operation). Аналогична предыдущей.

Условием выполнения операции выявления идентичных, входящих и конфликтующих неатомарных формул является наличие конструкции вида:



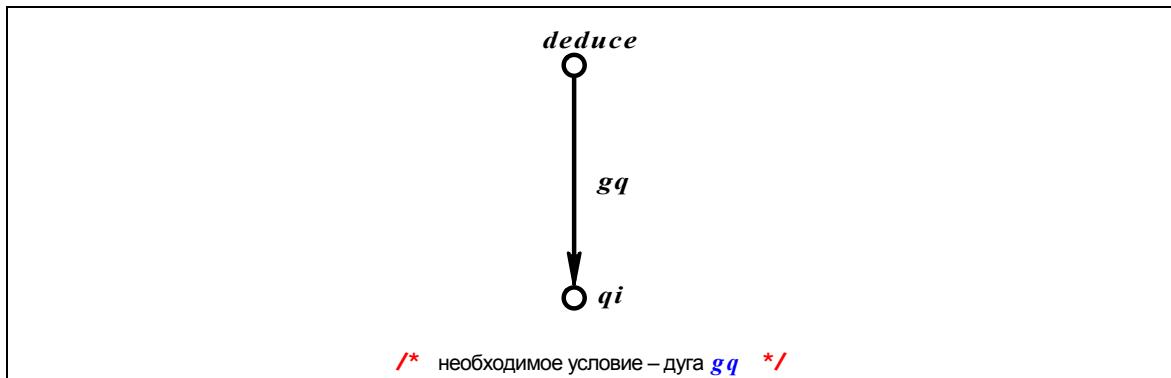
Операция анализирует структуру неатомарных формул входящих в связку *qi* отношения *associative* и формирует связи отношения бинарного структурного включения *inclusion* между формулами. Если две формулы взаимно включают друг друга по структуре, и кванторы, которые навешены на переменные формул совпадают, то между такими формулами формируется связка отношения синонимии.

Результатом применения операции могут быть конструкции вида:

```
inclusion → ((formula → Formula1) , in_ : (formula → Formula2)) ;
Formula1=Formula2 ;
```

Операция выполнения логических рассуждений (SCL reasoning operation).

Условием выполнения операции выполнения логических рассуждений является наличие конструкции вида:



Операция находит элемент *vx* включённой в связку *qi* отношения *reasoning*, помеченный атрибутом *expectations_*, проверяет являются ли все элементы найденного множества *vx* запросами, на которые получен ответ и если это так, то операция находит *ga* – компонент связки *qi*, помеченный атрибутом *effect_*. Если *ga* включён во множество *deny_*, то он также включается во множество *denied_*. Если *ga* включён во множество *confirm_*, то он также включается во множество *confirmed_*. Если связка *qi* включена во множество временных конструкций, то она удаляется.

Микропрограмма операции выполнения логических рассуждений имеет следующий вид:

Шаг 1. Вначале операция проверяет структуру запроса: в качестве допустимой разрешается структура запроса только следующего вида, когда узел запроса является знаком связки отношения *reasoning*:

<i>reasoning</i> → <i>qi</i> ;

Шаг 2. Если проверка структуры запроса осуществлена безуспешно, то операция завершается с возвратом ошибки, иначе – удаляется дуга *gq*.

Шаг 3. Отыскивается константная позитивная дуга *ge*, выходящая из узла *qi* и помеченная атрибутом *effect_*. Если поиск осуществлён безуспешно, то операция завершается с возвратом ошибки.

Шаг 4. Ищется константная позитивная дуга *ga*, выходящая из узла *Goals*, входящая в искомый константный узел *va*, в которую входит дуга *ge*. Если поиск осуществлён безуспешно, то операция завершается с возвратом ошибки.

Шаг 5. Ищется константная позитивная дуга, выходящая из узла *qi*, входящая в искомый константный узел *vx* и помеченная атрибутом *expectations_*. Если поиск осуществлён безуспешно, то осуществляется переход на **шаг 7**.

Шаг 6. Во множестве *vx* отыскиваются элементы, которые не входят ни во множество *denied_*, ни во множество *confirmed_*. Если найден хотя бы один такой элемент, то операция завершается успешно, иначе – удаляется узел *vx*.

Шаг 7. Если дуга *ga*, включена хотя бы в одно из множеств *denied_* и *confirmed_*, то операция завершается успешно.

Шаг 8. Если дуга включена во множество *confirm_*, то осуществляется генерация конструкции, удовлетворяющей образцу задаваемому узлом *va*. Генерируемая конструкция включается в формулу, являющуюся соответствующим фактографическим высказыванием (либо квазиатомарной логической формулой), формируется полное соответствие интерпретации между образцом *va* и сгенерированной формулой. В дугу *va* генерируется дуга из узла *confirmed_*.

Шаг 9. Если дуга включена во множество *deny_*, то осуществляется генерация квазиатомарной формулы *fj*, удовлетворяющей образцу задаваемому узлом *va*. Генерируемая конструкция включается в формулу *fj*, формируется полное соответствие интерпретации между образцом *va* и сгенерированной формулой *fj*. В дугу *va* генерируется дуга из узла *denied_*.

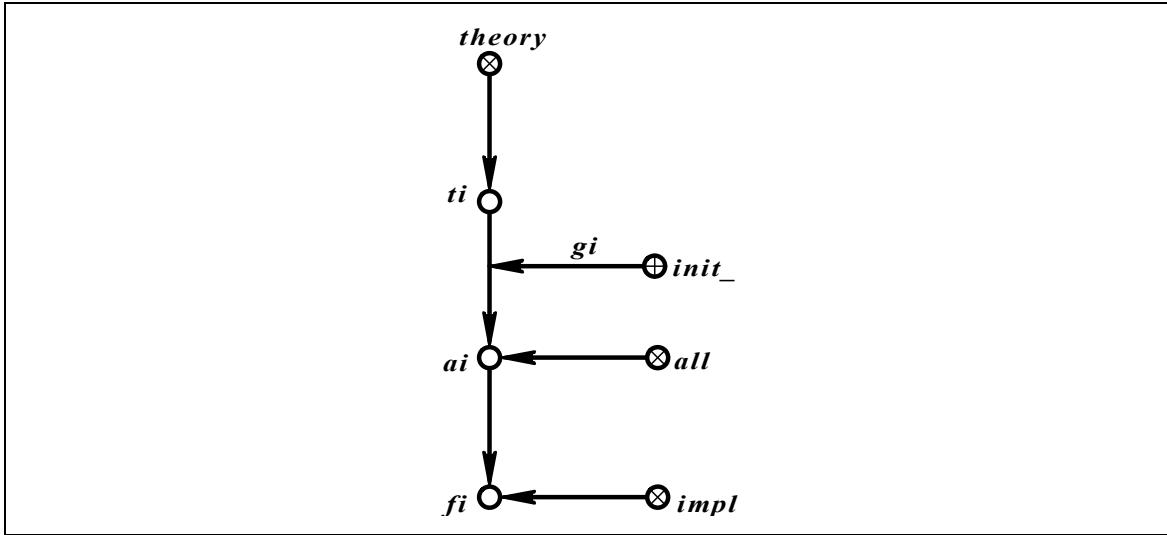
Шаг 10. При наличии константной позитивной дуги из узла *temporary* в узел *qi*. Генерируется константная позитивная дуга из узла *release* в узел *qi*.

Шаг 11. Операция завершается успешно.

Конец микропрограммы.

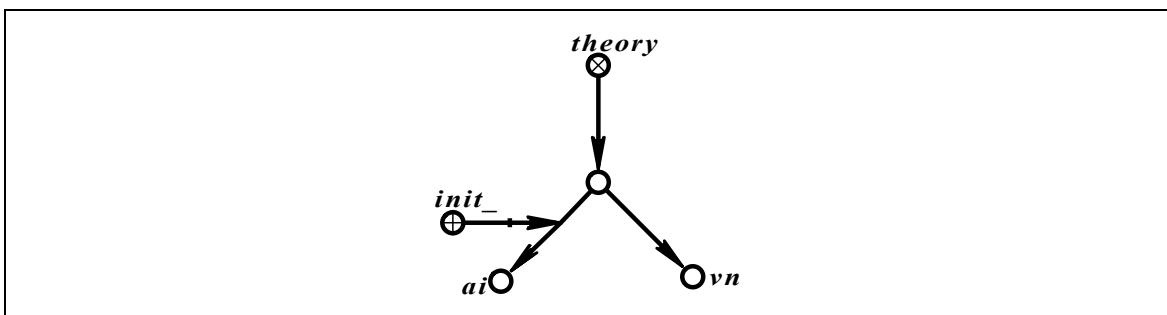
Операция реализации продукции (SCL producing operation).

Условием выполнения операции реализации продукции является наличие sc-конструкции вида:

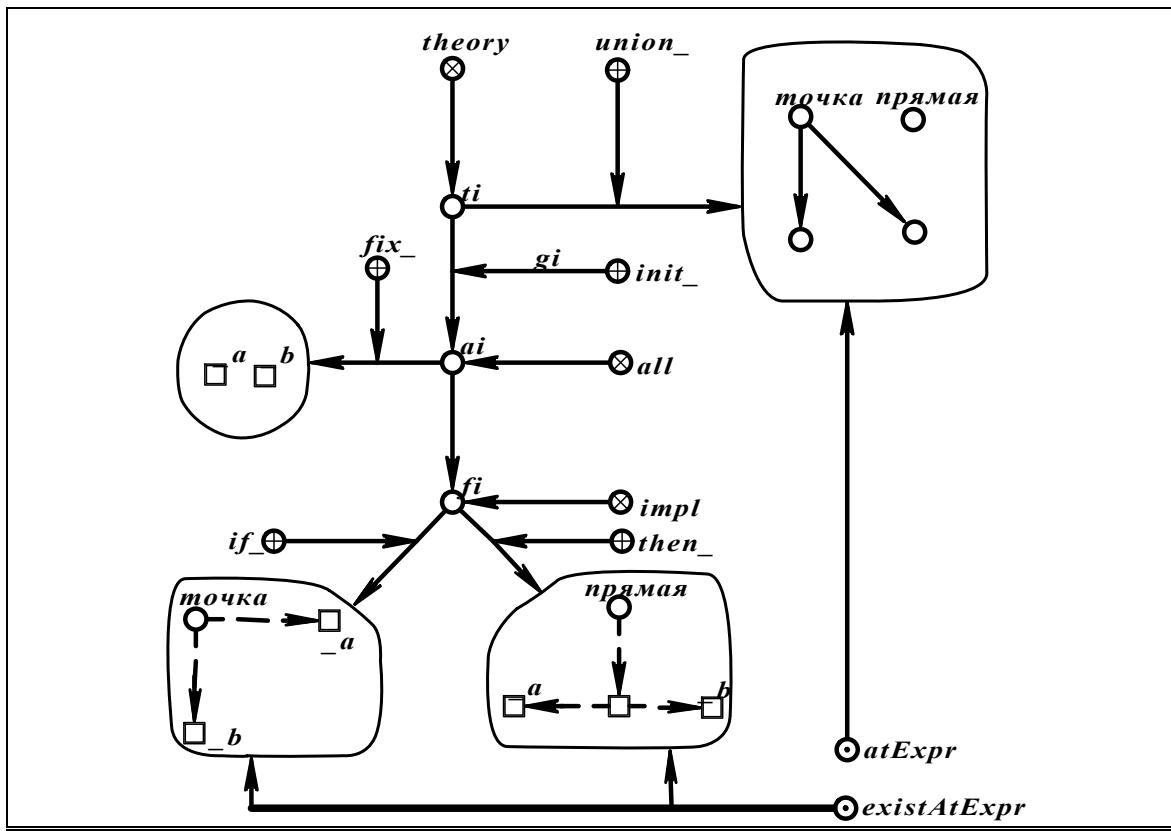


Операция находит левую часть импликативной формулы fi и ищет формулу ri релевантную этой части. Если левая часть является атомарной формулой, то поиск новой формулы заключается в поиске структуры, которая является фрагментом обобщённого атомарного (фактографического) высказывания теории ti , структуры, изоморфной этой атомарной формуле с учётом привязки констант, и формировании узла, из которого проводятся константные позитивные дуги во все найденные элементы структуры, являющегося знаком искомой формулы. Если же левая часть является сложной формулой, то осуществляется поиск релевантной ей формулы, которая включена во множество ti . Между левой частью импликации и найденной релевантной ей формулой ri формируется конструкция отношения релевантности, в рамках которой каждая переменная формулы, являющейся левой частью импликации получает своё значение – константу, включённую в какую-либо атомарную формулу формулы ri . Затем операция находит правую часть формулы fi , которая является атомарной формулой и формирует релевантную ей формулу vn , являющуюся фактографическим высказыванием, структура которого изоморфна правой части импликации с учётом привязки констант. Затем все значения тех переменных из найденных левой и правой частей импликации, которые связаны квантором общности в рамках кванторной формулы ai , склеиваются. Дуга gi удаляется. Элементы сформированной формулы vn добавляются в обобщённое атомарное (фактографическое) высказывание теории ti , а сама формула включается в теорию ti .

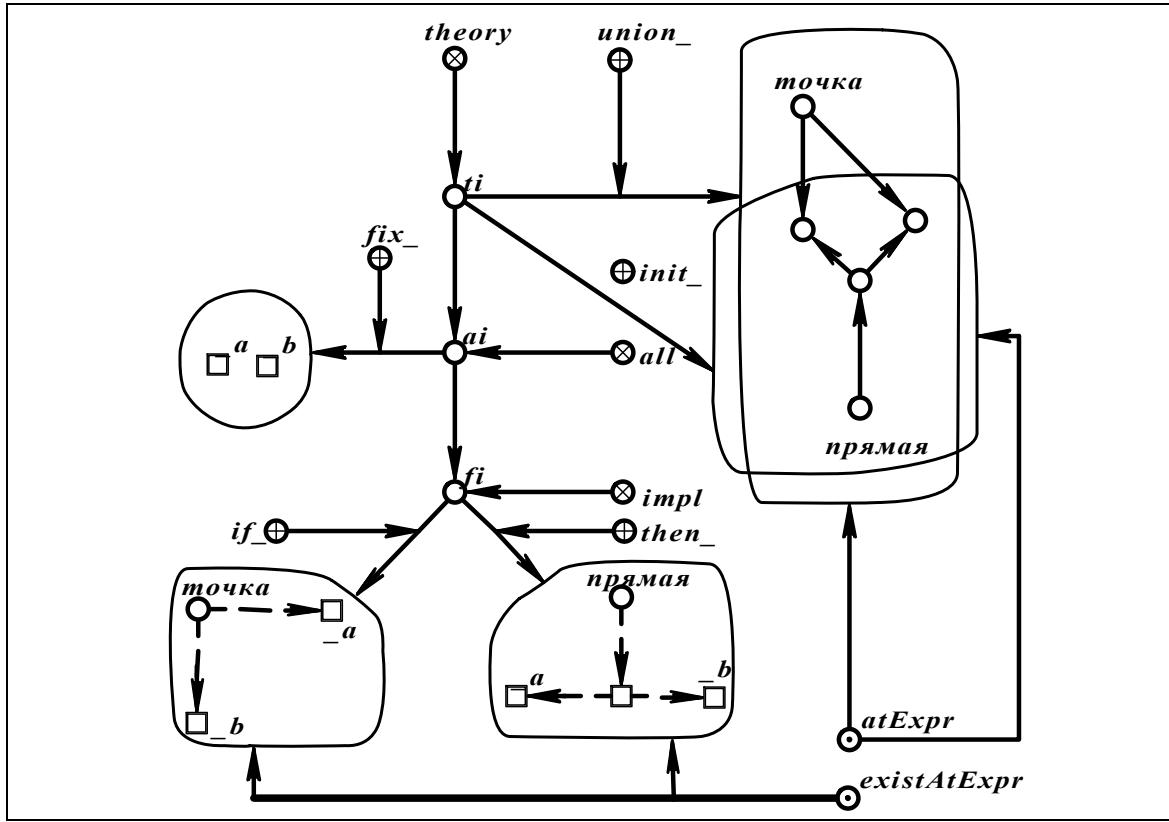
Результатом выполнения операции реализации продукции является следовательно конструкция вида:



Приведём пример выполнения данной операции. При наличии в базе знаний конструкции:



после выполнения операции реализации продукции мы получим следующую конструкцию:



Микропрограмма операции реализации продукции имеет следующий вид:

Шаг 1. Вначале операция осуществляет поиск неизвестных элементов конструкции, являющейся условием выполнения этой операции. Такими неизвестными элементами являются все элементы вышеприведённой конструкции-условия за исключением ключевых узлов: “*theory*”, “*init_*”, “*all*” и “*impl*”. В ходе выполнения этого шага операция находит, в частности, такие соответствующие элементы вышеприведённой конструкции-условия как: константную позитивную дугу *gi*, выходящую из ключевого узла “*init_*”, константный узел *ti*, являющийся знаком формальной теории, константный узел *ai*, являющийся знаком кванторной формулы, являющейся в рамках теории *ti* истинным высказыванием о всеобщности, а также константный узел *fi*, являющийся знаком импликативной формулы.

Шаг 2. Если хотя бы один из элементов *gi*, *ti*, *ai* и *fi* или хотя бы одна соответствующая константная позитивная дуга из вышеприведённой конструкции не найдена на предыдущем шаге, то операция завершает свою работу, иначе – удаляется дуга *gi*.

Шаг 3. Осуществляется поиск константного узла *vc*, в который входит константная позитивная дуга, выходящая из узла *fi*, которая является элементом ключевого множества “*if_*”. Из этого следует, что искомый узел *vc* является посылкой импликативной формулы *fi*. Таким образом формула *fi* и её посылка связаны между собой конструкцией вида:

fi → *if_* : *vc* ;

Шаг 4. Осуществляется поиск константного узла *ve*, в который входит константная позитивная дуга, выходящая из узла *fi*, которая является элементом ключевого множества “*then_*”. Из этого

следует, что искомый узел ***vc*** является следствием импликативной формулы ***fi***. Таким образом формула ***fi*** и её следствие связаны между собой конструкцией вида:

***fi* → *then_* : *ve* ;**

Шаг 5. Осуществляется поиск константного узла ***vf***, в который входит константная позитивная дуга, выходящая из узла ***ai***, которая является элементом ключевого множества “***fix_***”. Из этого следует, что искомый узел ***vf*** является множеством переменных связываемых квантором общности в рамках кванторного высказывания о всеобщности ***ai***. Таким образом искомое множество связываемых переменных и кванторная формула ***ai*** связаны между собой конструкцией вида:

***ai* → *fix_* : *vf* ;**

Шаг 6. Проверяется наличие константной позитивной дуги, выходящей из узла, являющегося одним из двух таких ключевых узлов как: “***atExpr***” и “***existAtExpr***”, и входящей в узел ***vc***: если такой дуги нет, то осуществляется переход на **шаг 30**, иначе – наличие такой дуги означает что, посылка ***vc*** является атомарной формулой, либо кванторной формулой о существовании интерпретации атомарной формулы.

Шаг 7. Осуществляется поиск связи ***ci*** отношения “**релевантность sc-конструкций**”, которая включает узел ***vs***, являющийся знаком связи, включающего формулу ***vc*** и узел “***i_***” под атрибутом “***атрибут_***”. Таким образом искомая связка и известные формула ***vc*** и отношение “**релевантность sc-конструкций**” связаны конструкцией вида:



Шаг 8. Если такая связка не найдена, то осуществляется переход на **шаг 14**.

Шаг 9. Осуществляется поиск узла ***ri*** отличного от узла ***vc***, который включен в одно из таких множеств как: “***atExpr***” и “***existAtExpr***”, который также включен в искомую пару ***sd*** одновременно с узлом “***i_***” под атрибутом “***атрибут_***”, в которую проведена константная позитивная дуга, выходящая из связки ***ci*** и не включённая во множество “***отношение_***”. Если узел ***ri*** не найден, то осуществляется переход на **шаг 14**.

Шаг 10. Осуществляется поиск узла ***ii***, в который проведена константная позитивная дуга из узла ***ci***, в которую входит константная позитивная дуга из ключевого узла “***отношение_***”. Если узел ***ii*** найден, то осуществляется переход на **шаг 18**.

Шаг 11. Элементы множества ***ri*** и только они включаются во множество, обозначенное узлом ***uf***.

Шаг 12. Множество *ri* очищается, то есть удаляются все выходящие из него константные позитивные дуги.

Шаг 13. Осуществляется переход на шаг 16.

Шаг 14. Осуществляется поиск всех элементов, в которые входят константные позитивные дуги из узла *vc*, если ни один такой элемент не найден, то операция завершается с возвратом ошибки.

Шаг 15. Осуществляется поиск константного узла *uf*, в который входит константная позитивная дуга, выходящая из узла *ti*, которая является элементом ключевого множества “*union_*”. Из этого следует, что искомый узел *uf* является обобщённым атомарным (факторграфическим) высказыванием формальной теории *ti*. Таким образом теория *ti* и формула *uf* связаны между собой конструкцией вида:

ti → *union_* : *uf* ;

Шаг 16. Осуществляется поиск конструкции, являющейся фрагментом конструкции, включенной во множество *uf*, – поиск фрагмента изоморфного (с учётом привязки констант) конструкции, которую образуют все элементы, найденные на четырнадцатом шаге. Искомый фрагмент – искомая конструкция будет являться изоморфной с учётом привязки констант в том и только в том случае, если каждому переменному узлу или переменному элементу неопределённого типа, найденному на четырнадцатом шаге соответствует некоторый константный узел или константный элемент соответственно, каждому константному узлу, дуге или элементу неопределённого типа – он сам, каждой переменной дуге – константная дуга аналогичной степени чёткости, выходящая из узла искомой конструкции, соответствующего узлу, из которого выходит переменная дуга, и входящая в элемент искомой конструкции, соответствующий элементу, в который входит переменная дуга, когда других элементов в искомой конструкции нет, причём для каждого элемента, найденного на четырнадцатом шаге соответствует один единственный элемент искомой конструкции. Все элементы искомой конструкции и только они включаются во множество, обозначенное узлом *ri*. Каждый элемент искомой конструкции *es*, соответствующий какому-либо найденному на четырнадцатом шаге переменному элементу *ei*, связывается с этим переменным элементом ориентированной парой – парой вида:

(*1_* : *ei* , *2_* : *es*) ;

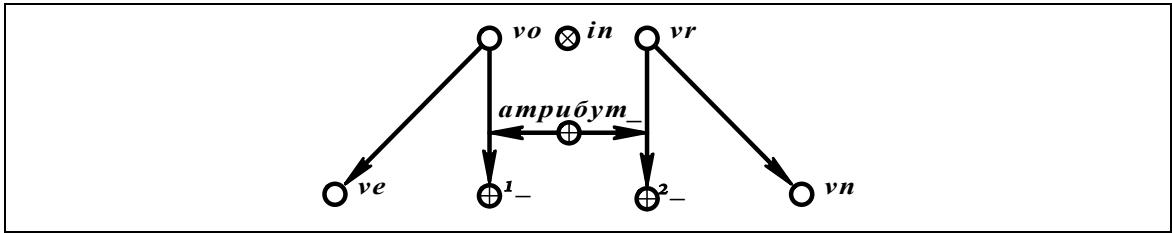
Шаг 17. Если поиск изоморфной конструкции прошёл безуспешно, то осуществляется переход на шаг 28, иначе – все сформированные на предыдущем шаге ориентированные пары и только они включаются во множество, обозначенное узлом *ii*.

Шаг 18. Если узел *ci* не был найден (на седьмом шаге), то генерируется связка отношения “*релевантность sc-конструкций*”, в которую включаются узлы *ri* и *vc* и только. Формула *ri* добавляется во множество *ti*.

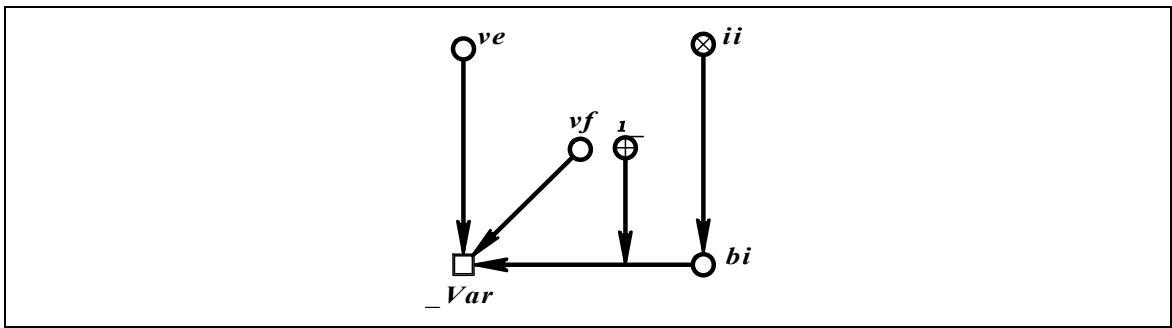
Шаг 19. В случае отсутствия, во множество *ci* добавляется узел *ii* под атрибутом “*отношение_*”.

Шаг 20. Проверяется наличие константной позитивной дуги, выходящей из узла, являющегося одним из двух таких ключевых узлов как: “*atExpr*”, “*existAtExpr*”, и входящей в узел *ve*: если такой дуги нет, то осуществляется переход на шаг 28, иначе – наличие такой дуги означает что, посылка *ve* является атомарной формулой, либо кванторной формулой о существовании интерпретации атомарной формулы.

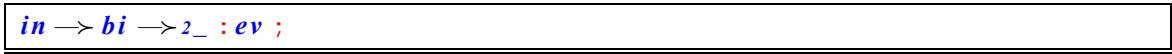
Шаг 21. Генерируются узлы **vn**, **vr**, **vo** и **in**. Генерируется конструкция вида:



Шаг 22. Во множество **in** включается каждая ориентированная пара **bi**, включённая во множество **ii** и имеющая в качестве первого элемента какую-либо переменную **_Var**, включённую во множества **vf** и **ve**, то есть пара, связанная конструкцией вида:



Шаг 23. Во множество **vn** включается каждый элемент **ev**, являющийся вторым элементом хотя бы одной ориентированной пары **bi**, включённой во множество **in**, то есть элемент, связанный конструкцией вида:



Шаг 24. Каждый константный элемент множества **ve** включается во множество **vn**.

Шаг 25. Для каждого переменного узла или переменного элемента неопределённого типа, включённого во множество **ve** и не включённого во множество **vf**, генерируется и включается во множество **vn** константный узел или константный элемент неопределённого типа соответственно. Для каждого генерируемого константного элемента генерируется включаемая во множество **in** ориентированная пара, включающая в качестве второго этот генерируемый элемент, а в качестве первого – переменный элемент, соответствующий генерируемому константному элементу.

Шаг 26. Для каждой переменной дуги, включённой во множество **ve** и не включённой во множество **vf**, генерируется и включается во множество **vn** константная дуга аналогичной степени чёткости, выходящая из узла, лежащего во множестве **vn**, соответствующего узлу, из которого выходит переменная дуга, и входящая в элемент, лежащий во множестве **vn**, соответствующий элементу, в который входит переменная дуга. Для каждой генерируемой константной дуги генерируется включаемая во множество **in** ориентированная пара, включающая в качестве второго эту генерируемую дугу, а в качестве первого – переменную дугу, соответствующую генерируемой константной дуге.

Шаг 27. Генерируется связка отношения “**релевантность sc-конструкций**”, в которую включаются узлы **vo** и **vr**, а также узел **in** под атрибутом “**отношение_**” и только. Формула **vn** добавляется во множества **ti** и “**atExpr**”. Все не добавленные элементы множества **vn** добавляются в обобщённое атомарное (фактографическое) высказывание **uf** формальной теории **ti**, связанное с теорией **ti** конструкцией вида:

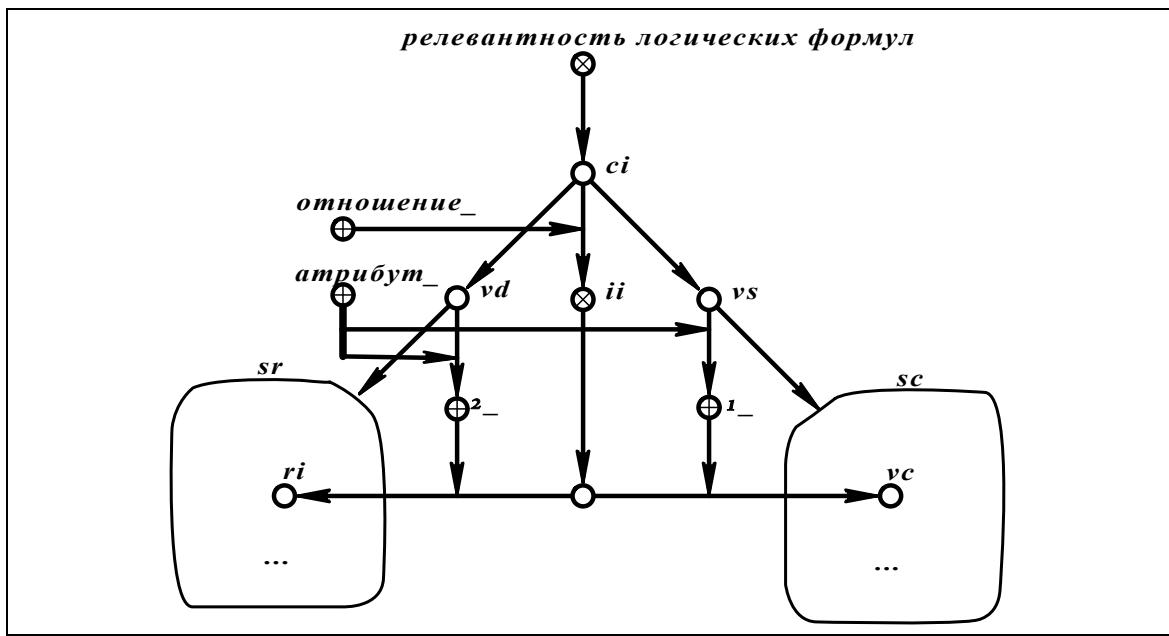
ti → *union_* : *uf* ;

Шаг 28. Осуществляется переход на **шаг 34**.

Шаг 29. Если узел *ri* был найден на **девятом шаге**, то операция завершается с возвратом ошибки, иначе – осуществляется переход на **шаг 28**.

Шаг 30. Проверяется наличие константной позитивной дуги, выходящей из узла, являющегося одним из двух таких ключевых узлов как: “*all*”, “*allEqExpr*”, “*allImpl*”, “*alt*”, “*conj*”, “*disj*”, “*exist*”, “*eqExpr*” и “*impl*”, и входящей в узел *ve*: если такой дуги нет, то осуществляется переход на **шаг 28**.

Шаг 31. Осуществляется поиск константного узла *ri*, в который проведена константная позитивная дуга из узла *ti*. Искомый узел *ri* кроме этого должен являться формулой релевантной формуле *vc*. В процессе поиска формируется связка *ci* отношения “**релевантность логических формул**”, в которую включены узел *vs*, являющийся знаком связки, в которую включен узел “*1_*” под атрибутом “*атрибут_*” и узел *sc*, включающий всю структуру формулы *vc*. В связку *ci* также должен быть включен узел *vd*, являющийся знаком связки, в которую включен узел “*2_*” под атрибутом “*атрибут_*” и узел *sr*, включающий всю структуру формулы *ri*. Кроме этого в связку *ci* также должен быть включен узел *ii* под атрибутом “*отношение_*”, являющийся знаком множества, ориентированных пар соответствия релевантности, которые выходят из элементов множества *sc* в соответствующие элементы множества *sr*. Таким образом искомые на данном шаге элементы в итоге будут связаны конструкцией вида:



Шаг 32. Если такой узел *ri* не найден, то осуществляется переход на **шаг 28**.

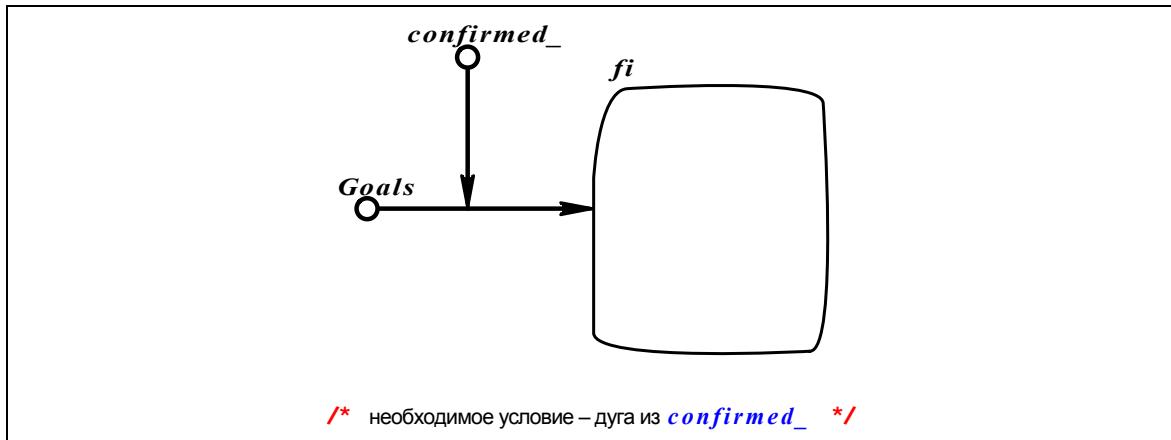
Шаг 33. Осуществляется переход на **шаг 20**.

Шаг 34. Работа операции успешно завершается.

Конец микропрограммы.

Операция обработки положительного ответа на запрос (SCL positive answer processing operation) .

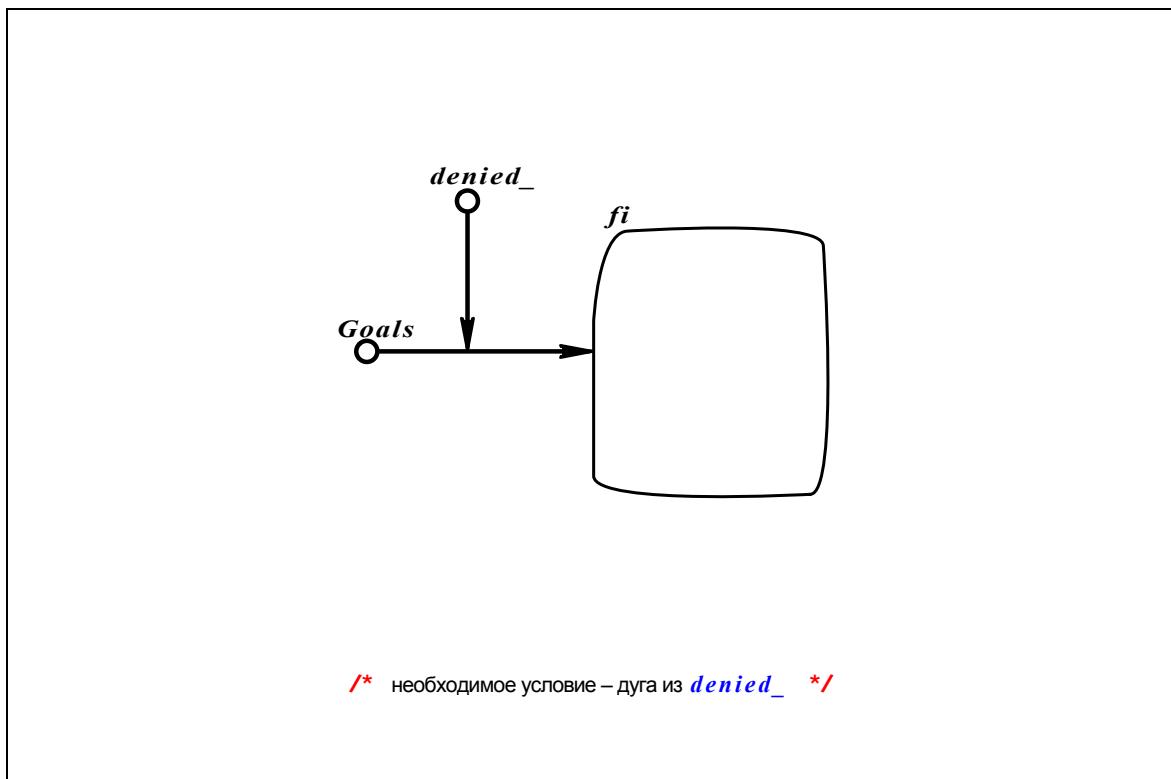
Условием выполнения операции обработки положительного ответа на запрос является наличие конструкции вида:



Операция анализирует: не является подтверждённый запрос какой-либо ожидаемой И-подцелью и если это так, то операция переносит подцель из разряда ожидаемых (из компонента соответствующей связки отношения *reasoning*, помеченного атрибутом *expectations_*) в достигнутые (в компонент этой связки, помеченный атрибутом *causes_*), создавая условия для применения операции выполнения логических рассуждений.

Операция обработки отрицательного ответа на запрос (SCL negative answer processing operation) .

Условием выполнения операции обработки отрицательного ответа на запрос является наличие конструкции вида:



Операция анализирует: не является опровергнутый запрос какой-либо ожидаемой И-подцелью и если это так, то операция переносит подцель из разряда ожидаемых (из компонента соответствующей связки отношения *reasoning*, помеченного атрибутом *expectations*_) в достигнутые (в компонент этой связки, помеченный атрибутом *causes*_), создавая условия для применения операции выполнения логических рассуждений.

Операция уничтожения промежуточных конструкций (SCL structure releasing operation) . Операция уничтожает все конструкции, которые включены во множество удаляемых конструкций.

Операция склейки идентичных формул (SCL formula optimizing operation) . Операция склеивает две синонимичные формулы.

Операция добавления факта (SCL fact adding operation) . Операция добавляет факт в SCL теорию, если он не противоречит другим фактам и высказываниям.

Операция добавления высказывания (SCL expression adding operation) . Операция добавляет высказывание в SCL теорию, если оно не противоречит фактам и другим высказываниям.

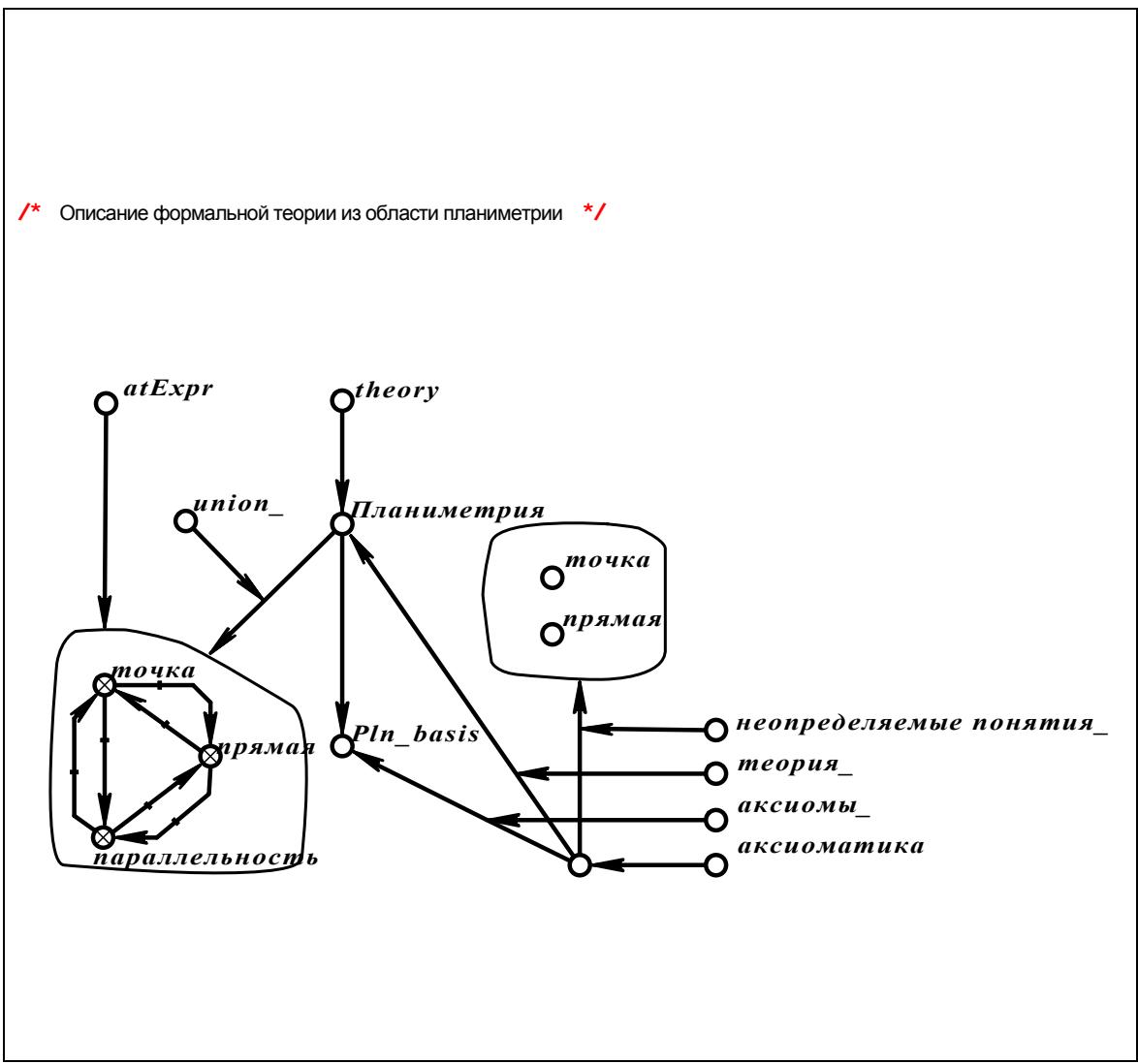
Более подробное рассмотрение операций абстрактной scl-машины приведено в работах [148; 150, 153] ([Голенков В.В..1995мо-ОпераЯSCLδОПЗ](#) ; [Голенков В.В..1995мо-РешенНЯSCLЗиОГ](#) ; [Голенков В.В..1996мо-БазовПТЯSCL](#)).

1.2. Решение задач в графодинамических ассоциативных машинах вывода

Ключевые понятия: задача, решение задачи, аксиома, определение, вывод.

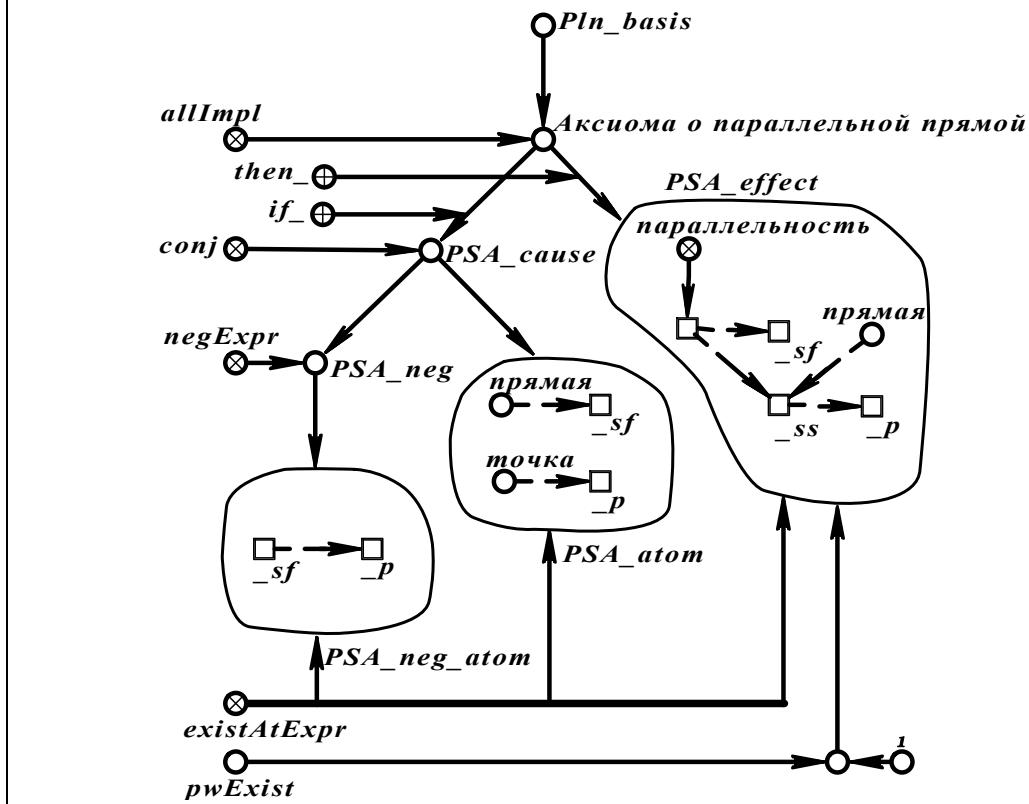
Рассмотрим решение задач (см. уточнение понятия задача в подразделе 6.3) на SCL. Пусть нам необходимо доказать, что существуют две параллельные прямые. Мы обладаем следующей исходной информацией.

Условие задачи:

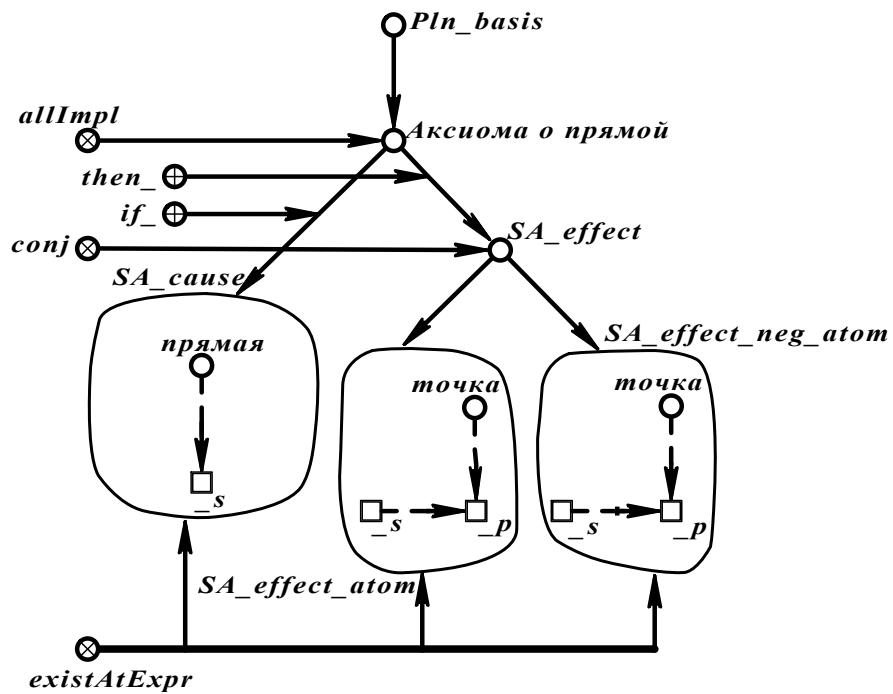


/* Формирование аксиоматики */

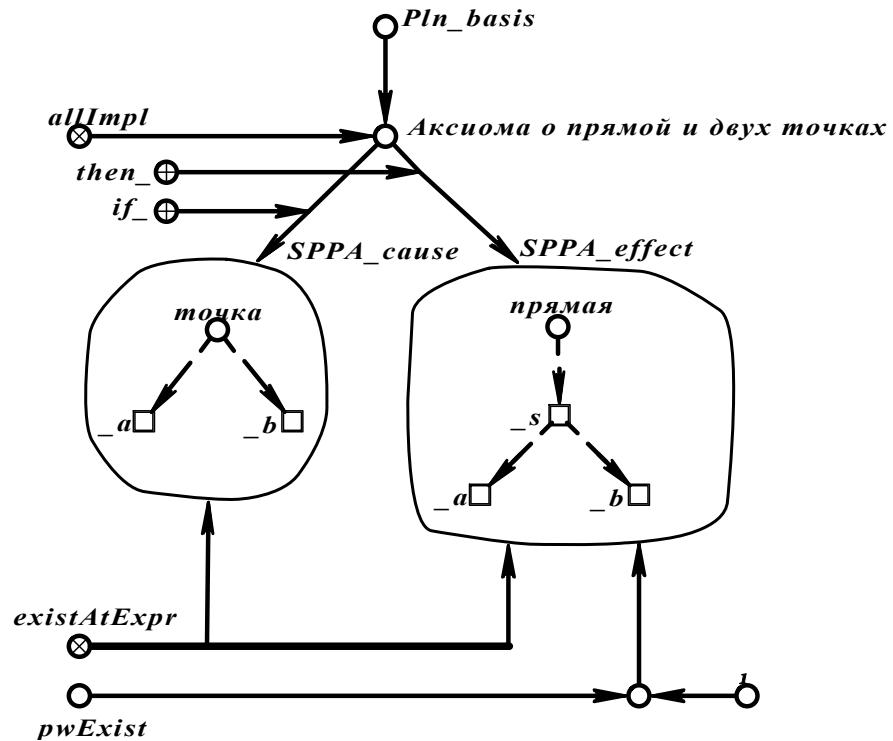
/* "через не лежащую на прямой точку можно провести параллельную этой прямой прямую и только одну" */



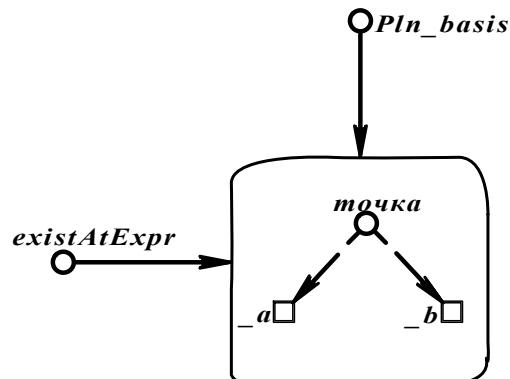
/* “для каждой прямой существуют точки принадлежащие ей и не принадлежащие ей” */



/* “через любые две точки можно провести прямую и только одну” */

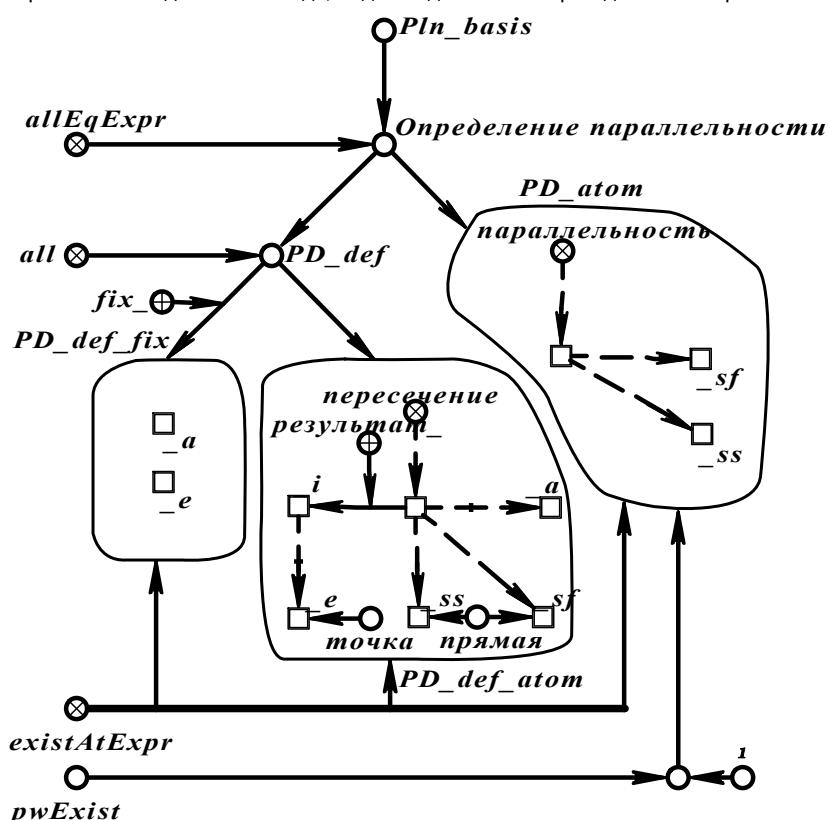


/* "существуют две точки" */

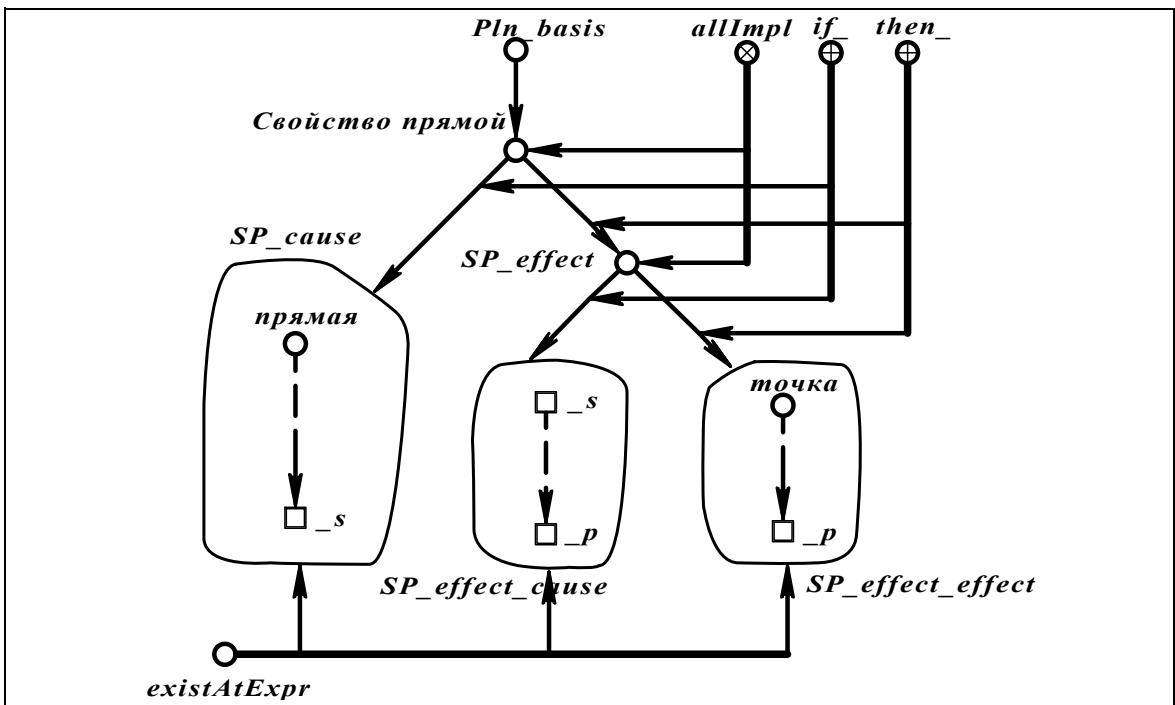


Аксиома о существовании двух точек

/* "две прямые параллельны тогда и только тогда, когда ни одна точка не принадлежит их пересечению" */



/* "если элемент принадлежит прямой, то этот элемент – точка" */



/* "фигура прямолинейна тогда и только тогда, когда существует прямая включающая все элементы фигуры" */

```

PIn_basis → Определение прямолинейной фигуры ;
allEqExpr → Определение прямолинейной фигуры ;
Определение прямолинейной фигуры → SFD_def, SFD_atom ;
existAtExpr → SFD_atom ;
SFD_atom = [прямолинейная фигура →> f] ;
exist → SFD_def ;
existAtExpr → SFD_def ;
SFD_def = [подмножество →> ( подмножество_ : f, множество_ : s) ;
прямая →> s] ;

```

/* "пересечением произвольного количества множеств является множество ,включающее такие и только такие элементы, каждый из которых принадлежит каждому множеству , образующему пересечение" */

```

PIn_basis → Set_basis ;
Set_basis → Определение пересечения ;
conj → Определение пересечения ;
Определение пересечения → DD_forward, DD_backward ;
allImpl → DD_forward, DD_backward ;
DD_forward → if_ : DD_def_f, then_ : DD_atom ;
DD_backward → if_ : DD_atom, then_ : DD_def_b ;
existAtExpr → DD_atom ;
DD_atom = [пересечение →> c] ;
exist → DD_def_f ;

```

```

DD_def_f → fix_ : DD_fix , DD_def_conj ;
exist → DD_def_b ;
DD_def_b → fix_ : DD_fix , DD_def_equivalence ;
DD_fix = [ _d ] ;
allEqExpr → DD_def_equivalence ;
conj → DD_def_conj ;
DD_def_conj → DD_def_equivalence_equivalence , DD_def_atom ;
DD_def_equivalence → DD_def_equivalence_equivalence , DD_def_atom ;
existAtExpr → DD_def_atom ;
DD_def_atom = [ _c →> результат_ :: _d ] ;
allEqExpr → DD_def_equivalence_equivalence ;
DD_def_equivalence_equivalence → DD_equivalence_atom , DD_implication ;
existAtExpr → DD_equivalence_atom ;
DD_equivalence_atom = [ _d →> _e ] ;
allImpl → DD_implication ;
DD_implication → if_ : DD_cause , then_ : DD_effect ;
existAtExpr → DD_cause , DD_effect ;
DD_cause = [ _c →> _s ] ;
DD_effect = [ _s →> _e ] ;

```

```

/*
 * "множество ss является подмножеством множества s, если каждый элемент ss принадлежит множеству s"
 */
Set_basis → Определение подмножества ;
conj → Определение подмножества ;
Определение подмножества → SSD_forward , SSD_backward ;
allImpl → SSD_forward , SSD_backward ;
SSD_forward → if_ : SSD_def_f , then_ : SSD_atom ;
SSD_backward → if_ : SSD_atom , then_ : SSD_def_b ;
existAtExpr → SSD_atom ;
SSD_atom = [ подмножество →> _c ] ;
exist → SSD_def_f ;
SSD_def_f → fix_ : SSD_fix , SSD_def_conj ;
SSD_fix = [ _ss , _s ] ;
conj → SSD_def_conj ;
allEqExpr → SSD_def_b ;
SSD_def_b → SSD_def_atom , SSD_implication ;
SSD_def_conj → SSD_def_atom , SSD_implication ;
existAtExpr → SSD_def_atom ;
SSD_def_atom = [ _c →> подмножество_ :: _ss , множество_ :: _s ] ;
allImpl → SSD_implication ;
SSD_implication → if_ : SSD_cause , then_ : SSD_effect ;
existAtExpr → SSD_cause , SSD_effect ;

```

```
SSD_cause = [ ss →> e ; ] ;
SSD_effect = [ s →> e ; ] ;
```

Далее следует формулировка запроса.

```
/* "существуют параллельные прямые" */
existAtExpr → query_atom1 ;
query_atom1 = [ параллельность →> ( a , c ) : прямая →> a , c ; ] ;
```

Приведём описание решения задачи на естественном языке.

Процесс решения состоит из следующих этапов:

- 1) определение множества формальных теорий описывающих закономерности свойств понятий, обозначенных константными элементами в запросе;
- 2) для конкретной теории заранее формируется множество целей (запросов) и множество предположений;
- 3) начинается обработка запроса, определяется тип высказывания запроса;
- 4) проверяется выводимость отрицания высказывания в рамках формальной теории, перебираются все истинные интерпретации (этапы 5, 6), каждая интерпретация на каждом этапе и только на нём включается во множество предположений;
- 5) осуществляется поиск семантически близких истинных высказываний формальной теории, при необходимости используются факты формальной теории, в данном примере такими высказываниями изначально являются: **"Аксиома о параллельной прямой"**, **"Аксиома о прямой"**, **"Определение параллельности"**, **"Свойство прямой"**, **"Определение прямолинейной фигуры"** и т. п.;
- 6) осуществляется протоколирующийся логический вывод с помощью отобранных на предыдущем этапе высказываний, этапы 5 и 6 повторяются, пока не закончатся семантически близкие высказывания или будет выявлено противоречие, в последнем случае высказывание ложно и при условии возникновения противоречия на каждой истинной интерпретации осуществляется переход на этап 10;
- 7) проверяется выводимость высказывания в рамках формальной теории, перебираются все ложные интерпретации (этапы 8, 9), каждая интерпретация на каждом этапе и только на нём включается во множество предположений;
- 8) осуществляется поиск семантически близких истинных высказываний формальной теории, при необходимости используются факты формальной теории;
- 9) осуществляется протоколирующийся логический вывод с помощью отобранных на предыдущем этапе высказываний, этапы 8 и 9 повторяются, пока не закончатся семантически близкие высказывания или будет выявлено противоречие, в последнем случае высказывание истинно;
- 10) на основании результатов полученных на этапах 6 и 9, формируется ответ: истинно ли, должно или невыводимо высказывание в рамках текущей формальной теории; уничтожается множество предположений, выводится протокол решения;
- 11) процесс повторяется со второго этапа для каждой найденной формальной теории.

Рассмотрим более детально этот процесс для данного примера.

На первом этапе анализируются все константные элементы, входящие хотя бы в одно из атомарных высказываний запроса. В данном случае это элементы: **"прямая"**, **"параллельность"**, то есть те элементы, которые также входят во множество, помеченное атрибутом **"union_"** в рамках какой-либо формальной теории. На основе этого анализа осуществляется построение множества **"theory_set"** всех таких формальных теорий. В данном примере множество **"theory_set"** будет содержать элемент **"Планиметрия"**.

На втором этапе формулируется запрос к каждой формальной теории:

```
Goals ← gq1 ← query1 ;
```

```
active_ , confirm_ → gq1 ;
query1 = [ theory → Планиметрия →> query_atom1 ] ;
```

На третьем этапе начинается обработка запроса (распишем последующие этапы по шагам) :

Шаг 1. К запросу “*query1*” применяется операция классификации и управления запросами, которая формирует конструкцию следующего вида:

```
Goals → search : query1 ;
```

Шаг 2. К запросу “*query1*” применяется операция эпизодического информационного поиска. Которая завершается безуспешно:

```
Goals → active_ : searched : query1 ;
factual → query1 ;
```

Шаг 3. К запросу “*query1*” применяется операция классификации и управления запросами, которая формирует конструкцию следующего вида:

```
Goals → traceUp : query1 ;
```

Шаг 4. К запросу “*query1*” применяется операция трассировки запроса снизу-вверх в сложном высказывании, которая завершается безуспешно:

```
Goals → tracedUp : active_ : query1 ;
```

Шаг 5. К запросу “*query1*” применяется операция классификации и управления запросами, которая формирует конструкцию следующего вида:

```
Goals → traceDown : query1 ;
```

Шаг 6. К запросу “*query1*” применяется операция трассировки запроса сверху-вниз, которая формирует конструкцию следующего вида:

```
Goals ≈ gq1 ≈ query1 ;
tracedDown , active_ → gq1 ;
reasoning → (effect_ : gq1 , expectations_ : (gx1) , causes_ : C1 ) ;
Goals ≈ gx1 ≈ query_atom1 ;
tracedUp , confirm_ , active_ → gx1 ;
```

Шаг 7. К запросу “*query_atom1*” применяется операция классификации и управления запросами, которая формирует конструкцию следующего вида:

```
Goals → search : query_atom1 ;
```

Шаг 8. К запросу “*query_atom1*” применяется операция эпизодического информационного поиска. Которая завершается безуспешно:

```
Goals → active_ : searched : query_atom1 ;
factual → query_atom1 ;
```

Шаг 9. К запросу “*query_atom1*” применяется операция классификации и управления запросами, которая формирует конструкцию следующего вида:

```
Goals → associateSimply : (associative →> (
  formula → query_atom1 ), (theory → Планиметрия ), _as · );
```

Шаг 10. К атомарному высказыванию “*query_atom1*” применяется операция простого ассоциирования атомарных формул. Среди таких формул будут формулы: “*PSA_atom*”, “*PSA_effect*”, “*SA_cause*”, “*SPPA_effect*”, “*PD_def_atom*”, “*PD_atom*”, “*SP_cause*”, “*SFD_def*”. Будет построена конструкция:

```
associative → ((formula → query_atom1 ), (theory → Планиметрия ),
  (PSA_atom , PSA_effect , SA_cause , SPPA_effect , PD_def_atom , PD_atom , SP_c
ause , SFD_def )) ;
Goals → active_ : query_atom1 ;
```

Шаг 11. К атомарному высказыванию “*query_atom1*” применяется операция выявления идентичных, входящих и пересекающихся атомарных формул. Результатом применения операции будут конструкции:

```
inclusion → ((formula → PSA_atom ), in_ : (formula → SP_cause )) ;
inclusion → ((formula → PSA_atom ), in_ : (formula → SA_cause )) ;
SA_cause = [прямая →> _s ] ; /* qat_PSAa_intersection */
inclusion → ((formula → query_atom1 ), in_ : (formula → PD_atom )) ;
inclusion → ((formula → PSA_effect ), in_ : (formula → PD_atom )) ;
inclusion → ((formula → PSA_effect ), in_ : (formula → SP_cause )) ;
inclusion → ((formula → PSA_effect ), in_ : (formula → SA_cause )) ;
PD_atom = [параллельность →> (_sf , _ss·) ;] /* qat_PSAe_intersection */
inclusion → ((formula → SPPA_effect ), in_ : (formula → SP_cause )) ;
inclusion → ((formula → SPPA_effect ), in_ : (formula → SA_cause )) ;
SA_cause = [прямая →> _s ] /* qat_SPPAe_intersctn */
inclusion → ((formula → query_atom1 ), in_ : (formula →
qat_PDda_intersection )) ;
inclusion → ((formula → PD_def_atom ) ,
  in_ : (formula → qat_PDda_intersection )) ;
qat_PDda_intersection = [прямая →> _s1 , _s2 ;];
inclusion → ((formula → qat_PDda_intersection ), in_ : (formula →
SP_cause )) ;
inclusion → ((formula → qat_PDda_intersection ), in_ : (formula →
SA_cause )) ;
inclusion → ((formula → SFD_def ), in_ : (formula → SP_cause )) ;
inclusion → ((formula → SFD_def ), in_ : (formula → SA_cause )) ;
(formula → SP_cause ) = (formula → SA_cause ) ;
```

Шаг 12. К атомарным формулам “*SA_cause*” и “*SP_cause*” применяется операция склейки атомарных формул: в результате чего формируется формула “*S__cause*”, а формулы “*SA_cause*” и “*SP_cause*” уничтожаются, а сгенерированные на предыдущем шаге конструкции преобразуются следующим образом:

```

inclusion → ((formula → PSA_atom), in_ : (formula → S__cause));
inclusion → ((formula → PSA_effect), in_ : (formula → S__cause));
inclusion → ((formula → SPPA_effect), in_ : (formula → S__cause));
inclusion → ((formula → qat_PDda_intersection), in_ : (formula →
S__cause));
inclusion → ((formula → SFD_def), in_ : (formula → S__cause));

```

Шаг 13. К атомарному высказыванию “*query_atom*” применяется операция декомпозиции запроса на вывод атомарной формулы на запросы к семантически близким атомарным формулам. Запрос будет выставлен к формулам “*PSA_atom*”, “*PSA_effect*”, “*S__cause*”, “*SPPA_effect*”, “*PD_def_atom*”, “*PD_atom*”, “*SFD_def*”.

```

decomposed , searched , confirm_ , active_ → gx1 ;
reasoning → !effect_ : gx1 , expectations_ : !gx2 , causes_ : C2 );
Goals ← gx2 ← PD_atom ;
searched , confirm_ , active_ → gx2 ;
reasoning → !effect_ : gx1 , expectations_ : !gx3 , causes_ : C3 ) ;
Goals ← gx3 ← qat_PDda_intersection ;
searched , confirm_ , active_ → gx3 ;
reasoning → !effect_ : gx3 , expectations_ : !gx4 , causes_ : C4 ) ;
Goals ← gx4 ← S__cause ;
searched , confirm_ , active_ → gx4 ;
reasoning → !effect_ : gx4 , expectations_ : !gx5 , causes_ : C5 ) ;
Goals ← gx5 ← PSA_atom ;
searched , confirm_ , active_ → gx5 ;
reasoning → !effect_ : gx2 , expectations_ : E6 , causes_ : C6 ) ;
reasoning → !effect_ : gx4 , expectations_ : E6 , causes_ : C6 ) ;
E6 → gx6 ;
Goals ← gx6 ← PSA_effect ;
searched , confirm_ , active_ → gx6 ;
reasoning → !effect_ : gx4 , expectations_ : !gx7 , causes_ : C7 ) ;
Goals ← gx7 ← SPPA_effect ;
searched , confirm_ , active_ → gx7 ;
reasoning → !effect_ : gx3 , expectations_ : !gx8 , causes_ : C8 ) ;
Goals ← gx8 ← PD_def_atom ;
searched , confirm_ , active_ → gx8 ;
reasoning → !effect_ : gx4 , expectations_ : !gx9 , causes_ : C9 ) ;
Goals ← gx9 ← SFD_def ;
searched , confirm_ , active_ → gx9 ;

```

```
factual ->
PD_atom , qai_PDda_intersection , S__cause , PSA_atom , PSA_effect , SPPA_eff
ct , PD_def_atom , SFD_def ;
```

Шаг 14. Остальные операции к высказыванию “*query_atom1*” не применимы, что выясняется в процессе применения к нему операции классификации и управления запросами.

Шаг 15. Начиная с формулы “*PSA_cause*” ко всем входящим формулам высказывания “*Аксиома о параллельной прямой*” ставится запрос на применение и применяется операция формирования множества свободных или связываемых переменных. Формируются, в частности, конструкции:

```
unassigned -> ((formula -> PSA_cause) , (theory -> Планиметрия) ,
PSA_cause_uVars) ;
PSA_cause_uVars = [_sf , _p ] ;
unassigned -> ((formula -> PSA_neg) , (theory -> Планиметрия) ,
PSA_neg_uVars) ;
PSA_neg_uVars = [_sf , _p ] ;
unassigned -> ((formula -> Аксиома о параллельной прямой) , PSA_uVars ,
(theory -> Планиметрия)) ;
PSA_uVars = [_sf , _p ] ;
```

Шаг 16. После операции классификации и управления запросами к атомарной формуле “*PSA_atom*” применяется операция трассировки запроса снизу-вверх, которая находит формулу “*PSA_cause*”. В результате формируется конструкция:

```
reasoning -> !effect_ : gx5 , expectations_ : !gx10) , causes_ : C10 ) ;
unassigned -> ((formula -> PSA_cause) , (theory -> Планиметрия) ,
PSA_cause_uVars) ;
Goals -> gx10 -> PSA_cause_uVars ;
searched , confirm_ , active_ -> gx10 ;
tracedUp , active_ -> gx5 ;
```

Шаг 17. Затем к формуле “*PSA_cause*” применяется операция трассировки запроса сверху-вниз, которая находит формулу “*PSA_neg*”. В результате формируется конструкция:

```
reasoning -> (effect_ : gx10 , expectations_ : (gx5 , gx11) , causes_ : C11 ) ;
unassigned -> ((formula -> PSA_neg) , (theory -> Планиметрия) ,
PSA_neg_uVars) ;
Goals -> gx11 -> PSA_neg_uVars ;
searched , confirm_ , active_ -> gx11 ;
tracedDown , active_ -> gx10 ;
```

Шаг 18. К формуле “*PSA_cause*” применяется операция трассировки запроса снизу-вверх, которая находит априори истинное импликативное высказывание “*Аксиома о параллельной прямой*”, следствием которого является формула “*PSA_effect*”. В результате формируются конструкции:

```
reasoning -> (effect_ : gx14 , expectations_ : (gx12 , gx13) , causes_ : C12 ) ;
```

```

unassigned → ((formula → Аксиома о параллельной прямой ) , PSA_uVars ,
(theory → Планиметрия ) );
Goals ← gx12 ← PSA_uVars ;
searched , confirm_ , active_ → gx12 ;
Goals ← gx13 ← PSA_effect ;
deny_ , active_ → gx13 ;
Goals ← gx14 ← [refuse →> gx10 ] ;
confirm_ → gx14 ;
tracedUp , active_ → gx10 ;

```

Шаг 19. К формуле “*Аксиома о параллельной прямой*” применяется операция трассировки запроса снизу-вверх. В результате формируется конструкция:

```
confirmed_ → gx12 ;
```

Шаг 20. К формуле “*PSA_neg*” применяется операция трассировки запроса сверху-вниз, которая находит формулу “*PSA_neg_atom*”. В результате формируется конструкция:

```

reasoning → (effect_ : gx11 , expectations_ : (gx15 ) , causes_ : C15 ) ;
Goals ← gx15 ← PSA_neg_atom ;
deny_ , active_ → gx15 ;
tracedDown , active_ → gx11 ;

```

Шаг 21. К атомарной формуле “*PSA_effect*” применяется операция эпизодического информационного поиска. Которая завершается безуспешно:

```
factual →> PSA_effect ;
searched , active_ → gx13 ;
```

Шаг 22. К атомарной формуле “*PSA_effect*” применяется операция трассировки запроса снизу-вверх, которая находит априори истинное импликативное высказывание “*Аксиома о параллельной прямой*”, посылкой которого является формула “*PSA_cause*”. В результате формируется конструкция:

```

reasoning → (effect_ : gx6 , expectations_ : (gx10 ) , causes_ : C13 ) ;
C13 → gx12 ;
tracedUp , active_ → gx6 ;

```

Шаг 23. К формуле “*PSA_cause*” применяется операция трассировки запроса сверху-вниз, которая находит формулы “*PSA_atom*” и “*PSA_neg*”. В результате формируется конструкция:

```

reasoning → (effect_ : gx10 , expectations_ : (gx11 , gx5 ) , causes_ : C14 ) ;
tracedDown , active_ → gx10 ;

```

Шаг 24. К атомарной формуле “**PSA_atom**” применяется операция простого ассоциирования атомарных формул. Будет построена конструкция:

```
associative → ((formula → PSA_atom), (theory → Планиметрия),
(query_atom1, PSA_effect, S__cause, SA_effect_atom, SA_effect_neg_atom, SP
PA_cause, SPPA_effect, Аксиома о существовании двух
точек, PD_def_atom, SFD_def, SP_effect_effect));
searched, confirm_, active_ → gx5;
```

Шаг 25. К атомарной формуле “**PSA_atom**” применяется операция выявления идентичных, входящих и пересекающихся атомарных формул. Результатом применения операции будет:

```
inclusion → ((formula → PSA_atom), in_ : (formula → S__cause));
inclusion → ((formula → qat_PDda_intersection), in_ : (formula →
S__cause));
inclusion → ((formula → query_atom1), in_ : (formula →
qat_PDda_intersection));
inclusion → ((formula → PSA_effect), in_ : (formula → S__cause));
inclusion → ((formula → PSA_atom), in_ : (formula → SP_effect_effect));
inclusion → ((formula → SA_effect_atom), in_ : (formula →
SP_effect_effect));
inclusion → ((formula → SA_effect_neg_atom),
in_ : (formula → SP_effect_effect));
SP_effect_effect = [точка →> _p]; /* PSAa_SAea_intersection */
inclusion → ((formula → SPPA_cause), in_ : (formula → SP_effect_effect));
inclusion → ((formula → Аксиома о существовании двух точек),
in_ : (formula → SP_effect_effect));
inclusion → (in_ : (formula → Аксиома о существовании двух точек),
in_ : (formula → SPPA_cause));
inclusion → ((formula → SPPA_effect), in_ : (formula → S__cause));
inclusion → ((formula → PD_def_atom),
in_ : (formula → qat_PDda_intersection));
inclusion → ((formula → PD_def_atom), in_ : (formula → PSA_atom));
inclusion → ((formula → SFD_def), in_ : (formula → S__cause));
```

Шаг 26. К атомарному формуле “**PSA_atom**” применяется операция декомпозиции запроса на вывод атомарной формулы на запросы к семантически близким атомарным формулам. В результате будут добавлены следующие конструкции:

```
reasoning → heffect_ : gx5, expectations_ : hgx4, causes_ : C15);
decomposed, searched, confirm_, active_ → gx5;
reasoning → heffect_ : gx4, expectations_ : hgx3, causes_ : C16 );
searched, confirm_, active_ → gx4;
reasoning → heffect_ : gx3, expectations_ : hgx1), causes_ : C17 );
searched, confirm_, active_ → gx3;
reasoning → heffect_ : gx5, expectations_ : hgx18), causes_ : C18 );
Goals ← gx18 ← SP_effect_effect;
```

```

searched , confirm_ , active_ → gx18 ;
reasoning → ℩ effect_ : gx18 , expectations_ : ℩ gx19 ) , causes_ : C19 ) ;
Goals ← gx19 ← SA_effect_atom ;
searched , confirm_ , active_ → gx19 ;
reasoning → ℩ effect_ : gx18 , expectations_ : ℩ gx20 ) , causes_ : C20 ) ;
Goals ← gx20 ← SA_effect_neg_atom ;
searched , confirm_ , active_ → gx20 ;
reasoning → ℩ effect_ : gx18 , expectations_ : ℩ gx21 ) , causes_ : C21 ) ;
Goals ← gx21 ← SPPA_cause ;
searched , confirm_ , active_ → gx21 ;
reasoning → ℩ effect_ : gx18 , expectations_ : ℩ gx22 ) , causes_ : C22 ) ;
Goals ← gx22 ← «Аксиома о существовании двух точек» ;
searched , confirm_ , active_ → gx22 ;
reasoning → ℩ effect_ : gx22 , expectations_ : ℩ gx21 ) , causes_ : C21 ) ;
reasoning → ℩ effect_ : gx21 , expectations_ : ℩ gx22 ) , causes_ : C22 ) ;
reasoning → ℩ effect_ : gx5 , expectations_ : ℩ gx8 ) , causes_ : C23 ) ;
factual →
SP_effect_effect , SA_effect_atom , SA_effect_neg_atom , SPPA_cause , Аксиома
о существовании двух точек ;

```

Шаг 27. К атомарной формуле “*PSA_effect*” применяется операция простого ассоциирования атомарных формул. Будет построена конструкция:

```

associative → ((formula → PSA_effect) , (theory →
Планиметрия) , (query_atom1 , PSA_atom , S__cause , SPPA_effect , PD_def_ato
m , PD_atom , SFD_def)) ;
searched , deny_ , active_ → gx13 ;

```

Шаг 28. К атомарной формуле “*PSA_effect*” применяется операция выявления идентичных, входящих и пересекающихся атомарных формул. Результатом применения операции будут найденные конструкции:

```

inclusion → ((formula → qat_PDda_intersection) , in_ : (formula →
S__cause)) ;
inclusion → ((formula → query_atom1) , in_ : (formula →
qat_PDda_intersection)) ;
inclusion → ((formula → PSA_effect) , in_ : (formula → S__cause)) ;
inclusion → ((formula → PSA_atom) , in_ : (formula → S__cause)) ;
inclusion → ((formula → PD_def_atom) ,
in_ : (formula → qat_PDda_intersection)) ;
inclusion → ((formula → PSA_effect) , in_ : (formula → PD_atom)) ;
inclusion → ((formula → SFD_def) , in_ : (formula → S__cause)) ;

```

Шаг 29. К атомарной формуле “*PSA_effect*” применяется операция декомпозиции запроса на вывод атомарной формулы на запросы к семантически близким атомарным формулам. В результате будут добавлены следующие конструкции:

```

decomposed , searched , deny_ , active_ → gx13 ;
reasoning → ℩effect_ : gx13 , expectations_ : ℩gx24) , causes_ : C24 ) ;
Goals ← gx24 ← S_cause ;
searched , deny_ , active_ → gx24 ;
reasoning → ℩effect_ : gx13 , expectations_ : ℩gx25) , causes_ : C25 ) ;
Goals ← gx25 ← PSA_effect ;
searched , deny_ , active_ → gx25 ;
factual → S_cause , PSA_effect ;

```

Шаг 30. Ко всем входящим формулам высказывания “*Аксиома о прямой*” ставится запрос на применение и применяется операция формирования множества свободных или связываемых переменных. Формируются, в частности, конструкции:

```

unassigned → ((formula → SA_effect) , (theory → Планиметрия) ,
SA_effect_uVars) ;
SA_effect_uVars = [ _s ] ;
unassigned → ((formula → Аксиома о прямой) , SA_uVars ,
(theory → Планиметрия) ) ;
SA_uVars = [ _s ] ;

```

Шаг 31. Применяется операция трассировки запроса снизу-вверх, которая находит априори истинное импликативное высказывание “*Аксиома о прямой*”, следствием которого является конъюнктивная формула “*SA_effect*”. В результате формируется конструкция:

```

reasoning → ℩effect_ : gx28 , expectations_ : ℩gx26 , gx27) , causes_ : C26 ) ;
unassigned → ((formula → Аксиома о прямой) , SA_uVars ,
(theory → Планиметрия) ) ;
Goals ← gx27 ← SA_uVars ;
searched , confirm_ , active_ → gx27 ;
Goals ← gx26 ← SA_effect_uVars ;
searched , deny_ , active_ → gx26 ;
Goals ← gx28 ← [ refuse →> gx4 ] ;
confirm_ → gx28 ;
tracedUp , active_ → gx4 ;

```

Шаг 32. К формуле “*Аксиома о прямой*” применяется операция трассировки запроса снизу-вверх. В результате формируется конструкция:

```

confirmed_ → gx27 ;

```

Шаг 33. К формуле “*SA_effect*” применяется операция трассировки запроса сверху-вниз, которая находит формулы: “*SA_effect_neg_atom*” и “*SA_effect_atom*”. В результате формируется конструкция:

```

reasoning → ℩effect_ : gx26 , expectations_ : ℩gx29) , causes_ : C29 ) ;
Goals ← gx29 ← SA_effect_atom ;
searched , deny_ , active_ → gx29 ;

```

```

reasoning → h_effect_ : gx26 , expectations_ : h gx30 ) , causes_ : C30 ) ;
Goals ← gx30 ← SA_effect_neg_atom ;
searched , deny_ , active_ → gx30 ;

```

Шаг 34. Для атомарной формулы “**SA_effect_atom**” выполняется операция эпизодического информационного поиска. Которая завершается безуспешно:

```

factual → SA_effect_atom ;

```

Шаг 35. Для атомарной формулы “**SA_effect_neg_atom**” выполняется операция эпизодического информационного поиска. Которая завершается безуспешно:

```

factual → SA_effect_neg_atom ;

```

Шаг 36. Для атомарной формулы “**SA_effect_atom**” выполняется операция трассировки запроса снизу-вверх, которая находит конъюнктивную формулу “**SA_effect**”. В результате формируется конструкция:

```

reasoning → h_effect_ : gx19 , expectations_ : h gx026 ) , causes_ : C026 ) ;
Goals ← gx026 ← SA_effect_uVars ;
searched , confirm_ , active_ → gx026 ;

```

Шаг 37. Для атомарной формулы “**SA_effect_neg_atom**” выполняется операция трассировки запроса снизу-вверх, которая находит конъюнктивную формулу “**SA_effect**”. В результате формируется конструкция:

```

reasoning → h_effect_ : gx20 , expectations_ : h gx026 ) , causes_ : C027 ) ;

```

Шаг 38. Для атомарной формулы “**SA_effect**” выполняется операция трассировки запроса снизу-вверх, которая находит априори истинное импликативное высказывание “**Аксиома о прямой**”, посылкой которого является формула “**SA_cause**”. В результате формируется конструкция:

```

reasoning → h_effect_ : gx026 , expectations_ : h gx4 ) , causes_ : h gx27 ) ;

```

Шаг 39. К атомарной формуле “**SA_effect_atom**” применяется операция простого ассоциирования атомарных формул. Будет построена конструкция:

```

associative → ((formula → PSA_effect_atom) , (theory →
Планиметрия) , (PSA_atom , SA_effect_neg_atom , SPPA_cause , Аксиома о
существовании двух точек , PD_def_atom , SP_effect_effect)) ;
searched , deny_ , active_ → gx29 ;

```

Шаг 40. К атомарной формуле “**SA_effect_atom**” применяется операция выявления идентичных, входящих и пересекающихся атомарных формул. Результатом применения операции будут конструкции:

```

inclusion → ((formula → PSA_atom) , in_ : (formula → SP_effect_effect)) ;
inclusion → ((formula → SA_effect_atom) , in_ : (formula →
SP_effect_effect)) ;

```

```

inclusion → ((formula → SA_effect_neg_atom) ,
    in_ : (formula → SP_effect_effect)) ;
inclusion → ((formula → SPPA_cause) , in_ : (formula → SP_effect_effect)) ;
inclusion → ((formula → Аксиома о существовании двух точек) ,
    in_ : (formula → SP_effect_effect)) ;
inclusion → (in_ : (formula → Аксиома о существовании двух точек) ,
    in_ : (formula → SPPA_cause)) ;
inclusion → ((formula → PD_def_atom) , in_ : (formula → PSA_atom)) ;

```

Шаг 41. К атомарному формуле “*SA_effect_atom*” применяется операция декомпозиции запроса на вывод атомарной формулы на запросы к семантически близким атомарным формулам. В результате будут добавлены следующие конструкции:

```

decomposed , searched , deny_ , active_ → gx29 ;
reasoning → heffect_ : gx29 , expectations_ : hgx31 , causes_ : C31 ) ;
Goals ← gx29 ← SA_effect_atom ;
Goals ← gx31 ← SP_effect_effect ;
searched , deny_ , active_ → gx31 ;
factual → SP_effect_effect ;

```

Шаг 42. Начиная с формулы “*SP_effect*” ко всем входящим формулам высказывания “*Свойство прямой*” ставится запрос на применение и применяется операция формирования множества свободных или связываемых переменных. Формируются, в частности, конструкции:

```

unassigned → ((formula → SP_effect) , (theory → Планиметрия) ,
    SP_effect_uVars) ;
SP_effect_uVars = [ _s , _p ] ;
unassigned → ((formula → Свойство прямой) , SP_uVars ,
    (theory → Планиметрия) ) ;
SP_uVars = [ _s ] ;

```

Шаг 43. Применяется операция трассировки запроса снизу-вверх, которая находит импликативную формулу “*SP_effect*”. В результате формируется конструкция:

```

reasoning → heffect_ : gx18 , expectations_ : hgx32 , gx33) , causes_ : C32 ) ;
reasoning → heffect_ : gx31 , expectations_ : hgx34 , gx33) , causes_ : C34 ) ;
unassigned → ((formula → SP_effect) , SP_effect_uVars ,
    (theory → Планиметрия) ) ;
Goals ← gx32 ← SP_effect_uVars ;
searched , confirm_ , active_ → gx32 ;
Goals ← gx34 ← SP_effect_uVars ;
searched , deny_ , active_ → gx34 ;
Goals ← gx33 ← SP_effect_cause ;
confirm_ , active_ → gx33 ;
tracedUp , active_ → gx31 ;
tracedUp , active_ → gx18 ;

```

Шаг 44. Применяется операция трассировки запроса снизу-вверх, которая находит априори истинное импликативное высказывание “*Свойство прямой*”, посылкой которого является атомарная формула “*SP_cause*”. В результате формируется конструкция:

```

reasoning → h effect_ : gx32 , expectations_ : h gx35 , gx4) , causes_ : C35 ) ;
reasoning → h effect_ : gx34 , expectations_ : h gx36 , gx4) , causes_ : C36 ) ;
unassigned → ((formula → Свойство прямой) , SP_uVars ,
               (theory → Планиметрия) ) ;
Goals ← gx35 ← SP_uVars ;
searched , confirm_ , active_ → gx35 ;
Goals ← gx36 ← SP_uVars ;
searched , deny_ , active_ → gx36 ;
Goals ← gx4 ← S__cause ;
tracedUp , active_ → gx32 ;
tracedUp , active_ → gx34 ;

```

Шаг 45. К атомарной формуле “*S__cause*” применяется операция простого ассоциирования атомарных формул. Будет построена конструкция:

```

associative → ((formula → S__cause) , (theory →
Планиметрия) , (query_atom1 , PSA_atom , PSA_effect , SA_cause , SPPA_effect ,
PD_def_atom , SFD_def)) ;
searched , confirm_ , active_ → gx4 ;

```

Шаг 46. К атомарной формуле “*S__cause*” применяется операция выявления идентичных, входящих и пересекающихся атомарных формул. Результатом применения операции будут конструкции:

```

inclusion → ((formula → query_atom1) , in_ : (formula →
qa1_PDda_intersection)) ;
inclusion → ((formula → qa1_PDda_intersection) , in_ : (formula →
S__cause)) ;
inclusion → ((formula → PSA_atom) , in_ : (formula → S__cause)) ;
inclusion → ((formula → PSA_effect) , in_ : (formula → S__cause)) ;
inclusion → ((formula → SPPA_effect) , in_ : (formula → S__cause)) ;
inclusion → ((formula → PD_def_atom) , in_ : (formula → PSA_atom)) ;
inclusion → ((formula → SFD_def) , in_ : (formula → S__cause)) ;

```

Шаг 47. К атомарному высказыванию “*S__cause*” применяется операция декомпозиции запроса на вывод атомарной формулы на запросы к семантически близким атомарным формулам. В результате будут добавлены следующие конструкции:

```

decomposed , searched , confirm_ , active_ → gx4 ;

```

Шаг 48. Ко всем формулам высказывания “*Аксиома о прямой и двух точках*” ставится запрос на применение и применяется операция формирования множества свободных или связываемых переменных. Формируется, в частности, конструкция:

```

unassigned → ((formula → Аксиома о прямой и двух точках) ,
    SPPA_uVars ,
    (theory → Планиметрия)) ;
SPPA_uVars = [_a , _b] ;

```

Шаг 49. Применяется операция трассировки запроса снизу-вверх, которая находит априори истинное импликативное высказывание “**Аксиома о прямой и двух точках**”, которое включает формулу “**SPPA_cause**” в качестве посылки. В результате формируется конструкция:

```

reasoning → ℩effect_ : gx7 , expectations_ : ℩gx37 , gx21) , causes_ : C37 ) ;
unassigned → ((formula → SP_effect) , SPPA_uVars ,
    (theory → Планиметрия)) ;
Goals ← gx37 ← SPPA_uVars ;
searched , confirm_ , active_ → gx37 ;
Goals ← gx21 ← SPPA_cause ;
searched , confirm_ , active_ → gx21 ;

```

Шаг 50. К атомарной формуле “**SPPA_cause**” применяется операция простого ассоциирования атомарных формул. Будет построена конструкция:

```

associative → ((formula → SPPA_cause) , (theory →
Планиметрия) , (PSA_atom , SA_effect_atom , SA_effect_neg_atom , Аксиома о
существовании двух точек , PD_def_atom , SP_effect_effect)) ;
searched , confirm_ , active_ → gx21 ;

```

Шаг 51. К атомарной формуле “**SPPA_cause**” применяется операция выявления идентичных, входящих и пересекающихся атомарных формул. Результатом применения операции будут конструкции:

```

inclusion → ((formula → PSA_atom) , in_ : (formula → SP_effect_effect)) ;
inclusion → ((formula → SPPA_cause) , in_ : (formula → SP_effect_effect)) ;
inclusion → ((formula → SA_effect_atom) , in_ : (formula →
SP_effect_effect)) ;
inclusion → ((formula → SA_effect_neg_atom) ,
    in_ : (formula → SP_effect_effect)) ;
inclusion → ((formula → Аксиома о существовании двух точек) ,
    in_ : (formula → SP_effect_effect)) ;
inclusion → (in_ : (formula → Аксиома о существовании двух точек) ,
    in_ : (formula → SPPA_cause)) ;
inclusion → ((formula → PD_def_atom) , in_ : (formula → PSA_atom)) ;

```

Шаг 52. К атомарной формуле “**SPPA_cause**” применяется операция декомпозиции запроса на вывод атомарной формулы на запросы к семантически близким атомарным формулам. В результате будут добавлены следующие конструкции:

```

decomposed , associatedSimply , searched , confirm_ , active_ → gx21 ;
reasoning → ℩effect_ : gx21 , expectations_ : ℩gx18) , causes_ : C33 ) ;
reasoning → ℩effect_ : gx18 , expectations_ : ℩gx5) , causes_ : C39 ) ;

```

Шаг 53. Для атомарного формулы “*Аксиома о существовании двух точек*” выполняется операция эпизодического информационного поиска. Которая завершается безуспешно:

```
factual -> Аксиома о существовании двух точек ;
```

Шаг 54. Применяется операция трассировки запроса снизу-вверх, которая выявляет истинность высказывания “*Аксиома о существовании двух точек*”.

```
reasoning -> !effect_ : gx22 , expectations_ : E39 , causes_ : !gx39)) ;
Goals <- gx39 <- query2 ;
confirmed_ -> gx39 ;
query2 = [ theory -> Планиметрия ->> Аксиома о существовании двух
точек ] ;
searched , active_ -> gx22 ;
```

Шаг 55. Применяется операция формирования фиктивной интерпретации атомарной формулы, которая формирует следующую конструкцию:

```
interpretation -> ((formula -> Аксиома о существовании двух
точек) , interpretation_ : ((var_ : _a , value_ : av) , !var_ : _aa , value_ : aav) , !var_
:_b , value_ : bv) , !var_ : _ba , value_ : bav) ) , (formula -> f)) ;
f = [ точка >- aav >- av ; точка >- bav >- bv ; ] ;
confirmed_ -> gx22 ;
```

где *_aa* и *_ba* соответственно переменные дуги, включённые в высказывание и входящие в *_a* и в *_b*.

Шаг 56. К формуле “*SPPA_cause*” применяется операция формирования интерпретации формулы, которая формирует аналогичную конструкцию:

```
interpretation -> ((formula -> SPPA_cause) , (formula ->
f) , interpretation_ : ((var_ : _a , value_ : av) , !var_ : _aa , value_ : aav) , !var_ : _b ,
value_ : bv) , !var_ : _ba , value_ : bav) ) ;
confirmed_ -> gx21 ;
```

Шаг 57. К формуле “*SPPA_uVars*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation -> ((formula -> SPPA_uVars) , (formula ->
f2) , interpretation_ : ((var_ : _a , value_ : av) , !var_ : _b , value_ : bv)) ;
f2 = [ av , bv ] ;
confirmed_ -> gx37 ;
```

Шаг 58. К высказыванию “*SPPA_effect*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation -> ((formula -> SPPA_effect) , (formula ->
f3) , interpretation_ : ((var_ : _a , value_ : av) , !var_ : _b , value_ : bv) , !var_ : _ba ,
value_ : bav) , !var_ : _aa , value_ : aav) , !var_ : _s , value_ : sv) , !var_ : _s , value_ :
sav))) ;
f3 = [ прямая -> sv ; sv -> av , bv ; ] ;
confirmed_ -> gx7 ;
```

Шаг 59. К высказыванию “*S__cause*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula →
  S__cause) , interpretation_ : ((var_ : _s , value_ : sv) , (var_ : _s , value_ : sav)) ,
  (formula → f4));
f4 = [ прямая → sv ; ];
confirmed_ → gx4;
```

Шаг 60. К высказыванию “*SA_uVars*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula →
  SA_uVars) , interpretation_ : ((var_ : _s , value_ : sv)) , (formula → f5));
f5 = [ sv ];
confirmed_ → gx027;
refuse → gx27;
```

Шаг 61. К высказыванию “*SP_uVars*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → SP_uVars) ,
  interpretation_ : ((var_ : _s , value_ : sv)) , (formula → f6));
f6 = [ sv ];
confirmed_ → gx35;
```

Шаг 62. К высказыванию “*SA_effect_uVars*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → SA_effect_uVars) , (formula →
  f7) , interpretation_ : ((var_ : _s , value_ : sv)));
f7 = [ sv ];
confirmed_ → gx026;
refuse → gx26;
```

Шаг 63. К высказыванию “*SA_effect_neg_atom*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → SA_effect_neg_atom) , (formula →
  f8) , interpretation_ : ((var_ : _s , value_ : sv) , !var_ : _sn , value_ : snv) , !var_ : _p ,
  value_ : pv) , !var_ : _pa , value_ : pav));
f8 = [ sv → pv ; точка → pv ; ];
confirmed_ → gx20;
refuse → gx30;
```

Шаг 64. К высказыванию “**SA_effect_atom**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → SA_effect_neg_atom) , (formula →
f8) , interpretation_ : ((var_ : s , value_ : sv) , !var_ : sa , value_ : sav) , !var_ : p ,
value_ : bv) , !var_ : pa , value_ : bav))) ;
f8 = [sv → bv ; точка → bv ;] ;
confirmed_ → gx19 ;
refuse → gx29 ;
```

Шаг 65. К высказыванию “**SP_effect_effect**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → SP_effect_effect) , (formula → f9) ,
interpretation_ : ((var_ : p , value_ : pv) , !var_ : pa , value_ : pav))) ;
f9 = [точка → pv ;] ;
confirmed_ → gx18 ;
```

Шаг 66. К высказыванию “**PSA_atom**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → SA_effect_neg_atom) , (formula → f10) ,
interpretation_ : ((var_ : p , value_ : pv) , !var_ : pa , value_ : pav))) ;
f10 = [точка → pv ;] ;
confirmed_ → gx5 ;
```

Шаг 67. К высказыванию “**PSA_atom**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_atom) , (formula → f11) ,
interpretation_ : ((var_ : p , value_ : pv) , !var_ : pa , value_ : pav))) ;
f11 = [точка → pv ;] ;
interpreted → gx5 ;
```

Шаг 68. К высказыванию “**PSA_cause**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_cause_uVars) , (formula → f12) ,
interpretation_ : ((var_ : p , value_ : pv))) ;
f12 = [pv] ;
interpreted → gx10 ;
```

Шаг 69. К высказыванию “**PSA_neg**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_neg_uVars) , (formula → f13) ,
interpretation_ : ((var_ : p , value_ : pv))) ;
f13 = [pv] ;
interpreted → gx11 ;
```

Шаг 70. К высказыванию “**PSA_neg_atom**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_neg_atom), (formula → f14),
interpretation_ : ((var_ : p, value_ : pv)));
f14 = [pv];
interpreted → gx15;
```

Шаг 71. К высказыванию “**PSA_neg_atom**” применяется операция расширенного ассоциирования атомарных формул. Будет построена временная конструкция:

```
temporary, associative → ((formula → PSA_neg_atom),
(theory → Планиметрия), (SA_effect_neg_atom, SP_effect_effect));
searched, confirm_, active_ → gx15;
```

Шаг 72. К атомарному высказыванию “**PSA_neg_atom**” применяется операция выявления идентичных, входящих и пересекающихся атомарных формул. Результатом применения операции будут генерируемые конструкции:

```
inclusion → ((formula → SA_effect_neg_atom),
in_ : (formula → PSA_neg_atom));
inclusion → ((formula → SP_effect_effect),
in_ : (formula → PSA_neg_atom));
```

Шаг 73. К атомарному высказыванию “**PSA_neg_atom**” применяется операция декомпозиции запроса на вывод простого высказывания на запросы к семантически близким простым высказываниям. В результате будут добавлены следующие конструкции:

```
reasoning → !effect_ : gx15, expectations_ : E40, causes_ : !gx19);
```

Шаг 74. К высказыванию “**PSA_neg_atom**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_neg_atom), (formula → f15),
interpretation_ : ((var_ : s, value_ : sv),
!var_ : sn, value_ : snv), !var_ : p, value_ : pv));
f15 = [sv →> pv];
denied_ → gx15;
```

Шаг 75. К высказыванию “**PSA_neg**” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_neg_uVars), (formula → f13),
interpretation_ : ((var_ : s, value_ : sv), !var_ : p, value_ : pv));
f13 = [pv, sv];
confirmed_ → gx11;
```

Шаг 76. К высказыванию “*PSA_cause*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_cause_uVars), (formula → f12));
interpretation_ : ((var_ : _s, value_ : sv), (var_ : _p, value_ : pv)));
f12 = [pv, sv];
interpreted → gx10;
```

Шаг 77. К высказыванию “*PSA_atom*” применяется операция формирования интерпретации формулы, которая на основании формул “*S__cause*” и “*PSA_cause*” формирует конструкцию:

```
interpretation → ((formula → PSA_cause_uVars), (formula → f11));
interpretation_ : ((var_ : _s, value_ : sav), (var_ : _s, value_ : sv),
(var_ : _pa, value_ : pav), (var_ : _p, value_ : pv)));
f11 = [прямая → sv; точка → pv];
confirmed_ → gx5;
```

Шаг 78. К высказыванию “*PSA_cause*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_cause_uVars), (formula → f12));
interpretation_ : ((var_ : _s, value_ : sv), (var_ : _p, value_ : pv));
f12 = [pv, sv];
confirmed_ → gx10;
```

Шаг 79. К высказыванию “*Аксиома о параллельной прямой*” применяется операция формирования интерпретации формулы, которая формирует конструкцию:

```
interpretation → ((formula → PSA_uVars), (formula → f16));
interpretation_ : ((var_ : _s, value_ : sv), (var_ : _p, value_ : pv));
f16 = [pv, sv];
confirmed_ → gx12;
```

Шаг 80. К высказыванию “*PSA_effect*” применяется операция формирования интерпретации формулы, которая формирует соответствующую конструкцию.

Шаг 81. К высказыванию “*PD_atom*” применяется операция формирования интерпретации формулы, которая формирует соответствующую конструкцию.

Шаг 82. К высказыванию “*qat_PDda_intersection*” применяется операция формирования интерпретации формулы, которая формирует соответствующую конструкцию.

Шаг 83. К высказыванию “*query_atom1*” применяется операция формирования интерпретации формулы, которая на основании формул “*S__cause*” и “*PSA_cause*” формирует соответствующую конструкцию.

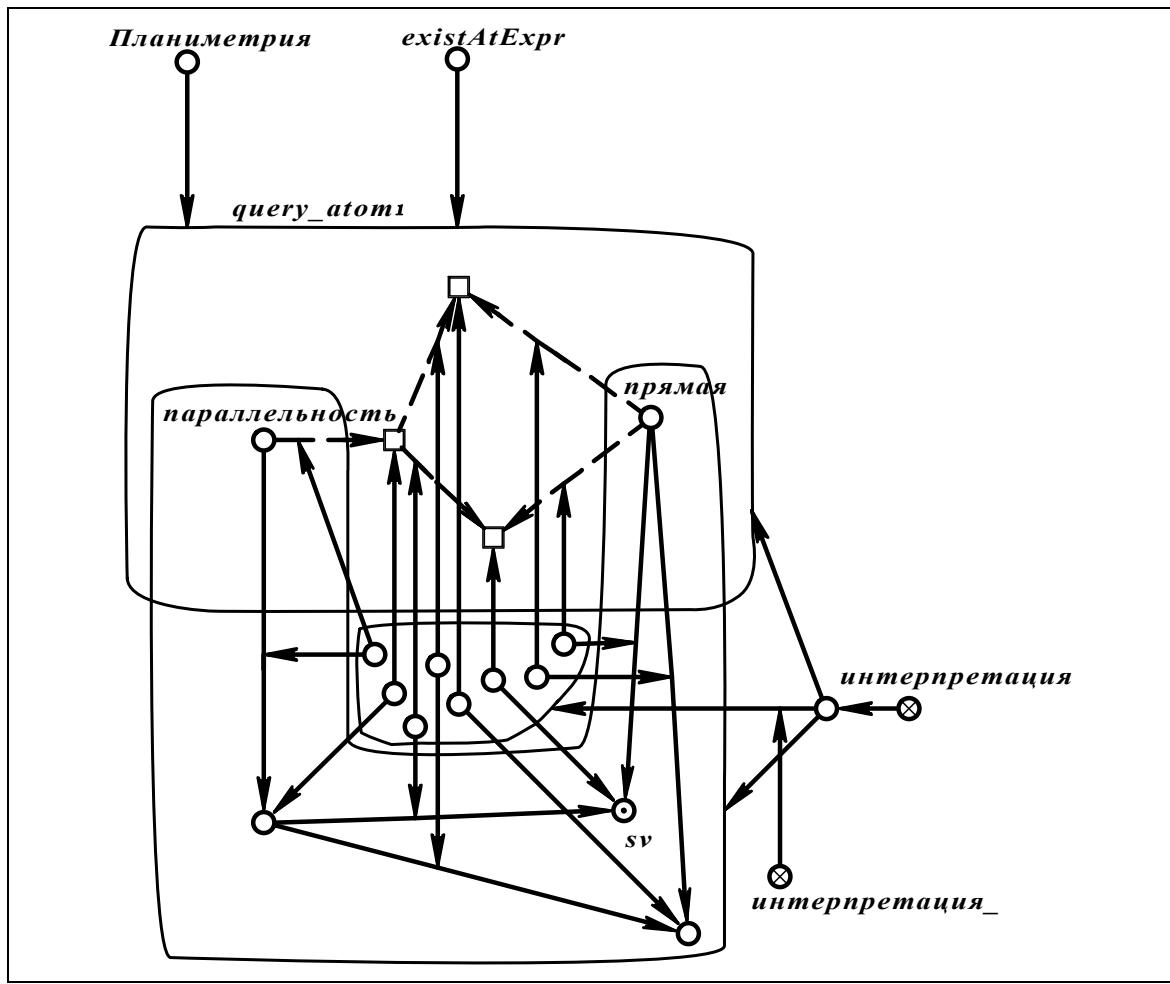
Шаг 84. К высказыванию “*query_atom1*” применяется операция вычисления значения высказывания. Высказывание приобретает истинностное значение *истина*:

```
Планиметрия → query_atom1;
```

Шаг 8.5. Ко всем высказываниям входящим во множество “*temporary*” применяется операция уничтожения промежуточных конструкций, которая удаляет интерпретации и различные служебные конструкции используемые исключительно для вывода.

Конец решения задачи

Приведём фрагмент результирующего состояния памяти графодинамической ассоциативной машины вывода, являющийся положительным ответом на поставленный нами запрос.



В процессе решения будет сформирована конструкция, поясняющая результат решения.

Выходы к разделу 8

Рассмотренная в данном разделе графодинамическая ассоциативная машина вывода является основой систем управления базами знаний, в которых информация представляется на языке SCL. Такие системы баз знаний являются неотъемлемой частью любой интеллектуальной системы. Это справедливо для интеллектуальных обучающих систем, систем построения виртуальных организаций и любых прикладных экспертных систем. Система операций машины логического вывода открыта для дополнения, и в случае реализации новых механизмов вывода, такие механизмы легко могут быть интегрированы (особенно в случае использования языка SCP в качестве языка микропрограммирования) с описанной выше графодинамической ассоциативной машиной вывода.

Заключение

Завершая данную книгу, сделаем ряд замечаний.

1. В книге акцент сделан не на разработку конкретного языка представления знаний и конкретной машины обработки знаний, а на разработку языка-ядра для целого семейства языков представления знаний, разработку ядра для целого семейства абстрактных машин обработки знаний, разработку принципов построения различных легко интегрируемых языков представления знаний и абстрактных машин обработки знаний. Это своего рода метаинформационный взгляд на принципы построения интеллектуальных систем.

Основными практическими результатами, описанными в данной книге, являются:

- фактографический язык SCB (Semantic Code Basic), описанный в разделах 2 и 3;
- язык SC (Semantic Code), являющийся основой для построения различных логических языков и языков представления знаний, описанный в разделе 4;
- язык SCL (Semantic Code Logic), являющийся одним из возможных (но не единственным возможным) логических языков, построенных на основе языка SC, описанный в разделах 5 и 6;
- язык описания целей (заданий) в графодинамических ассоциативных машинах, описанный в разделе 6;
- навигационно-поисковая графодинамическая ассоциативная машина, описанная в разделе 7;
- графодинамические ассоциативные машины вывода, описанные в разделе 8.

2. Аспекты реализации графодинамических ассоциативных машин обработки знаний рассмотрены в книге [411] ([ПрогрВАМ-2001кн](#)), которая является естественным продолжением данной книги.

В основе реализации графодинамических ассоциативных машин обработки знаний лежит язык программирования SCP (Semantic Code Programming), специально ориентированный на обработку информации в графодинамической (структурно перестраиваемой) ассоциативной памяти.

3. Применение графодинамических моделей обработки знаний в области автоматизации обучения рассмотрено в книге [236] ([ИнтелОСиВУО-2001кн](#)). В ней рассмотрены вопросы проектирования интеллектуальных обучающих систем и интеллектуальных виртуальных учебных организаций.

Литература

1. **Абдрахманов Б.К. 1990 ст-ИсслеВРА**
Абдрахманов Б.К. Исследование возможностей распараллеливания алгоритмов функционирования экспертизных систем и реализация их на однородной вычислительной среде //Методы и средства параллельной обработки данных в системах связи. - М.: Моск. ин-т связи, 1990. - с. 60-74.
2. **Авербух А.Б.. 1990 ст-оПринциПВя**
Авербух А.Б., Коваль В.Н., Рабинович З.Л. О принципах построения внутреннего языка ЭВМ, ориентированной на решение широкого класса задач искусственного интеллекта // II Всесоюзная конференция "Искусственный интеллект-90". - Минск, 1990. - Т. 2. - с. 121-124.
3. **Агафонов В.Н.. 1988 ст-ЛогичП**
Агафонов В.Н., Борщев В.Б., Воронков А.А. Логическое программирование в широком смысле (обзор) // Логическое программирование. - М.: Мир, 1988. - с. 298-366.
4. **Агафонов В.Н. 1982 ст- ТипыИАД**
Агафонов В.Н. Типы и абстракция данных в языках программирования (обзор) // Данные в языках программирования. Абстракция и типология. - М.: Мир, 1982. - с. 265-327.
5. **Агафонов В.Н.1990 кн-СпециП**
Агафонов В.Н. Спецификация программ: понятийные средства и их организация. 2-е издание. - Новосибирск: Наука, 1990.
6. **Агафонов В.Н.ред.1982 сб-АбстрИТ**
Данные в языках программирования. Абстракция и типология: Сб. статей / Пер. с англ. под ред. Агафонова В.Н. - М.: Мир, 1982.
7. **Агафонов В.Н.ред.1988 сб-ЛогичП**
Логическое программирование: Сб. статей / Пер. с англ. и франц.; Под ред. Агафонова В.Н. - М.: Мир, 1988.
8. **Агеев В.Н.1984 ст- ПримеГиГС**
Агеев В.Н. Примеры гипертекстовых и гипермедиа систем (обзор)// Компьютерные технологии в высшем образовании: Сб. статей. М.: Изд-во МГУ, 1994.-с.225-229.
9. **Айелло Л.. 1986 ст-ПредсИИМ**
Айелло Л., Чекки К., Сартини В. Представление и использование метазнаний // ТИИЭР. - 1986. - Т. 74. – N 10. - с. 12-32.
10. **Айзерман М.А.. 1988 ст-ДинамПкАС**
Айзерман М.А., Гусев Л.А., Петров С.В., Смирнова И.М., Тененбаум Л.А. Динамический подход к анализу структур, описываемых графами (основы графодинамики) // Исследования по теории структур. - М.: Наука, 1988. - с. 5-76.
11. **Айзерман М.А.. 1977 ст-ДинамПкАС**
Динамический подход к анализу структур, описываемых графиками / Айзерман М.А., Гусев Л.А., Петров С.В., Смирнова И.М. // Автоматика и телемеханика. - 1977. - N 7. - с.136-151; - N 8. - с. 123-136.
12. **Айлиф Дж. 1973 кн-ПринциПБМ**
Айлиф Дж. Принципы построения базовой машины / Пер. с англ. - М.: Мир, 1973.
13. **Акофф Р.. 1974 кн-оЦелеуc**
Акофф Р., Эмери Ф. О целеустремленных системах. Пер. с англ. - М.: Сов.радио, 1974.
14. **Александров В.В. 1992 ст-ВоспрВИ**
Александров В.В. Восприятие визуальной информации в системах искусственного интеллекта // Новости искусственного интеллекта. - 1992. - N 1. - с.25-45.
15. **Алешина Н.А.. 1990 кн-ЛогикИК**
Логика и компьютер. Моделирование рассуждений и проверка правильности программ / Н.А. Алешина, А.М. Анисов, П.И. Быстров и др.- - М.: Наука, 1990.
16. **Амамия М.. 1993 кн-АрхитЭВМ**
Амамия М., Танака Ю. Архитектура ЭВМ и искусственный интеллект: Пер. с япон.- М.: Мир, 1993.

17. **Амосов Н.М.. 1975 ст-АктивСС**
Амосов Н.М., Касаткин А.М., Касаткина Л.М. Активные семантические сети в роботах с автономным управлением // Труды IV Международной объединенной конференции по искусственному интеллекту: Т.9. - М., 1975.
18. **Анастази А. 1982 кн-ПсихоТ**
Анастази А. Психологическое тестирование. Т.1.М., 1982.
19. **Арбіб М. 1968 кн- МозгМиM**
Арбіб М. Мозг, машина и математика. - М.: Наука, 1968.
20. **Арбіб М.А. 1976 кн-МетафM**
Арбіб М.А. Метафорический мозг / Пер. с англ. - М.: Мир, 1976.
21. **Арлазаров В.Л.. 1998 ст-ТеорияИМСИКС**
Арлазаров В.Л., Журавлев Ю.И., Ларичев О.И., Лохин В.М., Макаров И.М., Рахманкулов В.З., Финн В.К. Теория и методы создания интеллектуальных компьютерных систем// Информационные технологии и вычислительные системы . - М, №1, 1998. - с.3-13.
22. **Асельдеров З.М. 1987 пр-ПредсГ**
Асельдеров З.М. Представление графов и операции над ними. - Киев, 1987. (Препринт / АН УССР. Ин-т кибернетики им. В.М.Глушкова; N 87-49).
23. **Асратян Р.Э.. 1979 ст-АппарРСА**
Асратян Р.Э., Волков А.Ф., Лысиков В.Т. Аппаратная реализация синтаксического анализатора с помощью ассоциативной памяти // Автоматика и телемеханика. - 1979. - N 4.
24. **Ахо А.. 1979 кн-ПострИАВА**
Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. - М.: Мир, 1979.
25. **Ачасова С.М.. 1990 кн-КоррепПВП**
Ачасова С.М., Бандман О.Л. Корректность параллельных вычислительных процессов. - Новосибирск: Наука. Сиб. отд-ние, 1990.
26. **Бабаев И.О.. 1981 ст- ЯзыкДекарт**
Бабаев И.О., Новиков Ф.А., Петрушина Т.И. Язык Декарт - входной язык системы СПОРА // Прикладная информатика. М.: Финансы и статистика, 1981. - Вып.1. - с. 35-73.
27. **Бабаян Б.А.. 1986 ст-УровенP**
Бабаян Б.А. Уровень программирования и архитектурные принципы построения ЭВМ // Кибернетика и вычислительная техника. М.: Наука, 1986. - Вып. 2. - с. 18-27.
28. **БазисРЕФАЛ-1977 уч**
Базисный РЕФАЛ и его реализация на вычислительных машинах. Методическое пособие М.: ЦИИПИАС, 1977.
29. **Байков А.М.. 1980 ст-УправКЦАИ**
Управляемый контекстом целенаправленный анализ изображений на основе использования семантической сети / Байков А.М., Кузин Е.С., Хахалин Г.К., Шамич А.П. // Вопросы радиоэлектроники. Сер. ЭВТ. - 1980. - Вып. 1. - с. 50-58.
30. **Бакаев А.А.. 1989 пр-ОболоЕС**
Бакаев А.А., Гриценко В.Н., Козлов Д.Н. Оболочка экспертной системы со смешанной стратегией логического вы-вода. - Киев, 1989. (Препринт / АН УССР Ин-т кибернетики им. В.М. Глушкова; N 89-15).
31. **Балашов Е.П.. 1978 кн-МногоРВС**
Многофункциональные регулярные вычислительные структуры/ Балашов Е.П., Смолов В.Б., Петров Г.А., Пузанков Д.В. - М.: Сов. радио, 1978.
32. **Бандман О.Л.. 1988 кн-СпециП**
Специализированные процессоры для высокопроизводительной обработки данных / Бандман О.Л., Миленков Н.Н., Седухин С.Г. и др. - Новосибирск: Наука, 1988.
33. **Бандман О.Л.. 1983 ст-оПаралA**
Бандман О.Л. О параллельных алгоритмах // Программирование. - 1983. - N 4. - с. 9-15.

34. **Бандман О.Л. 1984 ст-АсинхИПМ**
Бандман О.Л. Асинхронная интерпретация параллельных микропрограмм // Кибернетика. - Киев, 1984. - N 2. - с. 14-20.
35. **Бандман О.Л. ред. 1981 кн-МетодПМ**
Методы параллельного микропрограммирования / под ред. Бандман О.Л. - Новосибирск: Наука, 1981.
36. **Барздин Я.М.. 1973 ст-о ОдномЯПГ**
Барздин Я.М., Калниньш Я.Я. Об одном языке преобразования графов, ориентированном на задание автоматов // Авт- томат. и выч. техн. - Рига, 1973. - N 5. - с. 22-28.
37. **Барздин Я.М. 1982 ст-НекотПИВ**
Барздин Я.М. Некоторые правила индуктивного вывода и их применение // Семиотика и информатика. - 1982. - Вып. 19. - с. 59-89.
38. **Батыршин И.З. 1996 ст-МетодПиОНИ**
Батыршин И.З. Методы представления и обработки нечеткой информации в интеллектуальных системах // Новости искусств, интеллекта. - 1996. - №2. - С.9-65.
39. **Башлыков А.А.. 1995 ст-ЭксперС**
Башлыков А.А., Вагин В.Н., Еремеев А.П. Экспертные системы поддержки интеллектуальной деятельности операторов АЭС, Вестник МЭИ, М.: Изд-во МЭИ, 1995. - С. 27-36.
40. **Белецкая И.П. 1990 отч-ГиперС**
Белецкая И.П. Гипертекстовые системы. Современное состояние // Знания. Диалог. Решение. Часть I. Теоретические основы и методы. Отчёт РГ-26 КНВВТ. - Киев, 1990.
41. **Белецкий В.Н. 1988 кн-МногоИПС**
Белецкий В.Н. Многопроцессорные и параллельные структуры с организацией асинхронных вычислений. - Киев: Наукова думка, 1988.
42. **Белнап Н.. 1981 кн-ЛогикВиО**
Белнап Н., Стил Т. Логика вопросов и ответов. - М.: Прогресс, 1981.
43. **Белов В.В.. 1976 кн-ТеориГ**
Белов В.В., Воробьев В.М., Шаталов В.Е. Теория графов. - М.: Высшая школа, 1976.
44. **Белоглазов В.Н.. 1974 ст-СистеAP3**
Белоглазов В.Н., Скоробогатов В.А. Система автоматизации решения задач на графах // Вычислительные системы. Вып.69. Вопросы обработки информации при проектировании систем. - Новосибирск: Ин-т матем. СО АН СССР, 1974. - с. 24-39.
45. **Беляев И.П.. 1980 ст-КомбиП**
Беляев И.П., Капустян В.М., Медведев Б.Г. Комбинаторная память // Изв. АН СССР. Техн. кибернет. - 1980. - N 2. - с. 114-127.
46. **Бениаминов Е.М. 1979 ст-АлгебрП**
Бениаминов Е.М. Алгебраический подход к моделям баз данных реляционного типа // Семиотика и информатика. - 1979. - Вып. 14. - с. 44-80.
47. **Бениаминов Е.М. 1988 ст-ОсновыКП**
Бениаминов Е.М. Основания категорного подхода к представлению знаний. Категорные средства // Изв. АН СССР. Техн. кибернет. - 1988. - N 2. - с. 21-33.
48. **Берж К. 1962 кн-ТеориГ**
Берж К. Теория графов и ее применения. - М.: ИЛ, 1962.
49. **Берзтисс А.Т. 1974 кн-СтрукцД**
Берзтисс А.Т. Структуры данных. - М.: Статистика, 1974.
50. **Берков В.Ф. 1972 кн-ВопроКФМ**
Берков В.Ф. Вопрос как форма мысли. - Мн.: БГУ, 1972.
51. **Беркович С.Я.. 1975 ст-о ЭффекПАП**
Беркович С.Я., Кочин Ю.Я., Молчанов В.А. Об эффективности применения ассоциативной памяти в вычислительных процедурах // Вычислительные системы. - Новосибирск: ИМ СО АН СССР, 1975. - Вып. 62. - с. 97-105.

52. **Берштейн Л.С.. 1975 ст-ОднородПС**
Берштейн Л.С., Лисяк В.В., Рабинович В.А. Однородная программируемая структура для решения комбинаторно-логических задач на графах и гиперграфах // Методы расчета и автоматизация проектирования устройств в микроэлектронных ЦВМ. - Киев: ИК АН УССР, 1975. - с. 39-52.
53. **Берштейн Л.С.. 1988 ст-ИсполРОГ**
Берштейн Л.С., Башмаков Д.М. Использование расплывчатых ориентированных гиперграфов второго рода для представления фреймовых моделей // Методы построения алгоритмических моделей сложных систем. - Таганрог, 1988. - Вып. 7.- с. 64-68.
54. **Берштейн Л.С.. 1989 тез-НечетГ**
Берштейн Л.С., Башмаков Д.М. Нечеткий граф L-того рода - модель представления знаний в ЭВМ // Тезисы доклада научно-технической конференции молодых ученых и специалистов "Архитектура ЭВМ и машинное моделирование". - Таганрог, 1989. - с. 77-78.
55. **Бикешева Г.Р.. 1997 ст-ГенерНПП**
Бикешева Г.Р., Борисов А.Н. Генерация нечетких продукционных правил и функций принадлежности, основанная на использовании нейронных сетей // VI Междунар. конф. "Знание-Диалог-Решение" KDS-97: Сб. науч. тр. Т. I.- Ялта. -1997. -С.136-145.
56. **Богомолов С.Е.. 1993 ст-ПредсИОДЗ**
Богомолов С.Е., Янковский А.Г. Представление и обработка динамических знаний в экспертных системах // Изв. АН СССР. Техн. кибернет. - 1993. - N 5. - с. 17-23.
57. **Борисов А.Н.. 1989 кн-ОбрабНИ**
Обработка нечеткой информации в системах принятия решений / А.Н.Борисов, А.В.Алексеев, Г.В.Меркульева и др. - М.: Радио и связь, 1989.
58. **Борщев В.Б.. 1965 ст-АлгорЯП**
Борщев В.Б., Шрейдер Ю.А. Алгоритмы, языки программирования и диспозиции // Кибернетика. - 1965. - N 4. - с. 45-54.
59. **Борщев В.Б.. 1976 ст-КлубнС**
Борщев В.Б., Хомяков М.В. Клубные системы // Научно - техническая информация, сер. 2. - 1976. - N 8. - с. 3 - 6.
60. **Борщев В.Б. 1981 ст-ЛогичПкОРБД**
Борщев В.Б. Логический подход к описанию реляционных баз данных // Семиотика и информатика. - 1981. - Вып. 16. - с. 78-122.
61. **Борщев В.Б. 1986 ст-Пролог**
Борщев В.Б. Пролог - основные идеи и конструкции // Прикладная информатика. - 1986. - Вып. 2. - с. 49-76.
62. **Борщев В.Б. 1990 ст-СеманяЛП**
Борщев В.Б. Семантика языков логического программирования // Итоги науки и техники. Сер. Вычислительные науки. Том 4.- 1990.- с. 3-113.
63. **Борщев В.Б. 1993 ст-ПримеПП**
Борщев В.Б. Примеры параллельных программ (вегетативные методы программирования) // Семиотика и информатика. - 1993. - Вып. 33. - с. 234-255.
64. **Борщев В.Б. 1989 ст-ВегетМ**
Борщев В.Б. Вегетативная машина // Программирование. - 1989. - N 5. - с. 16-28.
65. **Борщев В.Б. 1983 ст-СхемыНКС**
Борщев В.Б. Схемы на клубных системах и вегетативная машина // Семиотика и информатика. - 1983. - Вып. 22. - с. 3-44.
66. **Борщев В.Б. 1990 ст-ПаралАВ**
Борщев В.Б. Параллельные асинхронные вычисления и моделирование параллельного выполнения логических программ (вегетативная модель вычислений и её применения) // Итоги науки и техники. Сер. Вычислительные науки. Том 4.- М.: ВИНИТИ, 1990. - с. 114-197.
67. **Бояров О.Д. 1980 пр- НовыеНвЯП**
Бояров О.Д. Новые направления в языках программирования для искусственного интеллекта. - Владивосток, 1980.(Препринт / ИАПУ ДВНЦ АН СССР).

68. **Братко И.1990кн-ПрогрНЯПролог**
Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. с англ. - М.: Мир, 1990.
69. **Братчиков И.Л.1975кн-СинтаксИП**
Братчиков И.Л. Синтаксис языков программирования. - М.: Наука, 1975.
70. **Бринтон Д.Б.1979ст-НоваяКА**
Бринтон Д.Б. Новая компьютерная архитектура, основанная на потоке данных / Пер. с англ. // Электроника. - 1979.- N 9. - с. 88-90.
71. **Брукс Ф.П. 1979кн-кПроекИСПК**
Брукс Ф.П. Как проектируются и создаются программные комплексы. Мифический человеко - месяц: очерки по системному программированию. М.: Наука, 1979.
72. **Брябрин В.М.1981ст-Ф-язык**
Брябрин В.М. Ф-язык - формализм для представления знаний в интеллектуальной диалоговой системе // Прикладная информатика / Сб. статей под ред. Савинкова В.М. - М.: Финансы и статистика, 1981. - Вып. 1. - с. 73-103.
73. **Брябрин В.М..1983кн-ДиалоС**
Диалоговые системы в АСУ / Брябрин В.М., Любарский Ю.А., Микулич Л.И. и др.; Под ред. Поспелова Д.А. - М.: Энергоатомиздат, 1983.
74. **Бублик В.В..1978ст-АлгебТСД**
Бублик В.В., Горюховский С.С. Алгебраическая трактовка структур данных. - Кибернетика. - 1978. - N 2. - с.10-15.
75. **Бублик В.В.1979ст-ФункцСО**
Бублик В.В. Функциональная семантика операторов преобразования структур данных // Программирование. - 1979. - N 3. - с. 44-53.
76. **Бублик В.В.1980ст-кОбоснО**
Бублик В.В. К обоснованию оператора анализа структур данных // Программирование. - 1980. - N 2. - с. 38-44.
77. **Бурбаки Н.1965кн-ТеориM**
Бурбаки Н. Теория множеств / Пер. с фр. под ред. Успенского В.А. - М.: Мир, 1965.
78. **Бухараев Р.Г..1986ст-оРазраAOС**
Бухараев Р.Г., Сулейманов Д.Ш. О разработке автоматизированных обучающих систем с интеллектуальными возможностями // Кибернетика. - 1986. - N 3. - с. 42-49.
79. **Буч Г. 1992кн-ОбъекОП**
Буч Г. Объектно-ориентированное проектирование с примерами применения. М.: Конкорд, 1992.
80. **Вагин В.Н.1989кн-ДедукИО**
Вагин В.Н. Дедукция и обобщение в системах принятия решений. - М.: Наука, 1989.
81. **Вагин В.Н..1997ст-РеалиКРИИ**
Вагин В.Н., Еремеев А.П. Реализация концепции распределенного искусственного интеллекта и многоагентности в системах поддержки принятия решений на базе инструментального комплекса G2+GDA // Proc. of the Internat. Workshop Distributed Artificial Intelligence Multi-Agent Systems "DAIMAS"97, June 15-18, 1997, St.-Peterburg, Russia, с. 262-268.
82. **Вагин В.Н..1985ст-ЗадачО**
Вагин В.Н., Викторова Н.П. Задачи обобщения в системах принятия решений: формирование классов объектов и отношений выбора на семантических сетях // Изв. АН СССР. Техн. кибернет. - 1985. - N 5. - с. 3-17.
83. **Вагин В.Н..1988ст-АбстрСПМ**
Вагин В.Н., Васильев М.Ю. Абстрактная схема параллельной машины логического вывода // Изв. АН СССР. Техн. кибернет. - 1988. - N 5. - с. 195-206.
84. **Вагин В.Н..1988ст-ПроекП**
Проект "Памир" / Вагин В.Н., Захаров В.Н., Поспелов Д.А. и др. // Изв. АН СССР. - 1988. - N 2. - с. 161-170.

85. **Вагин В.Н..1990ст-АрхитСЛВ**
Вагин В.Н., Васильев М.Ю. Архитектура спецпроцессора логического вывода // Вторая Всесоюзная конференция по искусственному интеллекту. - Минск, 1990. - с. 136-140.
86. **Вагин В.Н..1999ст-КонстИС**
Вагин В.Н., Еремеев А.П. Конструирование интеллектуальных систем поддержки принятия решений реального времени. - Третья международная конференция "Интеллектуальное управление: Новые интеллектуальные технологии в задачах управления" (1С1Т'99).Переславль-Залесский, 6-9 декабря 1999. - М.: Наука. Физматлит, 1999, с.27-32.
87. **Вагин В.Н.1986ст-ПаралД**
Вагин В.Н. Параллельная дедукция на семантических сетях // Изв. АН СССР. Техн. кибернет. - 1986. - N 5. - с.51-61.
88. **Вайнцвайг М.Н..1987ст-МеханМ**
Вайнцвайг М.Н., Полякова М.П. Механизм мышления и моделирование его работы в реальном времени // Интеллектуальные процессы и их моделирование. - М.: Наука, 1987. - с. 208-229.
89. **Вайнцвайг М.Н.1980пр-ОбучаС**
Вайнцвайг М.Н. Обучающаяся система искусственного интеллекта с ассоциативной памятью-процессором. - М., 1980.(Препринт / АН СССР, Научн. сов. по комплекс. пробл. "Кибернетика").
90. **Вальковский В.А..1990кн-ЭлемеСП**
Вальковский В.А., Малышкин В.Э. Элементы современного программирования в суперЭВМ. - Новосибирск: Наука. Сиб. отд-ние., 1990.
91. **Вальковский В.А..1983кн-ЭлемеПП**
Элементы параллельного программирования / Вальковский В.А., Котов В.Е., Марчук А.Г., Миренков Н.Н.; Под ред. Котова В.Е. - М.: Радио и связь, 1983.
92. **Варшавский В.И.1988ст-МоделДС**
Модели для спецификации и анализа асинхронных процессов в схемах / Варшавский В.И., Кишиневский М.А., Кондратьев А.Ю. и др. // Изв. АН СССР. Техн. кибернет. - 1988. - N 2. - с. 171-190.
93. **Варшавский В.И.ред.1986кн-АвтомУАП**
Автоматное управление асинхронными процессами в ЭВМ и дискретных системах / Под ред. В.И. Варшавского. - М.: Наука, 1986.
94. **Васильев В.В..1987кн-ЭлектрМ3**
Васильев В.В., Ралдугин Е.А. Электронные модели задач на графах. - Киев: Наукова думка, 1987.
95. **Ващенко Н. Д.1983ст-ФормиПвСС**
Ващенко Н. Д. Формирование понятий в семантической сети // Кибернетика. - 1983. - N 2. - С. 101-107.
96. **Вельбицкая О.Н.1990ст-оНекотОФСС**
Вельбицкая О.Н. О некоторых особенностях формирования семантической сети предметной области на основе автоматического анализа текста // II Всесоюзная конференция "Искусственный интеллект-90". 21-24 октября 1990 года. - Т. 1 - с. 56-59.
97. **Вертгеймер М.1987кн-ПродуМ**
Вертгеймер М. Продуктивное мышление. - М.: Прогресс, 1987.
98. **Видоменко В.П.1980ст-оСтрукИЯ**
Видоменко В.П. О структурной интерпретации языков информационно-логического программирования // Программирование. - 1980. - N 5. - с. 56-62.
99. **Виленкин Н.Я..1980кн-СовреOШКМ**
Современные основы школьного курса математики: Пособие для студентов пед. ин-тов/ Н.Я. Виленкин, К.И. Дуничев, Л.А. Калунин, А.А. Столляр.- М. Просвещение, 1980.
100. **Виленкин Н.Я.1969кн-РасскОМ**
Виленкин Н.Я. Рассказы о множествах. – 2-е изд. – М.: Наука, 1969.
101. **Виньков М.М.. 2000ст-ФормаP**
Виньков М.М., Фоминых И.Б. Формализация рассуждений с умолчаниями в интеллектуальных системах реального времени на основе временной логики //Сб. научных трудов VII Национальной конференции с международным участием "Искусственный интеллект - 2000", Переславль-Залесский, 2000.-т. I, с. 197-206.

102. **Вирт Н. 1989кн-АлгорИСД**
Вирт Н. Алгоритмы и структуры данных / Пер. с англ. - М.: Мир, 1989.
103. **Воеводин В.В. 1986кн-МатемМ**
Воеводин В.В. Математические модели и методы в параллельных процессах. - М.: Наука, 1986.
104. **Войшвилло Е.К.. 1974ст-ЯзыкИЛВ**
Войшвилло Е.К., Петров Ю.А. Язык и логика вопросов // Логика и методология научного познания. - М.: МГУ, 1974.- с. 147-158.
105. **Вольвачёв Р.Т. 1986уч-ЭлемеМЛ**
Вольвачёв Р.Т. Элементы математической логики и теории множеств: Учеб. Пособие. – Мин.: Изд-во «Университетское», 1986.
106. **Вольдман Г.Ш.. 1977пр-НекомС**
Вольдман Г.Ш., Задыхайло И.Б. Некоторые соображения об определении непроцедурности языков программирования. - М., 1977. (Препринт / ИПМ АН СССР; N 51).
107. **Вольдман Г.Ш. 1980ст-ОбъяснH**
Вольдман Г.Ш. Объяснение непроцедурности средствами теории категорий // Программирование. - 1980. - N 4.
108. **Вольфенгаген В.Э.. 1979ст-СистеPЗ**
Вольфенгаген В.Э., Воскресенская О.В., Горбанев Ю.Г. Система представления знаний с использованием семантических сетей// Вопросы кибернетики: Интеллектуальные банки данных. М.:АН СССР,1979.-с.49-69.
109. **Вольфенгаген В.Э.. 1988ст-КомпъЛ**
Вольфенгаген В.Э., Шаргатова З.И. Компьютерная логика на основе applicативных вычислительных систем // Машинная реализация систем искусственного интеллекта: сб. науч. трудов / под ред. Кузина Л.Т. - М.: Энергоатомиздат, 1988. - с. 21-26.
110. **Воронков А.А.. 1987ст-АвтомДТ**
Воронков А.А., Дегтярев А.И. Автоматическое доказательство теорем // Кибернетика. - 1986. - N 3. - с.27-33; - 1987. - N 4. - с. 88-96.
111. **ВысокВ-1988кн**
Высокоскоростные вычисления. Архитектура, производительность, прикладные алгоритмы и программы супер-ЭВМ. - М.: Радио и связь, 1988.
112. **Гаазе-Рапопорт М.Г.. 1987кн-оАмебыДР**
Гаазе-Рапопорт М.Г., Поспелов Д.А. От амебы до робота: модели поведения. - М.: Наука, 1987.
113. **Гаврилова Т.А.. 1992кн-ИзвлеИСЗ**
Гаврилова Т.А., Червинская К.Р. Извлечение и структурирование знаний для экспертных систем. М.: Радио и связь,1992.
114. **Гаврилова Т.А.. 1999ст-ВИКОНТ**
Гаврилова Т.А., Лещева И.А.. ВИКОНТ: Визуальный Конструктор ОНТологий для структурирования семантической информации// Труды Первой Всероссийской научной конференции "Электронные библиотеки: перспективные методы и технологии, электронные коллекции".СПб,1999.-с.97-98.
115. **Гаврилова Т.А.. 1999ст-АктивСкИСА**
Гаврилова Т.А., Котова Е.Е., Писарев А.С. Активные схемы как инструмент семантического анализа// Труды международного семинара "ДИАЛОГ 99" Таруса, 1999.-с.26-27.
116. **Гаврилова Т.А.. 2000кн-БазыЗИС**
Базы знаний интеллектуальных систем/ Т.А. Гаврилова, В.Ф. Хорошевский - СПб: Питер, 2000.
117. **Гаек П.. 1984кн-АвтомОГ**
Гаек П., Гавранек Т. Автоматическое образование гипотез: математические основы общей теории. / Пер. с англ. Финна В.К. и др.; Под ред. Поспелова Д.А. - М.: Наука, 1984.
118. **Галаган Н.И.. 1986ст-ИнтелРС**
Галаган Н.И., Рабинович З.Л. Интеллектуальные решающие системы // Кибернетика. - 1986. - N 3. - с. 18-26.

119. **Гастев Ю.А. 1975 кни-ГомомИМ**
Гастев Ю.А. Гомоморфизмы и модели. Логико-алгебраические аспекты моделирования. - М.: Наука, 1975.
120. **Георгиев В.О. 1993 ст-МоделПЗ**
Георгиев В.О. Модели представления знаний предметных областей диалоговых систем (обзор) // Изв. АН СССР. Техн. кибернет. - 1993. - N 5. - с. 24-44.
121. **Гергей Т.. 1986 ст-ПроектЛИВС**
Гергей Т., Поспелов Д.А. Проект ЛИВС - логическая информационно-вычислительная система // Изв. АН СССР. Техн. кибернет. - 1986. - N 5. - с. 128-138.
122. **Гильберт Д.. 1979 кни-ОсновыМ**
Гильберт Д., Бернаис П. Основание математики. Логические исчисления и формализация арифметики. - М.: Наука, 1979.
123. **Гладкий А.В.. 1961 кни-СхемаСЯ**
Гладкий А.В., Рыбакова М.В., Шедько Т.И. Схема семантического языка для записи математических текстов. - М.: ИНИ АН СССР, 1961.
124. **Гладкий А.В. 1973 кни-ФормаGия**
Гладкий А.В. Формальные грамматики и языки. - М.: Наука, 1973.
125. **Гладун В.П. 1987 кни-ПланиР**
Гладун В.П. Планирование решений. - Киев: Наукова думка, 1987.
126. **Гладун В.П. 1994 кни-ПроцеФНВ**
Гладун В.П. Процессы формирования новых знаний. - София: СД "Педагог 6", 1994.
127. **Гладун В.П. 1977 кни-ЭвристисПвСС**
Гладун В.П. Эвристический поиск в сложных средах. - Киев: Наукова Думка, 1977.
128. **Глушков В.М.. 1970 кни-ВычисМ**
Вычислительные машины с развитой системой интерпретации / Глушков В.М., Баранович А.А., Калиниченко Л.А., Михновский С.Д., Рабинович З.Л. - Киев: Наукова думка, 1970.
129. **Глушков В.М.. 1974 пр-РекуроМиВТ**
Глушков В.М., Игнатьев М.Б., Мяников В.А., Торгашев В.А. Рекурсивные машины и вычислительная техника. - Киев, 1974. (Препринт / ИК АН УССР; N 74-57).
130. **Глушков В.М.. 1978 кни-АлгебЯП**
Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. - К.: Наукова думка, 1978.
131. **Глушков В.М.. 1978 ст-оРазвиМЭВМ**
Глушков В.М., Погребинский С.Б., Рабинович З.Л. О развитии структур мультипроцессорных ЭВМ с интерпретацией языков высокого уровня // Управляющие машины и системы. - 1978. - N 6. - с. 61-66.
132. **Глушков В.М. 1978 ст-КиберИИ**
Глушков В.М. Кибернетика и искусственный интеллект // Кибернетика и диалектика. - М.: Наука, 1978. - с. 162 - 182.
133. **Глушков В.М.. 1980 кни-МетодСМ**
Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Методы символьной мультиобработки. - Киев: Наукова думка, 1980.
134. **Глушков В.М. 1980 ст-ФундаИ**
Глушков В.М. Фундаментальные исследования и технология программирования // Программирование. - 1980. - N 2. - с. 3-13.
135. **Глушков В.М.. 1981 ст-оПострСЯ**
Глушков В.М., Капитонова Ю.В., Летичевский А.А. О построении семейства алгоритмических языков для программирования и проектирования многопроцессорных вычислительных систем // Кибернетика. - 1981. - N 1. - с. 1-7.
136. **Голенков В.В., 1995 мо-ОписаЯSCP**
Голенков В.В., Гулякина Н.А., Елисеева О.Е. Описание языка SCP (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1995.

137. Голенков В.В.. 1998уч-МоделПиП3

Голенков В.В., Гулякина Н.А. Модели представления и переработки знаний. Методические указания. - Мин.: БГУИР, 1998.

138. Голенков В.В.. 1998ки-ЯзыкПП

Голенков В.В., Гулякина Н.А. Язык параллельного программирования, ориентированный на переработку сложноструктурированных знаний в структурно перестраиваемой ассоциативной памяти. – Мин.: БГУИР, 1998.

139. Голенков В.В.. 1989пр-ОбщаяХПис

Голенков В.В., Голенков Викт.В., Гарустович Л.В. Общая характеристика прикладных интеллектуальных систем. - Минск, 1989.(Препринт / Ин-т технической кибернетики АН БССР; N 2).

140. Голенков В.В.. 1992ст-АппарПГМ

Голенков В.В., Королев В.Г., Татаренко В.А. Аппаратная поддержка графовых моделей представления и переработки знаний // III конф. по искусственному интеллекту КИИ-92: Сб. науч. трудов.- Тверь, 1992.- Т.2.- С. 163-166.

141. Голенков В.В.. 1994мо-ОписаЯSCPас1

Голенков В.В., Гулякина Н.А., Королев В.Г., Татаренко В.А., Золотой С.А.. Описание семантики языка SCPas. Операторы преобразования состояния SC-графа (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1994.

142. Голенков В.В.. 1994мо-ОписаЯSCPас2

Голенков В.В., Гулякина Н.А., Татаренко В.А., Гапонов П.А., Кузьмицкий В.М.. Описание семантики языка SCPas. Операторы поиска и проверки условий. Операторы управления вычислительным процессом (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1994. – 98 с.

143. Голенков В.В.. 1994мо-ТермиМПГК

Голенков В.В., Павловский Р.М., Хижняк И.А., Космылев Л.В., Приходько Е.Ю. Терминальный модуль параллельного графового компьютера (PGC): Интерфейс с пользователем и процессорными модулями, структура, логическая организация (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1994.

144. Голенков В.В.. 1994пр-ОписаЯSCPас1

Голенков В.В., Гулякина Н.А., Королев В.Г., Татаренко В.А., Елисеева О.Е. Описание языка SCPas. Операторы преобразования состояния SC-графа.- Минск, 1994.(Препринт / Ин-т техн. кибернетики АН Беларуси; N 7).

145. Голенков В.В.. 1994пр-ОписаЯSCPас2

Голенков В.В., Гулякина Н.А., Кузьмицкий В.М., Малевич И.Е., Соловьев А.С. Описание языка SCPas. Операторы по- иска и проверки условий. - Минск, 1994.(Препринт / Ин-т техн. кибернетики АН Беларуси; N 9).

146. Голенков В.В.. 1994пр-ОписаЯSCPас3

Голенков В.В., Гулякина Н.А., Королев В.Г., Гапонов П.А., Ногина Н.Э.. Описание языка SCPas. Операторы управления вычислительным процессом. Примеры программ. - Минск, 1994. (Препринт / Ин-т техн. кибернетики АН Беларуси; N 11).

147. Голенков В.В.. 1994пр-ПредслВнЯSCL

Голенков В.В., Королев В.Г., Малевич И.Е. Представление логических высказываний на языке SCL - Semantic Code Logic. - Минск, 1994. (Препринт / Ин-т техн. кибернетики АН Беларуси; N 19).

148. Голенков В.В.. 1995мо-ОпераЯSCLδОПЗ

Голенков В.В., Королев В.Г., Елисеева О.Е. Операции языка SCL для обработки простых запросов (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1995.

149. Голенков В.В.. 1995мо-ОписаЯSCPbs

Голенков В.В., Гулякина Н.А., Малевич И.Е. Описание языка SCPbs (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1995.

150. Голенков В.В.. 1995мо-РешенНЯSCLЗиОГ

Голенков В.В., Королев В.Г., Елисеева О.Е. Решение на языке SCL задач из области геометрии (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1995.

151. Голенков В.В.. 1995пр-ПредсЗРВ

Голенков В.В., Королев В.Г., Малевич И.Е. Представление знаний различного вида на языке SCL - Semantic Code Logic. - Минск, 1995. (Препринт/ Ин-т техн. кибернетики АН Беларуси; N 4).

152. Голенков В.В.. 1995пр-ПредстМиАО

Голенков В.В., Королев В.Г., Елисеева О.Е., Малевич И.Е. Представление теоретико-множественных и арифметических отношений на языке SCL - Semantic Code Logic. - Минск, 1995. (Препринт / Ин-т техн. кибернетики АН; N 3).

153. Голенков В.В.. 1996мо-БазовПТЯSCL

Голенков В.В., Королев В.Г. Базовые преобразования текстов языка SCL для реализации механизмов дедуктивного логического вывода (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1996.

154. Голенков В.В. 1992ст-ИскуссИвРБ

Голенков В.В. Искусственный интеллект в Республике Беларусь // Новости искусственного интеллекта. - 1992. - N 1. - С.54-59.

155. Голенков В.В. 1993ст-ПострПИС

Голенков В.В. Построение прикладных интеллектуальных систем в мультитранспьютерной среде // Новости искусственного интеллекта. - 1993. - N 4. - С.128-148.

156. Голенков В.В. 1994мо-ОписаГЯSC

Голенков В.В. Описание графового языка SC (Материалы по математическому обеспечению ЭВМ). - Минск: Ин-т техн. кибернетики АН Беларуси, 1994.

157. Голенков В.В. 1994пр-ПаралГК

Голенков В.В. Параллельный графовый компьютер (PGC), ориентированный на решение задач искусственного интеллекта, и его применение. - Минск, 1994.(Препринт / Ин-т техн. кибернетики АН Беларуси; N 2).

158. Голенков В.В. 1996уч-ГрафоМ

Голенков В.В. Графодинамические методы и средства параллельной асинхронной переработки информации в интеллектуальных системах: Учебное пособие по специальности "Искусственный интеллект". - Минск: Бел. гос. ун-т информатики и радиоэлектроники, 1996.

159. Голицын Г.А.. 1996ст-ОтборПИ

Голицын Г.А., Фоминых И.Б. Отбор полезной информации в нейронной сети. // Динамические интеллектуальные системы в управлении и моделировании. Материалы семинара. - М., 1996, с.76-81.

160. Голицын Г.А.. 1996ст-НейроСиЭС

Голицын Г.А., Фоминых И.Б. Нейронные сети и экспертные системы: перспективы интеграции// Новости искусственного интеллекта.-М.,1996, №4.- с.121-145.

161. Головань Э.Т.. 1970ст-АссоцСМП

Головань Э.Т., Лук А.Н. Ассоциативно-сетевая модель памяти // Проблемы бионики. - Харьков: Харьковский университет, 1970. - Вып. 4. - с. 54-60.

162. Головкин Б.А. 1980кн-ПаралВС

Головкин Б.А. Параллельные вычислительные системы. - М.: Наука, 1980.

163. Гончаров С.С. 1985ст-ЛогичПвШС

Гончаров С.С. Логическое программирование в широком смысле // Вычислительные системы. - 1985 - с. 3 - 48.

164. Горбань А.Н. 1990кн-ОбучеHС

Горбань А.Н. Обучение нейронных сетей. - М.: ПароГраф, 1990.

165. Гордиенко Е.К.. 1985ст-УправПвБ3

Гордиенко Е.К., Захаров В.Н. Управление процессами в базах знаний // Изв. АН СССР. Техн. кибернет. - 1985. - N 5. - с. 175-193.

166. Гордиенко Е.К.. 1986ст-РеалиПАЛВ

Гордиенко Е.К., Захаров В.Н., Миронов А.Ю. Реализация параллельных алгоритмов логического вывода в матричной однородной среде // Изв. АН СССР. Техн. кибернет. - 1986. - N 5. - с. 153-197.

167. Гордиенко Е.К. 1988 ст-РеалиПП

Гордиенко Е.К., Кириллов В.Ю. Реализация поисковой процедуры языка ВОЛНА-0 в однородной вычислительной среде// Изв. АН СССР. Техн. кибернет. - 1988. - N 5. - с. 218-223.

168. Гордиенко Е.К. 1990 ст-ПолноИЭНО

Гордиенко Е.К., Захаров В.Н., Кириллов В.Ю., Миронов А.Ю. Полнота и эффективность навигационных операций в децентрализованных интеллектуальных системах // Изв. АН СССР. Техн. кибернет. - 1990. - N 5. - с. 212-220.

169. Гордиенко Е.К. 1986 ст-РеалиПФ

Гордиенко Е.К. Реализация поисковых функций языка FRL с применением двухтеговой ассоциативной памяти // Изв. АН СССР. Техн. кибернет. - 1986. - N 2. - с. 71-88.

170. Гордиенко Е.К. 1990 ст-АссоцПП

Гордиенко Е.К. Ассоциативные параллельные процессоры // Искусственный интеллект. Кн. 3. Программные и аппаратные средства. - М.: Радио и связь, 1990. - с. 235-249.

171. Гостев Ю.Г. 1981 ст-ОписаСД

Гостев Ю.Г. Описание структур данных с помощью граffопорождающих грамматик // Программирование. - 1981. - N 2. - с. 44-51.

172. Гостев Ю.Г. 1981 ст-ОписаСП

Гостев Ю.Г. Описание семантики программ с помощью подстановок на графах // Программирование. - 1981. - N 5. - с. 11-17.

173. Гoto A.. 1988 кн-ВысокМВ

Гото А., Танака Х., Мотоока Т. Высокопараллельный механизм вывода - модель трансформации целей и архитектура машины // Язык Пролог в пятом поколении ЭВМ. - М.: Мир, 1988.

174. Григорьев Д.Ю. 1976 ст-АлгорКСМ

Григорьев Д.Ю. Алгоритмы Колмогорова сильнее машин Тьюринга // Зап. научн. семинаров Ленинград. отд. математ. ин-та АН СССР. - 1976. - Вып. 60. - с. 29-37.

175. Григорьев Д.Ю. 1977 ст-ТеореВдМТ

Григорьев Д.Ю. Теоремы вложения для машин Тьюринга разных размерностей и алгоритмов Колмогорова // Докл. АН СССР. - 1977. - Т. 234. - N 1. - с. 15-18.

176. Гринченко Т.А. 1990 отч-ЯзыкДОТиГ

Гринченко Т.А. Язык для обработки текстов и гипертекстов // Знания. Диалог. Решение. Часть I. Теоретические основы и методы. Отчёт РГ-26 КНВВТ. - Киев, 1990.

177. Грис Д. 1984 кн-НаукаП

Грис Д. Наука программирования / Пер. с англ. под ред. А.П.Ершова. - М.: Мир, 1984.

178. Гросс М.. 1971 кн-ТеориФГ

Гросс М., Лантен А. Теория формальных грамматик. - М.: Мир, 1971.

179. Грундспенъкис Я.А. 1981 ст-оИсполМПВ

Грундспенъкис Я.А. Об использовании модели параллельных вычислений для исследования структур ориентированных графов // Микроэлектроника. - 1981. - Т. 10. - вып. 4. - с. 302-308.

180. Гудырева Е.М.. 1988 ст-АлгебРИСД

Гудырева Е.М., Цаленко М.Ш. Алгебра иерархических структур данных // Изв. АН СССР. Техн. кибернет. - 1988. - N 5. - с. 51-71.

181. Гуляев В.А.. 1982 кн-ОрганЖВС

Гуляев В.А., Додонов А.Г., Пелехов С.П. Организация живучих вычислительных структур. - Киев: Наукова думка, 1982.

182. Гуляева Д.М. 1989 мо-РешенПЗ

Гуляева Д.М. Решение прикладных задач на расширенных семантических сетях. // Математическое обеспечение ЭВМ и систем программирования. - М., 1989.

183. Гусаков В.Я.. 1981 ст-ДинамАС

Гусаков В.Я., Гусакова С.М. Динамические алгебраические системы как математическая модель банка данных // Семиотика и информатика. - 1981. - Вып. 17. - с. 43-52.

184. **Гусакова С.М. 1979 ст-Модели ПС**
Гусакова С.М. Моделирование информационно-поисковых систем алгебраическими моделями бинарного типа // НТИ. Сер. 2. - 1979. - N 2. - с. 1-5.
185. **Гусев Л.А.. 1968 ст-Языки Г**
Гусев Л.А., Смирнова И.М. Языки, грамматики и абстрактные автоматные модели (обзор) // Автоматика и телемеханика. - 1968. - N 4. - с. 72 - 94; - N 5. - с. 73 - 94.
186. **Данилов Г.А.. 1987 ст-Машин БД**
Данилов Г.А., Шмид А.В. Машины баз данных // Итоги науки и техники. - М.: ВИНИТИ. - 1987. - Т. 22. - с. 3-43.
187. **Дейт К. 1980 кн-Введение в СБД**
Дейт К. Введение в системы баз данных / Пер. с англ. - М.: Наука, 1980.
188. **Денисюк Ю.Н. 1982 ст-Некоторые проблемы голографии в трехмерных средах**
Денисюк Ю.Н. Некоторые проблемы и перспективы голографии в трехмерных средах. В кн.: Оптическая голография, под ред. Г.Колфилда, т.2, М.: Мир, 1982.
189. **Джорджеф М.П.. 1982 ст-Процез**
Джорджеф М.П., Лэнски Э.Л. Процедурные знания // ТИИЭР.- 1986. - Т. 74. - N 10. - с. 101-118.
190. **Джоунз Г. 1989 кн-Программирование на языке Оккам**
Джоунз Г. Программирование на языке Оккам: Пер. с англ. - М.: Мир, 1989.
191. **Дийкстра Э. 1972 кн-Взаимодействие последовательных процессов**
Дийкстра Э. Взаимодействие последовательных процессов // Языки программирования. - М.: Мир, 1972.
192. **Диринг М.Ф. 1987 ст-Архитектура машин для искусственного интеллекта**
Диринг М.Ф. Архитектура машин для искусственного интеллекта // Реальность и прогнозы искусственного интеллекта. - М.: Мир, 1987. - с. 209-230.
193. **Додонов А.Г.. 1979 ас-Устройство для исследования графов**
Додонов А.Г., Федотов В.В., Фёдоров В.Н., Хаджинов В.В., Шишмарев В.М. Устройство для исследования графов: а.с. 643880 (СССР) // Б. И. - 1979. - N 3.
194. **Донаху Дж. 1980 кн-Взаимодополняющие определения семантики языков программирования**
Донаху Дж. Взаимодополняющие определения семантики языков программирования // Семантика языков программирования. - М.: Мир, 1980.
195. **Доорс Дж.. 1990 кн-Пролог**
Доорс Дж., Рейблейн А.Р., Вадера С. Пролог - язык программирования будущего: Пер. с англ./ Предисловие А.Н. Волкова. - М.: Финансы и статистика, 1990.
196. **Дорфман Я.Г.. 1987 ст-Нейро ИММ**
Дорфман Я.Г., Сергеев В.М. Нейроморфогенез и модели мира в сетях нейронных процессоров // Интеллектуальные процессы и их моделирование. - М.: Наука, 1987. - с. 39-65.
197. **Дрейфус Х. 1978 кн-Чего не могут вычислительные машины**
Дрейфус Х. Чего не могут вычислительные машины. Критика искусственного разума / Пер. с англ.; Постскр. Б.В. Бирюкова. - М.: Прогресс, 1978.
198. **Дрибас В.П. 1982 кн-Реляционные модели баз данных**
Дрибас В.П. Реляционные модели баз данных. - Мин.: БГУ, 1982.
199. **Дунин-Борковский В.Л. 1978 кн-Информационные процессы в нейронных структурах**
Дунин-Борковский В.Л. Информационные процессы в нейронных структурах. - М.: Наука, 1978.
200. **Евреинов Э.В. 1981 кн-Однородные вычислительные системы, структуры и среды**
Евреинов Э.В. Однородные вычислительные системы, структуры и среды. - М.: Наука, 1981.
201. **Евстигнеев В.А. 1985 кн-Применение теории графов в программировании**
Евстигнеев В.А. Применение теории графов в программировании / Под ред. Ершова А.П.- М.: Наука, 1985.
202. **Емеличев В.А.. 1990 кн-Лекции по теории графов**
Лекции по теории графов / Емеличев В.А., Мельников О.М., Сарванов В.И., Тышкевич Р.И. - М.: Наука, 1990.

203. ***Епифанов М.Е. 1984ст-ИндукоВАС***
 Епифанов М.Е. Индуктивное обобщение в ассоциативных сетях// Известия АН СССР. Техническая кибернетика. №5, 1984.- с.132-146.
204. ***Еремеев А.П. 1998ст-ОрганСППР***
 Еремеев А.П. Организация систем поддержки принятия решений семиотического типа для динамических проблемных областей // Сборник докладов Международной конференции по мягким вычислениям и измерениям SCM'98. Санкт-Петербург, 22-26 июня 1998 г. Т.2.
205. ***Еремеев А.П. 1996ст-РеалиПСПР***
 Еремеев А.П., Симонов Д.Н., Чибизова Н.В. Реализация прототипа системы поддержки принятия решений реального времени на основе инструментального комплекса G2 // Программные продукты и системы, N 3, 1996, с.21-26.
206. ***Еремеев А.П. 1989ст-оРеалиППБЗ***
 Еремеев А.П. О реализации параллельной производственной базы знаний // ЭВМ новых поколений и перспективы их использования в народном хозяйстве. - М.: МДНТП, 1989. - с. 103-108.
207. ***Еремеев А.П. 1990ст-ПроблПвОЗ***
 Еремеев А.П. Проблемы параллелизма в обработке знаний // II Всесоюзная конференция "Искусственный интеллект-90". - Минск, 1990. - Т. 2. - с. 151-154.
208. ***Ершов А.П.ред.1992кн-АлгорМОиА***
 Алгоритмы, математическое обеспечение и архитектура многопроцессорных вычислительных систем / Отв. ред. Ершов А.П. - М.: Наука, 1982.
209. ***Ершов А.П. 1978ст-Смешав***
 Ершов А.П. Смешанные вычисления: потенциальные применения и проблемы исследования // Теория и практика программного обеспечения ЭВМ / Труды советско-французского симпозиума. - Новосибирск, 1978. - с. 5-40.
210. ***Ершов А.П. 1982ст-Вычис***
 Ершов А.П. Вычислимость в произвольных областях и базисах. // Семиотика и информатика. - 1982. - Вып. 19. - с. 3-58.
211. ***Ефимова С.М.. 1986ст-МоделПГ***
 Ефимова С.М., Суворов Е.В. Модель П-графов для представления знаний и способ ее аппаратной реализации на основе метода М // Изв. АН СССР. Техн. кибернет. - 1986. - N2.- с. 32-47.
212. ***Ефимова С.М.. 1988кн-ПоискВБЗ***
 Ефимова С.М., Суворов Е.В. Поиск в базах знаний, опирающихся на модель П-графов, и его аппаратная реализация на основе метода М 53 0 . - М.: ВЦ АН СССР, 1988.
213. ***Ефимова С.М. 1983ст-оОднойФМ***
 Ефимова С.М. Об одной формальной модели представления информации в базе данных // НТИ. Сер. 2. - 1983. - N 1.- с. 16-21.
214. ***Ефимова С.М. 1985кн- П-графы***
 Ефимова С.М. П-графы для представления знаний. - М.: ВЦ АН СССР, 1985.
215. ***Загорулько Ю.А. 1983ст-ПрогрС***
 Загорулько Ю.А. Программные средства для эффективного оперирования семантической сетью, допускающей вложенность // Методологические проблемы искусственного интеллекта. Уч. зап. Тартусского гос. универ. - Тарту, 1983. - N 654. - с. 101-110.
216. ***Загорулько Ю.А. 1987пр-ТехноКСОЗ***
 Загорулько Ю.А. Технология конструирования средств обработки знаний на основе семантических сетей. Общая схема и базовые средства. - Новосибирск, 1987. (Препринт / ВЦ СО АН СССР; N 749)
217. ***Загорулько Ю.А. 1988пр-ТехноКСОЗ***
 Загорулько Ю.А. Технология конструирования средств обработки знаний на основе семантических сетей. Средства спецификации и настройки. - Новосибирск, 1988. (Препринт / ВЦ СО АН СССР; N 793).
218. ***Заде Л. 1972кн-ЛингвП***
 Заде Л. Лингвистическая переменная. М.: Физматиз., 1972.

219. **Задыхайло И.Б.. 1981пр-Систек**
Задыхайло И.Б., Мельников Б.Ф., Садыхов Я.А. Система команд и алгоритмы реализации запросов для специального процессора реляционных баз данных. - М., 1981. (Препринт / ИПМ АН СССР; N 109).
220. **Замулин А.В. 1987кн-ТипыД**
Замулин А.В. Типы данных в языках программирования и базах данных. - Новосибирск: Наука, 1987.
221. **Замулин А.В. 1990кн-СистеПБД**
Замулин А.В. Системы программирования баз данных и знаний. - Новосибирск: Наука. Сиб. отд., 1990.
222. **Затуливер Ю.С.. 1981кн-ВопроПИМР**
Затуливер Ю.С., Медведев И.Л. Вопросы построения и многопроцессорной реализации языка структурно-параллельного программирования с управлением потоками данных // Вопросы кибернетики. Многопроцессорные вычислительные системы с перестраиваемой структурой (Архитектура. Структура. Применение) / Под ред. Прангисвili И.В. - М.: АН СССР, 1981. - с. 123-166.
223. **Захаров В.Н. 1981ст-ОсновНИ**
Захаров В.Н. Основные направления исследований в проекте "ПАМИР" // Материалы семинара "ЭВМ новых поколений и перспективы их применения в народном хозяйстве". - М.: МДНТП, 1989. - с. 20-28.
224. **Захаров В.Н.. ред. 1990спр-ИскуссИ-Кз**
Искусственный интеллект: В 3 кн. Кн. 3. Программные и аппаратные средства: Справочник /Под ред. В.Н. Захарова, В.Ф. Хорошевского. - М.: Радио и связь, 1990.
225. **Земляченко В.М.. 1982ст-ПроблИГ**
Земляченко В.М., Корнеенко Н.М., Тышкевич Р.И. Проблема изоморфизма графов // Зап. науч. семинаров ЛОМИ. - Л.: Наука. Ленингр. отд-ние, 1982. - Т. 118, ч. 1. - с. 83-152.
226. **Зенкин А.А. 1991кн-КогниКК**
Зенкин А.А. Когнитивная компьютерная графика. - М.: Наука, 1991.
227. **ЗнакоС-1990кн**
Знаковые системы в социальных и когнитивных процессах. - Новосибирск: Наука, 1990.
228. **Зыков А.А. 1969кн-ТеориКГ**
Зыков А.А. Теория конечных графов. - Новосибирск: Наука. Сиб. отд-ние, 1969.
229. **Зыков А.А. 1974ст-Гипер**
Зыков А.А. Гиперграфы // Успехи математических наук. - 1974. - Т. 29. - N 6. - с. 89-154.
230. **Зыков А.А. 1987кн-ОсновТГ**
Зыков А.А. Основы теории графов. - М.: Наука, 1987.
231. **Иванов В.В. 1972ст-БинарС**
Иванов В.В. Бинарные структуры в семантических системах// Системные исследования. Ежегодник. - 1972.
232. **Иванов Ю.Н. 1988кн-ТеорииО**
Иванов Ю.Н. Теория информационных объектов и системы управления базами данных. - М.: Наука, 1988.
233. **Иванс Д.ред. 1985кн-СистеPO**
Системы параллельной обработки / Под ред. Иванса Д. - М.: Мир, 1985.
234. **Игнатющенко В.В. 1981ст-кПостазПЭВС**
Игнатющенко В.В. К постановке задачи повышения эффективности вычислительных систем на основе ассоциативных методов обработки информации // Вопросы кибернетики. Многопроцессорные вычислительные системы с перестраиваемой структурой (Архитектура. Структура. Применение) / Под ред. Прангисвili И.В. - М.: АН СССР, 1981. - с. 14-21.
235. **Игнатьев М.Б. 1980кн-Введение ВТВП**
Игнатьев М.Б. Введение в теорию вычислительных процессов. - Л.: ЛЭТИ, 1980.
236. **ИнтелОСиВУО-2001кн**
Интеллектуальные обучающие системы и виртуальные учебные организации/ В.В.Голенков, В.Б.Тарасов, О.Е.Елисеева и др.; Под ред. В.В.Голенкова, В.Б.Тарасова.-Мн.:БГУИР,2001.

237. ИскусСИ-1990сб

Искусственный интеллект - основа новой информационной технологии. Сборник научных трудов./Под ред. Поспелова Д.А., Семенова Н.А. - Калинин, 1990.

238. Калиниченко Л.А.. 1990кн-МашинБДиЗ

Калиниченко Л.А., Рывкин В.М. Машины баз данных и знаний.- М.: Наука, 1990.

239. Калиниченко Л.А. 1983кн-МетодИСИ

Калиниченко Л.А. Методы и средства интеграции неоднородных баз данных. - М.: Наука, 1983.

240. Калужнин Л.А. 1973кн-ВведениеВОА

Калужнин Л.А. Введение в общую алгебру. – М.: Наука, 1973.

241. Каляев А.В. 1984кн-МногоG

Каляев А.В. Многопроцессорные системы с программируемой архитектурой. - М.: Радио и связь, 1984.

242. Кандрашина Е.Ю.. 1989кн-ПредсЗоВиП

Кандрашина Е.Ю., Литвинцева Л.В., Поспелов Д.А. Представление знаний о времени и пространстве в интеллектуальных системах. - М.: Наука, 1989.

243. Карпов В.Я. 1987ст-РазрабИИП

Карпов В.Я. Разработка и использование пакетов прикладных программ// Информатика и научно-технический прогресс. - М.: Наука, 1987. - с. 104-120

244. Кауфман В.Ш. 1993кн-ЯзыкиП

Кауфман В.Ш. Языки программирования. Концепции и принципы. - М.: Радио и связь, 1993.

245. Кахро М.И.. 1981кн-ИнструСП

Кахро М.И., Калья А.П., Тытугу Э.Х. Инstrumentальная система программирования ЕС ЭВМ (ПРИЗ). - М.: Финансы и статистика, 1981.

246. Кислюк О.С. 1983пр-ЯзыкOOГ

Кислюк О.С. Язык обработки ориентированных графов. - Владивосток, 1983. (Препринт / ИАПУ ДВНЦ АН СССР)

247. Клацки Р. 1978кн-ПамятЧ

Клацки Р. Память человека. Структуры и процессы: Пер.с англ./Под ред. Е.Соколова. - М.: Мир, 1978.

248. Клещев А.С.. 1982ст-СистеППОБ3

Клещев А.С., Черняховская М.Ю. Системы представления проблемно-ориентированных баз знаний. // Изв. АН СССР. Техн. кибернет. - 1982. - N 5. - с. 43-63.

249. Клещев А.С. 1980ст-РеляцМВ

Клещев А.С. Реляционная модель вычислений // Программирование. - 1980. - N 4. - с. 20-29.

250. Клещев А.С. ред. 1984сб-ЯзыкиПЗиВРЭС

Языки представления знаний и вопросы реализации экспертных систем: Сб. науч. тр. / Отв. ред. Клещев А.С. - Владивосток: ДВНЦ АН СССР, 1984.

251. Клещев А.С. 1986пр-СеманПМ

Клещев А.С. Семантические порождающие модели. Общая точка зрения на фреймы и продукции в экспертных системах. Владивосток, 1986. (Препринт / ИАПУ с ВЦ ДВНЦ АН СССР).

252. Клоксин У.. 1987кн-Прогр

Клоксин У., Меллиш К. Программирование на языке Пролог / Пер. с англ. - М.: Мир, 1987.

253. Клыков Ю.И.. 1980кн-БанкиД

Клыков Ю.И., Горьков Л.Н. Банки данных для принятия решений. - М.: Сов. радио, 1980.

254. Ковальски Р. 1990кн-ЛогикВРП

Ковальски Р. Логика в решении проблем: Пер. с англ. - М.: Наука, 1990.

255. Козинцев И.Б.. 1988ст-SIMD

Козинцев И.Б., Ильинский Н.И., Мальцев В.И. SIMD для ИЛИ-параллельного выполнения логических программ // Машинная реализация систем искусственного интеллекта: Сб. науч. тр. / Под ред. Кузина Л.Т. - М.: Энергоиздат, 1988. - с. 26-33.

256. Козырев В.П.. 1985кн-ТеориГ

Козырев В.П., Юшманов С.В. Теория графов (алгоритмические, алгебраические и метрические проблемы) // Итоги науки и техники. Сер. теория вероятностей, математическая статистика, теоретическая кибернетика. - М.: ВИ- НИТИ, - Т.23. 1985.

257. Кокун Л.М. 1975ст-Семанс

Кокун Л.М. Семантические сети как средство представления информации // Эвристические модели в психологии и социологии. Т.3. - Киев, 1975. - с. 18-26.

258. Колмогоров А.Н.. 1982кн-Введение ВМЛ

Колмогоров А.Н., Драгалин А.Г. Введение в математическую логику. – М.: Изд-во МГУ, 1982.

259. Колмогоров А.Н.. 1958ст-кОпредA

Колмогоров А.Н. К определению алгоритма // Успехи математических наук. - 1958. - Т.13. - N 4(82). - с. 3-28.

260. Кондрашена Е.Ю.. 1989кн-ПредсЗоВиП

Кондрашена Е.Ю., Литвинцева Л.В., Поспелов Д.А. Представление знаний о времени и пространстве в интеллектуальных системах / Под ред. Д.А.Поспелова. - М.: Наука, 1989.

261. Корнеев В.В. 1985кн-АрхитeВС

Корнеев В.В. Архитектура вычислительной системы с программируемой структурой. - Новосибирск: Наука. Сиб.отд-ние, 1985.

262. Котов В.Е.. 1966ст-АсинхВП

Котов В.Е., Нариняни А.С. Асинхронные вычислительные процессы над общей памятью // Кибернетика. - 1966. - N 3. - с. 64-71.

263. Котов В.Е.. 1991кн-ТеориСП

Котов В.Е., Сабельфельд В.К. Теория схем программ. - М.:Наука, 1991.

264. Котов В.Е. 1974ст-ТеориПП

Котов В.Е. Теория параллельного программирования. Прикладные аспекты // Кибернетика. - 1974. - N 1. - с. 1-16.

265. Котов В.Е. 1979пр-ФормамПВ

Котов В.Е. Формальные модели параллельных вычислений. - Новосибирск, 1979. (Препринт / ВЦ СО АН СССР; N 165).

266. Котов В.Е. 1980ст-оПараля

Котов В.Е. О параллельных языках // Кибернетика. - 1980. - N 3. - с. 1-12; - N 4. - с. 1-10.

267. Котов В.Е. 1991ст-МАРС

Котов В.Е. МАРС: Архитектуры и языки для реализации параллелизма, модульности и программируемости // Системная информатика. Вып. 1. Проблемы современного программирования. - Новосибирск: Наука. Сиб. отд-ние, 1991. - с. 174-194.

268. Коул Б.К. 1988ст-РазвиР

Коул Б.К. Развитие разработок и производства ассоциативных ЗУ // Электроника. - 1988. - N 12. - с. 31-36.

269. Кохонен Т. 1980кн-АссоцП

Кохонен Т. Ассоциативная память. - М.: Мир, 1980.

270. Кохонен Т. 1982кн-АссоциЗУ

Кохонен Т. Ассоциативные запоминающие устройства. - М.: Мир, 1982.

271. Кочин Ю.Я. 1970ст-ЛогичААСП

Кочин Ю.Я. Логический анализ ассоциативных структур памяти // Кибернетика. - 1970. - N 4. - с. 135-137.

272. Краснов С.А. 1990ст-Транс

Краснов С.А. Транспьютеры, транспьютерные вычислительные системы и Оккам // Вычислительные процессы и системы. - М.: Наука, 1990. - Вып. 7. - с. 3-93.

273. Кристальный Б.В.. 1985ст-оИнтел

Кристальный Б.В., Войскунский В.Г., Раскина А.А., Сидоров И.С., Шарыгин В.И. Об интеллектуальности автоматизированных фактографических ИПС // НТИ. Сер.2. - 1985. - N 12. - с.4-11.

274. Кристофицес Н. 1978 кн-ТеориГ

Кристофицес Н. Теория графов. Алгоритмический подход / Пер. с англ. Вершкова Э.В., Коновальцева И.В. - М.: Мир, 1978.

275. Касабов Н.. 1989 ст-ПаралКОБ3

Касабов Н., Тришина Е. Параллельно-конвейерная обработка баз знаний, основанных на системах продукции, модель и ее реализация на транспьютерных сетях // 2 Международный семинар "Теория и применение искусственного интеллекта". - Созополь, 1989. - с. 187-193.

276. Кузнецов В.Е. 1989 кн-ПредсВЭВМНП

Кузнецов В.Е. Представление в ЭВМ неформальных процедур. М.: Наука, 1989.

277. Кузнецов О.П. 1995 ст-НеклаПвИИ

Кузнецов О.П. Неклассические парадигмы в искусственном интеллекте// Известия РАН. Теория и системы управления. - М: Наука, 1995 - № 5.- С. 76-84.

278. Кузнецов О.П. 1998 ст-ОбразМиБП

Кузнецов О.П. Образное мышление и быстрые процессы // Новости искусств. интеллекта. - 1998. - №2. - С. 117-130.

279. Кузнецов О.П.. 1988 кн-ДискрМдИ

Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – 2-е изд. – М.: Энерготомиздат, 1988.

280. Кузнецов О.П.. 2000 ст-ПсевдНС

Кузнецов О.П., Шипилина Л.Б. Псевдооптические нейронные сети - полная прямолинейная модель и методы расчета ее поведения. //Теория и системы управления, 2000г., №5, с.168-176.

281. Кузнецов О.П. 1992 ст-ГологМОИвНС

Кузнецов О.П. Голографические модели обработки информации в нейронных сетях. Докл. АН, 1992, т.324, № 3.

282. Кузнецов О.П. 1993 ст-МоделГПОИвНС

Кузнецов О.П. Модели голографических процессов обработки информации в нейронных сетях. Автоматика и телемеханика, 1993, № 7.

283. Кузнецов О.П. 1995 ст-оПовышЭООИ

Кузнецов О.П., Шипилина Л.Б. О повышении эффективности обработки образной информации в псевдооптических нейронных сетях//Труды седьмой национальной конференции по искусственному интеллекту, 2000.-с.578-587.

284. Кузнецов О.П. 1996 ст-ПсевдНСПМ

Кузнецов О.П. Псевдооптические нейронные сети - прямолинейные модели. Автоматика и телемеханика, 1996, № 12.

285. Кузнецов С.О.. 1988 ст-РаспрПЭС

Кузнецов С.О., Финн В.К. Распространение процедур экспертизных систем типа ДСМ на графы // Изв. АН СССР. Техн. кибернет. - 1988. - N 5. - с. 4-11.

286. Кузнецов В.Е. 1989 кн-ПредсВЭВМ

Кузнецов В.Е. Представление в ЭВМ неформальных процедур: продукционные системы. - М.: Наука, 1989.

287. Кузнецов И.П. 1978 кн-МеханОСИ

Кузнецов И.П. Механизмы обработки семантической информации. - М.: Наука, 1978.

288. Кузнецов И.П. 1986 кн-СеманП

Кузнецов И.П. Семантические представления. - М.: Наука, 1986.

289. Куратовский К.. 1970 кн-ТеориM

Куратовский К., Мостовский А. Теория множеств / Пер. с англ. - М.: Мир, 1970.

290. Курочкин В.М. ред. 1989 кн-СеманЯП

Семантика языков программирования. Сб. статей: Пер. с англ. /под ред. Курочкина В.М. - М.: Мир, 1989.

291. **Лавров С.С. 1982кн-ОсновыП**
Лавров С.С. Основные понятия и конструкции языков программирования. - М.: Финансы и статистика, 1982.
292. **Лавров С.С. 1982ст-СинтезП**
Лавров С.С. Синтез программ // Кибернетика. - 1982. - с. 11-16.
293. **Лавров С.С. 1984ст-Расширимость**
Лавров С.С. Расширимость языков. Подходы и практика // Прикладная информатика. - М.: Финансы и статистика, 1984. - Вып. 2. - с. 17-22.
294. **Ладенко И.С.. 1988кн-ЛогикаЦУ**
Ладенко И.С., Тульчинский Г.Л. Логика целевого управления. - Новосибирск: Наука. Сиб. отд-ние, 1988.
295. **Ларичев О.И.. 1987ст-СистемыПР**
Ларичев О.И., Петровский А.Б. Системы поддержки принятия решений: современное состояние и перспективы развития//Итоги науки и техники. Техн. кибернетика.- 1987.- т.21.- с.131-165.
296. **Ларичев О.И.. 1996кн-КачесМПР**
Ларичев О.И., Мошкович Е.М. Качественные методы принятия решений. Вербальный анализ решений.- М.:Наука. Физматлит, 1996.
297. **Левин Д.Я. 1976ст-СЕТЛ**
Левин Д.Я. СЕТЛ - язык программирования весьма высокого уровня // Программирование. - 1976. - N 5. - с. 3-9.
298. **Левин Д.Я. 1983кн-ЯзыкСУ**
Левин Д.Я. Язык сверхвысокого уровня СЕТЛ и его реализация (для ЭВМ БЭСМ-6). - Новосибирск: Наука, 1983.
299. **Левин Д.Я. 1987кн-ИнструКП**
Левин Д.Я. Инstrumentальный комплекс программирования на основе языков высокого уровня / Под ред. А.П.Ершова. - М.: Наука, 1987.
300. **Лельчук Т.И. 1990ст-ПаралВОЯ**
Лельчук Т.И. Параллельные возможности объектно-ориентированных языков // Программирование. - 1990. - N 6. - с.33-45.
301. **Лельчук Т.И. 1991ст-ПаралООЯ**
Лельчук Т.И. Параллельные объектно-ориентированные языки // Системная информатика. Вып. 1. Проблемы современного программирования. - Новосибирск: Наука. Сиб. отд-ние, 1991. - с. 195-229.
302. **Лефевр В.А. 1973кн-КонфлС**
Лефевр В.А. Конфликтующие структуры. - М.: Сов. радио, 1973.
303. **Ливчак А.Б. 1984ст-о СеманСЯЗ**
Ливчак А.Б. О семантической силе языка запросов // НТИ, сер. 2. - 1984. - N 2. - с. 30-31.
304. **Лимантов Ф.С. 1971ст-о ПриродВ**
Лимантов Ф.С. О природе вопроса// Вопрос. Мнение. Человек. Уч. Записки ЛГПИ им.Герцена. Т.497, 1971.-с.4-20.
305. **Линдсей П.. 1974кн-ПерерИ**
Линдсей П., Норман Д. Переработка информации у человека.- М.: Мир, 1974.
306. **Лисков Б.. 1989кн-ИсполА**
Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ: Пер. с англ. - М.: Мир, 1989.
307. **Лозовский В.С. 1978пр-ЗаданРБД**
Лозовский В.С. Задание реляционной базы данных в виде мультисети и реализация поиска по образцу // Информационное и программное обеспечение систем ситуационного управления. - Киев, 1978. (Препринт / ИКАН УССР; N 78-14).
308. **Лозовский В.С. 1979ст-СеманС**
Лозовский В.С. Семантические сети // Представление знаний в человеко-машинных и робототехнических системах. т.А. - М.: ВЦ АН СССР, ВНИТИ,1979. - с. 84-121.

309. **Лозовский В.С. 1979 ст-СитуаИДС**
Лозовский В.С. Ситуационная и дефиниторная семантика системы представления знаний // Кибернетика. - 1979. - N 2. - с. 98-101.
310. **Лозовский В.С. 1982 ст-ЭкстенБ3**
Лозовский В.С. Экстенсиональная база знаний на основе семантических сетей // Известия АН СССР. Техн. кибернет. - 1982. - N 5. - с. 23-42.
311. **Лузин Н.Н. 1958 кн-СобрАС**
Лузин Н.Н. Собрание сочинений. – М.: Изд. АН СССР, 1958.
312. **Лупичев Л.Н. ред. 1992 сб-ОбрабДИ**
Обработка динамической информации в интеллектуальных системах: Сб. науч. тр.; Под ред. акад. Лупичева Л.Н. - М.: Инт-физ.-техн. пробл., 1992.
313. **Любарский Ю.Я. 1980 кн-ИнтелИС**
Любарский Ю.Я. Интеллектуальные информационные системы. - М.: Наука, 1980.
314. **Майерс Г. 1985 кн-АрхитСЭВМ**
Майерс Г. Архитектура современных ЭВМ: в 2-х книгах / Пер. с англ. - М.: Мир, 1985.
315. **Маллас Дж. 1990 кн-РеляцЯПролог**
Маллас Дж. Реляционный язык Пролог и его применение: Пер. с англ. /Под ред. В.Н. Соболева. - М.: Наука, 1990.
316. **Мальцев А.И. 1970 кн-АлгебРС**
Мальцев А. И. Алгебраические системы. – М.: Наука, 1970.
317. **Манин Ю.И. 1979 кн-ДоказИН**
Манин Ю.И. Доказуемое и недоказуемое. - М.: Сов. радио, 1979.
318. **Мануель Т. 1988 ст-Нейро**
Мануэль Т. Нейропроцессор, обеспечивающий возможность вычислений в реальном масштабе времени на персональных компьютерах / Пер. с англ. // Электроника. - 1988. - т.61. - N 3. - с. 73-74.
319. **Мануель Т. 1988 ст-РазвиРК**
Мануэль Т. Развитие рынка компьютеров, ориентированных на языки ЛИСП и ПРОЛОГ / Пер. с англ. // Электроника. - 1988. - т.61. - N 4. - с. 57-65.
320. **Маркевичус Р. 1977 ст-ЯзыкПдОГ**
Маркевичус Р. Язык программирования для обработки графов // Автоматизация процессов планирования и управления. - 1977. - Вып. 5. - с. 9-30.
321. **Мартин Дж. 1980 кн-ОрганБД**
Мартин Дж. Организация баз данных в вычислительных системах / Пер. с англ. - М.: Мир, 1980.
322. **Мартынов В.В. 1989 ст-УСК-4**
Мартынов В.В. УСК-4 - особого рода язык представления и преобразования знаний // Изв. АН СССР. Техн. кибернет. - 1989. - N 5. - с. 71-75.
323. **Мартынов С.А. 1990 ст-СеманН**
Мартынов С.А. Семантические нейросети: средство построения интеллектуальных систем // II Всесоюзная конференция "Искусственный интеллект-90". Круглые столы. - Минск, 1990. - с. 23-267.
324. **Марчук Г.И.. 1979 ст-ПроблВТ**
Марчук Г.И., Котов В.Е. Проблемы вычислительной техники и фундаментальные исследования // Автоматика и вычислительная техника. - 1979. - N 2. - с. 3-14.
325. **Маслов С.Ю. 1986 кн-ТеоридС**
Маслов С.Ю. Теория дедуктивных систем и ее применение. М.: Радио и связь, 1986.
326. **МатемИАОПВ-1985 сб**
Математическое и архитектурное обеспечение параллельных вычислений (Вычислительные системы, 109): Сб. науч. тр./ Под ред. Миренкова Н.Н., Косатева Ю.Г. - Новосибирск: ИМ СО АН СССР, 1985.
327. **МатемЛвП-1991 сб**
Математическая логика в программировании: Сб. статей / Пер. с англ. - М.: Мир, 1991.

328. *МашинРСИИ-1988сб*

Машинная реализация систем искусственного интеллекта: Сб. науч. тр. / Под ред. Кузина Л.Т. - М.: Энергоатомиздат, 1988.

329. *Мейер Б..1982кн-МетодП*

Мейер Б., Бодуэн К. Методы программирования. В 2-х томах. - М.: Мир, 1982.

330. *Мейер Д.1987кн-ТеориРБД*

Мейер Д. Теория реляционных баз данных / Пер. с англ. Под ред. Цаленко М.Ш. - М.: Мир, 1987.

331. *Мейер Д.1997кн-ТеориРБД*

Мейер Д. Теория реляционных баз данных: Пер. с англ. – М.: Мир, 1997.

332. *Мейтус В.Ю..1977пр-СтрукЛА*

Мейтус В.Ю., Танчер И.В. Структурно-алгоритмические алгебры и параллельные вычисления. - Киев, 1977.(Препринт / ИК АН УССР; N 77-11).

333. *Мелихов А.Н..1975кн-РазраНИОС*

Мелихов А.Н., Берштейн Л.С., Лисяк В.В., Рабинович В.А. Разработка и исследование однородной структуры для решения комбинаторно-логических задач на графах и гиперграфах // Однородные вычислительные системы и среды. Часть 1. - Киев: Наукова думка, 1975.

334. *Мелихов А.Н..1977ст-ОперанГ*

Мелихов А.Н., Берштейн Л.С. Операции над гиперграфами и их свойства // Изв. АН СССР. Техн. кибернетики N 4.. - 1977.

335. *Мелихов А.Н..1990кн-СитуаCC*

Мелихов А.Н., Берштейн Л.С., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой. - М.: Наука, 1990.

336. *Мельчук И.А.1974кн- ОпытТЛМ*

Мельчук И.А. Опыт теории лингвистических моделей "Смысл - Текст". Семантика, синтаксис. М.:Наука, 1974.

337. *Месарович М..1973кн-ТеориИМС*

Месарович М., Мако Д., Такахара Я. Теория иерархических многоуровневых систем. - М.: Мир, 1973.

338. *Месарович М..1978кн- ОбщаяTC*

Месарович М., Такахара Я. Общая теория систем: математические основы. М.:Мир, 1978.

339. *Метлицкий Е.А..1989кн-СистемПП*

Метлицкий Е.А., Каверзnev В.В. Системы параллельной памяти: Теория, проектирование, применение / Под ред. Тимохина В.И. - Л.: Изд-во Ленинградского университета, 1989.

340. *Минский М.1979кн-ФреймДПЗ*

Минский М. Фреймы для представления знаний. - М.: Энергия, 1979.

341. *Минский М.1978кн-СтрукДПЗ*

Минский М. Структура для представления знаний / Пер. с англ. под ред. Уинстона П. - М.: Мир, 1978.

342. *Минский_М.1971кн-ВычисИА*

Минский М. Вычисления и автоматы. - М.: Мир, 1971.

343. *Миренков Н.Н.1989кн-ПаралП*

Миренков Н.Н. Параллельное программирование для многомодульных вычислительных систем. - М.: Радио и связь, 1989.

344. *Митчелл Д.А..1993кн-ВнутрT*

Митчелл Д.А., Томпсон Дж.А., Мансон Г.А., Брукс Г.Р. Внутри транспьютера: Пер. с англ.- М.: Мейкер, 1993.

345. *Мишин А.И..1979ст-ВычисC*

Мишин А.И., Седухин С.Г. Вычислительные системы и параллельные вычисления с локальными взаимодействиями. // Вычислительные системы (Новосибирск). - 1979. - N 78. - с.90-103.

346. **Мищенко В.А.. 1988кн-ИнтелСАП**

Интеллектуальные системы автоматизированного проектирования больших и сверхбольших интегральных микросхем / Мищенко В.А., Городецкий Л.М., Гурский Л.И. и др.; Под ред. Мищенко В.А. - М.: Радио и связь, 1988.

347. **Молокова О.С. 1992ст-МетодАП3**

Молокова О.С. Методология анализа предметных знаний // Новости искусственного интеллекта. - 1992. - N 3. - с.11-60.

348. **Мотоока Т.ред. 1984кн-ЭВМПП**

ЭВМ пятого поколения: Концепции, проблемы, перспективы / Под ред. Мотоока Т.; Пер. с англ.; Предисл. Велихова Е.П. - М.: Финансы и статистика, 1984.

349. **Мотоока Т. 1988ст-ЯпонсПВС**

Мотоока Т. Японский проект вычислительных систем пятого поколения // Высокоскоростные вычисления. Архитектура, производительность, прикладные алгоритмы и программы су- пер-ЭВМ. - М.: Радио и связь, 1988. - с. 90-104.

350. **Наринъяни А.С.. 1987ст-ПростСУА**

Наринъяни А.С., Загорулько Ю.А. Простые средства управления активацией для производственных систем // Информатика. Технологические аспекты. - Новосибирск: ВЦ СО АН СССР, 1987. - с. 118-126.

351. **Наринъяни А.С.. 1994пр-Неточч**

Наринъяни А.С. Неточность как НЕ-фактор. Попытка доформального анализа. - Москва-Новосибирск, 1994. (Препринт / ВЦ СО АН СССР; N 2).

352. **Наринъяни А.С. 1974ст-ТеориПП**

Наринъяни А.С. Теория параллельного программирования. Формальные модели // Кибернетика. - 1974. - N 3. - с.1-15; N 5. - с. 1-14.

353. **Наринъяни А.С. 1982ст-СистеП**

Наринъяни А.С. Системы продукции как модульный программный комплекс // Прикладные и экспериментальные лингвистические процессы. - Новосибирск, 1982. - с.125-152.

354. **Наринъяни А.С. 1986кн-ПаралОЗ**

Наринъяни А.С. Параллельность обработки знаний и технологии виртуальных машин // Разработка ЭВМ нового поколения: архитектура, программирование, интеллектуализация. - Новосибирск: ВЦ СО АН СССР, 1986.

355. **Непейвода Н.Н.. 1982ст-кТеориСП**

Непейвода Н.Н., Свириденко Д.И. К теории синтеза программ // Труды Института математики СО АН СССР. - 1982. - Т.2. - с. 159-175.

356. **Непейвода Н.Н. 1978ст-СоотнМПЕВ**

Непейвода Н.Н. Соотношение между правилами естественного вывода и операторами алгоритмических языков // Доклады АН СССР. - 1978. - Т. 239. - N 3. - с. 526-529.

357. **Непейвода Н.Н. 1979ст-ООдномМПП**

Непейвода Н.Н. Об одном методе построения правильной программы из правильных подпрограмм // Программирование 1979. - N 1. - с. 15-25.

358. **Непейвода Н.Н. 1983ст-СеманАЯ**

Непейвода Н.Н. Семантика алгоритмических языков // Итоги науки и техники. - Сер.: Теория вероятностей. Математическая статистика. Теоретическая кибернетика. - М.: ВИ- НИТИ АН СССР, 1983. - Т. 20. - с. 95-166.

359. **Нестеренко Б.Б.. 1989кн-ОсновАМПВ**

Нестеренко Б.Б., Марчук В.А. Основы асинхронных методов параллельных вычислений. Киев: Наук. думка, 1989.

360. **Нечепуренко М.И.. 1990кн-АлгорИПР3**

Алгоритмы и программы решения задач на графах и сетях / Нечепуренко М.И., Попков В.К., Майнагашев С.М., Кауль С.В., Проскуряков В.А., Кохов В.А., Грязунов А.Б. - Новосибирск: Наука. Сиб. отд., 1990.

361. **Никитченко Н.С. 1982 ст-КомпоСЯП**
Никитченко Н.С. Композиционная семантика языков программирования // Программирование. - 1982. - N 6. - с. 9-18.
362. **Нильсон Н. 1985 кн-Принципы**
Нильсон Н. Принципы искусственного интеллекта. - М.: Радио и связь, 1985.
363. **Новиков А.И. 1983 кн-СеманТИЕФ**
Новиков А.И. Семантика текста и ее формализация. - М.: Наука, 1983.
364. **Новиков П.С. 1973 кн-ЭлемеMЛ**
Новиков П.С. Элементы математической логики. - М.: Наука, 1973.
365. **Норман Д. 1985 кн-ПамятИН**
Норман Д. Память и обучение. - М.: Мир, 1985.
366. **ОбщаяА-Т1-1990кн**
Общая алгебра. Т.1 / Мельников О.В., Ремесленников В.Н., Романьев В.А. и др. Под общ. ред. Л.А.Скорнякова – М.: Наука, 1990.
367. **ОбщаяА-Т2-1991кн**
Общая алгебра. Т.2 / Артамонов В.А., Салий В.Н., Скорняков Л.А. и др. Под общ. ред. Л.А.Скорнякова – М.: Наука, 1991.
368. **Озкарахан Э. 1989 кн-МашинБД**
Озкарахан Э. Машины баз данных и управление базами данных. - М.: Мир, 1989.
369. **Оллонгрен А. 1977 кн-Определение языков программирования интерпретирующими автоматами**
Оллонгрен А. Определение языков программирования интерпретирующими автоматами: пер. с англ. / Под ред. Кауфмана В.Ш. - М.: Мир, 1977.
370. **ОписаниIPO-1980сб**
Описание и распознавание объектов в системах искусственного интеллекта. Сб. статей. - М.: Наука, 1980.
371. **Оре О. 1980 кн-ТеорияГ**
Оре О. Теория графов. - М.: Наука, 1980.
372. **Осипов Г.С. 1997 кн-Приобретение знаний**
Осипов Г.С. Приобретение знаний интеллектуальными системами. -М.: Наука, 1997.
373. **Осипов Г.С. 1993 ст-Информационные технологии, основанные на знаниях**
Осипов Г.С. Информационные технологии, основанные на знаниях // Новости искусственного интеллекта. - 1993. - N 1. - с. 7-41.
374. **Осуга С..ред. 1990 кн-Приобретение знаний**
Приобретение знаний: Пер. с япон. / Под ред. С Осуги, Ю. Саэки. - М.: Мир, 1990.
375. **Осуга С. 1989 кн-Обработка знаний**
Осуга С. Обработка знаний: Пер. с япон. - М.: Мир, 1989.
376. **Петров С.В.. 1977 ст-ГрафоГиЗГ**
Петров С.В. Графовые грамматики и задачи графодинамики // Автоматика и телемеханика. - 1977. - N 10. - с. 133-138.
377. **Петров С.В.. 1977 ст-НормаФГГ**
Петров С.В. Нормальная форма графовых грамматик // Автоматика и телемеханика. - 1977. - N 6. - с. 153-157.
378. **Петров С.В.. 1977 ст-СтандФГГ**
Петров С.В. Стандартная форма графопорождающих грамматик// Актуальные вопросы теории и практики управления. - М.: Наука, 1977.- с. 71-75.
379. **Петров С.В.. 1978 ст-ГрафоГиA**
Петров С.В. Графовые грамматики и автоматы (обзор) // Автоматика и телемеханика. - 1978. - N 7. - с. 116-136.

380. **Плесневич Г.С. 1982ст-ПредсЗвАС**
Плесневич Г.С. Представление знаний в ассоциативных сетях // Изв. АН СССР. Техн. кибернет. - 1982. - N 5. - с.6-22.
381. **Плесневич Г.С. 1983ст-ДенотСАС**
Плесневич Г.С. Денотационная семантика ассоциативных сетей // Семиотика и информатика. - 1983. - Вып. 21.
382. **Плесневич Г.С. 1984ст-КонцептЯ**
Плесневич Г.С. Концептуальные языки и модели данных // Изв. АН СССР. Техн. кибернет. - 1984. - N 5. - с. 23-39.
383. **Пойа Д. 1976кн-МатемО**
Пойа Д. Математическое открытие. Решение задач: основные понятия: изучение и преподавание. - М.: Наука, 1976.
384. **Пономарёв В.М. 1982пр-РаспрВ**
Пономарёв В.М., Плюснин В.У. Торгашёв В.А. Распределённые вычисления и машины с динамической архитектурой. - Л., 1982. (Препринт / ЛИАП; N 54).
385. **Попков В.К. 1986ст-ГиперИиХС**
Попков В.К. Гиперсети и их характеристики связности // Исследования по прикладной теории графов. - Новосибирск: Наука, 1986. - с.25-58.
386. **Попов Э.В. 1976кн-АлгорОИР**
Попов Э.В., Фирдман Г.Р. Алгоритмические основы интеллектуальных роботов и искусственного интеллекта. - М.: Наука, 1976.
387. **Попов Э.В. 1982кн-ОбщениСЭВМиЕЯ**
Попов Э.В. Общение с ЭВМ на естественном языке. - М.: Наука, 1982.
388. **Попов Э.В. ред. 1990спр-ИскусСИ-К1**
Искусственный интеллект: В 3 кн. Кн. 1. Системы общения и экспертные системы: Справочник /Под ред. Э.В. Попова.- М.: Радио и связь, 1990.
389. **Попов Э.В.. 1996кн-СтатиИДЭС**
Попов Э.В., Фоминых И.Б., Кисель Е.Б., Шапот М.Д. Статистические и динамические экспертные системы. М.: Финансы и статистика, 1996.
390. **Поспелов Д.А. 1981кн-ЛогикЛМ**
Поспелов Д.А. Логико-лингвистические модели в системах управления. - М.: Энергоатомиздат, 1981.
391. **Поспелов Д. А. 1982кн-ИскусСИФиН**
Поспелов Д. А. Искусственный интеллект: фантазия или наука?. М.: Радио и связь, 1982
392. **Поспелов Д.А. 1986кн-СитуаУ**
Поспелов Д.А. Ситуационное управление: Теория и практика. М.: Наука, 1986.
393. **Поспелов Д.А. 1986кн-НечетМ**
Нечеткие множества в моделях управления и искусственного интеллекта / Под. ред. Д.А. Поспелова. - М.: Наука, 1986.
394. **Поспелов Д.А. 1986ст-ПредсЗ**
Поспелов Д.А. Представление знаний. Опыт системного анализа // Системные исследования. Методологические проблемы. Ежегодник. - М.: Наука, 1986. - с. 83-102.
395. **Поспелов И.Г.. 1987ст-ДинамОСП**
Поспелов И.Г., Поспелова Л.Я. Динамическое описание систем продукции и проверка непротиворечивости производственных экспериментальных систем // Изв. АН СССР. Техн. кибернет. - 1987. - N 1. - с. 184-192.
396. **Поспелов Д.А. 1989кн-МоделР**
Поспелов Д.А. Моделирование рассуждений. Опыт анализа мыслительных актов. М.: Радио и связь, 1989.
397. **Поспелов Д.А. 1989кн-МоделР**
Поспелов Д.А. Моделирование рассуждений. Опыт анализа мыслительных актов. - М.: Радио и связь. - 1989.

398. *Поспелов Д.А. 1989 ст-ИнтелИ*

Поспелов Д.А. Интеллектуальные интерфейсы для ЭВМ новых поколений // Электронная вычислительная техника. - М.: Радио и связь, 1989. - Вып. 3. - с. 4-20.

399. *Поспелов Д.А. 1989 ст-ПродуВИС*

Поспелов Д.А. Послесловие: продукции в интеллектуальных системах // Кузнецов В.Е. Представление в ЭВМ неформальных процедур. - М.: Наука, 1989. - с. 146-156.

400. *Поспелов Д.А. 1989 ст-Экспеc*

Поспелов Д.А. Предисловие // Экспертные системы: состояние и перспективы. - М.: Наука, 1989. - с. 3-8.

401. *Поспелов Д.А. ред. 1990 спр-ИскусИ-К2*

Искусственный интеллект: В 3 кн. Кн. 2. Модели и методы: Справочник /Под ред. Д.А. Поспелова. - М.: Радио и связь, 1990.

402. *Поспелов Д.А. 1994 слв-ИнфорЭС*

Информатика: Энциклопедический словарь для начинающих/ Сост. Д.А. Поспелов. - М.:Педагогика-Пресс, 1994.

403. *Поспелов Д.А.. 1996 ст- КакСлиП*

Поспелов Д.А., Литвинцева Л.В. Как совместить левое и правое?//Новости искусственного интеллекта. - 1996. - № 2. - С.66-71.

404. *Поспелов Д.А. 1998 ст-МетафOиC*

Поспелов Д.А. Метафора, образ и символ в познании мира // Новости искусственного интеллекта. - 1998 - №1. - С.94-114.

405. *Прангивили И.В.. 1981 ст- СовреСПСЭВМ*

Прангивили И.В., Стецюра Г.Г. Современное состояние проблемы создания ЭВМ с нетрадиционной структурой и архитектурой, управляемых потоком данных // Измерение, контроль, автоматизация. - 1981. - N 1. - с. 36-48.

406. *Прангивили И.В.. 1983 кн-ПаралВС*

Прангивили И.В., Виленкин С.Я., Медведь И.П. Параллельные вычислительные системы с общим управлением. - М.: Энергоатомиздат, 1983.

407. *Прангивили И.В. 1981 ст-АрхитКВВС*

Прангивили И.В. Архитектурные концепции высокопроизводительных вычислительных систем 80-х годов // Вопросы кибернетики. Многопроцессорные вычислительные системы с перестраиваемой структурой. (Архитектура. Структура. Применения)/Сб. статей под ред. Прангивили И.В.- М.: АН СССР, 1981.- с. 3 - 14.

408. *Пратт Т. 1979 кн-ЯзыкиП*

Пратт Т. Языки программирования: разработка и реализация: Пер. с англ. под ред. Ю.М.Баяковского - М.:Мир, 1979.

409. *ПредсМСвГМ-2001уч*

Представление математических структур в графодинамических моделях: Учеб. пособие по курсу "Математические основы искусственного интеллекта" для студентов специальности "Искусственный интеллект". В 3 ч. Ч.1.- Мин.: БГУИР, 2001.

410. *Прибрам К. 1975 кн-ЯзыкиМ*

Прибрам К. Языки мозга. М.:Прогресс, 1975.

411. *ПрогрВАМ-2001кн*

Программирование а ассоциативных машинах/ В.В.Голенков, Г.С.Осипов, Н.А.Гулякина и др. - Мин.:БГУИР, 2001.

412. *Пэранек Г.В. 1991 кн-РаспРИИ*

Пэранек Г.В. Распределенный искусственный интеллект// В кн.: Искусственный интеллект: применение в интегрированных производственных системах, Э.Кьюсиак (ред.). М.: Машиностроение, 1991.

413. *Рабинович З.Л. 1976 ст-ОрганСЭВМ*

Рабинович З.Л. Организация структур ЭВМ в связи с повышением машинного интеллекта // Управляющие системы и машины. - 1976. - N 6. - с. 12-19.

414. Рабинович З.Л. 1979 ст-НекомБП

Рабинович З.Л. Некоторый бионический подход к структурному моделированию целенаправленного мышления // Кибернетика. - 1979. - N 2. - с. 115-118.

415. Рабинович З.Л. 1979 ст-РазвиСУЭВМ

Рабинович З.Л. Развитие структур универсальных ЭВМ в связи с проблемами автоматизации научных исследований // Автоматика. - 1979. - N 5. - с. 63-72.

416. Рабинович З.Л. 1984 ст-МашинИ

Рабинович З.Л. Машинный интеллект и структуры ЭВМ пятого поколения // Кибернетика. - 1984. - N 3. - с. 95-107.

417. Рабинович З.Л. 1988 ст-ИдеолИЭВМ

Рабинович З.Л. Идеология интеллектуализации ЭВМ в пятом поколении // УСиМ. - 1988. - N 5. - с. 3-9.

418. Рабинович З.Л. 1990 ст-оРазвиАПР

Рабинович З.Л. О развитии аппаратной поддержки рассуждений в интеллектуальных системах // II Всесоюзная конференция "Искусственный интеллект-90". Круглые столы. - Минск, 1990. - с. 45-50.

419. Рабинович З.Л. 1993 ст-оМеханМ

Рабинович З.Л. О механизмах мышления и интеллектуальных ЭВМ // Кибернетика и системный анализ. - 1993. - N 3. - с.69-78.

420. Рахилина Е.В. 1986 ст-кОписаВОО

Рахилина Е.В. К описанию вопросно-ответного отношения // НТИ. Сер. 2. - 1986. - N 2. с. 24-28.

421. РеальИПИИ-1987кн

Реальность и прогнозы искусственного интеллекта / Пер. с англ. - М.: Мир, 1987.

422. Редъко В.Н. 1979 ст-ОсноввКП

Редъко В.Н. Основания композиционного программирования// Программирование. - 1979. - N 3. - с. 3-13.

423. Редъко В.Н. 1981 ст-СеманСП

Редъко В.Н. Семантические структуры программ // Программирование. - 1981. - N 1. - с. 3-19.

424. Резанов С.Н. 1989 ст-оОдномМО

Резанов С.Н. Об одном методе обобщения на семантических сетях в системе управления энергообъединением // Изв. АН СССР. Техн. кибернет. - 1989. - N 5. - с. 55-62.

425. Рейнгольд Э.. 1980 кн-КомбиАТиП

Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. / пер.с англ. под ред. В.Б.Алексеева. - М.: Мир, 1980.

426. Рейтман У.Р. 1968 кн-ПознайМ

Рейтман У.Р. Познание и мышление. Моделирование на уровне информационных процессов. - М.: Мир, 1968.

427. Родрига Г.ред. 1986-ПаралВ

Параллельные вычисления. Под. ред. Родрига Г. - М.: Наука, 1986.

428. Розенкрэнц Д. 1970 ст-ПрогрГ

Розенкрэнц Д. Программные грамматики и классы формальных языков // Сборник переводов по вопросам информационной теории и практики. - М.: ВИНИТИ, 1970. - N 16. - с.117-146.

429. Рубашкин В.Ш. 1989 кн-ПредсIAS

Рубашкин В.Ш. Представление и анализ смысла в интеллектуальных информационных системах. - М.: Наука, 1989.

430. Рыбина Г.В.. 1979 ст-СистеППЗ

Системы представления понятийных знаний с использованием фреймов / Рыбина Г.В., Строганова Н.А., Фардзинова М.И., Хромов А.А. // Интеллектуальные банки данных. - М.: Сов. радио, 1979. - с. 25-48.

431. Савушкин С.А. 1992 ст-НейроЭС

Савушкин С.А. Нейросетевые экспертные системы // Нейрокомпьютер. - 1992. - N 2.

432. **Сапатый П.С. 1983 ст-о ЭффекСРО**
Сапатый П.С. Об эффективности структурной реализации операций над семантическими сетями // Техн. кибернет. - 1983. - N 5. - с. 128-134.
433. **Сапатый П.С. 1984 ст-АктивИП**
Сапатый П.С. Активное информационное поле как модель структурного решения задач на графах и сетях // Изв. АН СССР. Техн. кибернет. - 1984. - N 5. - с. 184-208.
434. **Сапатый П.С. 1986 ст-Язык ВОЛНА-0**
Сапатый П.С. Язык ВОЛНА-0 как основа навигационных структур для баз знаний на основе семантических сетей // Изв. АН СССР. Техн. кибернет. - 1986. - N 5. - с. 198-210.
435. **Свами М.. 1984 кн- Графы СиА**
Свами М., Тхуласираман К. Графы, сети и алгоритмы. - М.: Мир, 1984.
436. **Семенов В.В. 1980 ст-СеманФС**
Семенов В.В. Семантические фреймовые сети как модели предметной области для САПР САУ // Представление знаний в системах искусственного интеллекта. - М.: МДНТП, 1980. - с. 117-122.
437. **СемиоАФИД-1985 кн**
Семиотические аспекты формализации интеллектуальной деятельности. - М.: Наука, 1985.
438. **Сергейчук И.М. 1977 ст-ВычисЛУ**
Сергейчук И.М. Вычислительное устройство для оперативного решения сетевых задач одного класса. // Управляющие системы и машины. - 1977. - N 3. - с. 84-90.
439. **Симонс Дж. 1985 кн-ЭВМПП**
Симонс Дж. ЭВМ пятого поколения: компьютеры 90-х годов.- М.: Финансы и статистика. - 1985.
440. **Скворцов Д.П.. 1981 ст-ЗамечООРЯ**
Скворцов Д.П., Финн В.К. Замечание об одном расширении языка многосортной логики предикатов // НТИ. - Сер. 2. - 1981. - N 8. - с. 25-26.
441. **Скворцов Д.П. 1982 ст-о НекотСПЛЯ**
Скворцов Д.П. О некоторых способах построения логических языков с кванторами по кортежам // Семиотика и информатика. - 1982. - Вып. 20. - с. 102-126.
442. **Скорняков Л.А. ред. 1990 кн-Общая**
Скорняков Л.А. ред. Общая алгебра. Т.1 / Мельников О.В., Ремесленников В.Н., Романьков В.А. и др. – М.: Наука, 1990.
443. **Скорняков Л.А. ред. 1991 кн-Общая**
Скорняков Л.А. ред. Общая алгебра. Т.2 / Артамонов В.А., Салий В.Н., Скорняков Л.А. и др. – М.: Наука, 1991.
444. **Скороходъко Э.Ф. 1983 кн-СеманСиАОТ**
Скороходъко Э.Ф. Семантические сети и автоматическая обработка текста. - Киев: Наук. думка, 1983.
445. **Скрагг П. 1983 ст-СеманСкМП**
Скрагг П. Семантические сети как модели памяти. // Новое в зарубежной лингвистике. - М.: Прогресс, 1983. - Вып. 12. - с.228-271.
446. **Скриган Н.И.. 1979 пр-СредсОИГС**
Скриган Н.И., Никитин А.С. Средства обработки иерархических графовых структур на базе расширения языка ПЛ/1 ОС (язык ГРАСПЛ). - Минск, 1979. (Препринт / Ин-т математики АН БССР; N 6-62).
447. **Смальян Р. 1981 кн-ТеорияФС**
Смальян Р. Теория формальных систем. - М.: Наука, 1981.
448. **Смирнова Е.Д. 1990 кн-ОсновыЛС**
Смирнова Е.Д. Основы логической семантики.-М.:Высш.шк.,1990.
449. **Смирнов В.А. 1972 кн-Форма Вили**
Смирнов В.А. Формальный вывод и логические исчисления. - М.: Наука, 1972. - 272 с.

450. ***Смирнов В.К. 1988ст- ЛИСПП***

Смирнов В.К., Рубин А.Г. ЛИСП-процессоры - аппаратная база для создания систем искусственного интеллекта // Машинная реализация систем искусственного интеллекта: Сб. науч. тр. / Под ред. Кузина Л.Т. - М.: Энергоатомиз-дат, 1988. - с. 48-58.

451. ***Соловьев В.А. 1990ст-ФормиНССПиС***

Соловьев В.А. Формирование на семантической сети понятий и суждений с помощью рассуждений по аналогии // II Всесоюзная конференция "Искусственный интеллект-90". Секционные и стеновые доклады. - Минск, 1990. - Т. 1. - с. 166-169.

452. ***Степанов А.М. 1981пр-ФреймИПСВ***

Степанов А.М. Фреймы и параллельные смешанные вычисления. - Новосибирск, 1981. (Препринт / СО ВЦ АН СССР; N 297).

453. ***Степанов А.М. 1981пр-ЭкспеСП***

Степанов А.М. Экспериментальная система программирования. - Новосибирск, 1981. (Препринт / СО ВЦ АН СССР; N 305).

454. ***Стерлинг Л.. 1990кн-ИскусП***

Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог. - М.: Мир, 1990.

455. ***Стогний А.А. 1987ст- ЭВМ Серии МИР***

Стогний А.А., Гринченко Т.А. ЭВМ серии МИР и пути повышения машинного интеллекта // Кибернетика. - 1987. - N 6. - с. 72-80.

456. ***Столяр А.А. 1982кн-КакМУвПП***

Столяр А.А. Как математика ум в порядок приводит. - Мн.: Выш. школа, 1982

457. ***Суворов Е.В.. 1981пр-СпециП***

Суворов Е.В., Фет Я.И. Специализированный процессор для аппаратной поддержки реляционных баз данных. - Новосибирск, 1981. (Препринт / Ин-т матем. СО АН СССР).

458. ***Суворов Е.В.. 1985ст-ПроцeБД***

Суворов Е.В., Фет Я.Н. Процессоры баз данных // Изв. АН СССР. Техн. кибернет. - 1985. - N 6. - с. 63-75.

459. ***Суворов Е.В. 1984пр-ОбобщПкРИЛП***

Суворов Е.В. Обобщенный подход к реализации информационно-логических процедур. - Новосибирск, 1984. - (Преп-rint / Ин-т матем. СО АН СССР; N 517).

460. ***Суворов П.Ю. 1979ст-ПредсДРГ***

Суворов П.Ю. Представление доказательств раскрашенными графами и гипотеза Хадвигера // Записки научных семинаров Ленинградского отделения Математического института АН СССР им. В.А.Стеклова. - 1979. - Т.88. - N 8. - с. 209-217.

461. ***Суньюань Г. 1984ст-СистоИВМП***

Суньюань Г. Системы и волновые матричные процессоры / Пер. с англ. // ТИИЭР. - 1984. - Т. 72. - N 7. - с.133-153.

462. ***Сырков Б.Ю.. 1992кн-ПрогрОМС***

Сырков Б.Ю., Матвеев С.В. Программное обеспечение мультитранспьютерных систем. -М.: ДИАЛОГ МИ-ФИ, 1992.

463. ***Такеути Г. 1978кн-ТеориД***

Такеути Г. Теория доказательств. - М.: Мир, 1978.

464. ***Таран Т.А. 1998кн-ОсновыДМ***

Таран Т. А. Основы дискретной математики. – К.: Просвіта, 1998.

465. ***Тарасов В.Б. 1998ст-АгентМСВС***

Тарасов В.Б. Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте// Новости искусственного интеллекта.№2, 1998.

466. ***Тейз А.. 1990кн-ЛогичПкИИ***

Логический подход к искусственному интеллекту: от классической логики к логическому программированию: Пер. с франц./Тейз А., Грибомон П., Луи Ж. и др. - М.: Мир, 1990.

467. Тейлор Р.. 1982. ст-ЯзыкOnП

Тейлор Р., Уилсон П. Язык, ориентированный на процессы и удовлетворяющий требованиям распределённой обработки // Электроника. - 1982. - N 24. - с. 26-34.

468. Тербер К.Дж. 1985кн-АрхитВВС

Тербер К.Дж. Архитектура высокопроизводительных вычислительных систем / Пер. с англ. - М.: Наука, 1985.

469. Тик Э.. 1988кн-КонвейПрологП

Тик Э., Уоррен Д.Г.Д. Конвейерный Пролог-процессор // Язык Пролог в пятом поколении ЭВМ. - М.: Мир, 1988.

470. Тондл Л. 1975кн-ПроблС

Тондл Л. Проблемы семантики. - М.: Прогресс, 1975.

471. Торгашев В.А. 1982ст-УправВП

Торгашев В.А. Управление вычислительными процессами в машинах с динамической архитектурой // Вычислительные системы и методы автоматизации исследований и управления. - М.: Наука, 1982.

472. Торгашев В.А. 1982ст-ЭВМСДА

Торгашев В.А. ЭВМ с динамической архитектурой // Вычислительные системы и методы автоматизации исследований и управления. - М.: Наука, 1982.

473. Торгашев В.А. 1983ст-РЯД

Торгашев В.А. РЯД - развивающийся язык программирования динамического типа для распределенных вычислений // Информационно-вычислительные проблемы автоматизации научных исследований. - М.: Наука, 1983. - с. 215-224.

474. Трахтенгерц Э.А. 1987кн-ПрогрОПП

Трахтенгерц Э.А. Программное обеспечение параллельных процессоров. - М.: Наука, 1987.

475. Тузов В.А. 1984кн-МатемМЯ

Тузов В.А. Математическая модель языка. - Л.: Изд-во ленингр. ун-та, 1984.

476. Тузов В.А. 1986ст-оФормаПЗ

Тузов В.А. О формализации понятия задачи // Методы и системы автоматизации в задачах науки и производства. - М.: Наука, 1986. - с. 73-83.

477. Турчин В. Ф. 1968.пр-АлгорЯРФ

Турчин В.Ф. Алгоритмический язык рекурсивных функций (РЕФАЛ). - М., 1968.(Препринт / ИПМ АН СССР).

478. Турчин В. Ф. 1968ст-МетааЯ

Турчин В.Ф. Метаалгоритмический язык // Кибернетика. - 1968. - N 4.

479. Тыугу Э.Х.. 1985ст-ОбъектОЯНУТ

Тыугу Э.Х., Мацкин М.Б., Пеньям Я.Э., Эмойис П.В. Объектно-ориентированный язык НУТ // Прикладная информатика.- М.: Финансы и статистика, 1985. - Вып. 2(9). - с. 45-66.

480. Тыугу Э.Х. 1989ст-ИнтегЗ

Тыугу Э.Х. Интеграция знаний // Изв. АН СССР. Техн. кибернет. - 1989. - N 5. - с. 3-13.

481. Тыугу Э.Х. 1990ст-ОбъектОП

Тыугу Э.Х. Объектно-ориентированное программирование // Программирование. - 1990. - N 6. - с. 16-26.

482. Тыугу Э.Х. 1984кн-КонцепП

Тыугу Э.Х. Концептуальное программирование. - М.: Наука, 1984.

483. Тюхтин В. С. 1977ст-ОсобеПЗИ

Тюхтин В.С. Особенности переработки знаковой информации человеком и ЭВМ // Психологические проблемы переработки знаковой информации. - М.: Наука, 1977. - с. 57-69.

484. Уварова Т.Г.. 1987кн-ФормаООЯдСС

Уварова Т.Г., Лифшиц Л.Л. Формальное описание операционного языка для семантических сетей. - М.: ВЦ АН СССР, 1987.

485. Уварова Т.Г. 1987 ст-ОпераСВЯ

Уварова Т.Г. Операционная семантика волновых языков и метод ее описания // Изв. АН СССР. Техн. кибернет. - 1987. - с. 128-142.

486. Уинстон П.Г. 1980 кн-ИскусСИ

Уинстон П. Искусственный интеллект. Пер. с англ. - М.: Мир, 1980.

487. Уинстон П.Г. 1987 ст-ЛиспСР

Уинстон П.Г. Лисп совершает революцию / Пер. с англ. // Реальность и прогнозы искусственного интеллекта. - М.: Мир, 1987. - с. 71-84.

488. Уоссермен Ф. 1992 кн-НейроТ

Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. М.: Мир, 1992.

489. Уотермен Д. 1989 кн-РуковПЭС

Уотермен Д. Руководство по экспертным системам: Пер. с англ. - М.: Мир, 1989.

490. Успенский В.А.. 1987 кн-Теория

Успенский В.А., Семенов А.Л. Теория алгоритмов, основные открытия и приложения. - М.: Наука, 1987.

491. Уэно Х..ред. 1989 кн-ПредсИИЗ

Представление и использование знаний: Пер. с япон. / Под ред. Х. Уэно, М. Исидзука. - М.: Мир, 1989.

492. Файн В.С. 1987 кн-РаспоО

Файн В.С. Распознавание образов и машинное понимание естественного языка. - М.: Наука, 1987.

493. Фет Я.И. 1980 ст-ПреобСД

Фет Я.И. Преобразование структур данных в специализированных однородных процессорах // Управляющие системы и машины. - 1980. - N 4. - с. 32-38.

494. Фет Я.И. 1981 кн-ПаралПдУС

Фет Я.И. Параллельные процессоры для управляющих систем. - М.: Энергоиздат, 1981.

495. Фет Я.И. 1986 ст-ВертиО

Фет Я.И. Вертикальная обработка как основа крупноблочной архитектуры // Известия АН СССР. Техн. кибернет. - 1986. - N 5. - с. 139-152.

496. Филд А.. 1993 кн-ФункцП

Филд А., Харрисон П. Функциональное программирование / Пер. с англ. - М.: Мир, 1993.

497. Финн В.К. 1981 ст-кФормаОИПС

Финн В.К. К формальному определению информационно-поисковой системы // НТИ. Сер. 2. - 1981. - N 5. - с. 5-15.

498. Финн В.К. 1976 кн-ЛогичПИП

Финн В.К. Логические проблемы информационного поиска. - М.: Наука, 1976.

499. Финн В.К. 1983 ст-оМашинОФПР

Финн В.К. О машинно-ориентированной формализации правдоподобных рассуждений в стиле Ф.Бэкона-Д.С.Миля // Семиотика и информатика. - 1983. - Вып. 20. - с. 35-101.

500. Финн В.К. 1984 ст-ИнфорСиПИИ

Финн В.К. Информационные системы и проблемы их интеллектуализации // НТИ. Сер.2.-1984.- N 1. -с.1-14.

501. Финн В.К. 1986 ст-оЛогичИВС

Финн В.К. О логических информационно-вычислительных системах // НТИ. Сер. 2. - 1986. - N 1. - с. 7-10.

502. Финн В.К. 1991 ст-ПравдР

Финн В.К. Правдоподобные рассуждения в интеллектуальных системах типа ДСМ // Итоги науки и техники. Серия Информатика. Том 15. - М.: ВИНИТИ, 1991. - с. 54-101.

503. Фоминых И.Б. 1999 ст-АдаптС

Фоминых И.Б. Адаптивные системы и информационная модель эмоций// Сб. трудов Международной конференции "Интеллектуальное управление: новые интеллектуальные технологии в задачах управления (ICIT 99)".- Переславль-Залесский.- Наука, Физматлит, 6-9 декабря 1999.- с.103-107.

504. Фоминых И.Б. 2000ст-ИнтелгНиСМ

Фоминых И.Б. Интеграция нейронных и символьно-логических моделей в интеллектуальных технологиях// Сб. научных трудов VII Национальной конференции с международным участием "Искусственный интеллект - 2000", Переславль-Залесский, 2000.-т. 2, с.588-595.

505. Фоминых И.Б. 1996ст-КлассдЭС

Фоминых И.Б. Классификация динамических экспертных систем. // Динамические интеллектуальные системы в управлении и моделировании. Материалы семинара. - М., 1996. с.38-42.

506. Фоминых И.Б. 1998ст-оСоотнОилП

Фоминых И.Б. О соотношении образной и логической парадигм в системах искусственного интеллекта /Новости искусственного интеллекта. - М.: АИИ, №3, 1998.- С.76-85.

507. Фоминых И.Б. 1999ст-НекотФА

Фоминых И.Б. Некоторые формальные аспекты информационного подхода к построению нейроэкспертных систем//Известия РАН. Теория и системы управления.- М: Наука, 1999.- №5.- с.83-86.

508. Фостер К. 1981кн-АссоцПП

Фостер К. Ассоциативные параллельные процессоры. - М.: Энергоиздат, 1981.

509. Фреге Г. 1977ст-СмыслИД

Фреге Г. Смысл и денотат // Семиотика и информатика. - М., 1977. - Вып. 8. - с. 181-210.

510. Фридман Г. Ш. 1976ст-ИсслеОЗКнГ

Фридман Г. Ш. Исследование одной задачи классификации на графах // Методы моделирования и обработка информации. - Новосибирск: Наука, 1976.-с. 147-177.

511. Фролов А.А.. 1987кн-НейроМАП

Фролов А.А., Муравьев И.П. Нейронные модели ассоциативной памяти. - М.: Наука, 1987.

512. Фу К. С. 1977кн-СтрукМвРО

Фу К. С. Структурные методы в распознавании образов. - М.: Мир, 1977.

513. Фути К.. 1988кн-ЯзыкиПиССБИС

Фути К., Судзуки Н. Языки программирования и схемотехника СБИС / Пер. с япон. - М.: Мир, 1988.

514. Халим З. 1988ст-ИЛИППМ

Халим З. ИЛИ-параллельная потоковая Пролог-машина // Язык Пролог в пятом поколении ЭВМ. - М.: Мир, 1988. - с.276-309.

515. Хант Э. 1978кн-ИскусСИ

Хант Э. Искусственный интеллект / Пер. с англ. - М.: Мир, 1978.

516. Харари Ф. 1973кн-ТеориГ

Харари Ф. Теория графов.-М.:Мир, 1973.

517. Хейс-Рот Ф.. 1987кн.- ПострЭС

Хейс-Рот и др. Построение экспертных систем// Под ред. Ф.Хейес-Рота, Д.Уотермана, Д.Лената. М.: Мир, 1987.

518. Харп Г. М.ред. 1993кн-Транс

Транспьютеры. Архитектура и программное обеспечение: Пер. с англ./Под ред. Харпа. Г.-М.: Радио и связь, 1993.

519. Хельбиг Г. 1980ст-СеманП3

Хельбиг Г. Семантическое представление знаний в вопросно-ответной системе FAS-80 // Представление знаний и моделирование процессов понимания. - Новосибирск, 1980. - с. 97-123.

520. Хендерсон П. 1983кн-ФункцП

Хендерсон П. Функциональное программирование. Применение и реализация / Пер. с англ. - М.: Мир, 1983.

521. Хендрикс Г. 1975ст-оРасшиПССВР

Хендрикс Г. О расширении применимости семантических сетей введением разбиений // Труды IV Международной объ- единенной конференции по искусственному интеллекту. - М., 1975. - Т. 1. - с. 190-206.

522. Хигман Б. 1974кн-СравнИЯП

Хигман Б. Сравнительное изучение языков программирования: Пер. с англ. - М.: Мир, 1974.

523. Хиллис У.Д. 1987ст-Коммум

Хиллис У.Д. Коммуникационная машина // В мире науки. - М.: Мир, 1987. - N 8.

524. Хинтикка Я. 1974ст-кВопроВ

Хинтикка Я. К вопросу о вопросах // Философия в современном мире. Философия и логика. - М.: Наука, 1974.

525. Хинтикка Я. 1980ст-ЛогикоЭИ

Хинтикка Я. Логико-эпистемологические исследования: Сб. избранных статей / Составление, вступительная статья, общ. ред. Садовского В.Н., Смирнова В.А. - М.: Прогресс, 1980.

526. Хинтон Дж.Е. 1987ст-ОбучеВПС

Хинтон Дж.Е. Обучение в параллельных сетях / Пер. с англ // Реальность и прогнозы искусственного интеллекта. - М.: Мир, 1987. - с. 124-136.

527. Хоар Ч. 1989кн-ВзаимПП

Хоар Ч. Взаимодействующие последовательные процессы: Пер. с англ. - М.: Мир, 1989.

528. Хоггер К. 1988кн-ВведениеВЛП

Хоггер К. Введение в логическое программирование: Пер. с англ. - М.: Мир, 1988.

529. Хокни Р.. 1986кн-ПаралЭВМ

Хокни Р., Джессхоуп К. Параллельные ЭВМ. Архитектура, программирование, алгоритмы. - М.: Мир, 1986.

530. Холл П. 1978кн-ВычислС

Холл П. Вычислительные структуры. Введение в нечисленное программирование. - М.: Мир, 1978.

531. Холстед М.Х. 1981кн-НачалНоП

Холстед М.Х. Начала науки о программах / Пер. с англ. под ред. В.М.Юфы - М.: Финансы и статистика, 1981.

532. Хоор Ч.Э.Р.. 1980кн-НепротВТсяП

Хоор Ч.Э.Р., Лаяэр П.Е. Непротиворечивые взаимодополняющие теории семантики языков программирования // Семантика языков программирования. - М.: Мир, 1980.

533. Хорошевский В.Ф. 1999ст-ПоведИА

Хорошевский В.Ф. Поведение интеллектуальных агентов: модели и методы реализации // Труды 4-го междунар. семинара по прикладной семиотике, семиотическому и интеллектуальному управлению ASC/IC'99. - М. - 1999. - С.5-20

534. Хорошевский В.Ф.. 1989ст-ПрогрИПЗвЭС

Хорошевский В.Ф., Шерстнев В.Ю. Программный инструментарий представления знаний в экспертных системах // Экспертные системы: состояние и перспективы. М.: Наука, 1989. - с.38-46.

535. Хорошевский В.Ф. 1988уч-АвтомПЭС

Хорошевский В.Ф. Автоматизация программирования экспертовых систем: Учебное пособие. - М.: МИФИ, 1988.

536. Хоффман И. 1986кн-АктивП

Хоффман И. Активная память: Экспериментальные исследования и теории человеческой памяти. - М.: Прогресс, 1986.

537. Хювенен Э.. 1990-МирЛиспа-К1

Хювенен Э., Сеппяnen Й. Мир Лиспа. В 2-х т. Т.1: Введение в язык Лисп и функциональное программирование /Пер. с финск. - М.: Мир, 1990. - 447 с.

538. Хювенен Э.. 1990-МирЛиспа-К2

Хювенен Э., Сеппяnen Й. Мир Лиспа. В 2-х т. Т.2: Методы и системы программирования. Пер.с финск. - М.: Мир, 1990. - 319 с.

539. Цаленко М.Ш. 1989кн-МоделСвБД

Цаленко М.Ш. Моделирование семантики в базах данных. - М.: Наука, 1989.

540. **Цейтин Г.С. 1985кн-ПрогрНАС**
Цейтин Г.С. Программирование на ассоциативных сетях. - в сб.: ЭВМ в проектировании и производстве / Под ред. Орловского Г.В. - Л.: Машиностроение, Ленинградское отд., 1985.
541. **Целищев В.В.. 1982кн-ЛогикИЯНТ**
Целищев В.В., Карпович В.Н., Поляков И.В. Логика и язык научной теории. - Новосибирск: Наука, 1982.
542. **Цой С.. 1971кн-ПриклТГ**
Цой С., Цхай С.М. Прикладная теория графов. - Алма-Ата: Наука, 1971.
543. **Чапля А.А.. 1988кн-ВопроПА**
Чапля А.А.. Вопросы построения архитектуры для решения задач искусственного интеллекта. - М.: ВЦ АН СССР, 1988.
544. **Чень Ч.. 1983кн-МатемЛиАДТ**
Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. - М.: Наука, 1983.
545. **Чери С.. 1992кн-ЛогичПиБД**
Чери С., Готлоб Г., Танка Л. Логическое программирование и базы данных: Пер. с англ. - М.: Мир, 1992.
546. **Черконе Н.. 1975ст-кВопрОСПС**
Черконе Н., Шуберт Л. К вопросу о семантическом представлении состояния // Труды IV Международной объединенной конференции по искусственному интеллекту, т.2. - М., 1975.
547. **Черная И.С. 1977ст-НекомЗИП**
Черная И.С. Некоторые задачи информационного поиска на ассоциативном параллельном процессоре // Программирование. - 1977. - N 1. - с. 74-83.
548. **Черных А.Н. 1982пр-НекомВПВ**
Черных А.Н. Некоторые вопросы параллельных вычислений. - М., 1982. (Препринт / ИТМ и ВТ АН СССР; N 10).
549. **Черных А.Н. 1982ст-ПаралИИОЯ**
Черных А.Н. Параллельная интерпретация операторного языка в экспериментальной системе программирования // Многопроцессорные вычислительные системы и их математическое обеспечение / Под ред. Котова В.Е. - Новосибирск: ВЦ СО АН СССР, 1982. - с. 123-146.
550. **Чиркова Р.Ю.. 1996ст-ТехносЭС**
Чиркова Р.Ю., Мальковский М.Г. Технология создания экспертных систем для динамических задач управления процессами в критических ситуациях // Программирование, N 6, 1996, с.48-62.
551. **Шанин Н.А.. 1965кн-АлгорМП**
Шанин Н.А. и др. Алгоритм машинного поиска естественного логического вывода в исчислении высказываний. - М.: Наука, 1965.
552. **Шенк Р.. 1987кн-ПознамМ**
Шенк Р., Хантер Л. Познать механизмы мышления// Реальность и прогнозы искусственного интеллекта. М.: Мир, 1987.
553. **Шенк Р.. 1989кн-кИнтеgСиП**
Шенк Р., Бирнбаум Л., Мей Дж. К интеграции семантики и прагматики// Новое в зарубежной лингвистике. Компьютерная лингвистика. Вып.14. М.: Прогресс, 1989.
554. **Шенк Р. 1980кн-ОбрабКИ**
Шенк Р. Обработка концептуальной информации: Пер. с англ. - М.: Энергия, 1980.
555. **Широков Ф.В. 1986кн-нПутиКППК**
Широков Ф.В. На пути к пятому поколению компьютеров. - М.: Наука, 1986.
556. **Шпаковский А.С.. 1988кн-ПаралОСД**
Параллельная обработка структур данных / Г.И.Шпаковский, А.С. Липницкий, В.М. Вашкевич и др.; Под ред. В.А. Мищенко.- Мн.: Университетское, 1988.
557. **Шрейдер Ю.А. 1971кн-РавенСП**
Шрейдер Ю.А. Равенство, сходство, порядок. – М.: Наука, 1971.

558. ***Шрейдер Ю.А.. 1982кн-СистемИМ***
Шрейдер Ю.А., Шаров А.А. Системы и модели. - М.: Радио и связь, 1982.
559. ***Шуберт Л. 1979ст-УсилиЕМСС***
Шуберт Л. Усиление выразительной мощности семантических сетей / Пер. с англ. Кибернетический сборник. Новая серия. - 1979. - Вып. 16. - с. 171-212.
560. ***Шура-Бура М.Р.. 1981ст-УвеличиCHAB***
Шура-Бура М.Р., Вольдман Г.Ш. Увеличение степени непроцедурности арифметических выражений в языках программирования // Программирование. - 1981. - N 2.
561. ***ЭкспеC-1989сб***
Экспертные системы: Состояние и перспективы. Сборник научных трудов. - М.: Наука, 1989.
562. ***Ягера Р.Р. 1986кн-НечетМ***
Нечеткие множества и теория возможностей. Последние достижения: Пер. с англ./Под ред. Р.Р. Ягера. - М.: Радио и связь, 1986.
563. ***ЯзыкПАда-1981кн***
Язык программирования Ада (предварительное описание) / Пер. с англ. - М.: Финансы и статистика, 1981.
564. ***ЯзыкПрологВПП-1988сб-ЯзыкПрологППЭВМ***
Язык Пролог в пятом поколении ЭВМ / Сб. статей. - М.: Мир, 1988.
565. ***Янсон А. 1991кн-ТПрологВСИ***
Янсон А. Турбо-Пролог в сжатом изложении. Пер. с нем. - М.: Мир, 1991.
566. ***Ясухара Х.. 1988ст-ORBIT***
Ясухара Х., Нитадори К. ORBIT - параллельная модель выполнения Пролог-программы // Язык Пролог в пятом поколении ЭВМ. - М.: Мир, 1988. - с. 231-247.
567. ***Яхно Т.М. 1984пр-СистeП***
Яхно Т.М. Системы продукций как стиль программирования задач искусственного интеллекта. - Новосибирск, 1984. (Препринт / ВЦ СО АН СССР; N 499)
568. ***Яхно Т.М. 1988ст-ФормалMвCП***
Яхно Т.М. Формальная модель вычислений в системах продуктов // Изв. АН СССР. Техн. кибернет. - 1988. - N 2. - с. 106-109.
569. ***Amamiya M.. 1985art-NewAfKBM***
Amamiya M., Hakozaku K., Yokoi T. et al. New architecture for knowledge base mechanisms // Fifth generation computer Systems, edited by Motooka T. - Proceeding of the Intern. Conf. Tokyo. - Japan. - Oct. 1985. - P.179-187.
570. ***Artale A.. 1998art-aTempoDL***
Artale A. and Franconi E. A Temporal Description Logic for Reasoning about Actions and Plans. Journal of Artificial Intelligence Research 9 (1998) 463-506.
571. ***Backus J. 1978art-CanPbL***
Backus J. Can programming be liberated from the von Neumann Style? A functional Style and its algebra of programs // Com. ACM. - 1978. - V. 21. - N 8. - P. 613-641.
572. ***Bailey D.A.. 1987art-GraphGBSoIS***
Bailey D.A., Cuny J.E. Graph grammar based specification of interconnection structures for massively parallel computation // Graph Grammars and their Application to Computer Science, LNCS 291 / H.Ehrig, M.Nagl, G.Rosen-berg, A.Rosenfeld, ed. - Sprinnger-Verlag. - 1987. - P.73-85.
573. ***Beer J.. 1987art-POPE***
Beer J., Giloi W.K. POPE - a Parallel-Operating Prolog Engine // Future Generation Computer Systems. - 1987. - Vol. 3. - N 2. - P. 83-92.
574. ***Benker H.. 1989art-KnowlCM***
Benker H. et al. The Knowledge Crunching Machine at ECRC: A Joint R & D Project of a High Speed Prolog System // ICL Technical Journal. - November. - 1989. - Vol. 6. - N 4. - P. 737-753.

575. *Bibel W.. 1987 art-ParalIM*

Bibel W. et al. Parallel Inference Machines // Future Parallel Computers, Lecture Notes in Computer Science. - 1987. - N 272. - P. 185-226.

576. *Bibel W.. 1985 art-BibliOPIM*

Bibel W., Aspetsberger K. A Bibliography on Parallel Inference Machines // Journal of Symbolic Computations. - 1985. - Vol. 1. - N 1. - P. 115-118.

577. *Bic L. 1985 art-DatafAfKRS*

Bic L. Dataflow architecture for knowledge representation systems // AFIPS NCC. - Chicago. - 1985. - P.136-145.

578. *Bic L. 1985 art-ProceONoDA*

Bic L. Processing of nets on dataflow architectures // Artificial Intelligence. - 1985. - V. 27. - N 2. - P.219-227.

579. *Boral H.. 1983 art-DatabM*

Boral H., Dewitt D. Database machines: an idea whose time has passed? A critique of the future of database machines // Proc. Ant. Workshop on Database & Dataflow Machines, Munich. - 1983. - P. 166-187.

580. *Borghoff U.. 1998 bk-InforTfKM*

Borghoff U., Pareschi R., 1998. Information Technology for Knowledge Management. – Springer-Verlag, Bln.

581. *Bosygit M. 1987 art-DistrCSAS*

Bosygit M. Distributed computing system architectures: hardware // Distributed operating systems: theory and practice. - Springel-Verlag. - 1987. - P. 201-218.

582. *Bottger H. 1984 art-InferIR*

Bottger H. Inferential information retrieval over an ex- tended semantics network // Computers and Artificial Intelligence. - 1984. - V. 3. - N 2. - P. 115-126.

583. *Brachman R. J. 1979 art-oEpists*

Brachman R. J. On the Epistemological Status of Semantic Networks // Associative Networks Representation and Use of Knowledge by Computers.-New York: Academic Press, 1979.-P. 3-50.

584. *Brukle H.J. 1978 art-HighLL*

Brukle H.J. High level language oriented hardware and post - von Neumann era // Proc. 5-th Symp Computer Architecture. - 1978. - P. 60-65.

585. *Castellani I.. 1982 art-GraphGfDS*

Castellani I., Montanari U. Graph grammars for distributed systems // Graph Grammars and their Application to Computer Science. LNCS 153 / H.Ehrig, M.Nagl, G.Rozen- berg, ed. - Springer-Verlag. - 1982. - P. 20-38.

586. *Chu Y. 1976 art-EvoluOCMS*

Chu Y. Evolution of Computer Memory Structure // Proc. National Computer Conf. AFIPS Press. - 1976. - P.733-748.

587. *Ciepielewski A.. 1989 art-ORPPROLOG*

Ciepielewski A., Haridi S., Hausman B. OR-parallel PROLOG on shared memory multiprocessors // J. Logic Programming. - 1989. - N 7. - p. 125-147.

588. *Clark K.L. 1987 art-PARLOG*

Clark K.L. PARLOG: the language and its applications // Lecture Notes in Computer Science. - V. 259: PARLE - Parallel Architectures and Languages Europe. - Springer-Verlag. - 1987. - P. 30-53.

589. *Clocksin W.F.. 1988 art-MethoFEEHCP*

Clocksin W.F., Alshawi H. A Method for Efficiently Executing Horn Clause Programs Using Multiple Processors // New Generation Computing. - 1988. - Vol. 5. - N 4. - P.361-376.

590. *Clocksin W.F. 1987 art-PrincODelPhi*

Clocksin W.F. Principles of the DelPhi Parallel Inference Machine // Computer Journal. - 1987. - Vol. 30. - N 5. - P. 386-392.

591. *Conery J.S... 1981 art-ParalIoLP*

Conery J.S., Kibler D.F. Parallel interpreter of logic programs // Proc. of ACM Conf. on Functional Programming Languages and Computer Architecture, Portsmouth. - New Hampshire. - 1981. - P. 163-170.

592. ***Conery J.S.. 1985art-ANDP***
 Conery J.S., Kibler D.F. AND parallelism and nondeterminism in logic programs // New generation Computing. - 1985. - V. 3. - N 1.
593. ***Conklin J. 1987art-Hyper***
 Conklin J., 1987. Hypertext: An Introduction and Survey // Computer. Vol.20, No. 9. P. 17-41.
594. ***Conklin J. 1996bk-DesigOM***
 Conklin J., 1996, Designing organizational memory: Preserving intellectual assets in a knowledge economy // Electronic Publication by Corporate Memory Systems, Inc.
595. ***Davis A.L. 1988art-FAIM-1***
 Davis A.L. FAIM-1: An Architecture for Symbolic Multiprocessing // Parallel Computation and Computers for Artificial Intelligence. - Kluwer Academia Publishers. - 1988. - P. 223-246.
596. ***Degano P.. 1987art-ModelfDS***
 Degano P., Montarani U. A model for distributed systems based on graph rewriting // J. ACM. - 1987. - V. 34(2).- P. 411-449.
597. ***Deliyani A.. 1979art-LogicASM***
 Deliyani A., Kowalski R.A. Logic and semantic networks// Communications of the ACM. - 1979. - V. 22. - N 3. - P.184-192.
598. ***Diel H. 1984art-ConcuDAA***
 Diel H. Concurrent data access architecture // Proc. Int. Conf. Fifth generation computer Systems / ed. by ICOT. - 1984. - P. 373-382.
599. ***Diel H. 1986art-ParallP***
 Diel H. Parallel logic programming based on an extended machine architecture // Fifth generation computer architectures / ed. by J.W.Woods Proc. IFIP TC 10 Working Conf. - 1986. - P. 15-30.
600. ***Edmund D.H.ed. 1989art-TrendICDPS***
 Trends in cooperative distributed problem solving / Durfee Edmund H., Lesser Victor R., Corkill Daniel D./IEEE Trans. Knowledge and Data Eng.- 1989. - Vol.1. - N 1. - P.63-83.
601. ***Engels G.. 1987art-GraphGESSM***
 Engels G., Lewerentz C., Schafer W. Graph grammar engineering: a software specification method // Graph Grammars and their Application to Computer Science. LNCS 291 / H.Ehrig, M.Nagl, G.Rozenberg, A.Rosenfeld, ed. - Springer-Verlag. - 1987. - P. 186-201.
602. ***Farquhar A.. 1996art-tOntols***
 Farquhar A., Fikes R., Rice J., 1996. The Ontolingua Server: A Tool for Collaborative Ontology Construction // Knowledge System Laboratory, KSL-96-26, September, 1996.
603. ***Findler N. V.ed. 1979bk-AssocNraUoK***
 Associative Networks Representation and Use of Knowledge by Computers/Nic. V. Findler Ed.-New York: Academic Press, 1979.
604. ***Fisher K.M. 1990art-SemanN***
 Fisher K.M., 1990. Semantic networking: New kid of the block // Journal of Research of Science Teaching. No.27 (10). P. 1001-1018.
605. ***Fisher K.M. 1992art- SemNet***
 Fisher K.M., 1992. SemNet: A tool for personal knowledge construction // In P. Kommers, D. Jonassen, & T. Mayes (Eds.) Cognitive tools for Learning. Berlin: Springer Verlad. P. 63-76.
606. ***Fominykh I.B. 1999art- SomeFF***
 Fominykh I.B. Some Formal Fspects of the Informational Approach to the Construction of Neural Expert Systems//J. of Computer and System Sciences International ISSN 1064-2037, 1999.- No.5.- pp.83-86.
607. ***Fominykh I.B. 2000art-NeuroMiT***
 Fominykh I.B. Neurological Models in Intellectual Technologies//Proceeding of the International Conf. "Intelligent Systems and Information Technologies in Control"(IS&ITS-2000), Pscov, June 2000.- St.Peterburg/pskov.- SPbSTU Publishers 2000.- pp.50-52.

608. **Fridman N.. 1997bk-Ontold**
Fridman N., Hafner, 1997. Ontology Design: A Survey and Comparative Review // AI Magazine. No. 18(3). P. 53-74.
609. **Gabor D.1969art-AssocHM**
Gabor D. Associative Holographical Memories. IBM J. of research and development, 1969, vol.13, № 2.
610. **Gavrilova T..1999art-CogniAoVKBD**
Gavrilova T., Chernigovskaya T., 1999. Cognitive Aspects of Visual Knowledge Base Desing // Proceedings of the International Conference PEG, Intelligent Computer and communications technology (Teaching & Learning for the 21-st Century), Great Britain. P. 174-181.
611. **Gavrilova T..1999bk-AdaptDLC**
Gavrilova T., Kotova E., Stash N., 1999. Adaptive Distance Learning Course on Artificial Intelligence, ICL99, Austria.
612. **Goering S.K.1990rep-GraphGAtCP**
Goering Steven K. A Graph Grammar Approach to Concurrent Programming. - Technical Report UIUCDCS-R-90-1576. - University of Illinois, Department of Computer Science.- May 1990.
613. **Golenkov V. V..1992art-ProjeOPC**
Golenkov V.V., Korolyov V.G., Solovyov A.S., Tatarenko V.A. The Project of a Parallel Computer for Hardware Support of Processing Semantic and Neural Networks // Proc. RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers. Rostov-on-Don, Russia. October 7-10, 1992.- Vol.1.- P.623-634.
614. **Golenkov V. V..1993art-KnowlPTSaNLI**
Golenkov V.V., Korneyevskaya L.I., Korolev V.G., Lobanov B.M., Malevitch I.E. Knowledge Processing Tools With Speech and Natural Language Interface// Proc East-West conference on Artificial Intelligence. From Theory to Practice. Moscow, Russia. September 7-9, 1993.- P.198-200.
615. **Golenkov V. V..1993art-ProgrLfPPoK**
Golenkov V.V., Gaponov P.A., Kuzmitsky V.M., Solovyov A.S., Tatarenko V.A. A Programming Language for Parallel Processing of Knowledge, Represented by Graph Structures // Proc. East-West conference on Artificial Intelligence. From Theory to Practice. Moscow, Russia. September 7-9, 1993.- P.17-19.
616. **Golenkov V. V..1994bk-PGC**
Golenkov V.V., Korolyov V.G., Kuzmitsky V.M., Tatarenko V.A., Vasilevskaya I.V. Parallel Graph Computer (PGC) for Solving Problems in the Area of Artificial Intelligence and Its Applications. - Minsk, 1994. - 48 p. (Pre-print/Institute for Engineering Cybernetics of Belarusian Academy of Sciences; N 5).
617. **Goncalves G..1988art-NetwoOT**
Goncalves G. et al. A network of transputers to emulate a parallel simbolic processor // Microprocessing and Microprogramming. - 1988. - V. 23. - P. 149-152.
618. **Goncharov S.S..1986art-SemanP**
Goncharov S.S., Ershov Yu.L., Sviridenko D.I. Semantic programming // Information Processing, Congr., IFIP. - North Holland, 1986. - p. 1093-1100.
619. **Goto A..1984art-HighPIE**
Goto A., Tanaka H., Motooka T. Highly parallel inference engine PIE - goal rewriting model and machine architecture // New Generation Computing. - 1984. - V. 2. - N1.- P. 37-58.
620. **Goto A..1987art-HighPPIM**
Goto A., Ushida S., Toward A. High Performance Parallel Inference Machine - The Intermediate Stage Plan of PIM// Lecture Notes in Computer Science. - Bo 272. - 1987. - P. 299-320.
621. **Gruber T.R.1995art-Towar**
Gruber T.R., 1995. Toward Principles for the Desing of ontologies Used for Knowledge Sharing // International Journal of Human and Computer Studies. – No. 43(5/6), P. 907-928.
622. **Guarino N.. 1999art- OntoSeek**
Guarino N., Masolo C., Vetere G., 1999. OntoSeek: Content-Based Access to the Web // IEEE INTELLIGENT SYSTEMS. – May/June 1999.
623. **Guarino N.1996art-Ontol**
Guarino N., 1996. Ontologies: What Are They, and Where's The Research? // A panel held at KR'96, the Fifth In-

- ternational Conference on Principles of Knowledge Representation and Reasoning, November 5, 1996, Cambridge, Massachusetts. <http://www-ksl.stanford.edu/KR96/Panel.html>.
624. **Gupta A.. 1987 art-ParalAaa**
 Gupta A., Forgy C., Newell A., Weding R. Parallel algorithms and architectures for rule-based systems // Comp. Archit. News. - 1987. - V. 14. - N 2. - P. 28-37.
625. **Halim Z. 1986 art-DataDM**
 Halim Z. A data-driven machine for OR-parallel evaluation of logic programs // New Generat. Comput. - 1986. — N 1. - p. 5-33.
626. **Halstead R.H. 1985 art-MultiLfcSC**
 Halstead R.H. Multilisp: A language for concurrent symbolic computation // ACM Trans. Progr. Lang. and Systems. - 1985. - V. 7. - N 4. - P. 501-538.
627. **Harland J.. 1987 art-oParalUfProlog**
 Harland J., Jaffar J. On Parallel Unification for Prolog // New Generation Computing. - 1987. - Vol. 5. - N 3. - P. 259-279.
628. **Harsat A.. 1990 art-CARMEL-2**
 Harsat A., Ginosar R. CARMEL-2: a second generation VLSI architecture for flat concurrent Prolog // New Generat. Comput. - 1990. - N 7. - p. 197-218.
629. **Hayes-Roth B. 1985 art-BlackAfC**
 Hayes-Roth B. A blackboard architecture for control // Artificial Intelligence. - 1985. - V. 26. - P. 251-321.
630. **Heerden P.J. 1968 bk-FoundOEK**
 Heerden P.J.van. The Foundation of Empirical Knowledge. N.V.Uitgeverij Wistik-Wassenaar, Netherland, 1968.
631. **Hendrix G.G. 1979 art-EncodKiPN**
 Hendrix G.G. Encoding Knowledge in Partitioned Networks// Associative Networks. Representation and Use of Knowledge by Computers. - New York: Academic Press, 1979. - P. 51-92.
632. **Higuchi T.A. 1985 art-SemanNLM**
 Higuchi T. A semantic network language machine // Micro- computers, Usage and Design 11-th Euromicro Symp. Microproc. and Microcomp. - Brussels. - 3-6 Sept. 1985. - P. 95-104.
633. **Hillis W.D. 1984 art-ConneM**
 Hillis W.D. The Connection Machine: a computer Architecture Based on Cellular Automata // Physica. - 1984. - V.10D. - N 1,2.
634. **Ibanez M. 1988 art-ParalIiFOL**
 Ibanez M. Parallel Inferencing in First Order Logic Based on the Connection Method. - Artificial Intelligence III: Methodology, Systems, Applications (AIMSA) - North-Holland. - 1988. - P. 149-158.
635. **Jonassen D.H.. 1999 bk-DesigH**
 Jonassen D.H., Mandl H. (Eds). 1999. Designing Hypermedia for Learning, – Springer-Verlad N.Y., Incorporated.
636. **Kaplan S.M.. 1989 art-Garp**
 Kaplan S.M., Goering S.K. Garp - a graphical/textual language for concurrent programming. - in: Proceedins of the Workshop on Object-Based Concurrent Programming. ACM Sigplan Notices. - April 1989. - P. 184-186.
637. **Kascuk P.A. 1984 art-HighIPPrologI**
 Kascuk P.A. Highly parallel Prolog interpreter based on generalized data flow model // Proc. 2nd Int. Logic Programming Conf. - Uppsala, 1984. - p. 195-205.
638. **Kolodner J.L. 1981 art-KnowIBSOM**
 Kolodner J.L. Knowledge-Based Self-Organizing Memory // Proc. 1981 Int. Conf on Cybernetics and Society. Atlan- ta, Georgia. October 26-28, 1981. Vol.1.-P.289-295
639. **Kreowski H.-J.. 1987 art2-iParalACP**
 Kreowski H.-J., Wilharm A. Is parallelism already concurrency? Part 2: non sequential processes in graph grammars // Graph Grammars and their Application to Computer Science. LNCS 291 / H.Ehrig, M.Nagl, G.Rozenberg, A.Rosenfeld, ed. - Springer-Verlag. - 1987. - P.361-377.

640. **Kreowski H.-J. 1987art1-iParalACP**
Kreowski H.-J. Is parallelism already concurrency? Part 1: derivations in graph grammars // Graph Grammars and their Application to Computer Science. LNCS 291 / H.Ehrig, M.Nagl, G.Rozenberg, A.Rosenfeld, ed. - Springer-Verlag. - 1987. - P. 343-360.
641. **Kunen K. 1987art-NegatILP**
Kunen K. Negation in logis programming // J. Logis Programming . - 1987. - 4. - N 3. - p. 289-308.
642. **Kъhn O.. 1998bk-CorpOM**
Къhn O., Abecker A., 1998. Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Challenges.
643. **Levesque H.. 1979art-ProceSfSN**
Levesque H., Mylopoulos J. A Procedural Semantics for Semantic Networks // Associative Networks. Representation and Use of Knowledge by Computers. - New York: Academic Press, 1979.-P 93-120.
644. **Lowe D.. 1999bk-HyperAW**
Lowe D., Hall W., 1999. Hypermedia and the Web: An Engineering Approach. Wiley, John & Sons, Incorporated.
645. **Luke S.. 1996art-OntoIBKD**
Luke S., Spector L., Rager D., 1996. Ontology-Based Knowledge Discovery on the World-Wide Web // In: Proc. Of the Workshop on Internet-based Information Systems, AAAI-96, Portland, Oregon.
646. **Macintosh A. 1997art-KnowlAM**
Macintosh A., 1997. Knowledge asset management // Airing. No. (20), April.
647. **Macintosh D.J.. 1987art-DistrDEFDES**
Macintosh Douglas J., Conry Susan E. A distributed development environment for distributed expert systems // 3rd Annu. Expert Syst. Gov. Conf, Washington, Oct.19-23, 1987. - P.72-79.
648. **Maida A.S.. 1982art-IntenCiPSN**
Maida A.S., Shapiro S.C. Intensional concepts in propositional semantic networks // Cogn. Sci. - 1982. - V. 6. - P. 291-330.
649. **Maikevich N.V.. 1998art-KnowlDP**
Maikevich N.V., Khoroshevsky V.F., 1998. Knowledge Driven Processing of HTML-Based Information for Intellectual Spaces on Web // In: Proceedings of the Third Joint Conference on Knowledge-Based Software Engineering, Smolenice, Slovakia.
650. **Maikevich N.V.. 1999art-IntellP**
Maikevich N.V., Khoroshevsky V.F., 1999. Intelligent Processing of Web Resources: Ontology-Based Approach and Multiagent Support // In. Proceedings of 1st International Workshop of Central and Eastern Europe on Multiagent Systems (CEEMAS'99). -1-4 June, 1999, St.-Petersburg, Russia.
651. **Moldovan D.I.. 1985art-SNAP**
Moldovan D.I., Tung Y.-W. SNAP: A VLSI Architecture for Artificial Intelligence Processing // J. Parallel Distribut. Comput. - 1985. - V. 2. - N 2. - P. 109-131.
652. **Moldovan D.I.. 1992art-ParalPAtoAI**
Moldovan D.I., Lee W., Lin C., Chung M. SNAP: Parallel Processing Applied to AI// Computer.-May 1992.-P.39-49.
653. **Moldovan D.I.. 1986art-ModelFPPoPS**
Moldovan D.I. A Model for Parallel Processing of Production Systems // Proc. of the Int. Conf. on Syst., Man and Cybern., IEEE. - Oct. 1986. - V. 1. - P. 568-573.
654. **Murakami K.. 1984art-ArchiAHS**
Murakami K., Kakuta T., Onai R. Architectures and hardware Systems: parallel inference machine and knowledge base machine // Proc. Int. Conf. 5-th Generation Computer Systems. - ICOT, ed. - 1984. - P. 18-36.
655. **Murakami K.. 1985art-ReseaOPMA**
Murakami K. et al. Research on Parallel Machine Architecture for Fifth-Generation Computing Systems // IEEE Computer. - 1985. - Vol. 18. - N 6. - P. 76-92.
656. **Mylopoulos J.. 1975art-SemanNaGoC**
Mylopoulos J., Cohen P., Borgida A., Sugar L. Semantic Networks and the Generation of Context // IJCAI-IV. - 1975. - V. 1. - P. 134-142.

657. **Nagl M. 1987 art-SetTAtGG**
 Nagl M. Set theoretic approaches to graph grammars // Graph Grammars and their Application to Computer Science. LNCS 291 / H.Ehrig, M.Nagl, G.Rozenberg, A.Rosenfeld, ed. - Springer-Verlag. - 1987. - P. 41-54.
658. **Ohbuchi R. 1988 art-OvervOAIAOPPR**
 Ohbuchi R. Overview of AI Application-oriented Parallel Processing Research in Japan // Parallel Computation and Computers for Artificial Intelligence. - Kluwer Academia Publishers. - 1988. - P. 247-260.
659. **Oldfield J.. 1986 art-LogicPaEA**
 Oldfield J. et al. Logic programs and an experimental architecture for their execution // IEE Proceedings. - May 1986. - Vol. 133. - Pt. E. - N 3. - P. 163-167.
660. **Onai R.. 1985 art-ArchiORBPI**
 Onai R. et al. Architecture of a Reduction-Based Parallel Inference Machine: PIM-R // New Generation Computing. - 1985. - V. 3. - N 2. - P. 197-228.
661. **Ontols-1998art**
 FIPA, 1998. Ontology Service. FIPA 98 Specification. Part 12. October, 1998. <http://www.cset.it/fipa>
662. **Oshisanwo A.O.. 1987 art-ParalMaA**
 Oshisanwo A.O., Dasiewicz P.P. A Parallel Model and Architecture for Production Systems // Proc. of the Int. Conf. on Parallel Processing. - Aug. 1987. - P. 147-153.
663. **ParalPaAI-1989bk**
 Parallel processing and artificial intelligence. - Chichester, Wiley. - 1989.
664. **Plander I. 1987 art-CompuAfAI**
 Plander I. Computer architectures for artificial intelligence in new generation computer system // Artificial Intelligence and Information-Control System of Robots, I.Plander (ed.). - North-Holland, 1987. - P. 77-92.
665. **Pospelov D.A.. 1997 art-KnowlISM**
 Pospelov D.A., Osipov G.S. Knowledge in semiotic models // Seventh Intern, conf. Ar-tif. Intell. and Information-Control systems of robots: Second workshop on applied semiotics. - Smolenice Castle, Slovakia. - 1997. - P. 1-12.
666. **Quillian M.R. 1968 art-SemanM**
 Quillian M. R. Semantic memory // Semantic Information Processing/M. Minsky; MIT Press.-Cambridge, Mass., 1968.-P.227-270.
667. **Rapaport W.J.. 1984 art-QuasiIR**
 Rapaport W.J., Shapiro S.C. Quasi-indexical reference in propositional semantic networks // Proc. 10-th Int. Conf. on Computational Linguistics (COLING-84). - 1984. - P. 65-70.
668. **Robinson I. 1988 art-PatteAM**
 Robinson I. Pattern Addressable Memory: Hardware for Associative Proccesing // Int. Workshop VLSI for Artificial Intelligence. - 1988.
669. **Rogers G. 1988 art-VisuaPwOar**
 Rogers G. Visual programming with objects and relations// Proceedings IEEE Workshop on Visual Languages. - October 1988. - P. 45-51.
670. **Rumelhart D.E.. 1973 art-ActivSN**
 Rumelhart D.E., Norman D.A. Active Semantic Networks as a Model of Human Memory // International Joint Conference on Artificial Intelligence - III. - Stanford University. - 1973. - P. 450-457.
671. **Sapaty P.S. 1986 art-WareLfPP**
 Sapaty P.S. A ware language for parallel processing of semantic networks // Computers and artificial intelligence. - 1986. - V. 5. - N 4. - P. 289-314.
672. **Sapaty P.S. 1987 art-WAVE-1**
 Sapaty P.S. The WAVE-1: a new ideology and language for distributed processing on graphs and networks // Compu-ters and artificial intelligence. - 1987. - V. 6. - N5.- P. 421-436.
673. **Schnitzler M. 1982 art-IsomoPoPS**
 Schnitzler M. The isomorphism problem is polynomially solvable for certain graph languages // Graph Grammars and their Application to Computer Science. LNCS 153 / H.Ehrig, M.Nagl, G.Rozenberg, ed. - Springer-Verlag. - Heidelberg. - 1982. - P. 369-379.

674. **Schonhage A. 1980 art-StoraMM**
Schonhage A. Storage modification machines // SIAM Jo-urn. Comput. - 1980. - V. 9. - N 3. - P. 490-508.
675. **Shapiro E.. 1983 art-ObjecOPiCP**
Shapiro E., Takeuchi A. Object-oriented programming in Concurrent Prolog // New Generation Computing. - 1983. - V. 1. - P. 25-49.
676. **Shapiro S.C. 1979 art-SnePS**
Shapiro S.C. The SNePS Semantic Network Processing Sys- tem // Associative Networks / N.V.Findler, ed. - New York: Academic Press. - 1979. - P. 179-203.
677. **Shibayama K.. 1987 art-LogicPLOPM**
Shibayama K et al. KPR: A Logic Programming Language-Oriented Parallel Machine // Logic Programming'87, Lecture Notes in Computer Science. - 1987. - N 315. - P.113-131.
678. **Sowa J.F. 1984 bk-ConceSIP**
Sowa J.F. Conceptual Structures - Information Processing in Mind and Machines. Addison-Wesley Publ.Comp., 1984.
679. **Stanfill C.. 1986 art-ParalFTS**
Stanfill C., Kahle B. Parallel free-text search on the connection machine system // CACM. - 1986. - N 12. - P.1229-1239.
680. **Sugie M.. 1986 art-HardwSoRBIM**
Sugie M. et al. Hardware Simulator of Reduction-Based Parallel Inference Machine PIM-R // Logic Programming'85, Lecture Notes in Computer Science. - 1986. - N 221. - P. 13-24.
681. **Takeuchi A.. 1986 art-Paral**
Takeuchi A., Furukawa K. Parallel logic Programming languages // Lect. Notes Comput. Sci., 225.-1986.-p.242-254.
682. **Tanaka H.. 1988 art-ReseaODFM**
Tanaka H., Amamiya M. The research of data-flow machines// Japan annual reviews in electronics, computers & telecommunications. - 1988. - V. 18. - P.15-27.
683. **Tarui T.. 1987 art-PreliePIM**
Tarui T. et al. A preliminary Evaluation of a Parallel Inference Machine for Stream Parallel Languages // Logic Programming'87, Lecture Notes in Computer Science. - 1987. - N 315. - P. 132-147.
684. **Tolia D. 1990 art-ServeACoPARLOG**
Tolia D. Survey and Comparison of PARLOG and Concurrent Prolog // SIGPLAN Notes. - 1990. - Vol. 25. - N 1. - P.33-42.
685. **Treleaven Ph. 1987 artCompuAfAI**
Treleaven Ph. Computer Architecture for Artificial Intelligence // Future Parallel Computers, Lecture Notes in Computer Science. - 1987. - N 272. - P. 416-492.
686. **Uschold M.. 1996 art- ONTOLOGIES**
Uschold M., Gruninger M., 1996. ONTOLOGIES: Principles, Methods and Applications, // Knowledge Engineering Review. Vol. 11, No. 2.
687. **Vagin V.N.. 1998 art-ClassLiSS**
Vagin V.N. Non-Classical Logics in Semiotic Systems // Proc. of Worcshop Applied Semiotics and Abstracts of CAI'98 Reports. Vol III, 1998. Pushchino, Russia, pp. 34-39.
688. **Villemin F.-Y. 1999 art-OntolBRIR**
Villemin F.-Y., 1999. Ontologies-based relevant information retrieval. – <http://www.cnam.fr/f-yv>.
689. **VisuaLaA-1990bk**
Visual languages and applications. - N.-Y.: Academic Press, 1990.
690. **VisuaLaVP-1990bk**
Visual languages and visual programming. - N.-Y.: Academic Press, 1990.
691. **Wiig K. 1996 art-KnowlM**
Wiig K., 1996. Knowledge management is no illusion! // Proc. of the First International Conference on Practical Aspects of Knowledge Management. Zurich, Switzerland: Swiss Informaticians Society.

692. *Yeremeyev A.P. 1997 art-Organ OSTKRM*

Yeremeyev A.P. Organization of Semiotic Type Knowledge Representation Model for Dynamic Decision Support Systems // Proc. of Seventh Int. Conf. 'Artificial Intelligence and Information-Control Systems of Robots' AIICSR'97. Second Workshop on Applied Semiotics, Sept. 15, 1997. Smolenice Castle, Slovakia, pp. 77-81.

693. *Yokota H.. 1986 art-Model AAfRKB*

Yokota H., Itoh H. A model and architecture for a relation knowledge base // Computer Architecture News. - 1986. - V. 14. - N 2. - P. 2-19.

Научное издание

Голенков Владимир Васильевич,
Елисеева Ольга Евгеньевна,
Иващенко Валерьян Петрович,
Казан Валентин Михайлович,
Гулякина Наталья Анатольевна,
Беззубенок Наталья Вячеславовна,
Лемешева Татьяна Леонидовна,
Сердюков Роман Евгеньевич,
Фоминых Игорь Борисович

**ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА
ЗНАНИЙ В
ГРАФОДИНАМИЧЕСКИХ
АССОЦИАТИВНЫХ
МАШИНАХ**

Под редакцией В.В. Голенкова

Редакторы: Е.Н. Батурчик, Н.А. Бебель, Т.Н. Крюкова, Т.А. Лейко

Компьютерная верстка: О.Е. Елисеева, В.П. Иващенко

Подписано в печать 22.11.2001. Формат 60x84 1/8. Бумага офсетная.

Печать офсетная. Усл.печ.л. . . Уч.-изд.л. . . Тираж 100 экз. Зак. .

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП № 156 от 05.02.2001. Лицензия ЛВ № 509 от 03.08.2001.
220013, Минск, П.Бровки, 6.

Отпечатано в Издательском центре БГУ. Лицензия ЛВ № 315 от 14.07.1998.

220030, Минск, ул. Красноармейская, 6.