OSTIS-2012
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

СЕМАНТИЧЕСКАЯ ТЕХНОЛОГИЯ КОМПОНЕНТНОГО ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ РЕШАТЕЛЕЙ ЗАДАЧ

Заливако С.С., Д.В. Шункевич

Белорусский государственный университет информатики и радиоэлектроники, г.Минск, Республика Беларусь

zalivako@mail.ru

shu.dv@tut.by

В работе приводится описание открытой семантической технологии проектирования интеллектуальных решателей задач. Отдельное внимание уделяется методике проектирования решателей и операций, являющихся составными частями таких решателей. Также в работе рассмотрено несколько примеров использования технологии при проектировании конкретных интеллектуальных систем по различным предметным областям.

Ключевые слова: интеллектуальная система, интеллектуальный решатель задач, логический вывод

Введение

В настоящее время большую актуальность имеет переход от ориентирования проектировщиков интеллектуальных систем с навязываемой (предлагаемой) машины обработки знаний на проектирование решателей задач из готовых компонентов. В основе этого подхода лежат общие принципы организации машин обработки знаний, позволяющих осуществить интегрирование различных моделей решения задач, как существующих, так и новых. Это предоставляет возможность реализации данной технологии на основе любых моделей решения задач.

В работе рассматривается методика проектирования интеллектуальных решателей задач, которая входит в состав комплексной открытой технологии проектирования интеллектуальных систем OSTIS [OSTIS, 2011].

1. Цель и задачи предлагаемой технологии

В основе данной работы лежит тезис о том, что различных интеллектуальных системах могут применяться различные модели решения задач. Гипотетически возможно существование универсального решателя задач, объединяющего в себе все известные способы и методы решения задач. Однако использование такого решателя в прикладных целях не является реальным и тем более целесообразным. Таким образом, наиболее приемлемым вариантом становится создание библиотеки совместимых между собой компонентов, из которых впоследствии может быть скомпилирован решатель, удовлетворяющий необходимым требованиям.

Основным требованием, предъявляемым при таком подходе к реализации моделей решения задач, является представление их в соответствующем формальном виде – как многоагентной системы, работающей над общей памятью. Приведение существующих моделей к такому формализму является нетривиальной задачей, хоть и не привносит радикальных изменений в текущее состояние теории решения задач в целом.

На основании сказанного выше, можно выдвинуть следующую гипотезу: любые модели решения задач могут быть представлены в описанном ранее формализме с целью максимального упрощения интеграции данных моделей между собой.

Целью данной работы является создание открытой семантической технологии компонентного

проектирования интеллектуальных решателей задач, позволяющей осуществить компиляцию готового решателя из имеющихся компонентов.

К рассматриваемой технологии выдвинуты следующие требования:
- открытость (технология является частью открытого проекта OSTIS);
- расширяемость (возможность свободного добавления новых компонентов);
- гибкость (пользователь может использовать лишь часть компонентов из предложенной библиотеки для обеспечения требуемого функционала разрабатываемой интеллектуальной системы);
- доступность (сравнительно неподготовленный пользователь может использовать технологию для создания интеллектуального решателя задач по интересующей его предметной области);
- универсальность (технология может быть использована в любых предметных областях, обеспечивая при этом полный функционал в случае необходимости);
- модульность (технология включает структурированную библиотеку компонентов обработки знаний);
- полнота (технология должна обеспечивать решение как можно большего набора задач из заданной предметной области за конечное время).
- асинхронность и параллельность (технология должна обеспечивать возможность параллельной асинхронной работы операций и, как следствие, возможность параллельного решения ряда задач)

Таким образом, при достижении поставленной цели возникают следующие проблемы:
- обеспечение предметной независимости и универсальности компонентов технологии и всей технологии в целом;
- обеспечение синхронизируемости отдельных компонентов технологии между собой;
- обеспечение синхронизируемости всей технологии в целом с другими направлениями проекта OSTIS, в частности с технологией проектирования баз знаний;
- обеспечение самодостаточности компонентов (или групп компонентов) технологии, т.е. способности их функционировать отдельно от других компонентов без утраты целесообразности их использования;
- обеспечение антропоморфности, и как следствие, доступности широкому кругу пользователей принципов и методов, рассматриваемых в рамках технологии при решении задач;
- возрастание времени решения задачи при расширении функционала интеллектуального решателя;
- возрастание времени решения задачи при расширении базы знаний системы.

Рассмотрим задачи, детализирующие подход к преодолению описанных проблем:
- разработка четкой системы правил проектирования операций интеллектуального решателя;
- уточнение языка вопросов, используемого при взаимодействии операций с другими компонентами технологии, в том числе и с другими операциями;
- разработка систем операций, обеспечивающих поиск решения задачи по одной из известных стратегий решения или их комбинации;
- разработка систем операций, обеспечивающих поиск решения задачи по одному из известных правил логического вывода или их комбинации, в рамках некоторой стратегии решения;
- реализация разработанных операций и их отладка;
- разработка классификации операций решателя;

Более подробно приведенные задачи описаны ниже.

2. Аналоги и преимущества предлагаемой технологии

В качестве аналогов предлагаемой технологии компонентного проектирования интеллектуальных решателей задач можно назвать следующих представителей:
- GPS (General Problem-Solver)
- QA3
- STRIPS (Stanford Research Institute Problem Solver)
- ПРИЗ (Пакет прикладных инженерных задач)
- ППР (Программа принятия решений)
- УДАВ (Универсальный делатель алгоритмов Варламова)

При разработке GPS авторов в основном интересовали вопросы, связанные с поисковой деятельностью человека, решающего задачи. Это привело к созданию известной эвристической стратегии поиска решений, используемой в различных дальнейших модификациях решателей. Однако, стремясь создать теорию мышления на подобной основе, авторы не уделили должного внимания другому важному аспекту теории — представлению знаний. В результате GPS не оказался универсальным решателем задач, на что надеялись его создатели. Решатель по существу имел процедурный язык низкого уровня, на котором, как показали, например, шахматы, далеко не всегда оказалось возможным эффективное описание сложных сред в терминах априори упорядоченных различий, таблиц связей, операторов и других элементов проблемной среды GPS. Поэтому, несмотря на довольно эффективную саму по себе стратегию поиска (анализ целей и средств, планирование и др.), система решала задачи медленно. Здесь сказалась нерешенность проблемы совмещения эффективной стратегии поиска с эффективным представлением знаний. [Ефимов, 1982]

Вопросно-ответная система QA3 может быть также названа многоцелевой системой решения задач или общим решателем задач. Она рассчитана на произвольную предметную область и произвольные вопросы, ее действие основано на автоматическом доказательстве теорем с использованием принципа резолюций. Но так как в рамках формализма метода резолюций оказалось затруднительным описание эвристик, то это обстоятельство заставило отказаться в QA3 от эвристического поиска. Таким образом,

попытка построить дедуктивный решатель, используя в полной степени формализм принципа резолюции, оказалась, как показала система QA3, также неуспешной. [Ефимов, 1982]

Система STRIPS (Stanford Research Institute Problem Solver), использует декларативно-процедуральное представление знаний в сочетании с эвристическим поиском. Эта особенность в сочетании с использованием макрооператоров, формируемых на основе обучения решателя STRIPS, позволила значительно повысить его эффективность. Улучшив, таким образом, стратегию поиска решений, авторы STRIPSa тем не менее не сумели решить ряд возникших па их пути проблем. Наиболее серьезной из них оказалась проблема так называемых побочных эффектов. Оказалось, что принципиально невозможно, оставаясь в рамках подобного описания действий, априори предусмотреть и описать полный эффект действий, т. е. что действительно меняется в результате применения данного оператора к конкретной ситуации. [Ефимов, 1982]

Ядром системы ПРИЗ (Пакет прикладных инженерных задач) служит организующая программа, не ориентированная априори на какую-либо предметную область. В наиболее общем режиме решатель по задаче, заданной текстом, формирует ее описание и далее составляет и исполняет решение задачи. Знания о предметной области составляют содержание пакета системы ПРИЗ и в процедурной форме представляют собой множество вычислительных моделей и программных модулей. Система ПРИЗ не планирует вычислительный процесс, составляющий решение заданной задачи, в полном объеме. Обычно в текстовом описании задачи содержится информация, по которой формируется управляющая программа, представляющая собой последовательность требуемых подзадач. Таким образом, ПРИЗ планирует решения только типовых подзадач при заданном скелете решения задачи в целом. [Кахро и др., 1988]

В системе ППР (Программа принятия решений) знания о предметной области представлены в пространстве признаков в виде растущих пирамидальных сетей (РПС), которые строятся автоматически. С помощью таких сетей удается хранить в системе необходимую информацию в компактном виде (общие для нескольких объектов признаки соответствуют одной вершине РПС), естественным образом организовать процедуру обучения системы в пространстве признаков и формировать понятия, характеризуемые своим объемом. В ППР поиск решений включает в число процедур построение дерева возможностей, эвристический поиск на дереве наилучшей ветви, анализ достижимости целей и механизм возврата в случае неудачи. Для увеличения эффективности поиска введены: двунаправленный поиск; представление в виде РПС знаний, описываемых на языке предикатов; процедура формирования рабочей информации в зависимости от решаемой задачи и процедуры формирования и применения макрооператоров. При всем при этом ППР представляет одноуровневую систему планирования и не использует процедурные языки, что не позволяет считать успешно решенной в этой системе проблему эффективного поиска. [Ефимов, 1982]

В программном комплексе «УДАВ» реализован «универсальный делатель алгоритмов Варламова». Этот метод базируется на миварной логической сети правил и представляет возможность активного обучения логического вывода, управляемого потоком данных, со снижением вычислительной сложности с N! (факториал) до линейной. «Универсальный делатель алгоритмов Варламова» работает со знаниями, представленными в виде продукционных правил и процедур. [Владимиров, 2010]

Одним из основных преимуществ предлагаемой технологии является ее ориентация на параллельную обработку знаний. Широкие возможности для реализации параллелизма обусловлены следующими моментами:

- Основными компонентами решателя являются sc-операции, по сути представляющие собой автономные самостоятельные агенты над общей памятью;
- Процедуры, реализующие операции решателя могут быть описаны как параллельные программы. Внутренний язык программирования SCP (semantic code programming) [Голенков и др., 2001], являющийся основным языком реализации процедур решателя, изначально является языком параллельного программирования;
- Сама концепция использования графодинамической ассоциативной памяти как среды взаимодействия операций предоставляет широкие возможности для параллелизма. Единственным условием в данном случае является наличие в реализации памяти стандартных механизмов синхронизации, например, таких как блокировки.

3. Состав технологии проектирования интеллектуальных решателей задач

В состав технологии входят следующие компоненты:

- модель интеллектуального решателя задач;
- библиотека ip-компонентов (intelligent property components – компонентов интеллектуальной собственности) решателя;
- система автоматизации проектирования;
- методика проектирования интеллектуальных решателей задач;
- help-системы поддержки проектирования интеллектуальных решателей задач;
- система управления коллективным проектированием интеллектуальных решателей задач.

Одним из достоинств семантической технологии компонентного проектирования интеллектуальных решателей задач является предметная независимость системы операций, используемых решателем, что

позволяет не привязываться к конкретной предметной области.

Далеко не все задачи возможно решить путем применения классической дедуктивной логики. В связи с этим технология предполагает возможность применения различных логических подходов к решению задач в различных предметных областях.

Некоторые логические подходы, которые используются в технологии:
- классическая дедуктивная логика;
- методы индуктивного вывода;
- абдуктивный вывод;
- нечеткие логики;
- правдоподобные рассуждения;
- логика умолчаний;
- темпоральная логика.

Различные логические подходы позволяют проектировать решатели задач для интеллектуальных систем в разных предметных областях, учитывая их специфику.

4. Модель интеллектуального решателя задач

В предлагаемом подходе к преодолению приведенных проблем решатель задач рассматривается в неклассическом варианте. В данном случае решатель задач представляет собой графодинамическую sc-машину (память в качестве модели представления знаний использует семантическую сеть), состоящую из двух частей:
- графодинамической sc-памяти;
- систему sc-операций.

Система операций является агентно-ориентированной и представляет собой набор sc-операций, условием инициирования которых является появление в памяти системы некоторой определенной конструкции. При этом операции взаимодействуют между собой через память системы посредством генерации конструкций, являющихся условиями инициирования для другой операции. При таком подходе становится возможным обеспечить гибкость и расширяемость решателя путем добавления или удаления из его состава некоторого набора операций. Более подробно процесс проектирования операций и предъявляемые к ним требования рассмотрены в соответствующем разделе.

Отличительной особенностью решателя задач как многоагентной системы в рамках данного подхода является принцип взаимодействия операций-агентов. Агенты обмениваются сообщениями исключительно через общую память путем использования соответствующего языка взаимодействия (языка вопросов-ответов), в отличие от большинства классических многоагентных систем, в которых агенты обмениваются сообщениями непосредственно друг с другом. В рассматриваемом подходе каждый агент, формулируя вопросную конструкцию в памяти, априори не знает, какой из агентов будет обрабатывать указанную конструкцию, а лишь дожидается появления в памяти факта окончания обработки вопроса. При этом в решении поставленной таким образом задачи может принимать участие целый коллектив агентов. Аналогичным образом, реагируя на появление некоторой конструкции в памяти, агент в общем случае не знает, кто из его коллег поставил данный вопрос, а лишь может проверить соответствие сгенерированной конструкции своему условию инициирования. В случае наличия такого соответствия, агент начнет обработку указанного вопроса (решение поставленной задачи), и в результате работы сгенерирует некоторый ответ на поставленный вопрос.

Проверка соответствия сгенерированного вопроса условиям инициирования агентов происходит следующим образом: автору вопроса после его формулирования необходимо инициировать данный вопрос (включить его во множество инициированных вопросов). После инициирования вопроса каждый из агентов, работающих в памяти, переходит в активное состояние и начинает проверку условия инициирования. При этом проверка начинается с наиболее уникальных фрагментов условия (например, типа вопроса) с целью оптимизации данного процесса. В случае установления факта изоморфности вопросной конструкции и условия инициирования агент начинает решение поставленной задачи, в противном случае агент переходит в состояние пассивного ожидания.

Описанная модель взаимодействия агентов в общей памяти позволяет обеспечить максимальную расширяемость системы агентов и предельно упростить процесс добавления новых агентов в уже имеющийся коллектив.

5. Библиотека универсальных совместимых ip-компонентов решателя

Центральным элементом семантической технологии проектирования интеллектуальных решателей задач является библиотека совместимых ip-компонентов.

Опишем структуру этой библиотеки.
- Библиотека готовых решателей задач:
  - решатель задач для экспертных систем, построенных на основе продукционной модели представления знаний;
  - решатель задач по геометрии;
  - решатель задач по теории графов;
  - и другие.

- Библиотека стратегий решения задач:
  - разбиение задачи на подзадачи:
    - поиск критерия разбиения задачи;
    - разбиение задачи по заданному критерию;
    - соединение решений, полученных в результате разбиения;
    - удаление дублирующихся знаний;

●●● и другие;
●● решение задачи с конца (стратегия обратного вывода)
●●● унификация;
●●● проверка противоречивости базы знаний;
●●● соединение И-подцелей;
●●● соединение ИЛИ-подцелей;
●●● получение решения;
●●● удаление промежуточных утверждений;
●●● и другие;
●● упрощение задачи (переход от формулировки в терминах предметной области к формулировке на логическом языке):
●●● операция обобщения;
●●● вывод обобщенного логического высказывания;
●●● фаззификация;
●●● дефаззификация;
●●● и другие;
●● случайный поиск и метод проб и ошибок (применимо в тех случаях, когда известно, что задача имеет небольшое число путей решения):
●●● определения числа возможных путей решения;
●●● применения пути решения;
●●● оценка эффективности пути решения;
●●● получения решения из всех путей;
●●● и другие;
●● использование правил для решения типовых задач:
●●● поиск подходящего класса задач для данной задачи;
●●● решения задачи методами найденного класса задач;
●●● поиск наиболее близкого класса задач для данной задачи;
●●● и другие;
●● метод деления пополам:
●●● разделение множества предполагаемых решений пополам;
●●● принятие решения об отказе от одной из половин;
●●● восстановление решения;
●●● и другие;
●● применение аналогии:
●●● генерация логического утверждения по аналогии;
●●● восстановление решения после применения аналогии;
●●● генерация фактов по аналогии;
●●● и другие;
●● метод поиска в глубину:
●●● операция поиска в глубину;
●●● генерация знаний на заданной глубине поиска;
●●● попытка решения задачи после генерации знаний;
●●● восстановление шагов решения после применения поиска;
●●● и другие;
●● метод поиска в ширину:
●●● операция поиска в ширину;
●●● построение дерева решений поиска в ширину;
●●● нахождения наиболее короткого решения (за наименьшее число шагов);
●●● поддержка очереди фактов;
●●● и другие;
●● перебор вариантов решения:
●●● генерация варианта решения;
●●● проверка варианта решения;
●●● выполнение «отката»;
●●● применение решения;
●●● восстановление решения;
●●● и другие;
●● генерация всех возможных следствий (прямой логический вывод):
●●● поиск всех возможных правил для применения;
●●● применение найденных правил;
●●● проверка новых фактов на то, что они отсутствуют в базе знаний;
●●● дополнение базы знаний сгенерированными фактами;
●●● восстановление решения;
●●● и другие;

● Библиотека операций:
●● логического вывода:
●●● генерация знаний на основании определения (эквиваленции);
●●● генерация знаний на основании продукции (импликации);
●●● генерация определения на основании двух импликаций;
●●● генерация более частного импликативного высказывания на основании более общего;
●●● интерпретация арифметического выражения;
●●● вывод обобщенного высказывания;
●●● и другие;
●● поисковые операции:
●●● операция поиска значения;
●●● операция поиска формулы для нахождения значения искомой характеристики;
●●● операция поиска доказательства;
●●● операции других машин интеллектуального поиска;
●●● и другие;
●● интерпретации хранимых способов решения задач:
●●● алгоритмов;
●●● процедурных программ;
●●● логических программ;
●●● нейронных сетей;
●●● генетических алгоритмов;
●●● методических указаний к решению задач;
●●● и других;
●● «сборки мусора»:
●●● удаления всех шаблонов, по которым осуществлялся поиск;
●●● удаления всех сгенерированных промежуточных логических утверждений;
●●● и другие;
●● мониторинга качества базы знаний:

- ••• устранение избыточности;
- ••• устранение противоречивости;
- ••• проверка полноты;
- ••• и другие.

- Библиотека базовых преобразований:
  - •• поиск изоморфоной конструкции по образцу;
  - •• генерация изоморфной конструкции по образцу;
  - •• поиск всех выходящих из узла дуг;
  - •• генерация тройки (узел-дуга-узел);
  - •• генерация пятерки (узел-дуга с атрибутом-узел);
  - •• поиск пятерки (узел-дуга с атрибутом-узел);
  - •• и другие.

- Библиотека программ, реализованных на различных языках программирования и на различных платформах

В языке SCP часто используются такие платформенно независимые процедуры, как:
- ***find_link_end*** – процедура поиска второго компонента связки бинарного отношения, если первый компонент известен;
- ***gen_meta_atom*** – процедура, генерирующая конструкцию, изоморфную заданной, причем переменным элементам первой конструкции соответствуют метапеременные элементы второй. Константные элементы не изменяются;
- ***merge*** – процедура, выполняющая слияние двух узлов в памяти системы;
- и другие.
- Платформенно зависимые процедуры, написанные на языке C++:
- ***createElement*** – процедура, выполняющая генерацию узла или дуги в sc-памяти;
- ***getContent*** – процедура получения содержимого узла;
- ***createIterator*** – создание итератора для просмотра элементов множества;
- и другие.

6. Средства автоматизации проектирования интеллектуальных решателей задач

В состав семантической технологии так же входят и инструментальные средства, которые позволяют автоматизировать процесс создания интеллектуальных решателей задач. Эти средства позволяют работать на более высоком уровне абстракции, чем текст программы на языке SCP. Перечислим основные возможности, которые имеет данное средство автоматизации:
- возможность включить (исключить) операцию из проектируемой системы операций интеллектуального решателя;
- верификация спроектированного набора операций;
- проверка операций на предметную независимость, универсальность;
- возможность отладки системы операций на конкретной базе знаний;
- профайлер производительности (отслеживание процессорного времени работы операций);
- профайлер памяти (отслеживание текущего состояния sc-памяти);
- просмотр стека вызовов (последовательность сгенерированных в памяти вопросов, операций, обрабатывающих данные вопросы и сгенерированные ответы на данные вопросы);

Таким образом, средства автоматизации позволяют разработчику интеллектуального решателя задач создавать наборы операций, которые реализовывают различные подходы к решению задач в рамках различных логических подходов.

7. Методика применения технологии при проектировании конкретных решателей задач

Технология проектирования интеллектуальных решателей задач основана на задачно-ориентированной методологии. В связи с этим проектирование системы операций состоит из четырех основных этапов:
- создание тестового сборника задач, которые решаются в рамках исследуемой предметной области;
- определение набора предметно независимых операций, которые будут использоваться при решении задач из тестового сборника;
- уточнение семантической спецификации каждой из указанных операций;
- реализация и отладка операций.

В общем случае можно выделить следующие предметно независимые классы задач
- Задачи синтеза доказательства
- Задачи верификации
- Задачи синтеза способа (алгоритма) решения
- Задачи анализа
- Задачи классификации

В качестве примера предметной области рассмотрим геометрию Евклида. Тогда классификация задач тестового сборника будет выглядеть следующим образом:
- по способу решения:
  - •• вычислительные задачи;
  - •• задачи на доказательство;
  - •• задачи на построение;
  - •• задачи на уточнение;
  - •• комбинированные задачи;
- по объекту решения:
  - •• геометрические точки;
  - •• прямые и отрезки;
  - •• треугольники;
  - •• многоугольники;
  - •• окружности;
  - •• и другие;
- по размерности пространства:
  - •• планиметрические;
  - •• стереометрические.

Семантическая спецификация операции представляет собой sc-конструкцию, которая описывает интерфейс взаимодействия проектируемой операции с другими операциями.

Данный интерфейс включает в себя:

- условие инициирования;
- узел, обозначающий sc-операцию;
- возможные результаты работы операции.

Формально спецификация операции описывается следующим образом:

- Сама sc-операция описывается предметным узлом, при этом указывается принадлежность операции множеству sc-операций.
- Связкой отношения ***«условие инициирования*»*** операция соединяется с конструкцией, которая является условием инициирования данной операции
- Связкой отношения ***«результат работы*»*** операция соединяется с логическим высказыванием, представляющим собой строгую дизъюнкцию атомарных высказываний (конструкций), каждое из которых может являться результатом работы данной операции, т.е. отражает изменения, произошедшие в памяти в результате работы данной операции.

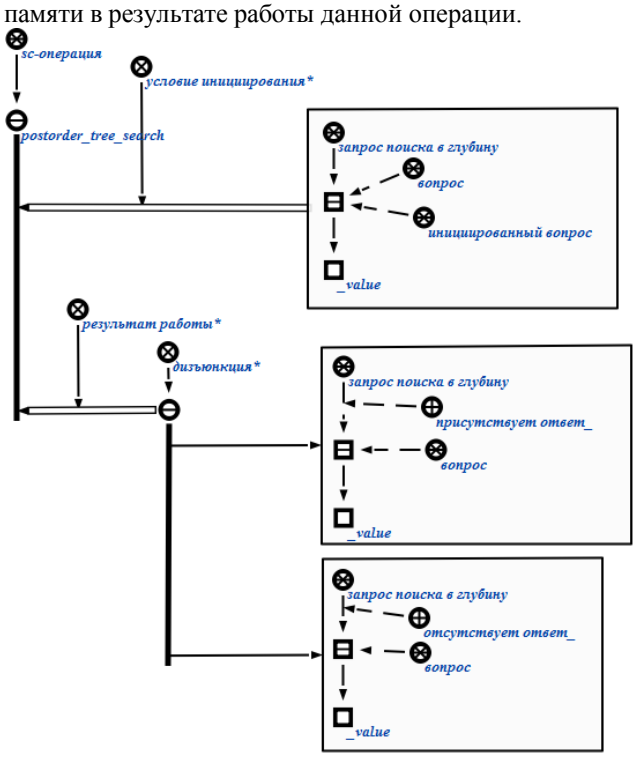


Рисунок 1 – Пример семантической спецификации операции поиска в глубину.

Реализацию операции условно можно разбить на 2 этапа:

- разработка алгоритма операции;
- реализация программы на специальном процедурном языке программирования SCP, предназначенном для обработки семантических сетей. При этом технология не ограничивает разработчика выборам именно данного языка для реализации, однако данный подход является наиболее оптимальным.

Тестирование sc-операций подразделяется на 3 этапа:

- подготовительный:
  - наполнение базы знаний (вносятся те знания, которые необходимы для работы операций);
- проверка совместимости:
  - проверка совместимости операции с прочими ip-компонентами системы. При этом ip-компонентами могут являться как определенные фрагменты базы знаний, так и другие операции, а также компоненты пользовательского интерфейса;
  - проверка правильности синхронизации с другими операциями;
- этап выявления и исправления ошибок:
  - локализация ошибки;
  - исправление ошибки.

8. Help-система для разработчиков интеллектуальных решателей

Семантическая технология предусматривает help-систему, которая позволяет обучать неподготовленного разработчика методике проектирования интеллектуальных решателей задач. Данная система представляет собой интеллектуальную справочную систему, построенную на базе открытой семантической технологии проекта OSTIS.

Help-система содержит знания о различных логических подходах и стратегиях, которые позволяют достигнуть решения той или иной задачи. Система помощи отслеживает действия пользователя и тем самым позволяет наилучшим образом подбирать подсказки и советы к каждому конкретному пользователю.

Help-систему по семантической технологии проектирования интеллектуальных решателей задач вполне можно считать полноценной справочной системой по логике. Что позволяет не только научить разработчика методике проектирования решателей, но и объяснить с формальной точки зрения процесс выбора операций, стратегий решения задач, логических подходов.

Help-система также содержит рекомендации по проектированию интеллектуальных решателей задач на основе имеющейся библиотеки, а также включает в себя подробную спецификацию указанной библиотеки и ее компонентов. Это позволяет считать рассматриваемую Help-систему одновременно руководством разработчика и пользователя.

9. Методика проектирования операций

Рассмотрим ряд принципов, соблюдение которых необходимо для корректности работы разрабатываемого решателя задач.

Каждая операция должна быть предметно независимой, т.е. в секции констант данной операции не должны быть описаны константы, имеющие отношение непосредственно к рассматриваемой предметной области. Исключение составляют понятия, которые могут использоваться

в различных предметных областях (например, отношения ***«включение*»*** и ***«часть-целое*»***). Данное правило может также быть нарушено в случае, если операция является вспомогательной и ориентирована на обработку какого-либо конкретного класса объектов (например, арифметические операции могут напрямую работать с конкретными отношениями ***«сложение*»*** и ***«умножение*»*** и т.п.). Всю необходимую для решения задачи информацию операция должна извлекать из семантической окрестности запроса.

Операция должна по возможности меньше ориентироваться на фиксированную форму представления фрагментов базы знаний, на работу с которыми она ориентирована. Степень глубины формализации и другие аспекты базы знаний определяются инженером по знаниям и не должны влиять на корректность работы операции. При обнаружении некорректности в представлении рассматриваемого фрагмента базы знаний операция, как и вся система не должна терять управления и, по возможности, сообщить пользователю о некорректности приведенных знаний.

Одна операция может состоять из ряда подпрограмм на выбранном языке программирования. Не стоит путать понятия ***«операция»*** и ***«программа»*** (***«подпрограмма»***). Подпрограмма не является агентом и должна вызываться другой подпрограммой. Операция представляет собой как минимум одну подпрограмму, которая имеет особый формат входных и выходных данных, автоматически реагирует на состояние sc-памяти и при необходимости запускает другие подпрограммы.

При проектировании подпрограмм следует учитывать возможность использования различными операциями одних и тех подпрограмм. Таким образом, появляется необходимость говорить не только о библиотеке sc-операций, но и библиотеках подпрограмм на различных языках программирования, например библиотеке scp-подпрограмм.

Каждая операция должна самостоятельно проверять полноту соответствия условия инициирования конструкции, имеющейся в памяти системы на данный момент. В процессе решения задачи может возникнуть ситуация, когда на появление одной и той же конструкции среагировали несколько операций. В таком случае выполнение продолжает только та операция, условие инициирования которой полностью соответствует сложившейся ситуации. Остальные операции обязаны в данном случае прекратить выполнение.

Выполнение предыдущего пункта достигается за счет тщательного уточнения спецификаций разрабатываемых операций. В общем случае условия инициирования у нескольких операций может совпадать, однако такая ситуация является очень нежелательной и может быть реализована в том случае, если операции не вносят критических изменений в ту область памяти, с которой работают остальные операции.

При проектировании систем операций интеллектуального решателя рекомендуется по возможности использовать операции, уже имеющиеся в библиотеке операций [1]. При необходимости реализации новой операции следует проектировать ее по возможности более общей, однако необходимо выделять в отдельные операции фрагменты рассуждений, которые могут быть использованы отдельно при решении другого класса задач.

Если в процессе работы операция генерирует в памяти какие-либо временные конструкции, то при завершении работы она обязана удалять всю информацию, использование которой в системе более нецелесообразно. Исключение составляют ситуации, когда подобная информация необходима нескольким операциям для решения одной задачи, однако после решения задачи информация становится бесполезной или избыточной и требует удаления. В данном случае ни одна из операций может оказаться не в состоянии удалить информационный мусор. В таком случае возникает необходимость говорить о специализированных вспомогательных операциях, задачей которых является уничтожение информационного мусора.

Операции необходимо объединять в группы для решения многоходовых задач, т.е. таких задач, для решения которых недостаточно всего одной операции. Очевидно, что под данное определение попадает большинство задач из практически любой предметной области. Группа операций является в некотором смысле самостоятельной подсистемой в рамках целостной системы операций.

При объединении операций в группы рекомендуется проектировать операции таким образом, чтобы они могли быть использованы не только в рассматриваемой группе. В случае, если это не представляется возможным и некоторые операции, будучи отделенными от группы, теряют смысл, необходимо указать данный факт при документировании рассматриваемых операций.

Инициатором запуска операции может бать как непосредственно пользователь системы, так и другая операция. При этом это никак не должно отражаться в работе самой операции. Необходимость вывода (трансляции) какого-либо фрагмента памяти пользователя отслеживается компонентами пользовательского интерфейса.

Язык взаимодействия операций через sc-память представляет собой подмножество языка вопросов. При расширении языка вопросов для введения какой-либо новой операции необходимо максимально сокращать sc-конструкцию, представляющую собой вопрос. В вопросе должна указываться только критически важная информация, все остальные знания операция должна находить самостоятельно в семантической окрестности вопроса. Это позволяет уменьшить сложность восприятия интерфейса операции потенциальным пользователем и унифицировать

форматы вопросов у большого числа различных операций.

При выделении в алгоритме решения задачи отдельных операций необходимо учитывать два фактора:

- операции должны быть как можно более универсальными, т.е. использоваться при решении как можно большего числа задач, что позволит избежать повторной реализации одних и тех же фрагментов рассуждений и уменьшит избыточность знаний в системе;
- операции должны быть по возможности антропоморфными, т.е. одна операция должна моделировать некий единый законченный акт мыслительной деятельности человека. Не следует искусственно увязывать ряд действий в одну операцию и наоборот, расчленять одно самодостаточное действие на поддействия. Это вызовет сложности восприятия принципов работы операции разработчиками и не позволит использовать операцию в ряде систем (например, в обучающих системах, которые должны объяснять ход решения пользователю);

Таким образом, в процессе разработке системы операций можно выделить следующие этапы:

- определение необходимого набора операций (с учетом уже имеющихся в библиотеке);
- определение ключевых узлов языка вопросов для связи операций через графодинамичекую память;
- составление спецификаций каждой из операций;
- реализация и тестирование операций;

10. Пример описания алгоритма выполнения операции

В качестве примера, описывающего особенности выполнения алгоритма, была выбрана операция получения значения продукции. Данная операция является одной из ключевых в системе операций прямого логического вывода.

Описание алгоритма работы операции:

1. Ищем узел связки конкретной теории.
   1.1. Если узел связки конкретной теории не найден, то переходим к шагу 2.
   1.2. Если узел связки конкретной теории найден, то получаем объект из запроса продукции.

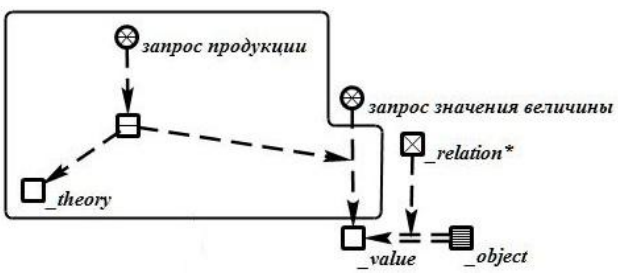


Рисунок 2 – Пояснение шага 1.2.

   1.2.1.Если объект из запроса продукции не найден, то переходим к шагу 2.
   1.2.2.Если объект из запроса продукции найден, то получаем условие (в конкретной теории находится под атрибутом 1_) и следствие (в конкретной теории находится под атрибутом 2_) из конкретной теории.
      1.2.2.1. Если условие и следствие не были получены, то переходим к шагу 2.
      1.2.2.2. Если условие и следствие получены, то получаем связанные узлы конкретной теории.
         1.2.2.2.1. Если связанные узлы конкретной теории не получены, то переходим к шагу 2.
         1.2.2.2.2. Если связанные узлы конкретной теории не получены, то происходит поиск фрагмента база знаний по шаблону (условие теории) с сохранением множества пар результатов поиска.
         1.2.2.2.3. Среди множества пар соответствия из шага 1.2.2.1.2. определяем ту, где под атрибутом 2_ имеем объект из шага 1.3.
         1.2.2.2.4. Из полученной пары из шага 1.2.2.2.3. получаем узел под атрибутом 1_.
         1.2.2.2.5. Ищем конкретный фрагмент базы знаний для объекта из шага 1.2. с сохранением множества пар результатов поиска.
         1.2.2.2.6. Формируем множество связок, где под атрибутом 1_ находится связанная переменная из условия теории, которая была найдена в шаге 1.2.2., а под атрибутом 2_ находится соответствующий узел из базы знаний.
         1.2.2.2.7. Для найденных пар из шага 1.2.2.2.5. ищем, где под атрибутом 1_ находятся связанные переменные из шага 1.2.2. и добавляем во множество связок из шага 1.2.2.2.6. связку, где под атрибутом 1_ находится узел из выше найденной пары под атрибутом 1_, под атрибутом 2_ находится узел из найденной пары под атрибутом 2_.
         1.2.2.2.8. Генерируем по следствию из теории константный контур, где заменяем связанные переменные из шага 1.2.2. на конкретные с помощью сформированного множества пар из шагов 1.2.2.2.6. и 1.2.2.2.7.
         1.2.2.2.9. Если множество связок к шагу 1.2.2.2.7. не сформировано, то переходим к шагу 2.
         1.2.2.2.10. Генерируем условие успешного завершения операции.

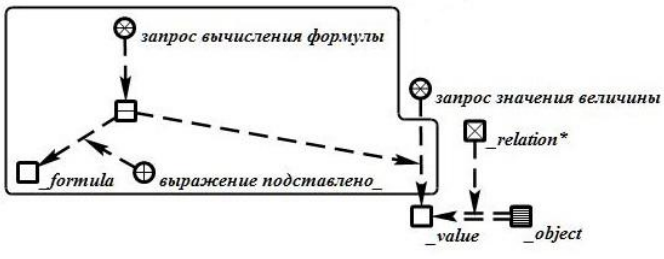


Рисунок 3 – Результат успешного завершения работы операции.

         1.2.2.2.11. Переходим к шагу 3.

2. Генерируем условие неудачного завершения операции.

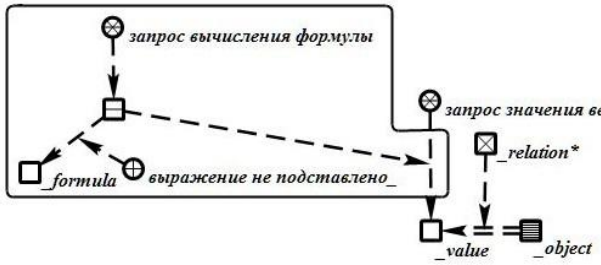


Рисунок 4 – Результат неудачного завершения работы операции.

3. Завершение работы операции.

11. Описание существующей реализации технологии

Рассмотрим более подробно текущую версию реализации предлагаемой технологии на примере разработанного группой авторов интеллектуального решателя задач. Как следует из самой концепции технологии, разработанный решатель является независимым от предметной области и представляет собой взаимосвязанную систему операций, работающих надо общей памятью.

На настоящий момент интеллектуальный решатель реализует смешанную стратегию решения, представляющую собой комбинацию традиционной продукционной модели и метода поиска в глубину.

Уровень логического вывода на данный момент представлен возможностью вывода на основе продукций вида «если…» - «то…» по правилу modus ponens. При этом возможен вариант преобразования найденной продукции в более частую форму, если это возможно с точки зрения исходных данных и не противоречит логической структуре утверждения. Также интеллектуальный решатель способен оперировать утверждениями, представленными в виде эквиваленции, такими как определения и утверждения о необходимости и достаточности. В указанном случае утверждение может быть применено как продукция, где в качестве посылки и заключения используются атомарные высказывания, входящие в указанное утверждение об эквиваленции. Роли данных высказываний определяются в зависимости от контекста в каждом конкретном случае.

Процесс решения задачи в текущей реализации можно разделить на следующие этапы:

• Этап работы поисковых операций.

Вне зависимости от типа задачи всегда имеется вероятность того, что данная задача уже была решена системой ранее или системе уже откуда-либо известен ответ на поставленный вопрос. На данном этапе работу осуществляет коллектив поисковых операций, каждая из которых, как правило, соответствует некоторому классу решаемых задач. Если ответ найден, решатель прекращает свою работу. В противном случае происходит переход на следующий этап решения.

• Этап применения стратегий решения задач.

На данном этапе осуществляется выбор между различными стратегиями решения задач, и, при необходимости, параллельный запуск различных стратегий. На данный момент, как уже было сказано, интеллектуальный решатель реализует комбинированную стратегию. Вначале рассматривается некоторый объект, для которого осуществляется поиск всех классов объектов, которым он принадлежит. Далее для каждого класса осуществляется поиск утверждений, справедливых для данного класса объектов (с целью оптимизации на настоящий момент данный факт должен быть явно указан проектировщиком базы знаний). При рассмотрении каждого утверждения осуществляется попытка применить его в рамках некоторой семантической окрестности рассматриваемого объекта, для чего осуществляется переход на следующий этап решения.

• Этап применения правил логического вывода.

На данном этапе происходит попытка применения утверждения, полученного на предыдущем шаге, с целью генерации в системе новых знаний. Если такое применение справедливо (например, посылка истинна) и имеет смысл (в результате применения будут сгенерированы новые знания), то осуществляется генерация новых знаний на основе одного из правил логического вывода. При этом применение происходит в контексте объекта, рассматриваемого на предыдущем этапе (в общем случае – ряда объектов). На данный момент реализован логический вывод на основе правила modus ponens. В будущем предполагается расширить набор подобных правил с целью увеличения количества различных типов утверждений, интерпретируемых решателем. Если в данном контексте вывод на основе данного утверждения невозможен или нецелесообразен, решение возвращается на предыдущий этап. В случае успешного применения утверждения происходит переход к следующему этапу решения.

• Этап оптимизации сгенерированных знаний и сборки мусора.

На данном этапе происходит интерпретация арифметических отношений, сгенерированных в процессе решения на предыдущем этапе, то есть попытка вычисления недостающих значений компонентов связок арифметических отношений (например, сложение величин и произведение величин) на основе имеющихся значений. Если вычислить все недостающие значения не представляется возможным, то все знания, сгенерированные на предыдущем этапе, уничтожаются и решение переходит на этап применения стратегий. В таком случае применение логического вывода для рассматриваемого на предыдущем шаге утверждения считается не целесообразным. Также на данном этапе происходит устранение синонимии, если таковая появилась на предыдущем этапе решения, например, сгенерирована связка отношения совпадения между некоторыми объектами. В конечном итоге происходит удаление конструкций, ставших ненужными и по каким-либо причинам не удаленных на предыдущих этапах решения.

Если все этапы решения выполнены успешно, то решение возвращается к первому этапу, и в случае, если ответ не получен, процесс повторяется еще раз. Стоит отметить, что в процессе решения один и тот же объект или одно и то же высказывание могут быть использованы многократно, если это целесообразно. Однако, очевидно, что применение одного и того же утверждения для одного объекта несколько раз не имеет смысла, при условии, что нужные знания из памяти не удаляются в процессе решения какими-либо сторонними операциями.

Одним из возможных вариантов стратегии решения задач является также использование интеллектуального пакета программ. В настоящее время данный подход в качестве эксперимента реализован в прототипе интеллектуальной справочной системы по теории графов. В указанном подходе после возникновения в памяти вопросной ситуации осуществляется просмотр спецификаций имеющихся в систему программ, ориентированных на решение какой-либо задачи. Программы могут быть реализованы как на внешних языках программирования, так и на языке SCP. В случае, если условие запуска программы соответствует вопросной конструкции, программа запускается на выполнение с соответствующими параметрами. При этом допускается возможность существования программ, необходимых для ответа на один и тот же вопрос, однако с различным количеством параметров. В результате работы программа генерирует в памяти некоторую ответную конструкцию. Применение интеллектуального пакета программ позволяет ускорить процесс решения задачи, однако увеличивает зависимость всего решателя от конкретной платформы или предметной области. Данный поход легко интегрируется с другими стратегиями решения задач и встраивается в общий процесс решения.

Как можно заключить из описанного процесса решения задачи, временная сложность решения напрямую не зависит от непосредственно количества объектов и утверждений, имеющихся в базе знаний, т.к. сам процесс решения осуществляется в некотором контексте вопроса, а не во всей базе знаний. Объем данного контекста определяется, во-первых, конкретной задачей, во-вторых, качеством проектирования базы знаний.

Практические результаты применения решателя показывают, что время решения одной задачи измеряется минутами. Существует несколько путей уменьшения времени решения задачи:

- Оптимизация исходных ходов реализованных операций и программ.
- Оптимизация процесса взаимодействия операций через общую память, модификация языка вопросов.
- Применение различных эвристик для оптимизации перебора и применения утверждений в базе знаний, а также выбора между различными стратегиями решения и моделями логического вывода.
- Оптимизация текущей реализации модели графодинамической ассоциативной памяти, а в конечном счете переход на аппаратную реализацию.

12. Примеры использования технологии

Рассмотрим процесс проектирования системы операций и использования технологии в рамках интеллектуальной справочной системы по геометрии Евклида.

Пример условия задачи:

Исходные данные:

- В треугольнике *Треугк(ТчкА;ТчкВ;ТчкС)* заданы три биссектрисы *Отр(ТчкА;ТчкА1)*, *Отр(ТчкВ;ТчкВ1)* и *Отр(ТчкС;ТчкС1)*.
- В треугольник *Треугк(ТчкА;ТчкВ;ТчкС)* вписана окружность *Окр(ТчкО;ТчкА2)*.
- Окружность *Окр(ТчкО;ТчкА2)* и треугольник *Треугк(ТчкА;ТчкВ;ТчкС)* имеют общие точки *ТчкА2*, *ТчкВ2*, *ТчкС2*.
- Длина отрезка *Отр(ТчкА;ТчкВ)* равна 12 см.
- Длина отрезка *Отр(ТчкА;ТчкС)* равна 10 см.
- Длина отрезка *Отр(ТчкА1;ТчкС)* равна 5 см.

Задача:

Определить длину радиуса окружности *Окр(ТчкО;ТчкА2)*.

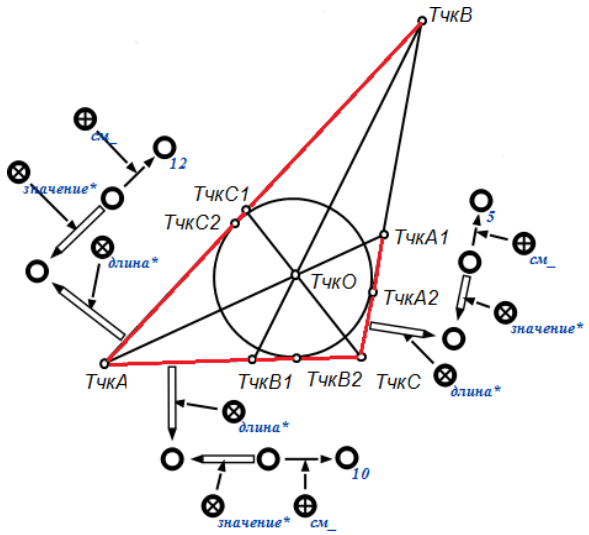


Рисунок 5 – Иллюстрация к задаче

Содержимое базы знаний системы (контекст решения задачи):

- теорема о биссектрисе (Биссектриса внутреннего угла треугольника делит противоположную сторону в отношении, равном отношению двух прилежащих сторон);
- формула вычисления длины отрезка как суммы длин двух отрезков, его составляющих;
- формула вычисления периметра треугольника как суммы длин трех его сторон;
- формула Герона для вычисления площади треугольника по длинам трех его сторон;

• формула вычисления радиуса вписанной в треугольник окружности как отношения площади треугольника к его полупериметру;

Далее представлено формальное описание условия задачи, а также формально описание некоторых фрагментов базы знаний. Полное описание всех фрагментов базы знаний можно найти на сайте OSTIS.

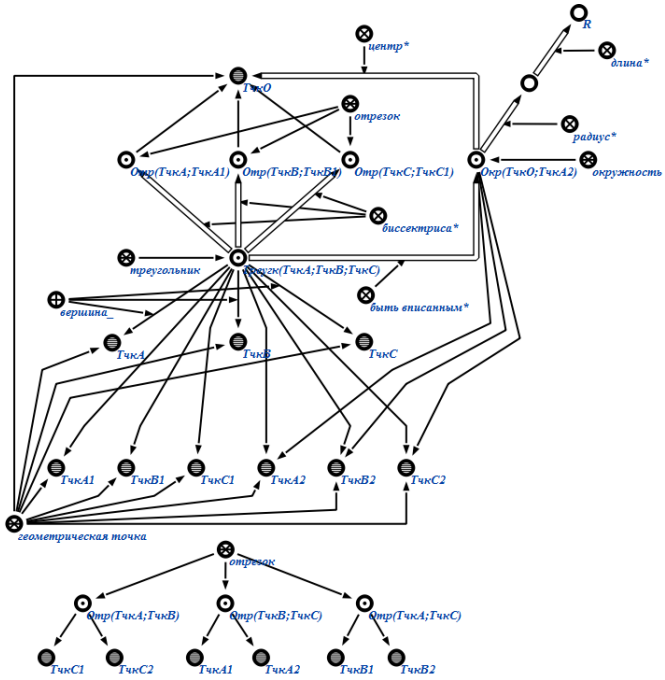


Рисунок 6 – Формальное описание условия задачи (фрагмент 1)

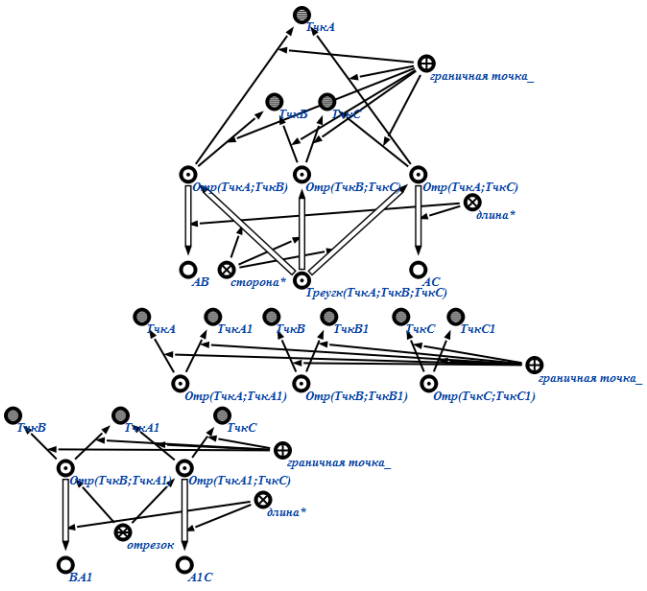


Рисунок 7 – Формальное описание условия задачи (фрагмент 2)

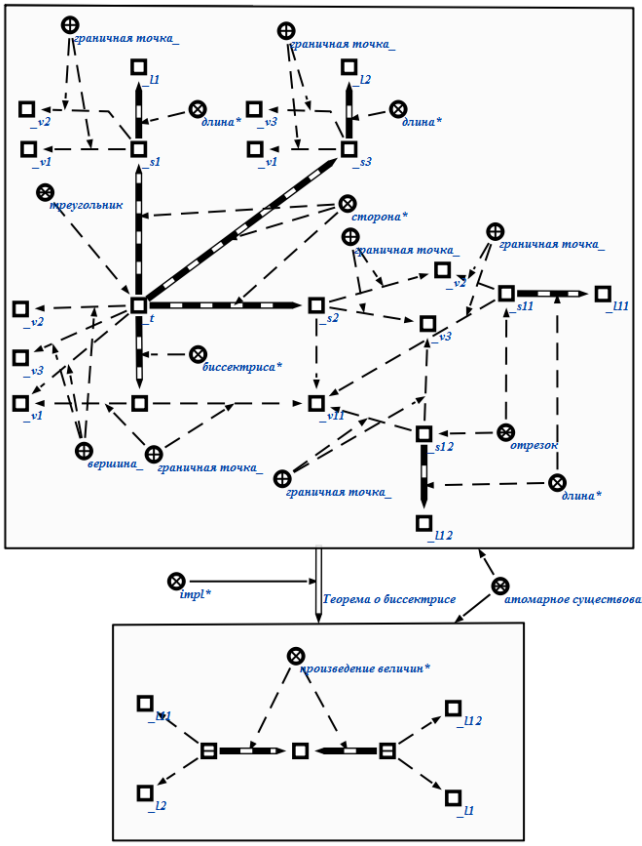


Рисунок 8 – Теорема о биссектрисе

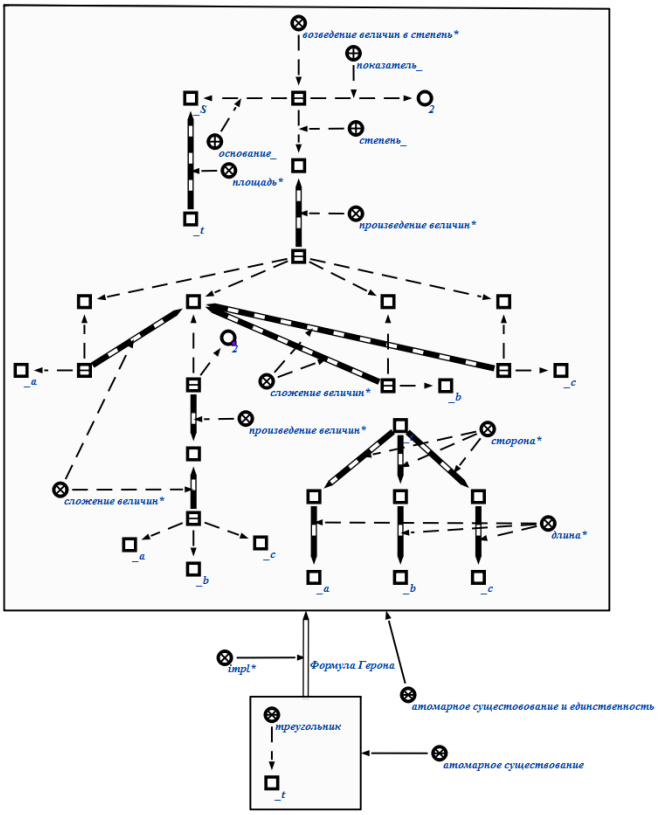


Рисунок 9 – Формула Герона

Для инициирования требуемого набора операций необходимо создать в памяти вопросную ситуацию:

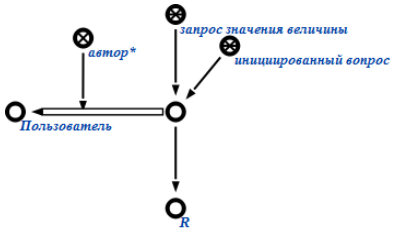


Рисунок 10 – Формат вопроса

Опишем краткий протокол решения задачи по шагам. Полную версию протокола решения данной задачи можно найти на сайте OSTIS.

- Шаг 1

Используемая операция

Операция поиска значения (find_value)

Пояснение

Операция пытается найти уже имеющееся значение требуемой величины

Требуемое значение отсутствует

Условие инициализации

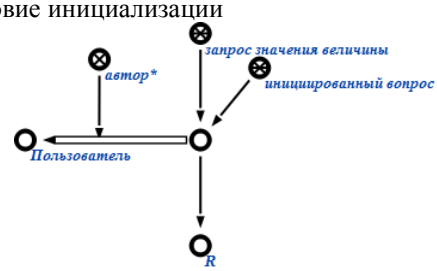


Рисунок 11 – Условие инициализации операции на шаге 1

Результат работы

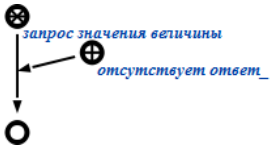


Рисунок 12 – Результат работы операции на шаге 1

- Шаг 2

Используемая операция

Операция, организующая поиск в глубину (postorder_tree_search_manager)

Пояснение

Операция организует запуск рекурсивной операции поиска в глубину.

Для этого на рассматриваемый объект устанавливается запрос поиска в глубину

Условие инициализации

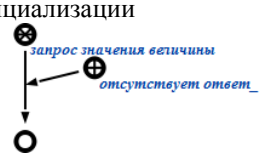


Рисунок 13 – Условие инициализации операции на шаге 2

Результат работы

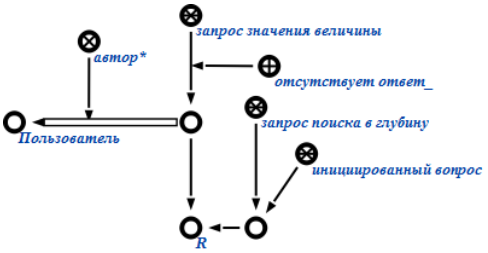


Рисунок 14 – Результат работы операции на шаге 2

- Шаг 3

Используемая операция

Операция поиска в глубину (postorder_tree_search)

Пояснение

Операция просматривает все объекты, связанные с исходным объектом и пытается сгенерировать новые знания. Если знания сгенерировать не удалось, запрос поиска в глубину устанавливается на объекты, связанные с данным. Просмотренные узлы добавляются в множество просмотренных узлов. В данном случае новые знания генерируются для объекта *Треугк(ТчкА;ТчкВ;ТчкС)*. При этом используется утверждение «*Теорема о биссектрисе*».

Условие инициализации

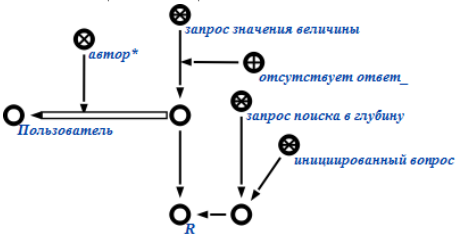


Рисунок 15 – Условие инициализации операции на шаге 3

Результат работы

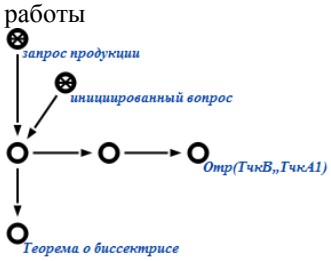


Рисунок 16 – Результат работы операции на шаге 3

- Шаг 4

Используемая операция

Операция получения значения продукции (find_value_production)

Операция интерпретации арифметического выражения (calculation)

Пояснение

На основании теоремы о биссектрисе вычисляется длина отрезка *Отр(ТчкВ;ТчкА1)* – 6см.

Условие инициализации

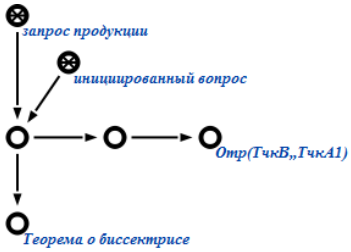


Рисунок 17 – Условие инициализации операции на шаге 4

Результат работы

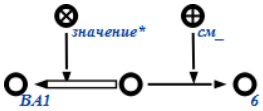


Рисунок 18 – Результат работы операции на шаге 4

- Шаг 5 Аналогичен шагу 1

  Используемая операция

  Операция поиска значения (find_value)
- Шаг 6 Аналогичен шагу 2

  Используемая операция

  Операция, организующая поиск в глубину (postorder_tree_search_manager)
- Шаг 7 Аналогичен шагу 3

  Используемая операция

  Операция поиска в глубину (postorder_tree_search)

  Пояснение
- В данном случае новые знания генерируются для объекта *Отр(ТчкВ;ТчкС)*. При этом используется утверждение «*Формула вычисления длины отрезка*»
- Шаг 8

  Используемая операция

  Операция получения значения продукции (find_value_production)

  Операция интерпретации арифметического выражения (calculation)

  Пояснение

  Вычисляется длина отрезка *Отр(ТчкВ;ТчкС)* как сумма длин отрезков *Отр(ТчкВ;ТчкА1)* и *Отр(ТчА1;ТчкС)* – 11 см.

  Результат работы

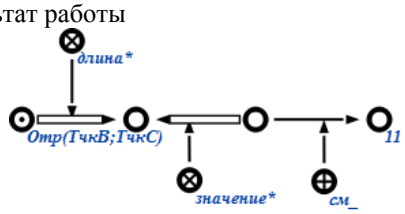


Рисунок 19 – Результат работы операции на шаге 8

- Шаг 9 Аналогичен шагу 1

  Используемая операция

  Операция поиска значения (find_value)
- Шаг 10 Аналогичен шагу 2

  Используемая операция

  Операция, организующая поиск в глубину (postorder_tree_search_manager)
- Шаг 11 Аналогичен шагу 3

  Используемая операция

  Операция поиска в глубину (postorder_tree_search)

  Пояснение

  В данном случае новые знания генерируются для объекта *Треугк(ТчкА;ТчкВ;ТчкС)*. При этом используется утверждение «*Формула вычисления периметра треугольника*»
- Шаг 12

  Используемая операция

  Операция получения значения продукции (find_value_production)

  Операция интерпретации арифметического выражения (calculation)

  Пояснение

  Вычисляется периметр треугольника *Треугк(ТчкА;ТчкВ;ТчкС)* – 33 см.

  Результат работы

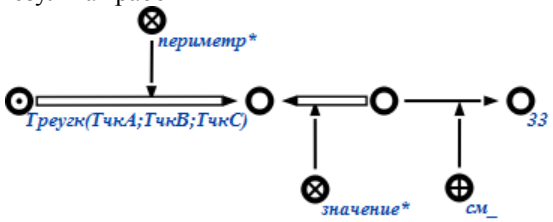


Рисунок 20 – Результат работы операции на шаге 12

- Шаг 13 Аналогичен шагу 1

  Используемая операция

  Операция поиска значения (find_value)
- Шаг 14 Аналогичен шагу 2

  Используемая операция

  Операция, организующая поиск в глубину (postorder_tree_search_manager)
- Шаг 15 Аналогичен шагу 3

  Используемая операция

  Операция поиска в глубину (postorder_tree_search)

  Пояснение

  В данном случае новые знания генерируются для объекта *Треугк(ТчкА;ТчкВ;ТчкС)*. При этом используется утверждение «*Формула Герона*»
- Шаг 16

  Используемая операция

  Операция получения значения продукции (find_value_production)

  Операция интерпретации арифметического выражения (calculation)

  Пояснение

  Вычисляется площадь треугольника *Треугк(ТчкА;ТчкВ;ТчкС)* – 51,52 см^2.

  Результат работы

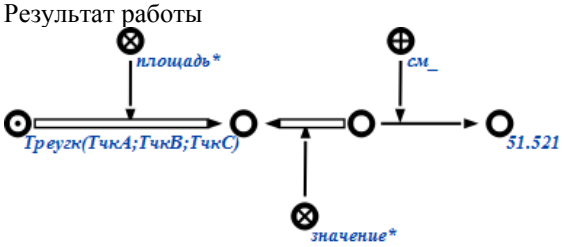


Рисунок 21 – Результат работы операции на шаге 16

- Шаг 17 Аналогичен шагу 1

  Используемая операция

Операция поиска значения (find_value)

- Шаг 18 Аналогичен шагу 2

Используемая операция

Операция, организующая поиск в глубину (postorder_tree_search_manager)

- Шаг 19 Аналогичен шагу 3

Используемая операция

Операция поиска в глубину (postorder_tree_search)

Пояснение

В данном случае новые знания генерируются для объекта *Окр(ТчкО;ТчкА2)*. При этом используется утверждение «*Соотношение площади треугольника и радиуса вписанной окружности*».

- Шаг 20

Используемая операция

Операция получения значения продукции (find_value_production)

Операция интерпретации арифметического выражения (calculation)

Пояснение

Вычисляется радиус окружности *Окр(ТчкО;ТчкА2)* – 3,1225 см.

Результат работы

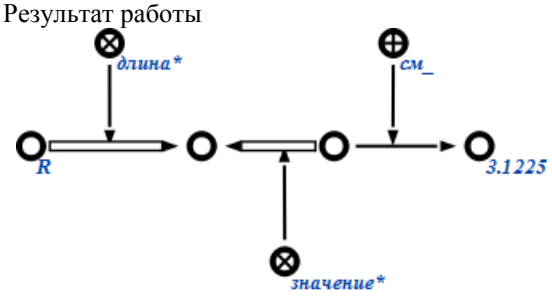


Рисунок 22 – Результат работы операции на шаге 20

- Шаг 21

Используемая операция

Операция поиска значения (find_value)

Пояснение

Операция пытается найти уже имеющееся значение требуемой величины

Требуемое значение присутствует

Результат работы

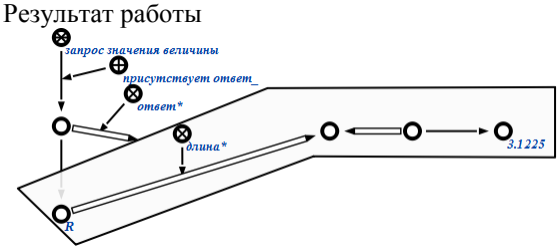


Рисунок 23 – Результат работы операции на шаге 21

Выбор операций, необходимых для решения задач в каждой конкретной прикладной интеллектуальной системе, определяется разработчиком интеллектуального решателя. В связи с этим некоторые операции, необходимые в одной предметной области будут избыточными в другой.

Например, операции нечеткого и правдоподобного вывода будут очень полезны в системах, где имеется много критериев для принятия решения, анализируется множество характеристик, которые просто невозможно описать с точки зрения однозначной истинности или ложности.

Эти же операции в геометрии Евклида, напротив, будут избыточно, т.к. решение задач осуществляется только по правилам классического вывода (дедуктивного, обратного и т.д.).

Заключение

Таким образом, семантическая технология проектирования интеллектуальных решателей задач позволяет проектировать такие системы предметно независимых операций, которые способны решать унифицированным образом множество различных задач одного класса. Каждая из операций представляет собой ip-компонент, который может быть использован в других прикладных системах. Многоагентная модель позволяет легко осуществлять интеграцию компонентов машин обработки знаний при условии корректной интеграции баз знаний. При этом никаких дополнительных действий по интеграции машин обработки знаний не требуется, т.к. грамотно разработанная операция многоагентной модели самостоятельно контролирует условие инициирования и текущее состояние памяти.

Основным практическим результатом изложенной работы является успешная попытка привести одну из существующих моделей решения задач к описанному выше формализму. Как показали результаты, данная задача вполне решаема. В процессе работы над проблемой был создан прототип универсального решателя задач, обладающий достаточно широким функционалом. Полученные результаты позволяют сделать предположение об истинности выдвинутой в разделе 1 гипотезы.

Основная цель подобной работы - позволить даже относительно неподготовленному (в области проектирования интеллектуальных систем) человеку создать интеллектуальную справочную систему по интересующей его предметной области, обладающую гибким функционалом, который определяется разработчиком на стадии проектирования.

Библиографический список

[Бенерджи, 1972] Бенерджи, Р. Теория решения задач/ Р. Бенерджи - М., Изд-во «Мир» 1972.

[Вагин, 1988] Дедукция и обобщение в системах принятия решений / Вагин В.Н; – М. :Изд-во «НАУКА», 1988.

[Вагин и др., 2008] Достоверный и правдоподобный вывод в интеллектуальных системах / Вагин В.Н. [и др.]; – М. : ФИЗМАТЛИТ, 2008.

[Вагин и др., 2010] Алгоритм абдуктивного вывода с использованием систем поддержки истинности на основе предположений / Вагин В.Н., Хотимчук К.Ю.; // Двенадцатая национальная конференция по искусственному интеллекту с международным участием. Труды конференции; - М. : ФИЗМАТЛИТ, 2010.

[Виноградов, 2010] Логика умолчаний как альтернатива модификационных исчислений / Виноградов Д.В.; // Двенадцатая национальная конференция по искусственному интеллекту с международным участием. Труды конференции; - М. : ФИЗМАТЛИТ, 2010.

[Владимиров и др., 2010] Владимиров А.Н., Варламов О.О., Носов А.В., Потапова Т.С. Программный комплекс "УДАВ": практическая реализация активного обучаемого логического ввода с линейной вычислительной сложностью на основе миварной сети правил //Труды научно-исследовательского института радио. - 2010.- №.1. С. 108-116.

[Гаврилова и др., 2001] Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. Учебник / Гаврилова Т.А.. [и др.]; – СПб. : Изд-во «Питер», 2001.

[Гладун, 1987] Гладун В.П. Планирование решений / В.П. Гладун; – Киев. : Изд-во «Наукова думка», 1987.

[Голенков и др., 2001] Голенков, В.В. Представление и обработка знаний в графодинамических ассоциативных машинах / Голенков В.В. [и др.]; под ред. В.В. Голенкова – Минск, 2001.

[Голенков и др., 2001] Программирование в ассоциативных машинах / Голенков В. В. [и др.]; под ред. В. В. Голенкова – Минск, 2001.

[Градштейн, 1965] Градштейн, И.С. Прямая и обратная теоремы / И.С. Градштейн; - М. : Наука, 1965.

[Грунский и др., 2010] Алгоритмы резолюции в логике высказываний при 0-1-ном представлении дизъюнктов / Грунский И.С., Волченко М.В.; // Двенадцатая национальная конференция по искусственному интеллекту с международным участием. Труды конференции; - М. : ФИЗМАТЛИТ, 2010.

[Гуц, 2003] Гуц, А.К. Математическая логика и теория алгоритмов/ А.К. Гуц ; - Омск : «Наследие. Диалог-Сибирь», 2003.

[Ефимов, 1982] Ефимов, Е. И. Решатели интеллектуальных задач / Е. И. Ефимов; - М. : Наука, 1982.

[Кахро и др., 1988] Инструментальная система программирования ЕС ЭВМ (ПРИЗ) / М.В. Кахро, А.П. Калья, Э.Х. Тыугу;– М., Изд-во «Финансы и статистика», 1988.

[Кулик, 2001] Кулик, Б. А. Логика естественных рассуждений / Б. А. Кулик; - СПб.: Изд-во «Невский диалект», 2001.

[Лакатос, 1967] Лакатос, И. Доказательство и опровержение / И. Лакатос; - М. : Наука, 1967. Пер. с английского.

[Маслов, 1986] Маслов, С.Ю. Теория дедуктивных систем и ее применения / С.Ю. Маслов; - М. : «Радио и связь», 1986.

[Найденова, 2010] Найденова К.А. Принципы организации правдоподобных рассуждений в интеллектуальных системах / Найденова К.А. // Двенадцатая национальная конференция по искусственному интеллекту с международным участием. Труды конференции; - М. : ФИЗМАТЛИТ, 2010.

[Непейвода, 2000] Непейвода Н.Н. Прикладная логика. Учебное пособие/ Непейвода Н.Н.; – Новосибирск. :НГУ, 2000.

[Нильсон, 1973] Нильсон Н. Искусственный интеллект. Методы поиска решений. Нильсон Н.; – М. :Мир, 1973.

[Нильсон, 1985] Нильсон Н. Принципы искусственного интеллекта. Н. Нильсон; – М. : «Радио и связь», 1985.

[Поспелов, 1981] Поспелов Д.А. Логико-лингвистические модели в системах управления / Д.А.Поспелов; – М. :Изд-во «Энергоиздат», 1981.

[Поспелов, 1989] Поспелов Д.А. Моделирование рассуждений. Опыт анализа мыслительных актов / Д.А.Поспелов; – М. :Изд-во «Радио и связь», 1989.

[Пойа, 1975] Пойа Д. Математика и правдоподобные рассуждения / Пойа Д.; – М. :Изд-во «НАУКА», 1975.

[Пойа, 1976] Пойа Д. Математическое открытие / Пойа Д.; – М. :Изд-во «НАУКА», 1976.

[Плесневич, 2006] Плесневич Г.С. Силлогистика для семантических сетей / Плесневич Г.С.; // Десятая национальная конференция по искусственному интеллекту с международным участием. Труды конференции; - М. : ФИЗМАТЛИТ, 2006, т.1. – с. 321-330.

[Плесневич, 2010] Плесневич Г.С. Нечеткая аристотелева логика / Плесневич Г.С.; // Двенадцатая национальная конференция по искусственному интеллекту с международным участием. Труды конференции; - М. : ФИЗМАТЛИТ, 2010.

[Рассел, Норвиг 2006] Рассел С., Норвиг П. Искусственный интеллект. Современных подход / Рассел С., Норвиг П. ; - М. : Вильямс, 2006.

[Розен, 1982] Розен В.В. Цель – оптимальность – решение / В.В. Розен; – М. :Изд-во «Радио и связь», 1982.

[Смирнов, 1972] Смирнов, В.А. Формальный вывод и логические исчисления / В.А. Смирнов; – М. : Изд-во «Наука», 1972

[Стефанюк, 2004] Стефанюк, В.Л. Локальная организация интеллектуальных систем / В.Л.Стефанюк; – М.: ФИЗМАТЛИТ, 2004.

[Столяр, 1965] Столяр А.А. Логические проблемы преподавания математики / Столяр А.А.; – Минск. : Изд-во «Вышэйшая школа», 1965.

[Столяр, 1971] Столяр А.А. Логическое введение в математику / Столяр А.А.; – Минск. :Изд-во «Вышэйшая школа», 1971.

[Столяр, 1982] Столяр А.А. Как математика ум в порядок приводит / Столяр А.А.; – Минск. :Изд-во «Вышэйшая школа», 1982.

[Столяр, 1987] Столяр А.А. Зачем и как мы доказываем в математике / Столяр А.А.; – Минск. :Изд-во «Народная асвета», 1987.

[Столяр, 1991] Математическая логика: Учеб. пособие / Л.А.Латонин, Ю.А.Макаренков, В.В.Николаева, А.А.Столяр. Под общ.ред. А.А.Столяра.; – Минск. :Изд-во «Вышэйшая школа», 1991.

[Тарасов, 2002] Тарасов, В.Б. От многоагентных систем к интеллектуальным организациям / В.Б. Тарасов; – М. :Изд-во УРСС, 2002.

[Теи и др., 1990] Логический подход к искусственному интеллекту / А. Теи, П. Грибомон, Ж. Луи, Д. Снийерс и др.; – М. :Изд-во «Мир», 1990.

[Туманов, 1969] Туманов, С.И. Поиск решения задач / С. И. Туманов; – М. :Изд-во «Просвещение», 1969.

[Тыугу, 1984] Тыугу, Э.Х. Концептуальное программирование / Э.Х. Тыугу;– М. :Изд-во «Наука», 1984.

[Хорошевский, 2008] Хорошевский, В.Ф. Пространства знаний в сети Интернет и Semantic Web (Часть 1) / В. Ф. Хорошевский // Искусственный интеллект и принятие решений. - 2008. - № 1. - С.80-97.

[Финн, 2008] Финн, В.К. Многозначные логики и их применение. Логические исчисления, алгебры и функциональные свойства / В.К. Финн. Том 1. М.: УРСС, 2008.

[Финн, 2008] Финн, В.К. Многозначные логики и их применения: Логики в системах искусственного интеллекта. / В.К. Финн. Том 2. М.: УРСС, 2008.

[Финн, 2010] Финн, В.К. Индуктивные методы милевского типа в системах искусственного интеллекта / Финн В.К.; // Двенадцатая национальная конференция по искусственному интеллекту с международным участием. Труды конференции; - М. : ФИЗМАТЛИТ, 2010.

[Финн, 2011] Финн, В.К. Искусственный интеллект. Методология, применение, философия / В.К. Финн. М.: Изд-во «Красанд», 2011.

[Эрдниев, 1998] Эрдниев, О.П. От задачи к задаче – по аналогии / Эрдниев О.П.; - Элиста : Калмыцкий государственный университет, изд. «Столетие», 2010.

[Kowalsky, 1975] Kowalsky, R. A. Proof procedure using connection graphs / R. A. Kowalsky // Journal of the ACM. – 1975 – № 22(4).

[OSTIS, 2011] Проект OSTIS [Электронный ресурс]. Минск, 2011. – Режим доступа: http://ostis.net/. – Дата доступа: 20.11.2011.

SEMANTIC TECHNOLOGY OF COMPONENTIAL DESIGNING OF INTELLECTUAL PROBLEM SOLVERS

Shunkevich D.V., Zalivako S.S.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

shu.dv@tut.by

zalivako@mail.ru

In this work there is a describing of open semantic technology of componential designing of intellectual problem solver. Single attention it is given to the designing technique of solvers and the operations which are components of such solvers. Also in this work there are several examples of technology using during the designing of concrete intellectual systems in different object domains.

INTRODUCTION

Now transition from orientation of designers of intellectual systems from the imposed (offered) knowledge processing machine on designing problem solvers from ready components has the big actuality. This approach is based on common principles of organization of knowledge processing machines, which allows performing integration of different problem solution models, both existing and new. This gives an opportunity of realization of given technology on the base of any problem solution model.

In this work it is represented the methodology of intellectual problem solvers designing, which is the part of complex open technology of intelligent systems designing OSTIS.

MAIN PART

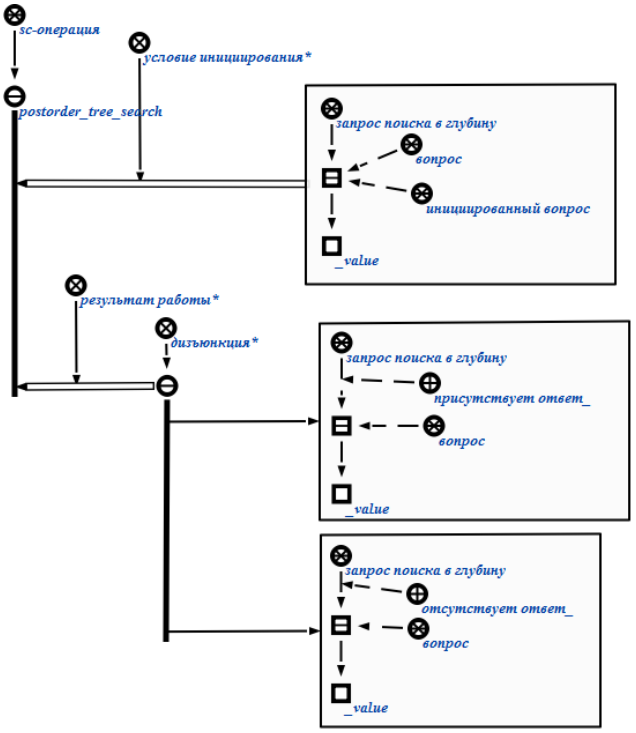
Here we will show the example of the semantic specification of the operation.

Semantic specification represents sc-construction, which describes an interface of interaction between the operation and other operations inside the system.

Interface includes:

- Initialization condition
- Node, designating the sc-operation
- Potential results of operation working

Formally, an operation is described in the next way:



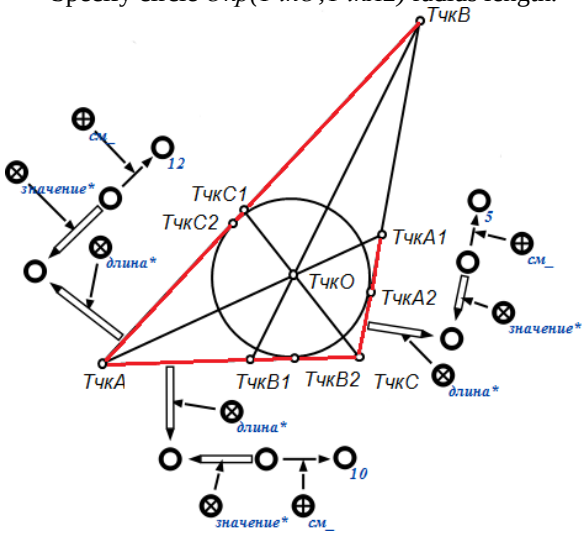
Drawing 24 – Semantic specification example. Postorder tree search operation.

Next, we will show an example of problem solution.
Problem situation:

- In triangle *Треугк(ТчкА;ТчкВ;ТчкС)* there are three bisectrixes *Отр(ТчкА;ТчкА1)*, *Отр(ТчкВ;ТчкВ1)* and *Отр(ТчкС;ТчкС1)*.
- In triangle *Треугк(ТчкА;ТчкВ;ТчкС)* circle *Окр(ТчкО;ТчкА2)* is inscribed.
- Circle *Окр(ТчкО;ТчкА2)* and triangle *Треугк(ТчкА;ТчкВ;ТчкС)* have common points *ТчкА2*, *ТчкВ2*, *ТчкС2*.
- Segment *Отр(ТчкА;ТчкВ)* length is 12 sm.
- Segment *Отр(ТчкА;ТчкС)* length is 10 sm.
- Segment *Отр(ТчкА1;ТчкС)* length is 5 sm.

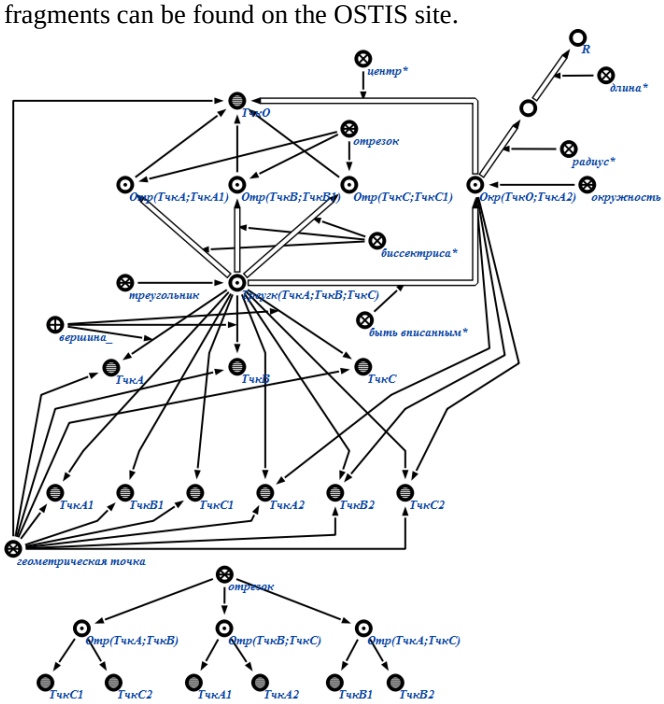
Problem:

Specify circle *Окр(ТчкО;ТчкА2)* radius length.

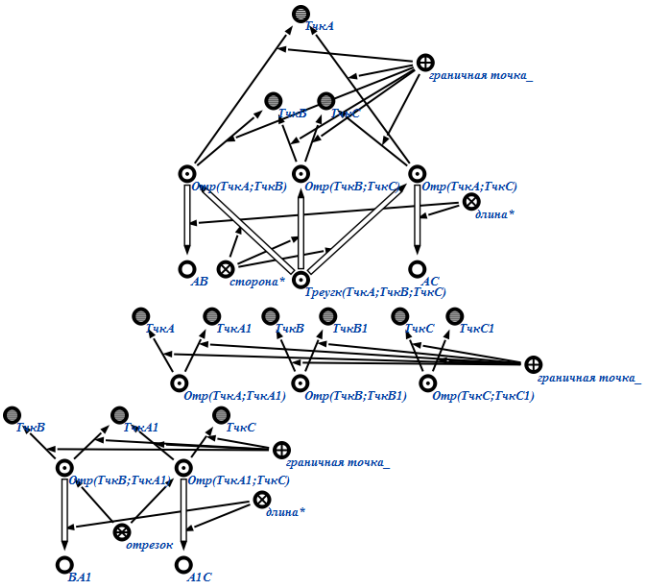


Drawing 25 – Problem illustration

Next there is represented formal problem situation description. Full description of all knowledge base fragments can be found on the OSTIS site.



Drawing 26 – Formal problem situation description (fragment 1)



Drawing 27 – Formal problem situation description (fragment 1)

CONCLUSION

Thereby, semantic technology of intellectual problem solvers designing allows to design such a systems of domain-independent operations, which are able to solve a number of problems from one class in unified way.