OSTIS-2013
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

МОДЕЛИ И СРЕДСТВА КОМПОНЕНТНОГО ПРОЕКТИРОВАНИЯ МАШИН ОБРАБОТКИ ЗНАНИЙ НА ОСНОВЕ СЕМАНТИЧЕСКИХ СЕТЕЙ

Шункевич Д.В.

Белорусский государственный университет информатики и радиоэлектроники,
г. Минск, Республика Беларусь
shu.dv@tut.by

В данной работе рассматриваются проблемы существующих методов, средств и технологий построения машин обработки знаний и ставится проблема отсутствия средств, позволяющих относительно неподготовленному разработчику в удовлетворительные сроки проектировать машины обработки знаний для прикладных интеллектуальных систем различного назначения. Далее рассматривается технология, призванная решить поставленную проблему путем интеграции различных методов и способов решения задач на общей формальной основе.
Ключевые слова: интеллектуальные системы; машины обработки знаний; решатель задач; компонентное проектирование; многоагентные системы.

Введение

В настоящее время особенно актуальными становятся проблемы обработки знаний в интеллектуальных системах. Вопросы представления знаний различного вида на настоящее время рассматриваются достаточно широко, существует большое количество языков представления знаний различной мощности и сложности, а также моделей представления знаний [Гаврилова и др., 2001].

Машина обработки знаний, включающая информационно-поисковую машину, интеллектуальный решатель задач и набор служебных операций обработки знаний (операции сборки мусора, выявления противоречий в базе знаний и т.д.), является важнейшей частью любой интеллектуальной системы, т.к. именно возможностями машины обработки знаний определяется функционал системы в целом, возможность давать ответы на нетривиальные вопросы пользователя и способность решать различные задачи.

Однако большинство прикладных интеллектуальных и экспертных систем [Гаврилова и др., 2001] имеют один и тот же недостаток – они не позволяют в должной мере обеспечить обработку тех знаний, которые в них содержатся.

Те же прикладные системы, которые обладают встроенной машиной обработки знаний, предоставляют пользователю жестко ограниченный функционал, заданный разработчиком на этапе проектирования системы. Примером таких машин обработки знаний может служить машина дедуктивного вывода, представленная в ряде экспертных систем [Ефимов, 1982], или машина нечеткого вывода.

Машина обработки знаний каждой конкретной системы во многом зависит от назначения данной системы, множества решаемых задач, предметной областью и другими факторами. Например, в системе, решающей задачи по геометрии, химии и другим естественным наукам обоснованным будет использование дедуктивных методов вывода, поскольку решение задач в таких предметных областях основывается только на достоверных правилах. В системах же медицинской диагностики, к примеру, постоянно возникает ситуация, когда диагноз может быть поставлен только с некоторой долей уверенности и абсолютно достоверным ответ на поставленный вопрос быть не может. В связи с этим ***возникает необходимость использования различных машин обработки знаний в различных системах***, при этом состав и возможности машины обработки знаний в конкретной системе определяется не только непосредственно разработчиком, а требует консультаций с экспертами в данной предметной области.

При проектировании машин обработки знаний интеллектуальных систем, как и при проектировании любых программных систем,

возникает ряд трудностей, связанных с переносимостью разработанного программного обеспечения на различные платформы, обеспечению возможности его последующей доработки (в том числе и сторонними разработчиками), универсализации методов принципов решения поставленных задач.

Основная проблема, рассматриваемая в данной работе, заключается в отсутствии средств, позволяющих относительно неподготовленному разработчику в удовлетворительные сроки проектировать машины обработки знаний для прикладных интеллектуальных систем различного назначения. Под неподготовленным разработчиком здесь понимается лицо, не имеющее специальной подготовки непосредственно в области разработки машин обработки знаний, однако имеющее представление об особенностях текущей предметной области и обладающее базовыми техническими навыками в работе с современными компьютерными средствами. Примером может служить эксперт-профессионал, тесно связанный с предметной областью, для которой разрабатывается система.

В связи с этим возникает необходимость создания универсальной технологии проектирования машин обработки знаний, обладающей следующими свойствами:

- Универсальность. Проектируемая технология должна обеспечивать возможности для обработки знаний и решения произвольных классов задач в различных предметных областях, не требуя при этом вмешательства пользователя данной разработки в ее внутреннее устройство.
- Модульность и расширяемость. Проектируемая технология должна предоставлять возможность расширения функционала системы, без изменения базовой модели машины обработки знаний.
- Кроссплатформенность. Проектируемая технология не должна зависеть от операционной системы и аппаратной архитектуры устройства, на котором предполагается работа информационной системы.
- Параллельность. Проектируемая технология должна обеспечивать возможность параллельного использования различных способов решения задач в рамках решения одной задачи, а также возможность параллельного решения сразу нескольких задач. При этом необходимо обеспечить согласованность и интегрируемость результатов применения различных методик решения задач.
- Обоснованность. Машина обработки знаний, в частности интеллектуальный решатель задач, построенный на базе предлагаемой технологии должен в случае необходимости указать пользователю правила вывода, на которых базируется решение той или иной задачи, другими словами построить алгоритм решения поставленной задачи в виде, понятном пользователю.

В данной работе рассматриваются основные принципы построения универсальной семантической технологии проектирования машин обработки знаний интеллектуальных систем, обладающей всеми описанными выше достоинствами. Данная технология является частью открытой семантической технологии проектирования интеллектуальных систем OSTIS [OSTIS, 2012].

1. Анализ существующих машин обработки знаний различного рода

В качестве наиболее заметных представителей машин обработки знаний в интеллектуальных системах можно указать следующих:

- 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)

1.1. Анализ системы GPS (General Problem-Solver)

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.

1.2. Анализ вопросно-ответной системы QA3

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.

1.3. Анализ системы STRIPS (Stanford Research Institute Problem Solver)

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]

1.4. Анализ системы ПРИЗ (Пакет прикладных инженерных задач)

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. [Ефимов, 1982], [Кахро и др., 1988]

1.5. Анализ системы ППР (Программа принятия решений)

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]

1.6. Анализ программного комплекса УДАВ (Универсальный делатель алгоритмов Варламова)

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 и др., 2010]

Следует отметить, что ни один из описанных примеров существующих машин обработки знаний не удовлетворяет всем требованиям, предъявленным к машинам обработки знаний во введении к данной статье.

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 [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.

Более подробно структура предлагаемой модели машины обработки знаний описана в следующем разделе данного проекта.

2. Общая модель и структура машин обработки знаний, построенных на основе технологии OSTIS

2.1. Унифицированная модель машин обработки знаний, разрабатываемых на основе технологии OSTIS.

В предлагаемом подходе к построению машин обработки знаний сама машина рассматривается в неклассическом варианте. В данном случае машина обработки знаний представляет собой графодинамическую sc-машину (память в качестве модели представления знаний использует семантическую сеть), состоящую из двух частей:

- графодинамическая sc-память;
- система sc-операций (sc-агентов).

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 функционала системы путем добавления или удаления из ее состава некоторого набора операций.

Отличительной особенностью машины обработки знаний как многоагентной системы в рамках данного подхода является принцип взаимодействия операций-агентов. По сути, предлагаемый подход реализует принцип «доски объявления», рассматриваемый в теории многоагентных систем [Тарасов, 2002]. Агенты обмениваются сообщениями исключительно через общую память путем использования соответствующего языка взаимодействия (языка вопросов-ответов, рассматриваемого далее), в отличие от большинства классических МАС, 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.

Следует также отметить немаловажный момент: для описания процедур, реализующих принципы работы того или иного агента (т.е. программ агента [Рассел, Норвиг]) используется специализированный язык SCP, построенный на базе SC-кода, как и в случае с представлением знаний, предназначенных для обработки. Такой подход имеет ряд преимуществ:

- И программа агента, и обрабатываемые знания, по сути, представлены на одном и том же языке. В связи с этим преобразование восприятий агента в его действия, описываемое функцией агента [Рассел, Норвиг 2006], значительно упрощается, т.к. отсутствует необходимость дополнительных преобразований во внутреннее представление агента;
- Так как алгоритм работы агента описан на том же языке, что и другие знания в системе, то появляется возможность модифицировать сам алгоритм того или иного агента прямо в процессе его работы. Это предоставляет широкие возможности для построения принципиально нового класса программ и, соответственно, агентов, способных к самоконфигурированию в процессе работы.

2.2. Иерархическая структуризация машин обработки знаний, разрабатываемых на основе технологии OSTIS.

Для определения структуры рассматриваемой модели машины обработки знаний рассмотрим

более подробно процесс поиска ответа на вопрос интеллектуальной системой.

Определим два основных понятия, используемых ниже.

Под ***стратегией решения задачи*** *понимается* общий, недетализированный план решения задачи, способ достижения поставленной цели

Стратегия решения задает принцип и порядок обхода объектов в рамках семантической окрестности вопроса.

Под ***операцией логического вывода*** понимается некоторый sc-агент, который получает на вход теоретико-множественную пару ***{S,O}***, где

S - логическое утверждение произвольной конфигурации

O - совокупность объектов, в семантической окрестности которых необходимо применить утверждение ***S***.

Целью такого агента является генерация в памяти новых знаний на основании уже имеющихся, т.е. по сути, применение утверждения ***S***.

Указанный процесс поиска ответа можно разделить на следующие этапы:

- Этап работы поисковых операций.

Вне зависимости от типа поставленного вопроса 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, подсистема обработки знаний 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. Целью работы каждой из стратегий является получение набора пар, связывающих некоторое множество объектов и логическое утверждение из базы знаний, которое справедливо для классов, которым принадлежат эти объекты в рамках некоторой теории. Впоследствии при рассмотрении каждого утверждения осуществляется попытка применить его в рамках некоторой семантической окрестности рассматриваемых объектов, 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). 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 верификации и 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 тоже 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.

Следует учитывать тот факт, что агенты сборки мусора, устранения синонимии и верификации знаний могут оказаться полезными и необходимыми не только на завершающем этапе работы интеллектуального решателя задач. В этом смысле 4-ый этап является несколько размытым и может быть частично интегрирован с какими-либо из предыдущих.

Таким образом, в структуре описываемой модели можно выделить 4 логических уровня, на

каждом из которых возможно использование методов параллельной обработки информации.

Следует также отметить, что использование такой многоуровневой модели позволяет улучшить производительность системы в целом за счет попеременного «выключения» некоторых уровней в процессе решения сложной задачи, т.е. искусственному переводу агентов, соответствующих некоторому уровню в неактивное состояние. К примеру, на этапе применения правил логического вывода могут быть отключены агенты, реализующие стратегии решения задач. Это уменьшит число ненужных срабатываний и проверок условий инициирования агентов при возникновении в памяти некоторого события. Однако такие действия могут негативно сказаться на возможностях системы в целом, к примеру, станет невозможным одновременное решение нескольких задач, поскольку разные задачи могут на один и тот же момент времени находиться на разных этапах решения, что делает отключение какого-либо этапа недопустимым.

Структуру такой параллельной асинхронной модели машины обработки знаний можно наглядно изобразить в виде следующей диаграммы:

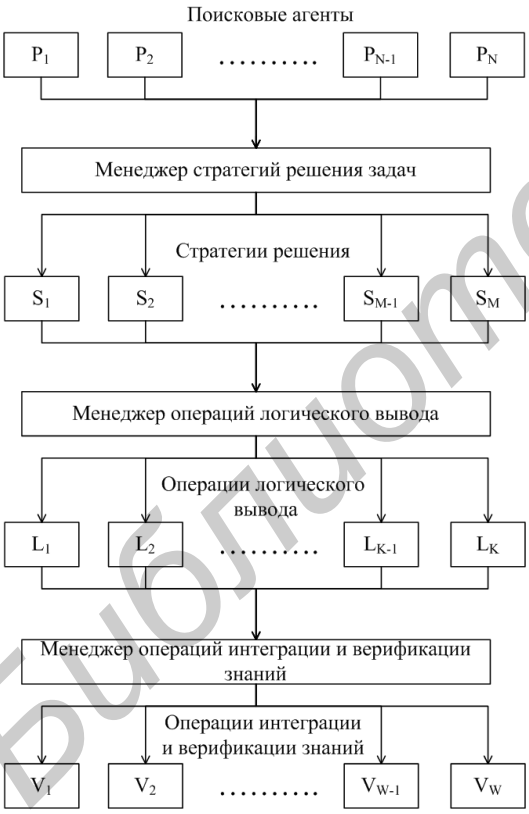

Рисунок 1 – Структура модели обработки знаний

В рамках каждого менеджера возможно использование принципа векторного параллелизма. Действительно, набор стратегий решения или операций логического вывода, по сути, представляет собой вектор. Каждый элемент данного вектора может быть рассмотрен по отдельности и активирован независимо от других в асинхронном режиме. Это позволяет говорить о возможности параллельного использования в рамках решения одной задачи нескольких стратегий, операций логического вывода, операций верификации и интеграции знаний. Таким образом, можно считать, что каждый из менеджеров реализует концепцию агента-супервизора, рассматриваемого в теории многоагентных систем, т.е. некоторого метаагента, основной задачей которого является координация действий других агентов. [Тарасов, 2002]

3. Классификация агентов

Совокупность агентов, осуществляющих обработку знаний в интеллектуальной системе, можно декомпозировать на основе различных критериев. В данной работе мы рассмотрим два основных способа классификации агентов.

По функциональному назначению агенты можно декомпозировать следующим образом:

- ***Поисковые агенты***. Основной задачей данного класса агентов, как следует из названия, является осуществление попытки поиска готового ответа на поставленный вопрос. Таким образом, для каждого класса решаемых системой задач необходимо наличие хотя бы одного соответствующего поискового агента.

- ***Агенты, реализующие стратегии решения***. К данному классу относятся агенты, которые реализуют принципы решения задач, заложенные в ту или иную стратегию решения. Для каждой имеющейся в системе стратегии необходимо наличие хотя бы одного агента, реализующего данную стратегию.

- ***Агенты логического вывода***. К данному классу относятся агенты, предназначенные для генерации новых знаний на основе некоторых логических утверждений. Количество и разнообразие таких агентов зависит от типологии логических утверждений, которые предполагается использовать в прикладной интеллектуальной системе.

- ***Агенты интерпретации программ***. Агенты данного класса предназначены для интерпретации программ, записанных как на внешних (с точки зрения системы), так и на внутренних языках системы. Количество таких агентов в конкретном случае зависит от интенсивности использования готовых программ в решении задачи и может равняться нулю, если готовые программы не предполагается использовать по каким-либо причинам.

- ***Агенты-мусорщики***. Агенты данного класса предназначены для удаления из памяти системы информационного мусора, т.е. конструкций, сгенерированных ранее каким-либо агентом, и потерявших актуальность либо семантическую ценность. Агенты-мусорщики могут ориентироваться либо на типовые конструкции,

либо на нетипичные конструкции, помеченные каким-либо специальным образом.

- ***Агенты интеграции знаний и устранения противоречий в базе знаний***. Данный класс агентов предназначен для обеспечения корректной интеграции в системе новых знаний с уже существующими, а также для верификации знаний. Агенты данного класса могут активироваться не в процессе решения системой конкретной задачи, а в произвольный момент при возникновении необходимости верификации знаний, как это указано в разделе 2.

На основании внутренней структуры агенты можно декомпозировать следующим образом:

- ***Атомарные агенты***. Под атомарным агентом понимается агент, не содержащий в своем составе других агентов.

- ***Составные агенты***. Под составным агентом понимается агент, в составе которого можно выделить ряд более простых агентов. Примером может служить агент, реализующий некоторую стратегию решения задач и имеющий в своем составе несколько программных агентов, выполняющих конкретные функции.

Следует отметить, что деление агентов на атомарные и неатомарные является достаточно условным и зависит от уровня детализации при анализе структуры машины обработки знаний.

4. Семантический унифицированный язык описания вопросов

Язык описания вопросов (или просто язык вопросов) предназначен для формулирования вопросов системе сторонними субъектами, каковыми могут быть как пользователи системы, так и сторонние интеллектуальные системы. Также и в рамках самой системы агенты обработки знаний могут формулировать вопросы, предназначенные для других подобных агентов.

Каждый вопрос в памяти системы описывается некоторой вопросной конструкцией, содержащей сам узел вопроса и дополнительную информацию, необходимую для поиска ответа на заданный вопрос – тип вопроса, аргументы и т.д.

Ниже приводятся описания ключевых узлов языка описания вопросов в рамках технологии OSTIS.

- Ключевой узел ***вопрос*** - знак множества вопросов произвольного вида, без уточнения конкретной семантики вопроса.

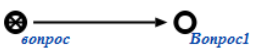

Рисунок 2 – Вопрос

- Семейство знаков множеств частных вопросов.

Примерами знаков множеств частных вопросов могут служить ***запрос значения величины***, ***запрос истинности*** и т.д.

Принадлежность какому-либо классу частных вопросов уточняет семантику данного конкретного вопроса. Классы вопросов можно разделить на предметно-зависимые (запрос яркости небесного тела) и предметно-независимые (запрос значения величины). Однако такое деление достаточно условно, потому как, например, запрос температуры объекта или запрос скорости объекта могут быть использованы далеко не только в системе по физике.

Любой класс частных вопросов является подмножеством множества вопросов.

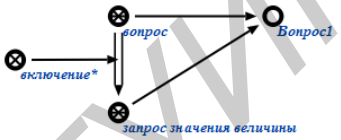

Рисунок 3 – Частный вопрос

- Аргументы вопроса.

Аргументами вопроса могут быть любые элементы базы знаний системы. В зависимости от конкретного класса вопроса их количество может варьироваться от 0 до десятка. Аргумент вопроса с теоретико-множественной точки зрения является его элементом, т.е. любой вопрос представляет собой множество аргументов (иногда – пустое множество).

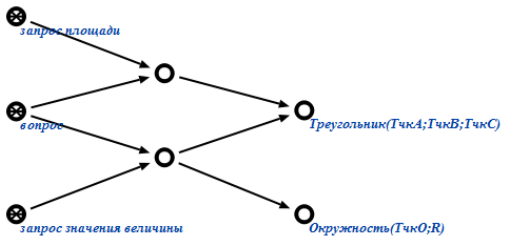

Рисунок 4 – Аргументы вопроса

При необходимости роль каждого аргумента может уточняться при помощи соответствующих ролевых отношений:

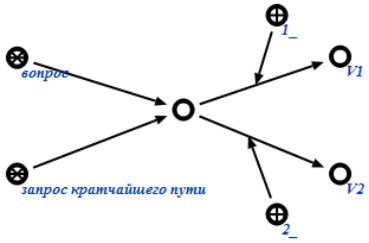

Рисунок 5 – Роль аргументов вопроса

- Ключевой узел ***автор****.

Является знаком отношения, связывающего вопрос автора данного вопроса (т.е. субъекта, сгенерировавшего в памяти соответствующую вопросную конструкцию).

Указание автора является необходимым в ряде случаев, например, когда вопрос задан пользователем через интерфейс, и ответ также должен быть выдан в нужное окно пользовательского интерфейса.

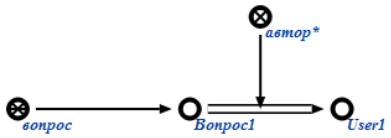

Рисунок 6 – Автор вопроса

- Ключевой узел ***ответ****.

Является знаком отношения, связывающего конкретный вопрос и некоторую конструкцию, являющуюся ответом на данный вопрос.

Вид этой конструкции определяется конкретным классом вопроса. Наличие ответа на вопрос (конечно, при условии возможности его получения) является необходимым независимо от автора, класса вопроса и прочих параметров.

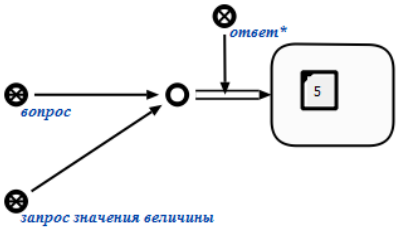

Рисунок 7 – Ответ на вопрос

- Ключевой узел ***решение****.

Является знаком отношения, связывающего конкретный вопрос решением. Решение представляет собой набор связок, каждая из которых содержит информацию о том, какое утверждение было использовано на данном шаге решения для какого объекта, и какая конструкция получена в итоге.

На указанных связках задано отношение строго порядка, обеспечивающее возможность определения последовательности шагов решения конкретной задачи.

В некоторых случаях решение может отсутствовать даже в случае успешного ответа на вопрос, к примеру, в том случае, если решение задачи ограничено простым поиском.

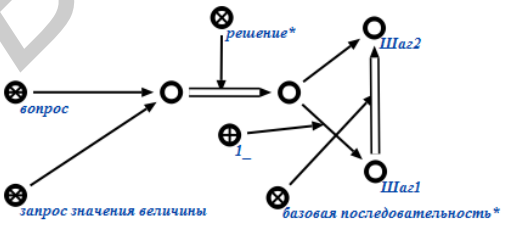

Рисунок 8 – Решение задачи

- Ключевой узел ***инициированный вопрос***.

Является знаком множества инициированных вопросов. Тот факт, что вопрос инициирован, свидетельствует о том, что вопросная конструкция полностью сформирована и машина обработки знаний может приступать к поиску ответа на поставленный вопрос. Таким образом, генерация дуги принадлежности вопроса множеству инициированных вопросов должна являться завершающим шагом построения любой вопросной конструкции.

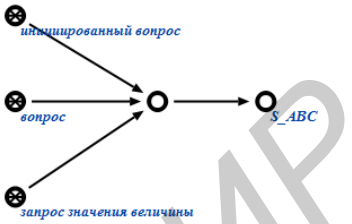

Рисунок 9 – Инициированный вопрос

- Ролевое отношение ***присутствует ответ'***.

Используется для указания того факта, что системе удалось найти ответ на поставленный вопрос. Роль указывается для вопроса в рамках некоторого класса частных вопросов.

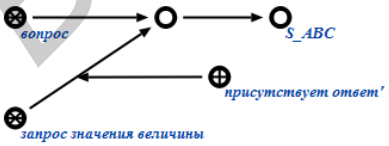

Рисунок 10 – Присутствие ответа на вопрос

Предполагается, что факт присутствия ответа указывается после того, как ответная конструкция полностью сформирована и сгенерирована связка отношения ***ответ****.

- Ролевое отношение ***отсутствует ответ'***.

Используется для указания того факта, что ответ на поставленный вопрос на настоящий момент найден быть не может. Роль указывается для вопроса в рамках некоторого класса частных вопросов.

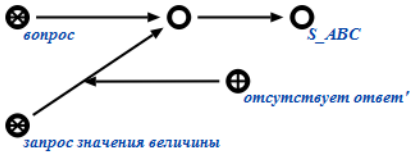

Рисунок 11 – Отсутствие ответа на вопрос

Для дополнительной синхронизации между агентами машины обработки знаний факт отсутствия ответа может уточняться более частными ролевыми отношениями, например, ***ответ отсутствует в явном виде'***.

5. Семантический унифицированный язык спецификации агентов машин обработки знаний

Семантическая спецификация агента машины обработки знаний позволяет менеджеру агентов определить необходимость запуска того или иного агента для решения конкретной задачи (подзадачи), а также установить порядок запуска агентов для решения одной и той же задачи для обеспечения большей эффективности.

Ниже приводятся описания ключевых узлов языка спецификации агентов машин обработки знаний в рамках технологии OSTIS.

- Ключевой узел ***стратегия решения задач***.

Является знаком множества стратегий решения задач. Используется менеджером стратегий для доступа к возможным стратегиям решения задач.

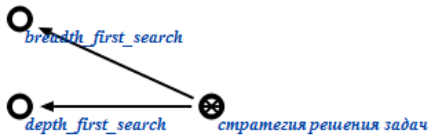

Рисунок 12 – Стратегии решения задач

- Ключевой узел ***операция логического вывода***.

Является знаком множества операций логического вывода. Используется менеджером операций логического вывода к возможным операциям логического вывода.

Рисунок 13 – Операции логического вывода

- Ключевой узел ***условие инициирования****.

Является знаком отношения, связывающего конкретную sc-операцию и условие ее инициирования, т.е. шаблон конструкции наличие которой в памяти системы является критерием активации данной sc-операции. Условие инициирования используется соответствующим менеджером операций для определения целесообразности запуска той или иной операции в данный момент.

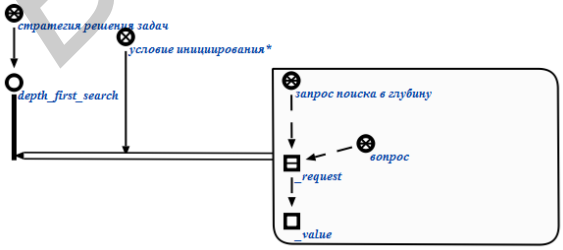

Рисунок 14 – Условие инициирования

- Ключевой узел ***результат работы****.

Является знаком отношения, связывающего конкретную sc-операцию и множество результатов ее выполнения. Указанное множество, как правило, представляет собой связку отношения ***строгая дизъюнкция****, поскольку несколько результатов работы не могут появиться в результате одного запуска sc-операции.

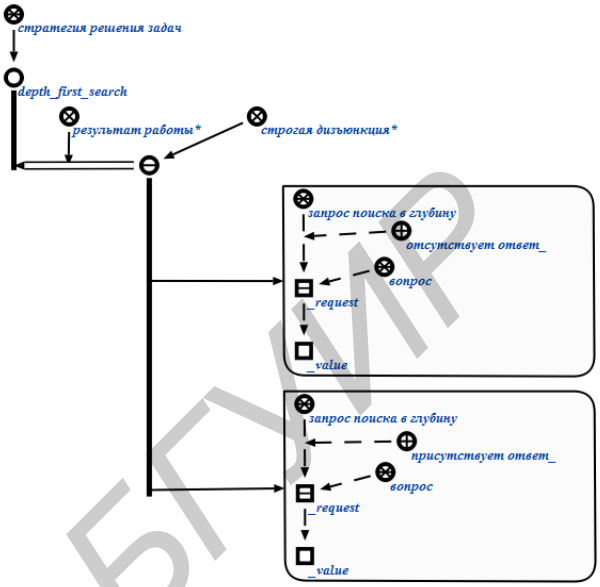

Рисунок 15 – Результат работы

Результат работы представляет собой шаблон, изоморфный конструкции, которая может быть сгенерирована агентом машины обработки знаний в семантической окрестности вопроса в процессе работы.

Результаты работы также могут быть использованы менеджерами операций для установления целесообразности запуска той или иной sc-операции в данный момент.

- Ключевой узел ***приоритет запуска****.

Является знаком отношения связывающего различные sc-операции с целью указания того факта, какая из операций является более приоритетной для запуска.

Отношение, как вытекает из определения, является транзитивным, антисимметричным и антирефлексивным, т.е. есть отношением строго порядка [Кузнецов, 2009].

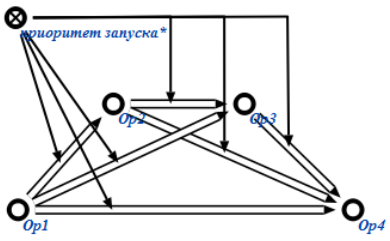

Рисунок 16 – Приоритет запуска агентов

Как видно, множество агентов, связанных данным отношением, образуют стандартный граф предпочтений [Поспелов, 1994], в котором узлам

соответствуют sc-операции, а дуги устанавливают приоритет запуска.

6. Классификация методов решения задач в интеллектуальных системах

Проблема автоматического решения задач достаточно давно рассматривается в работах по искусственному интеллекту. Приведем краткую классификацию существующих методов решения задач, рассмотренных в литературе:

- Классический дедуктивный вывод.

Классический дедуктивный вывод является наиболее популярным при построении автоматических решателей задач, так как всегда дает достоверный результат. Дедуктивный вывод включает в себя прямой и обратный и логический вывод (принцип резолюции, процедуру Эрбрана и др.) [Вагин и др., 2008], все виды силлогизмов [Малыхина, 2002] и т.д. Основной проблемой дедуктивного вывода является невозможность его использования в ряде случаев, когда отсутствуют достоверные правила вывода.

- Индуктивный вывод.

Индуктивный вывод предоставляет возможность в процессе решения использовать различные предположения, что делает его удобным для использования в слабоформализованных и трудноформализуемых предметных областях, например при построении систем медицинской диагностики. Подробно принципы индуктивного вывода рассмотрены в [Кулик, 2001], [Пойа, 1975].

- Абдуктивный вывод.

Под абдуктивным выводом в искусственном интеллекте, как правило, понимается вывод наилучшего абдуктивного объяснения, т.е. объяснения некоторого события, ставшего неожиданным для системы. Причем «наилучшим» считается такое объяснение, которое удовлетворяет специальным критериям, определяемым в зависимости от решаемой задачи и используемой формализации. Абдуктивный вывод подробно рассматривается в [Вагин и др., 2008].

- Нечеткие логики.

Теория нечетких множеств и, соответственно, нечетких логик, также применяется в системах, связанных с трудноформализуемыми предметными областями. Подробнее теория нечетких логик рассматривается в [Поспелов, 1989], [Батыршин, 2001], [Деменков, 2001] и других изданиях.

- Логика умолчаний.

Логика умолчаний применяется, в том числе, для того, чтобы оптимизировать процесс рассуждений, дополняя процесс достоверного вывода вероятностными предположениями в тех случаях, когда вероятность ошибки крайне мала. Подробнее логика умолчаний рассмотрена в статье [RRIAI, 2012].

- Темпоральная логика.

Применение темпоральной логики является очень актуальным для нестатичных предметных областей, в которых истинность того или иного утверждения меняется со временем, что существенно влияет на ход решения какой-либо задачи. Следует отметить, что используемый в данной работе язык представления знаний предоставляет все необходимые возможности для описания таких динамических предметных областей. Более подробно темпоральная логика рассмотрена в работе [Еремеев, 1997]

В заключение данного раздела, следует отметить один из важнейших принципов данной работы. Данная работа не ставит своей целью разработку нового метода решения задач, нового класса логик или отрицание существующих достижений в данной области. Целью работы является разработка технологии, позволяющей интегрировать ***любые модели решения задач*** и ***принципы логического вывода*** для решения задач в интеллектуальных системах на основе общей формальной модели. Для того, чтобы использовать какую-либо новую или существующую модель, необходимо привести ее предлагаемому в данной работе формализму, что позволить интегрировать и синхронизировать ее с уже имеющимися в соответствующей библиотеке совместимых компонентов.

7. Библиотека ip-компонентов машин обработки знаний

Центральным элементом всей технологии OSTIS и, соответственно, любой более частной технологии является библиотека совместимых ip-компонентов.

Рассмотрим подробнее структуру этой библиотеки, от общего к частному.

- Библиотека готовых машин обработки знаний

В данную библиотеку целиком попадают самодостаточные машины обработки знаний, подходящие для какой-либо предметной области. Примерами могут служить:

●●Машина обработки знаний для статичных предметных областей (геометрия, алгебра)

●●Машина обработки знаний для динамичных предметных областей (физика, химия, многие гуманитарные области)

●●Машина обработки знаний для систем классификации на основе признаков (медицинская диагностика, биология, другие классификационные области)

●●И другие

- Библиотека готовых подсистем машин обработки знаний

В данную библиотеку попадают взаимосвязанные коллективы агентов,

соответствующие одному уровню в структуре машины обрабоки знаний, описанной выше.

Таковыми, например, являются:

●●Поисковая машина, адаптированная под какой-либо класс предметных областей.

●●Совокупность стратегий решения, адаптированных под какую-либо предметную область.

●●Машина дедуктивного логического вывода

●●Машина индуктивного логического вывода

●●Машина логического вывода на основе темпоральной логики

●●И другие

- Библиотека агентов машин обработки знаний

В данную библиотеку попадают все возможные реализации агентов, предназначенных для обработки знаний в интеллектуальных системах.

Данная библиотека может быть декомпозирована различными способами. Рассмотрим некоторые из них.

По функциональному назначению агентов:

●●Библиотека агентов информационного поиска

●●Библиотека агентов, реализующих стратегии решения задач

●●Библиотека агентов логического вывода

●●Библиотека агентов сборки мусора

●●Библиотека агентов верификации знаний и устранения противоречий

По языку реализации агентов:

●●Библиотека агентов, реализованных на внутреннем языке SCP

●●Библиотека агентов, реализованных на внешнем языке программирования (может быть декомпозирована далее в зависимости от конкретных языков реализации).

По степени универсальности агентов (декомпозиция является достаточно условной, т.к. зависит от критериев универсальности):

●●Библиотека агентов, ориентированных на решение частной задачи (поиск площади объекта)

●●Библиотека агентов, ориентированных на решение некоторого класса задач (применение импликативного утверждения, применение утверждения об эквивалентности)

- Библиотека программ и процедур машин обработки знаний

В данную библиотеку попадают все возможные программы и процедуры, использованные для реализации агентов обработки знаний.

Так же как и библиотека агентов, данная библиотека может быть декомпозирована различными способами по схожим критериям. Рассмотрим некоторые из них.

По функциональному назначению программ:

●●Поисковые процедуры

●●Процедуры генерации заданных конструкций

●●Процедуры сравнения заданных конструкций

●●Процедуры удаления заданных фрагментов конструкций

●●Базовые теоретико-множественные процедуры (объединение, пересечение и т.д.)

●●Системные процедуры, предназначенные, например, для предварительной обработки знаний

Другие критерии классификации процедур и программ полностью аналогичны рассмотренным в библиотеке агентов, поэтому подробное их описание приводить здесь не будем.

8. Автоматические средства поддержки проектирования машин обработки знаний

Рассмотрим ряд средств, обеспечивающих дополнительные возможности при проектировании машин обработки знаний на основе библиотек, а также собственно компонентов, входящих в состав каждой из библиотек.

- Среда программирования для языка SCP.

Данная среда программирования должна обладать всеми стандартными возможностями, предоставляемыми средами разработки для существующих языков программирования. Среди таких возможностей можно отметить

●●Отладчик, поясняющий ошибки времени выполнения в виде, понятном пользователю, и позволяющий осуществлять пошаговую отладку программы с просмотром значений переменных и т.п.

●●Редактор исходных текстов программ, имеющий функции автодополнения, подсветки синтаксиса, выявления синтаксических ошибок, поиска необъявленных переменных и т.д.

●●Транслятор реального времени, позволяющий протранслировать составленную программу в память системы прямо в процессе ее работы для последующего тестирования и использования.

- Профайлер графодинамической памяти.

Задача профайлера состоит в подсчете объема памяти, используемого тем или иным агентом или группой агентов. Объем памяти оценивается соответственно на основе количества созданных узлов и дуг, с возможностью уточнения типа учитываемых элементов. Такой профайлер позволит отслеживать и предупреждать накопление информационного мусора, ложные срабатывания агентов обработки знаний и т.д.

- Визуализатор графодинамической памяти.

Задача визуализатора состоит в отображении в реальном времени средствами пользовательского интерфейса некоторого фрагмента памяти (например, семантической окрестности некоторого узла). Это позволяет разработчику лучше понять

процессы, происходящие в памяти, оценить правильность работы спроектированных программ и выяснить, действительно ли все происходит так, как он планировал. Использование визуализатора наиболее целесообразно одновременно с отладчиком в режиме пошаговой отладки, т.к. разработчик сможет визуально отследить реальный результат каждого шага составленной программы.

- Среда проектирования коллективов агентов над общей памятью.

В мире существуют средства, позволяющие осуществлять проектирование многоагентных систем. Наиболее значимые представители рассмотрены в статье А.В. Нарушева и В.Ф. Хорошевского [Нарушев, Хорошевский, 2000]. Однако все указанные средства ориентированы на непосредственное взаимодействие агент-агент, в то время как в рамках рассматриваемого подхода к проектированию машин обработки знаний взаимодействие агентов осуществляется через общую память.

Рассматриваемая среда проектирования коллективов агентов позволяет разработчику отслеживать срабатывания агентов при инициировании вопросов в памяти системы, минимизировать количество ложных или нецелесообразных срабатываний, выбрать наиболее рациональный порядок активации агентов в каждом случае.

Таким образом, речь идет о создании интегрированной среды разработки машин обработки знаний на основе мультиагентного подхода.

Библиографический список

[OSTIS, 2012] Проект OSTIS [Электронный ресурс]. Минск, 2012. – Режим доступа: http://ostis.net/. – Дата доступа: 30.11.2012.

[RRIAI, 2012] Искусственный интеллект. Системы и модели [Электронный ресурс]. Минск, 2012. – Режим доступа: http://www.rriai.org.ru/logika-kosvennogo-opisaniya-i-logika-umolchaniya.html. – Дата доступа: 30.11.2012.

[Батыршин, 2001] Батыршин И.З. Основные операции нечеткой логики и их обобщения / И.З. Батыршин; – Казань : Отечество, 2001.

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

[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.

[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.

[Деменков, 2001] Деменков Н.П. Нечеткое управление в технических системах / Н.П. Деменков; – М : Изд. им. Баумана, 2005.

[Еремеев, 1997] Еремеев А.П. Построение решающих функций на базе тернарной логики в системах принятия решений в условиях неопределенности // А.П. Еремеев Известия академии наук. Теория и системы управления, 1997. №5.

[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.

[Кузнецов, 2009] Кузнецов О.П. Дискретная математика для инженера. Изд. 6, стереотипное. / О.П. Кузнецов; - СПБ, Лань, 2009.

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

[Малыхина, 2002] Малыхина Г.И. Логика / Г.И. Малыхина; – Мн. : Вышейшая школа, 2002.

[Нарушев, Хорошевский, 2000] Нарушев Е.С., Хорошевский В.Ф. AgSDK: Инструментарий разработки мультиагентных систем // Труды 7-ой Национальной конференции по искусственному интеллекту с международным участием, Переславль Залесский, 24-27 октября 2000 – Москва: ИФМЛ, том 2.

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

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

[Поспелов, 1994] Поспелов Д.А. Информатика. Энциклопедический словарь. / Д.А.Поспелов; – М. : «Просвещение», 1994.

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

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

MODELS AND MEANS OF KNOWLEDGE PROCESSING MACHINES COMPONENT DESIGN ON BASIS OF SEMANTIC NETWORKS

Shunkevich D.V.

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

shu.dv@tut.by

This work is devoted to considering of problems of existing methods, means and technologies of knowledge processing machines design. The article describes the problem of lack of means, which allow relatively inexperienced developer to design knowledge processing machine for applied intelligent systems for different purposes. Next part of the paper considers the technology which was designed to solve given problem in the way of integration different problem situations solution methods and ways on the common formal basis.

Keywords: intelligent systems; knowledge processing machine; problem situations solver; component design; multiagent systems.