OSTIS-2014
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

МАШИНА ОБРАБОТКИ ЗНАНИЙ ИНТЕЛЛЕКТУАЛЬНОЙ МЕТАСИСТЕМЫ ПОДДЕРЖКИ ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Шункевич Д.В.

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

shu.dv@tut.by

Данная статья посвящена принципам организации и функционирования машин обработки знаний интеллектуальных систем в целом, лежащим в основе технологии проектирования машин обработки знаний в рамках технологии OSTIS. В качестве примера применения такой технологии рассматривается машина обработки знаний интеллектуальной метасистемы IMS.OSTIS направленной на поддержку проектирования интеллектуальных систем.

Ключевые слова: машина обработки знаний; интеллектуальная система; многоагентная система; технология проектирования.

Введение

Данная статья посвящена принципам организации и функционирования машин обработки знаний, построенных на базе технологии проектирования машин обработки знаний в рамках технологии OSTIS [OSTIS, 2013]. Более подробное формальное описание понятий и принципов, рассматриваемых в данной статье, можно найти на сайте метасистемы поддержки проектирования интеллектуальных систем IMS.OSTIS [IMS.OSTIS, 2013]

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

В предлагаемом подходе к построению машин обработки знаний сама машина представляет собой графодинамическую sc-машину (память в качестве модели представления знаний использует семантическую сеть с теоретико-множественной интерпретацией [Голенков и др., 2001]), состоящую из двух частей:

- графодинамическая sc-память;
- система sc-агентов над общей памятью.

Система sc-агентов над общей памятью представляет собой коллектив sc-агентов, 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 агентов, не внося при этом изменений, затрагивающих других агентов.

Отличительной особенностью машины обработки знаний как многоагентной системы в рамках данного подхода является принцип взаимодействия 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, 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, некоторые их которых могут выполнять определенные действия параллельно.

2. Предметная область sc-агентов

Под ***абстрактным sc-агентом*** понимается некоторый класс функционально эквивалентных *sc-агентов,* разные экземпляры (т.е. представители) которого могут быть реализованы по-разному.

Каждый ***абстрактный sc-агент*** имеет соответствующую ему спецификацию. В спецификацию каждого ***абстрактного sc-агента*** входит:

- указание ключевых *sc-элементов* этого *sc-агента*, т.е. тех *sc-элементов*, хранимых в *sc-памяти*, которые для данного *sc-агента* являются «точками опоры»
- формальное описание условий инициирования данного *sc-агента*, т.е. тех ситуация в *sc-памяти*, которые инициируют деятельность данного *sc-агента*.
- формальное описание первичного условия инициирования данного *sc-агента*, т.е. такой ситуации в *sc-памяти*, которая побуждает *sc-агента* перейти в активное состояние и начать проверку наличия своего полного условия инициирования.
- строгое, полное, однозначно понимаемое описание деятельности данного *sc-агента*, оформленное при помощи каких-либо понятных, общепринятых средств, не требующих специального изучения, например на естественном языке.
- описание результатов выполнения данного *sc-агента*.

Понятие ***абстрактного sc-агента*** разбивается на два подкласса:

- *неатомарный абстрактный sc-агент*
- *атомарный абстрактный sc-агент*

Под ***неатомарным абстрактным sc-агентом*** понимается *абстрактный sc-агент*, который декомпозируется на коллектив более простых *абстрактных sc-агентов*, каждый из которых в свою очередь может быть как *атомарным абстрактным sc-агентом*, так и ***неатомарным абстрактным sc-агентом***. При этом в каком либо варианте *декомпозиции абстрактного sc-агента* дочерний ***неатомарный абстрактный sc-агент*** может стать *атомарным абстрактным sc-агентом*, и реализовываться соответствующим образом.

Отношение ***декомпозиции абстрактного sc-агента**** трактует *неатомарные абстрактные sc-агенты* как коллективы более простых *абстрактных sc-агентов*, взаимодействующих через *sc-память*.

Другими словами, ***декомпозиция абстрактного sc-агента**** на *абстрактные sc-агенты* более низкого уровня уточняет один из возможных подходов к реализации этого *абстрактного sc-агента* путем построения коллектива более простых *абстрактных sc-агентов*.

Под ***атомарным абстрактным sc-агентом*** понимается *абстрактный sc-агент*, для которого уточняется платформа его реализации, т.е. существует соответствующая связка отношения *программа sc-агента**.

Понятие ***абстрактного sc-агента*** разбивается на два подкласса:

- *платформенно-независимый абстрактный sc-агент*
- *платформенно-зависимый абстрактный sc-агент*

К ***платформенно-независимым абстрактных sc-агентам*** относят *атомарные абстрактные sc-агенты*, реализованные на базовом языке программирования Технологии OSTIS, т.е. на *Языке SCP*. [Голенков и др., 2001]

При описании ***платформенно-независимых абстрактных sc-агентов*** под платформенной независимостью понимается платформенная независимость с точки зрения Технологии OSTIS, т.е реализация на специализированном языке программирования, ориентированном на обработку семантических сетей (*Языке SCP*), поскольку *атомарные sc-агенты*, реализованные на указанном языке могут свободно переноситься с одной платформы интерпретации *sc-моделей* на другую. При этом языки программирования, традиционно считающиеся платформенно-независимыми в данном случае не могут считаться таковыми [Петцольд, 2002], [Шилдт, 2004], [Хорстманн, 2004].

К ***платформенно-зависимым абстрактным sc-агентам*** относят *атомарные абстрактные sc-агенты*, реализованные ниже уровня sc-моделей, т.е. не на *Языке SCP*, а на каком-либо другом языке описания программ.

Существуют *sc-агенты*, которые принципиально должны быть реализованы на платформенно-зависимом уровне, например, собственно *sc-агенты* интерпретации *sc-моделей* или рецепторные и эффекторные *sc-агенты*, обеспечивающие взаимодействие с внешней средой.

Под ***sc-агентом*** понимается конкретный экземпляр (с теоретико-множественной точки зрения - элемент) некоторого *атомарного абстрактного sc-агента*, работающий в какой-либо конкретной интеллектуальной системе.

При этом каждый *sc-агент* инициируется при появлении в *sc-памяти* конкретной ситуации, соответствующей условию инициирования *атомарного абстрактного sc-агента*, экземпляром которого является заданный *sc-агент*. В данном случае можно провести аналогию между принципами объектно-ориентированного программирования, рассматривая *атомарный абстрактный sc-агент* как класс, а конкретный *sc-агент* – как экземпляр, конкретную имплементацию этого класса [Хорстманн, 2004], [Гамма, 2007].

Взаимодействие *sc-агентов* осуществляется только через *sc-память*. Как следствие, результатом

работы любого *sc-агента* является некоторое изменение состояния *sc-памяти*, т.е. удаление либо генерация каких-либо *sc-элементов*.

В общем случае один *sc-агент* может явно передать управление другому *sc-агенту*, если этот *sc-агент* априори известен. Для этого каждый *sc-агент* в *sc-памяти* имеет обозначающий его *sc-узел*, с которым можно связать конкретную ситуацию в текущем состоянии базы знаний, которую инициируемый *sc-агент* должен обработать.

Однако далеко не всегда легко определить того *sc-агента*, который должен принять управление от заданного *sc-агента*, в связи с чем описанная выше ситуация возникает крайне редко. Более того, иногда условие инициирования *sc-агента* является результатом деятельности непредсказуемой группы *sc-агентов*, равно как и одна и та же конструкция может являться условием инициирования целой группы *sc-агентов*.

При этом общаются через sc-память не *программы sc-агентов**, а сами описываемые данными программами *sc-агенты*.

Под ***активным sc-агентом*** понимается *sc-агент* интеллектуальной системы, который реагирует на события соответствующие его условию инициирования, и, как следствие, его *первичному условию инициирования**. Не входящие во множество ***активных sc-агентов*** *sc-агенты* не реагируют ни на какие события в *sc-памяти*.

Связки отношения ***ключевые sc-элементы sc-агента**** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и *sc-узел*, обозначающий множество *sc-элементов*, которые являются ключевыми для данного *абстрактного sc-агента*, то данные *sc-элементы* явно упоминаются в рамках программ, реализующих данный *абстрактный sc-агент*.

Связки отношения ***программа sc-агента**** связывают между собой *sc-узел*, обозначающий *атомарный абстрактный sc-агент* и *sc-узел*, обозначающий множество программ, реализующих указанный *атомарный абстрактный sc-агент*.

В случае *платформенно-независимого абстрактного sc-агента* каждая связка отношения *программа sc-агента** связывает *sc-узел,* обозначающий указанный *абстрактный sc-агент* с множеством *scp-программ*, описывающих деятельность данного *абстрактного sc-агента*. Данное множество содержит одну *агентную scp-программу*, и произвольное количество (может быть, и ни одной) *scp-программ*, которые необходимы для выполнения указанной *агентной scp-программы.*

В случае *платформенно-зависимого абстрактного sc-агента* каждая связка отношения *программа sc-агента** связывает *sc-узел,* обозначающий указанный *абстрактный sc-агент* с множеством файлов, содержащих исходные тексты программы на некотором внешнем языке программирования, реализующей деятельность данного *абстрактного sc-агента*.

Связки отношения ***первичное условие инициирования**** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и бинарную ориентированную пару, описывающую первичное условие инициирования данного *абстрактного sc-агента*, т.е. такой ситуации в *sc-памяти*, которая побуждает *sc-агента* перейти в активное состояние и начать проверку наличия своего полного условия инициирования.

Первым компонентом данной ориентированной пары является знак некоторого подмножества понятия *sc-событие*, например *sc-событие добавления выходящей sc-дуги*, т.е. по сути конкретный тип события в *sc-памяти*.

Вторым компонентом данной ориентированной пары является произвольный в общем случае *sc-элемент*, с которым непосредственно связан указанный тип события в *sc-памяти*, т.е., например, sc-элемент, из которого выходит либо в который входит генерируемая либо удаляемая *sc-дуга*, либо *sc-ссылка*, содержимое которой было изменено.

После того, как в *sc-памяти* происходит некоторое *sc-событие*, активизируются все *активные sc-агенты*, ***первичное условие инициирования**** которых соответствует произошедшему *sc-событию*.

Связки отношения ***условие инициирования и результат**** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и бинарную ориентированную пару, связывающую условие инициирования данного *абстрактного sc-агента* и результаты выполнения данного экземпляров данного *sc-агента* в какой-либо конкретной системе.

Указанную ориентированную пару можно рассматривать как логическую связку импликации, при этом на *sc-переменные*, присутствующие в обеих частях связки, неявно накладывается квантор всеобщности, на *sc-переменные*, присутствующие либо только в посылке, либо только в заключении неявно накладывается квантор существования.

Первым компонентом указанной ориентированной пары является логическая формула, описывающая условие инициирования описываемого абстрактного *sc-агента*, то есть конструкции, наличие которой в *sc-памяти* побуждает *sc-агент* начать работу по изменению состояния *sc-памяти*. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

Вторым компонентом указанной ориентированной пары является логическая формула, описывающая возможные результаты выполнения описываемого абстрактного *sc-агента*,

то есть описание произведенных им изменений состояния *sc-памяти*. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

sc-описание поведения sc-агента представляет собой *sc-окрестность,* описывающую деятельность sc-агента до какой-либо степени детализации, однако такое описание должно быть строгим, полным и однозначно понимаемым. Как любая другая *sc-окрестность*, ***sc-описание поведения sc-агента*** может быть протранслировано на какие-либо понятные, общепринятые средства, не требующие специального изучения, например на естественный язык.

Описываемый *абстрактный sc-агент* входит в ***sc-описание поведения sc-агента*** под атрибутом *ключевой sc-элемент'*.

Под ***sc-событием*** понимается элементарное изменение состояния *sc-памяти*, которое может стать *первичным условием инициирования** какого-либо *sc-агента*.

Класс *sc-событий* в целом разбивается на следующие подклассы:

- *sc-событие добавления выходящей sc-дуги*
- *sc-событие добавления входящей sc-дуги*
- *sc-событие добавления инцидентного sc-ребра*
- *sc-событие удаления выходящей sc-дуги*
- *sc-событие удаления входящей sc-дуги*
- *sc-событие удаления инцидентного sc-ребра*
- *sc-событие удаления sc-элемента*
- *sc-событие изменения содержимого sc-ссылки*

3. Библиотека компонентов машин обработки знаний

Одним из важнейших компонентов интеллектуальной метасистемы поддержки проектирования интеллектуальных систем является библиотека совместимых компонентов таких систем, в частности, библиотека многократно используемых компонентов машин обработки знаний. Для машин обработки знаний роль совместимых компонентов, легко интегрируемых в различные интеллектуальные системы, выполняют *абстрактные sc-агенты*, а также их реализации, в особенности, платформенно-независимые.

Библиотека многократно используемых компонентов машин обработки знаний может быть декомпозирована на следующие частные библиотеки:

- Библиотека агентов информационного поиска
- Библиотека агентов погружения интегрируемого знания в базу знаний
- Библиотека агентов выравнивания онтологии интегрируемого знания с основной онтологией базы знаний
- Библиотека агентов поиска решения явно сформулированных задач
- Библиотека агентов логического вывода
- Библиотека агентов обнаружения и удаления информационного мусора
- Библиотека агентов-координаторов

В свою очередь, каждая из рассмотренных библиотек может быть декомпозирована по различным признакам.

Заключение

В данной работе рассмотрены понятия предметной области агентов машин обработки знаний, лежащие в основе технологии проектирования машин обработки знаний интеллектуальных систем в рамках Технологии OSTIS.

Библиографический список

[IMS.OSTIS, 2013] Метасистема IMS.OSTIS [Электронный ресурс]. Минск, 2013. – Режим доступа: http://ims.ostis.net/. – Дата доступа: 30.12.2013.

[OSTIS, 2013] Проект OSTIS [Электронный ресурс]. Минск, 2013. – Режим доступа: http://ostis.net/. – Дата доступа: 30.12.2013.

[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.

[Петцольд, 2002] Петцольд, Ч. Программирование для Microsoft Windows на C#. В 2х томах / Ч. Петцольд; – М. : Издательско-торговый дом «Русская Редакция», 2002.

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

[Хорстманн, 2004] Хорстманн, К.С. Библиотека профессионала. Java 2. В 2х томах / К.С. Хорстманн, Корнелл Г; – М. : Издательский дом «Вильмс», 2004.

[Шилдт, 2004] Шилдт, Г. Полный справочник по C# / Г. Шилдт; – М. : Издательский дом «Вильмс», 2004.

[Гамма, 2007] Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес; – СПб. : «Питер», 2007.

THE KNOWLEDGE PROCESSING MACHINE OF THE INTELLIGENT DESIGN SUPPORT METASYSTEM FOR INTELLIGENT SYSTEMS

Shunkevich D.V.

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

shu.dv@tut.by

This arcticle is devoted to principles of organization and operating of knowledge processing machines of intelligents systems as whole, which underly the technology of knowledge processing machine design is borders of OSTIS technology. As an example of such a technology apllication the author considers knowledge processing machine of the intelligent metasystem IMS.OSTIS, which is oriented to the intelligent systems design support.