OSTIS-2016
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ВЗАИМОДЕЙСТВИЕ АСИНХРОННЫХ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ ОБРАБОТКИ ЗНАНИЙ В ОБЩЕЙ СЕМАНТИЧЕСКОЙ ПАМЯТИ

Шункевич Д.В.

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

shunkevichdv@gmail.com

В работе рассматривается базовая модель построения машин обработки знаний систем, управляемых знаниями, построенных по открытой семантической технологии проектирования интеллектуальных систем. В частности, описываются базовые принципы взаимодействия асинхронных параллельных процессов обработки знаний в общей семантической памяти, типология действий, выполняемых агентами в такой памяти, типология и средства описания блокировок, необходимых для синхронизации деятельности агентов над общей семантической памятью.

Ключевые слова: семантические технологии, формализация деятельности, многоагентные системы, проект OSTIS.

Введение

Системы, управляемые знаниями, построенные по открытой семантической технологии проектирования интеллектуальных систем (*Технологии OSTIS*) будем называть ostis-*системами* [Голенков, 2015]. В качестве модели представления знаний в данной технологии используется унифицированная семантическая сеть с теоретико-множественной интерпретацией [Голенков и др., 2001]. Такую модель представления будем называть SC-кодом (Semantic computer code) [Голенков, 2012]. Элементы такой семантической сети будем называть *sc-узлами* и *sc-коннекторами* (*sc-дугами*, *sc-ребрами*).

Каждая *ostis-система* состоит из не зависящей от платформы реализации унифицированной логико-семантической модели этой системы (*sc-модели компьютерной системы*) и платформы интерпретации таких моделей. В свою очередь каждая *sc-модель компьютерной системы* может быть декомпозирована на *sc-модель базы знаний*, *sc-модель машины обработки знаний*, *sc-модель интерфейса*. В данной работе внимание будет уделено *sc-модели машины обработки знаний*.

В рамках *Технологии OSTIS sc-модель машины обработки знаний* включает графодинамическую *sc-память*, в которой хранятся конструкции *SC-кода*, и коллектив агентов взаимодействующих через эту память (*sc-агентов*).

Данная модель предполагает, что каждый *sc-агент* реагирует на соответствующий ему класс ситуаций и/или событий, происходящих в *sc-памяти*, и осуществляет определенное преобразование *sc-текста* (текста *SC-кода*), находящегося в семантической окрестности обрабатываемой ситуации и/или события.

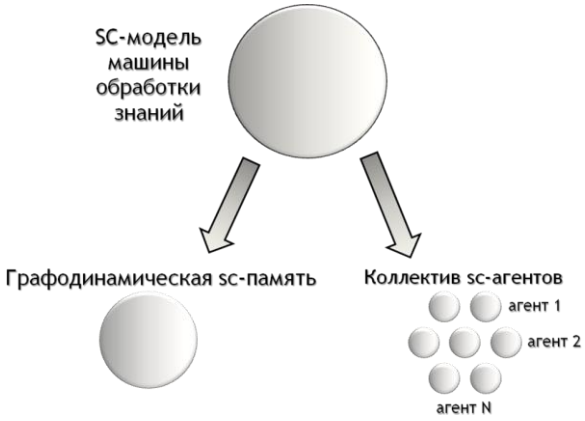

Рисунок 1 – Модель машины обработки знаний

Более подробно семантическая модель машины обработки знаний и преимущества такого подхода к ее построению рассмотрены в [Голенков, 2012], [Голенков, 2013], [Шункевич, 2013].

Перечислим некоторые достоинства предлагаемого подхода к построению машин обработки знаний:

- поскольку машина обработки знаний состоит из агентов, которые обмениваются сообщениями только через общую память, добавление нового агента или исключение (деактивация) одного или нескольких существующих агентов, как правило, не приводит к изменениям в других агентах, поскольку агенты не обмениваются сообщениями напрямую;
- часто агенты работают параллельно и независимо друг от друга, выполняя разные *действия в sc-памяти*; таким образом, даже существенное расширение числа агентов в рамках одной системы не приводит к ухудшению ее производительности;
- в качестве основного языка реализации программ агентов обработки знаний предполагается использовать *Язык SCP*, тексты программ которого записываются при помощи тех же семантических сетей, что и обрабатываемая информация, при этом подход к интерпретации *scp-программ* (*программ Языка SCP*) предполагает создание при каждом вызове *scp-программы* уникального *scp-процесса*. Таким образом, одновременно могут выполняться несколько операций, при этом разные копии агентов могут выполняться на разных серверах, за счет распределенной реализации интерпретатора sc-моделей (платформы реализации sc-моделей). Более того, язык реализации программ агентов позволяет осуществлять параллельные асинхронные вызовы подпрограмм с последующей синхронизацией, и даже параллельно выполнять операторы в рамках одной программы;
- как уже было сказано, программы, описывающие деятельность агентов, могут быть записаны на том же языке, что и обрабатываемые знания, и храниться в той же базе знаний, равно как и их спецификации. Таким образом, перенос агента из одной системы в другую заключается в простом переносе фрагмента базы знаний, без каких-либо дополнительных операций, зависящих от платформы интерпретации;
- спецификации агентов и их программы могут быть записаны на том же языке, что и обрабатываемые знания, что существенно сокращает перечень специализированных средств, предназначенных для проектирования машин обработки знаний, и упрощает их разработку за счет использования более универсальных компонентов;
- тот факт, что для интерпретации *scp-программы* создается соответствующий ей уникальный *scp-процесс*, позволяет по возможности оптимизировать план выполнения перед его реализацией и даже непосредственно в процессе выполнения без потенциальной опасности испортить общий универсальный алгоритм всей программы.

Следует отдельно отметить факт возможности параллельного выполнения процессов (как правило-целенаправленных действий в sc-памяти), а также возможности параллельного выполнения операторов в рамках одной *scp-программы*. При этом отнесение какой-либо конкретной машины обработки знаний к какому-либо классу параллельных систем, рассматриваемых в литературе [Хоар, 1989], [Шпаковский, 2013] представляется возможным только, во-первых, с учетом конкретных классов решаемых такой машиной задач, во-вторых, с учетом возможностей используемой *платформы реализации sc-моделей* касающихся параллельной обработки знаний.

1. Понятие процесса и действия в sc-памяти

Понятие ***процесса*** как изменения в целом более подробно описано в общих предметных областях в рамках базы знаний IMS [IMS, 2016].

Понятие *действия* вообще, как частного случая *воздействия*, и, соответственно, *процесса*, применительно к системам, управляемым знаниями, подробно рассматривается в [Шункевич, 2016].

В рамках данной работы основное внимание уделяется понятию процесса в *sc-памяти* и действия в *sc-памяти*, как целенаправленного изменения информации хранящейся в памяти *ostis-системы*.

Рассмотрим спецификацию понятия ***действие в sc-памяти*** в SCn-коде:

действие в sc-памяти
= внутреннее действие ostis-системы
= действие, выполняемое в sc-памяти
= действие, выполняемое в абстрактной унифицированной семантической памяти
= действие, выполняемое машиной обработки знаний ostis-системы
= действие, выполняемое агентом или коллективом агентов ostis-системы
= информационный процесс над базой знаний, хранимой в sc-памяти
= процесс решения информационной задачи в sc-памяти
<= включение:*
    процесс в sc-памяти

Каждое ***действие в sc-памяти*** обозначает некоторое преобразование, выполняемое некоторым *sc-агентом* (или коллективом *sc-агентов*) и ориентированное на преобразование *sc-памяти*. Спецификация действия после его выполнения может быть включена в протокол решения некоторой задачи.

Преобразование состояния базы знаний включает, в том числе и информационный поиск, предполагающий (1) локализацию в базе знаний ответа на запрос, явное выделение структуры ответа и (2) трансляцию ответа на некоторый внешний язык.

Во множество ***действий в sc-памяти*** входят знаки действий самого различного рода, семантика каждого из которых зависит от конкретного контекста, т.е. ориентации действия на какие-либо

конкретные объекты и принадлежности действия какому-либо конкретному классу действий.

Следует четко отличать:

- Каждое конкретное ***действие в sc-памяти***, представляющее собой некоторый переходный процесс, переводящий sc-память из одного состояния в другое;
- Каждый тип ***действий в sc-памяти***, представляющий собой некоторый класс однотипных (в том или ином смысле) действий;
- sc-узел, обозначающий некоторое конкретное ***действие в sc-памяти***;
- sc-узел, обозначающий структуру, которая является описанием, спецификацией, заданием, постановкой соответствующего действия.

2. Типология действий в sc-памяти

Приведем семантическую типологию действий в sc-памяти, представленную в SCn-коде:

действие в sc-памяти
=> включение:*

- *действие в sc-памяти, инициируемое вопросом*
- *действие редактирования базы знаний ostis-системы*
- *действие установки режима ostis-системы*
- *действие редактирования файла, хранимого в sc-памяти*
- *действие интерпретации программы, хранимой в sc-памяти*

действие в sc-памяти, инициируемое вопросом
= действие, направленное на формирование ответа на поставленный вопрос
=> включение:*

- *действие. сформировать заданный файл*
- *действие. сформировать заданную структуру*
  => включение:*
  - *действие. верифицировать заданную структуру*
    *=> включение**:
    - *действие. установить истинность или ложность указываемого логического высказывания*
    - *действие. установить корректность или некорректность указываемой структуры*
    - *действие. сформировать структуру, описывающую некорректности, имеющиеся в указываемой структуре*
  - *действие. установить тип заданного sc-элемента*
    *=> включение**:
    - *действие. установить позитивность/негативность указываемой sc-дуги принадлежности или непринадлежности*
  - *действие. сформировать семантическую окрестность*
    *=> включение**:
    - *действие. сформировать полную семантическую окрестность указываемой сущности*
    - *действие. сформировать базовую семантическую окрестность указываемой сущности*
    - *действие. сформировать частную семантическую окрестность указываемой сущности*
  - *действие. сформировать структуру, описывающую связи между указываемыми сущностями*
    *=> включение**:
    - *действие. сформировать структуру, описывающую сходства указываемых сущностей*
    - *действие. сформировать структуру, описывающую различия указываемых сущностей*
  - *действие. сформировать структуру, описывающую способ решения указываемой задачи*
  - *действие. сформировать план генерации ответа на указанный вопрос*
  - *действие. сформировать протокол выполнения указываемого действия*
  - *действие. сформировать обоснование корректности указываемого решения*
  - *действие. верифицировать обоснование корректности указываемого решения*
  - *действие, одним из аргументов которого является некоторая обобщенная структура*
  - *действие, направленное на установление темпоральных характеристик указываемой сущности*
  - *действие, направленное на установление пространственных характеристик указываемой сущности*

действие редактирования базы знаний
=> включение:*

- *действие. исправить ошибки в заданной структуре*
- *действие. преобразовать указанную структуру в соответствии с указанным правилом*
- *действие. отождествить два указанных sc-элемента*
- *действие. включить множество*
  = сделать все элементы множества si явно принадлежащими множеству sj, то есть

сгенерировать соответствующие sc-дуги принадлежности

- *действие генерации sc-элементов*
  => включение:*
  - *действие генерации, одним из аргументов которого является некоторая обобщенная структура*
    *=> включение**:
    - *действие. сгенерировать структуру, изоморфную указываемому образцу*
  - *действие. сгенерировать sc-элемент указанного типа*
    *=> включение**:
    - *действие. сгенерировать sc-коннектор указанного типа*
    - *действие. сгенерировать sc-узел указанного типа*
  - *действие. сгенерировать структуру, содержащую указанные sc-элементы*
  - *действие. сгенерировать файл с заданным содержимым*
  - *действие. обновить понятия*
    = действие. заменить неиспользуемое понятие на его определение через используемое понятие
  - *действие. установить указанный файл в качестве основного идентификатора указанного sc-элемента*
  - *действие. протранслировать содержимое указываемого файла в sc-память*
  - *действие. интегрировать указанную структуру в текущее состояние базы знаний*
  - *действие. сгенерировать структуру, описывающую историю эволюции ostis-системы*
  - *действие. сгенерировать структуру, описывающую историю эксплуатации ostis-системы*
- *действие удаления sc-элементов*
  => включение:*
  - *действие. удалить указанные sc-элементы*
    *=> включение**:
    - *действие. удалить указанный sc-элемент*
    - *действие. удалить sc-элементы, входящие в состав указанной структуры и не являющиеся ключевыми узлами каких-либо sc-агентов*
  - *действие. исключить указанные sc-элементы из клиентской части базы знаний*

Далее более детально рассмотрим некоторые из выделенных классов действий в sc-памяти.

Каждое ***действие. отождествить два указанных sc-элемента*** может быть выполнено как *действие. физически отождествить два указанных sc-элемента* или *действие. логически отождествить два указанных sc-элемента*. В случае логического отождествления в протоколе деятельности агентов сохраняется само действие с его спецификацией, включающей обязательное указание того, какие элементы были сгенерированы, а какие удалены. В случае физического отождествления протокол действия не сохраняется.

Каждое ***действие. обновить понятия*** обозначает переход от какой-то группы понятий, использовавшихся ранее, к другой группе понятий, которые будут использоваться вместо первых, и станут *основными понятиями*.

В общем случае ***действие. обновить понятия*** состоит из следующих этапов:

- Определить заменяемые понятия на основе заменяющих;
- Внести соответствующие изменения в программы sc-агентов, ключевыми узлами которых являются обновляемые понятия;
- Заменить все конструкции в базе знаний, содержащие заменяемые понятия, в соответствии с определениями этих понятий через заменяющие их понятия;
- При необходимости, *sc-элементы*, обозначающие замененные таким образом понятия, могут быть полностью выведены из текущего состояния базы знаний.

Первым аргументом (входящим в знак *действия* под атрибутом *1'*) ***действия. обновить понятия*** является знак множества *sc-узлов*, обозначающих заменяемые понятия, вторым (входящим в знак *действия* под атрибутом *2'*) - знак множества *sc-узлов*, обозначающих заменяющие понятия. В общем случае любое или оба этих множества могут быть *синглетонами*.

Каждое ***действие. удалить указанные sc-элементы*** может быть выполнено как *действие. физически удалить указанные sc-элементы* или *действие. логически удалить указанные sc-элементы*. В случае логического удаления в протоколе деятельности агентов сохраняется само действие с его спецификацией, включающей обязательное указание того, какие элементы были удалены, т.е. по сути, элементы просто исключаются из текущего состояния базы знаний. В случае физического удаления протокол действия не сохраняется.

В случае удаления какого-либо *sc-элемента*, инцидентные ему *связки*, в том числе *sc-коннекторы*, также удаляются.

Для того, чтобы выполнить ***действие. интегрировать указанную структуру в текущее состояние базы знаний***, необходимо склеить *sc-элементы*, входящие в интегрируемую *структуру* с синонимичными им *sc-элементами*, входящими в

текущее состояние базы знаний, заменить неиспользуемые (например, устаревшие) понятия, входящие в интегрируемую *структуру*, на используемые (т.е. заменить неиспользуемые понятия на их определения через используемые), явно включить все элементы интегрируемой *структуры* в число элементов утвержденной части базы знаний и явно включить все элементы интегрируемой *структуры* в число элементов одного из атомарных разделов утвержденной части базы знаний.

3. Принципы взаимодействия агентов в sc-памяти

Как было сказано ранее, все действия в sc-памяти независимо от конкретного класса выполняются sc-агентами, которые могут в том числе, инициироваться пользователями ostis-системы.

Понятие агента в sc-памяти (*sc-агента*) и их типология подробно рассмотрены в [Шункевич, 2014], теория многоагентных систем и типология агентов в целом описаны в [Тарасов, 2002].

Ниже в данной работе подробно рассматриваются принципы взаимодействия *sc-агентов* в памяти *ostis-системы*.

3.1. Общие принципы организации взаимодействия sc-агентов и пользователей ostis-системы через общую sc-память.

Каждая *ostis-система* представляет собой многоагентную систему, агенты которой взаимодействуют между собой только(!) через общую для них *sc-память*. При этом пользователи *ostis-системы* также считают агентами этой системы. Кроме того, *sc-агенты* делятся на внутренние, рецепторные и эффекторные. Взаимодействие между агентами через общую *sc-память* сводится к следующим видам действий:

(1) К использованию общедоступных для соответствующей группы агентов части хранимой базы знаний. В простейшем случае по уровню прав доступа агенты *ostis-системы* разбиваются на две группы – главные администраторы базы знаний (их может быть несколько) вместе с обслуживающими их *sc-агентами* и все остальные агенты;

(2) К формированию (генерации) новых фрагментов базы знаний и/или к корректировке (редактированию) каких-либо фрагментов доступной части базы знаний;

(3) К интеграции (погружению) новых и/или обновленных фрагментов в состав доступной части базы знаний;

Пользователь *ostis-системы* не может сам непосредственно выполнить какое-либо действие в sc-памяти, но он может средствами пользовательского интерфейса инициировать построение (генерацию, формирование в *sc-памяти*) *sc-текста*, являющегося спецификацией *действия в sc-памяти*, выполняемого либо одним атомарным *sc-агентом* за один акт, либо одним атомарным *sc-агентом* за несколько актов, либо коллективом *sc-агентов*. В спецификации каждого такого *действия в sc-памяти*, инициированного пользователем, этот пользователь указывается как заказчик этого действия. Таким образом, пользователь *ostis-системы* дает поручения (задания, команды) *sc-агентам* этой системы на выполнение различных специфицируемых им действий в sc-пам*ятью*.

Каждый *sc-агент*, выполняя некоторое *действие в sc-памяти*, должен «помнить», что *sc-память*, над которой он работает, является общим ресурсом не только для него, но и для всех остальных *sc-агентов*, работающих над этой же *sc-памятью*. Поэтому *sc-агент* должен соблюдать определенную этику поведения в коллективе таких *sc-агентов*, которая должна минимизировать помехи, которые он создает другим *sc-агентам*.

Деятельность каждого агента *ostis-системы* дискретна и представляет собой множество элементарных действий (актов). При этом при выполнении практически каждого акта агент выделяет некий фрагмент базы знаний, который вообще не должен быть «виден» другим агентам (только ему самому) и/или некоторый фрагмент базы знаний, который может быть «виден», но не может изменяться другими агентами. Указанная блокировка – это своего рода «забор» (ограждение) через который другим агентам перелезать запрещено. Эта блокировка устанавливается самим агентом при выполнении соответствующего акта и снимается им же на последнем этапе выполнения этого акта.

Если некий *sc-агент* выполняет некоторое действие в sc-пам*ятью*, то он на время выполнения этого действия может:

(1) Запретить другим *sc-агентам* изменять состояние некоторых sc-элементов, хранимых в *sc-памяти* – удалять их, изменять тип;

(2) Запретить другим *sc-агентам* добавлять или удалять элементы некоторых множеств, обозначаемых соответствующими *sc-узлами*;

(3) Запретить другим *sc-агентам* доступ на просмотр некоторых *sc-элементов*, то есть эти *sc-элементы* становятся полностью «невидимыми» (полностью заблокированными) для других *sc-агентов* но только на время выполнения соответствующего действия.

Указанные блокировки должны быть полностью сняты до завершения выполнения соответствующего действия. Подчеркнем, что число *sc-элементов*, блокируемых на время выполнения некоторого действия, в основном входят атомарные и неатомарные связки, и не должны входить *sc-узлы*, обозначающие бесконечные классы каких-либо сущностей, и, тем более, sc-узлы, обозначающие различные понятия (ключевые классы различных предметных областей).

Этичное (неэгоистичное) поведение *sc-агента*, касающееся блокировки *sc-элементов* (то есть ограничения к ним доступа другим *sc-агентам*) предполагает соблюдение следующих правил:

(1) Не следует блокировать больше *sc-элементов*, чем это необходимо, то есть не следует «жадничать»;

(2) Как только для какого-либо *sc-элемента* необходимость его блокировки отпадает до завершения выполнения соответствующего действия, этот *sc-элемент* желательно сразу деблокировать (снять блокировку);

Для того, чтобы *sc-агент* проверил возможность работы с каким-либо произвольным *sc-элементом*, он должен либо убедиться в том, что этот *sc-элемент* не входит во множество «полностью заблокированный sc-элемент», либо убедиться в том, что указанный *sc-элемент* входит в указанное множество, но при это связан отношением *полностью заблокированный sc-элемент** с действием, выполняемым этим *sc-агентом*. Очевидно, что больших затрат времени указанная проверка не потребует.

Особой группой полностью заблокированных *sc-элементов* (на время выполнения действия *sc-агентом*) являются вспомогательные *sc-элементы* («леса»), создаваемые только на время выполнения этого действия. Эти sc-элементы в конце выполнения действия должны не деблокироваться, а удаляться.

Если *действие в sc-памяти*, выполняемое *sc-агентом,* завершилось (т.е. стало прошлой сущностью), то *sc-агент* оформляет результат («сухой остаток») этого *действия*, указывая (1) удаленные *sc-элементы* и (2) сгенерированные sc-элементы. Это необходимо, если нам придется сделать откат этого *действия*, т.е возвратиться к состоянию базы знаний до выполнения указанного *действия*;

3.2. Средства спецификации блокировок в sc-памяти

Рассмотрим более подробно типологию и средства описания *структур*, заблокированных на время выполнения того или иного *действия в sc-памяти*.

Бинарное отношение ***блокировка**** связывает знаки *действий в sc-памяти* со знаками *структур* (ситуативных), которые содержат элементы, заблокированные на время выполнения данного действия или на какую-то часть этого периода. Каждая такая *структура* принадлежит какому-либо из *типов блокировки*.

Первым компонентом связок отношения ***блокировка**** является знак *действия в sc-памяти*, вторым – знак заблокированной *структуры*.

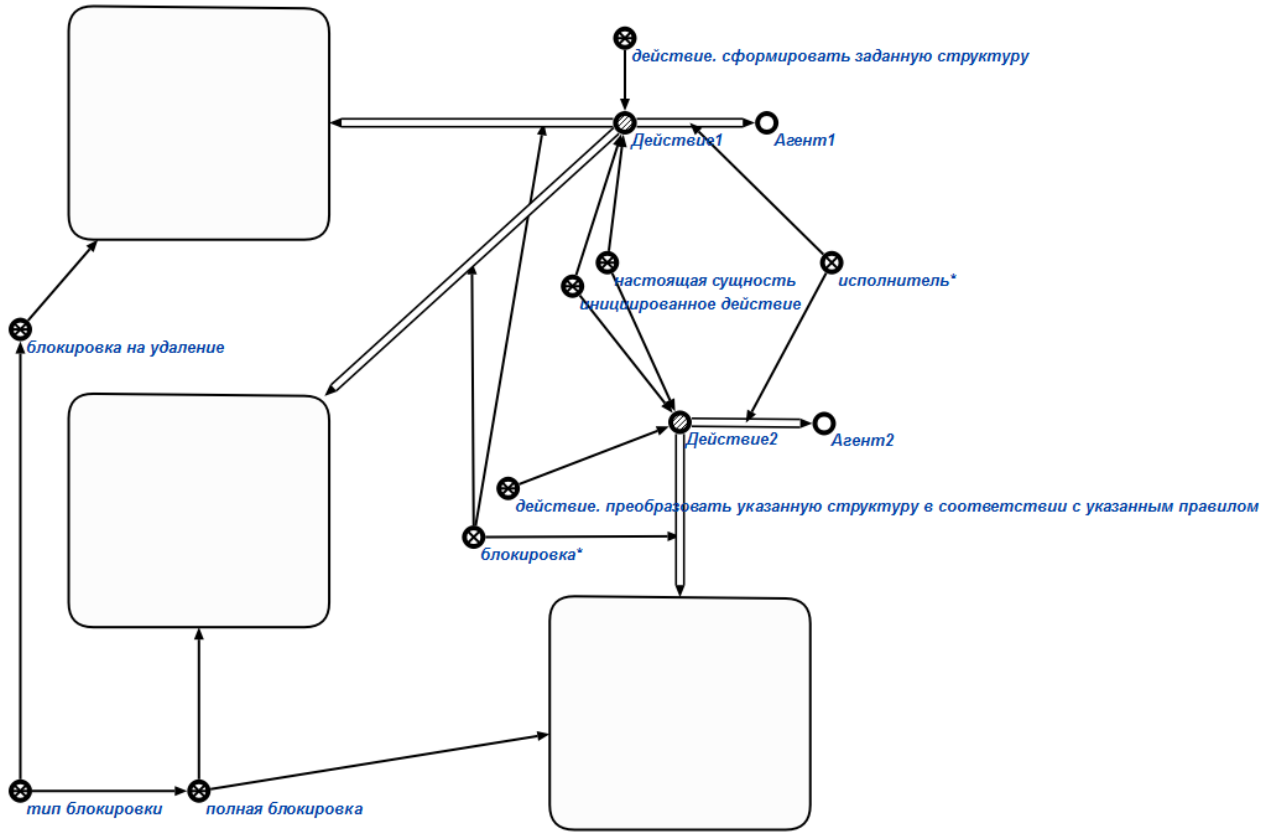

Рисунок 2 – Представление в sc-памяти заблокированных структур

Множество ***тип блокировки*** содержит все возможные классы блокировок, т.е. *структуры*, содержащие *sc-элементы*, заблокированные каким-либо *sc-агентом* на время выполнения им некоторого *действия в sc-памяти*.

тип блокировки
∋ *полная блокировка*
∋ *блокировка на любое изменение*
∋ *блокировка на удаление*

Каждая *структура*, принадлежащая множеству ***полная блокировка*** содержит *sc-элементы*, просмотр и изменение (удаление, добавление инцидентных *sc-коннекторов,* удаление самих *sc-элементов,* изменение содержимого в случае файла) которых запрещены всем *sc-агентам*, кроме собственно *sc-агента*, выполняющего соответствующее данной структуре *действие в sc-памяти*, связанное с ней отношением *блокировка**.

Для того, чтобы исключить возможность реализации *sc-агентов*, которые могут внести изменения в конструкции, описывающие блокировки других *sc-агентов*, все элементы этих конструкций, в том числе, сам знак *структуры*, содержащей заблокированные *sc-элементы* (принадлежащей как множеству ***полная блокировка***, так и любому другому *типу блокировки*) и связки отношения *блокировка**, связывающие эту *структуру* и конкретное *действие в sc-памяти*, добавляются в ***полную блокировку***, соответствующую данному *действию в sc-памяти*. Таким образом, каждой ***полной блокировке*** соответствует петля принадлежности, связывающая ее знак с самим собой.

Каждая *структура*, принадлежащая множеству ***блокировка на любое изменение*** содержит *sc-элементы*, изменение (физическое удаление, добавление инцидентных *sc-коннекторов,* физическое удаление самих *sc-элементов,* изменение содержимого в случае файла) которых запрещено всем *sc-агентам*, кроме собственно *sc-агента*, выполняющего соответствующее данной структуре *действие в sc-памяти*, связанное с ней отношением *блокировка**. Однако не запрещен просмотр (чтение) этих *sc-элементов* любым *sc-агентом*.

Каждая *структура*, принадлежащая множеству ***блокировка на удаление*** содержит *sc-элементы*, физическое удаление которых запрещено всем *sc-агентам*, кроме собственно *sc-агента*, выполняющего соответствующее данной структуре *действие в sc-памяти*, связанное с ней отношением *блокировка**. Однако не запрещен просмотр (чтение) этих *sc-элементов* любым *sc-агентом*, добавление инцидентных *sc-коннекторов*.

Заключение

В работе рассмотрена типология действий, выполняемых в *sc-памяти* агентами обработки знаний в рамках систем, управляемых знаниями, описаны принципы взаимодействия таких агентов и средства синхронизации их деятельности.

Работа выполнена при поддержке гранта БРФФИ-РФФИ-М Ф15РМ-074 «Методы и средства онтологического моделирования для семантических технологий проектирования интеллектуальных систем».

Библиографический список

[IMS, 2016] Метасистема IMS.OSTIS [Электронный ресурс]. Минск, 2016. – Режим доступа: http://ims.ostis.net/. – Дата доступа: 15.01.2016.

[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 /В. В. Голенков [и др.] – Мн. : БГУИР, 2001.

[Голенков, 2012] Голенков, В.В., Гулякина Н.А. Принципы Графодинамические модели парралельной обработки знаний: принципы построения, реализации и проевтирования . – В кн Междунар. научн.-техн. конф. . «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2012). Материалы конф. [Минск, 16-18 февр. 2012 г.]. – Минск: БГУИР, 2012, с. 23-52.

[Голенков, 2013] Голенков, В.В., Гулякина Н.А. Открытый проект, направленный на создание технологии компонентного проектирования интеллектуальных систем./В.В. Голенков, Н.А. Гулякина// Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013): материалы III междунар. науч.-техн.конф. – Минск: БГУИР, 2013 – с.55-78

[Голенков, 2015] Голенков, В.В. Семантическая технология компонентного проектирования систем, управляемых знаниями. Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015): материалы V Междунар.научн.-техн.конф./ В. В. Голенков, Н.А Гулякина// Мн.: БГУИР, 2015

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

[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.

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

[Хоар, 1989] Хоар, Ч. Взаимодействующие последовательные процессы / Ч. Хоар; – М.: «Мир», 1989.

[Шпаковский, 2013] Шпаковский Г.И. Коротко о параллельном программировании и аппаратуре / Г.И. Шпаковский; – Мн.: БГУ, 2013.

[Шункевич, 2013] Шункевич, Д.В. Модели и средства компонентного проектирования машин обработки знаний на основе семантических сетей. – В кн Междунар. научн.-техн. конф. . «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2013). Материалы конф. [Минск, 2013 г.]. – Минск: БГУИР, 2013.

[Шункевич, 2014] Шункевич, Д.В. Машина обработки знаний интеллектуальной метасистемы поддержки проектирования интеллектуальных систем. /Д.В. Шункевич// Открытые семантические технологии интеллектуальных систем (Ostis-2014): Материалы IV международной науч.-тех. конф -- Минск: БГУИР,2014. – с.93-96

[Шункевич, 2016] Шункевич, Д.В. Формальное семантическое описание целенаправленной деятельности различного вида субъектов / Д.В. Шункевич // Открытые семантические технологии интеллектуальных систем (Ostis-2016): Материалы VI международной науч.-тех. конф -- Минск: БГУИР,2016.

ASYNCHRONOUS AND PARALLEL KNOWLEDGE PROCESSING INTERACTION IN COMMON SEMANTIC MEMORY

Shunkevich D.V.

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

shunkevichdv@gmail.com

The paper deals with the basic model of construction machinery knowledge processing systems, knowledge-driven, built on open semantic technology of intelligent systems design. Particularly, it discribe the basic principles of asynchronous and parallel knowledge processing interaction in common semantic memory, typology of acts, agents run in this memory, typology and means of describing locks, necessary for synchronization of agents run on a common semantic memory.

Key words: semantic technologies, action formalisation, multy-agent systems, OSTIS project.

Introduction

The systems managed by knowledge, built on open semantic technology for intelligent systems, will be called *ostis-systems*. As a model of knowleg representation in this tecnology the unified semntic network with set-theoretic interpretation is used. This model of representation will be called SC-code (Semantic computer code). The elementsof this semantic network will be called *sc-nodes* and *sc-connectors* (sc-arches, sc- rib).

Every *ostis-system* consiset of platform-independent unified logical-semantic model of this system(*computer system sc-model*) and platform interpretation of such models. Every computer system sc-model might be decomposed on *sc- model of knolege base*, *sc-model of knowledge processing machine*, *sc-model of interfa*ce. In this work we will focused on *sc-model of knowledge processing machine*.

We list some of the advantages of the proposed approach to the construction of knowledge processing machines:

- Such as knowledge processing machine consiset of agents, which exchange messages only through the common memory, the addition of a new agent or exclusion one or some existing agents, doesn't mean any changes in other agents;
- Agents run parallel and independent of each other, performing different actions in *sc-memory.* Consequently, even a significant increase in the number of agents in the same system does not lead to a deterioration of its performance;
- Basic language for implementing software processing knowledge is the SCP language, and it's texts ate wrote by the same semantic networks as the processes information. At the same time every time you call the program created the unicum scp-process. That gives a possibility to perform multiple operations simultaneously;
- The transfer agent from one system to another is a simple transfer of a fragment of the knowledge base, without any additional operations that depend on the interpretation of the platform;
- The agent specifications and their programmes might be written at the same language with processed knowledge, which significantly reduces the list of special funds intended for knowledge processing machine design.

Conclusion

The paper considers the typology of actions performed in the sc-memory processing agent of knowledge within the system, manage knowledge, describes the principles of interaction between these agents and synchronization of their activities.

This work was supported by grant BRFFR-RFFR – Y F15 PM-074 "Methods and tools for the ontological modeling of semantic technologies for the intelligent systems design."