OSTIS-2012
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822

МОДЕЛЬ КЛАССОВ СЕМАНТИЧЕСКИХ СЕТЕЙ И ИХ ПРЕОБРАЗОВАНИЙ

Тимченко В.А.

Учреждение Российской академии наук Институт автоматики и процессов управления Дальневосточного отделения РАН,
г. Владивосток, Россия

rakot2k@mail.ru

В работе представлена модель для описания классов семантических сетей, а также модель для описания спецификаций преобразования семантических сетей – структурных проекций в терминах описания их классов, являющиеся инвариантными по отношению к технологическим пространствам. Модель классов семантических сетей включает в себя формализм для описания связи класса семантических сетей с элементами конкретного синтаксиса языка их текстового представления.
Ключевые слова: модели преобразования семантических сетей, представление информации семантическими сетями, семантические сети, структурные проекции.

ВВЕДЕНИЕ

Вопросы преобразования разных видов информации (данных, знаний, программ на формальных языках), обрабатываемых программными системами, остаются сложными и, вместе с тем, одними из первостепенных для исследования. С такими задачами приходится сталкиваться специалистам во многих сферах профессиональной деятельности. К таким областям можно отнести, например, инженерию программного обеспечения, инженерию знаний, онтологий, разработку систем для анализа и преобразования программ, моделей и т. д.

Семантические сети успешно используются для представления различного вида данных, знаний, онтологий в разных предметных областях. Поэтому многие содержательные задачи преобразования информации, возникающие в различных технологических пространствах [Kurtev, 2002] [Bezivin, 2005], а также на их стыке, можно сформулировать в терминах преобразования семантических сетей. Задача преобразования семантических сетей в общем случае может быть сформулирована следующим образом: по исходной семантической сети и спецификации преобразования сформировать новую семантическую сеть, удовлетворяющую этой спецификации.

Вместе с тем, в разных технологических пространствах исторически независимо сформировались свои эффективные модели и методы преобразования информации; разработаны программные системы их поддержки. Однако, как показывает практика и опыт использования, методы и формализмы, применяемые в одном технологическом пространстве, зачастую по многим причинам напрямую не могут быть “спроецированы” на другое [Czarnecki, 2006]. Поэтому если задача преобразования возникает на пересечении разных технологических пространств и данный класс задач не охвачен ни одной программной системой, то под этот случай необходимо разрабатывать новое специализированное средство преобразования, либо адаптировать одно из существующих.

В частности существующие модели описания метаинформации (фактически классов семантических сетей) предоставляют уровень абстракции и нотацию, общепринятую в том технологическом пространстве, на которое ориентирована данная модель (например, язык MOF и его разновидности в технологическом пространстве modelware, РБНФ и другие формализмы спецификации грамматик в технологическом пространстве grammarware) [Wimmer, 2007]. Поэтому описание в терминах таких моделей метаинформации из другого технологического пространства либо невозможно, либо сопряжено с большими трудностями.

В данной работе представлены модели классов семантических сетей и их преобразований, являющиеся инвариантными по отношению к технологическим пространствам. Данные модели позволяют определять как структурные

преобразования, так и преобразования из текстового представления информации в структурное (в виде семантической сети) и наоборот.

1. Основные понятия

Введем ряд определений терминов, которые будут в дальнейшем использоваться в работе.

Под *описанием класса семантических сетей информации* (*S*) будем понимать описание устройства информации из некоторой предметной области – ее структуры, свойств и ограничений на возможное содержание в виде семантической сети. Описания классов семантических сетей формируются в соответствии с *Моделью описания классов семантических сетей*.

Под *представлением информации в виде семантической сети* или просто *семантической сетью* (S_{inst}) будем понимать семантическую сеть, являющуюся экземпляром (информацией более низкого уровня общности) по отношению к сети *S*, представляющей собой информацию более высокого уровня общности – метаинформацию. Это означает, что вершины S_{inst}, представляющие понятия, входящие в объемы понятий, представленных вершинами из *S*, являются экземплярами этих вершин из семантической сети *S*, тогда как сами вершины семантической сети *S* являются прототипами для вершин из семантической сети S_{inst}. Таким образом, *S* определяет, в сущности, язык для описания множества S_{inst}.

Под *структурной проекцией* (μ) (*спецификацией преобразования*) класса семантических сетей исходной информации на класс семантических сетей целевой информации будем понимать множество правил (соответствий) $\mu = \{Rule_i\}_{i=1}^{rulescount}$. Каждое правило $Rule_i$ описывает структуру подсети семантической сети целевой информации, являющейся экземпляром некоторой подсети из описания класса семантических сетей целевой информации, сопоставляемое понятию из описания класса семантических сетей исходной информации. Структурные проекции описываются в терминах *Модели описания структурных проекций*.

Используя введенные термины, *преобразование семантических сетей* можно определить как генерацию целевой семантической сети, являющейся экземпляром класса семантических сетей целевой информации, на основе исходной семантической сети, являющейся экземпляром класса семантических сетей исходной информации, по описанию структурной проекции класса семантических сетей исходной информации на класс семантических сетей целевой информации.

2. Модель описания классов семантических сетей

Определим *модель описания классов семантических сетей* как тройку <*Concepts_Network*, *Id_Concept*, *Syntax_Restrictions*>. Рассмотрим далее каждый компонент данной модели.

1. *Concepts_Network* (S_{nc}) = <*Concepts*, *Relations*, *Initial_Concept*> – описание класса сетей понятий. Данное описание базируется на модели представления семантических сетей, предложенной в рамках архитектуры ИРУО [Орлов, 2006a], [Орлов, 2006b].

1.1. *Concepts* = $\{Concept_i\}_{i=1}^{conceptscount}$ – конечное непустое множество понятий (включая *Initial_Concept*). Каждое понятие $Concept_i$ описывается следующим образом: $Concept_i$ = <*Concept_Value*, *Concept_Kind*, *Concept_TerminalType*>.

1.1.1. *Concept_Value* = <*Concept_Value_Type*, *Concept_Value_Value*> – значение понятия описывается своим типом *Concept_Value_Type* и значением *Concept_Value_Value*. Значение понятия есть непустая последовательность символов, которая идентифицирует определенный класс объектов описываемой предметной области, множество значений (сорт) или представляет в ней конкретное (константное) значение.

Concept_Value_Type ∈ {“Строковое”, “Целое”, “Вещественное”, “Логическое”, “Бинарные данные”}.

Concept_Value_Value ∈ *String* ∪ *Integer* ∪ *Real* ∪ *Boolean*. *String* – множество строк. *Integer* – множество целых чисел. *Real* – множество вещественных чисел. *Boolean* – множество {“true”, “false”}. Значением понятия может быть последовательность символов, интерпретируемых как строковая константа (*Concept_Value_Type* = “Строковое”), целое число из множества целых чисел (*Concept_Value_Type* = “Целое”), вещественное число из множества вещественных чисел (*Concept_Value_Type* = “Вещественное”), элемент множества {“true”, ”false”} (*Concept_Value_Type* = “Логическое”), бинарные данные (*Concept_Value_Type* = “Бинарные данные”).

1.1.2. *Concept_Kind* ∈ {“Терминальное”, “Нетерминальное”} − тип понятия. Понятие может быть терминальным (*Concept_Kind* = “Терминальное”) или нетерминальным (*Concept_Kind* = “Нетерминальное”).

1.1.3. *Concept_TerminalType* ∈ {“Идентификатор”, “Константа”, “Сорт”, “Не определено”} − тип терминального понятия. Терминальное понятие может быть идентификатором (*Concept_TerminalType* = “Идентификатор”), константой (*Concept_TerminalType* = “Константа”), или обозначать некоторый сорт (*Concept_TerminalType* = “Сорт”). Если понятие $Concept_i$ является нетерминальным, то *Concept_TerminalType* = “Не определено”.

1.2. *Relations* = $\{Relation_i\}_{i=0}^{relationscount}$ – конечное, возможно пустое, множество отношений. Каждое отношение $Relation_i$ является направленным бинарным отношением (дугой), имеющим, возможно, спецификатор множественности и связывающим два понятия. Отношение описывается следующим образом: $Relation_i$ = <*Relation_EndSp*, *Begin_Concept*, *End_Concept*>.

1.2.1. *Relation_EndSp* ∈ {“Конкретность”, “Единственность”, “Множество”} − спецификатор множественности (кардинальность) – характеристика отношения, определяющая способ порождения понятия-экземпляра (понятия из S_{inst}) вместе с отношением к нему по понятию-концу данного отношения (*End_Concept* – понятие из S_{nc}). *Relation_EndSp* = “Конкретность” (или пустой спецификатор) означает, что может быть порождено только одно понятие-экземпляр и его значение (имя) должно совпадать со значением (именем) его понятия-прототипа (*End_Concept*), иначе говоря, в S_{inst} создается в точности такое же понятие как и понятие-прототип из S_{nc}. *Relation_EndSp* = “Единственность” означает, что по *End_Concept* может быть порождено только одно понятие-экземпляр, но его значение (имя) обязательно должно быть задано при порождении. Порожденное понятие при этом является сущностью из класса объектов или элементом из множества значений, идентифицируемого понятием-прототипом (*End_Concept*) (строка, целое значение, вещественное значение, логическое значение). *Relation_EndSp* = “Множество” означает, что по *End_Concept* может быть порождено любое количество (но, по крайней мере, одно) понятий-экземпляров и их значения (имена) обязательно должны быть заданы при порождении. Каждое порожденное понятие при этом является сущностью из класса объектов или элементом из множества значений, идентифицируемого понятием-прототипом (*End_Concept*).

1.2.2. *Begin_Concept* – понятие, из которого дуга исходит – понятие-начало отношения. *Begin_Concept* ∈ *Concepts* \ *Terminal_Concepts*, где *Terminal_Concepts* – множество терминальных понятий, *Terminal_Concepts* ⊂ *Concepts*. Понятием-началом отношения может быть любое нетерминальное понятие.

1.2.3. *End_Concept* – понятие, в которое дуга входит – понятие-конец отношения. *End_Concept* ∈ *Concepts* \ {*Initial_Concept*}. Понятием-концом отношения может быть любое понятие за исключением *начального понятия*.

1.3. *Initial_Concept* – начальное понятие в описании класса сетей понятий, оно единственно, и через него не может быть выражено ни одно другое понятие из описываемого класса сетей понятий. *Initial_Concept* ∈ *Concepts*.

2. *Id_Concept* – понятие, представляющее идентификатор в описании класса сетей понятий, оно единственно и не может быть выражено через другие понятия, т.е. является терминальным понятием типа “Сорт”. *Id_Concept* ∈ *Concepts*. Понятие выделяется отдельно (если требуется текстовое представление информации) для того, чтобы в абстрактном синтаксисе отличить идентификатор от строковой константы.

3. *Syntax_Restrictions* = <*Lexica*, *Syntax*> – описание синтаксических ограничений. Оно может отсутствовать, если текстовое представление информации не требуется.

3.1. *Lexica* = $\{<Lexem\_Type_i, Definition_i>\}_{i=1}^{5}$, где $Lexem\_Type_i$ ∈ {“Идентификатор”, “Целое число”, “Вещественное число”, “Строковая константа”, “Ограничитель строковой константы”}.

3.2. *Syntax* = $\{<Concept_i, Definition_i>\}$, где $Concept_i$ ∈ *Concepts* \ *Terminal_Concepts*. Синтаксические ограничения задаются для нетерминальных понятий. $Definition_i$ – строка символов, включающая метасимволы и представляющая конкретное лексическое или синтаксическое ограничение для конкретного понятия или вида лексемы.

Приведенная модель для описания синтаксических ограничений имеет свое представление в виде сети понятий, в терминах которой пользователь может формально описывать способ текстового представления информации. Далее подробно рассмотрена эта общая структура, которая может быть использована для представления синтаксических ограничений любого контекстно-свободного языка (рис. 1).

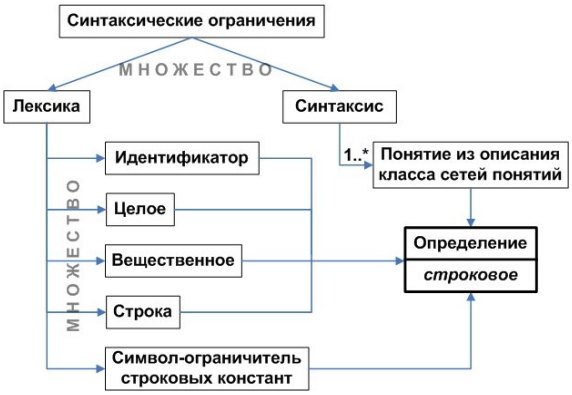

Рисунок 1 – Модель синтаксических ограничений

Синтаксические ограничения языка кроме, собственно, синтаксических определений, содержат еще и ограничения на вид лексем. Синтаксическое определение представляет собой определение объемлющего понятия из описания класса сетей понятий через объемлемые понятия и элементы конкретного синтаксиса языка текстового представления. В определении фиксируется также порядок следования объемлемых понятий и элементов конкретного синтаксиса.

Лексический словарь языка определяется

фиксированным набором вершин, соответствующих базовым типам данных – целое, вещественное и строковое и, кроме этого, определением вида идентификатора или имени. Все ограничения на вид лексем задаются с помощью регулярных выражений с использованием нотации perl5 [Christiansen, 1996]. Строка, представляющая синтаксическое определение, содержит специальные метасимволы, которые будут описаны ниже. Метасимволы не распространяются на лексические ограничения.

Синтаксические ограничения – это корневая вершина в сети понятий, соответствующей модели синтаксических ограничений. Корневой вершиной сети понятий, задающей синтаксические ограничения для конкретного вида информации, должна быть вершина-экземпляр этой вершины "Синтаксические ограничения" с точно таким же названием.

Лексика – в сети понятий, соответствующей синтаксическим ограничениям для конкретного вида информации, должна быть единственная вершина-экземпляр вершины "Лексика" с таким же точно названием. Эта вершина является корнем поддерева, задающего лексические ограничения конкретного вида информации.

Идентификатор – в сети понятий, соответствующей синтаксическим ограничениям для конкретного вида информации, должна быть единственная вершина-экземпляр вершины "Идентификатор" с таким же точно названием. Эта вершина задает ограничения на вид идентификаторов.

Строка – в сети понятий, соответствующей синтаксическим ограничениям для конкретного вида информации, должна быть единственная вершина-экземпляр вершины "Строка" с таким же точно названием. Эта вершина задает ограничения на вид строковых констант.

Символ-ограничитель строковых констант – символ, указывающий начало и конец строковой константы. При синтаксическом анализе эта информация не существенна, ограничители строки можно использовать наряду с другими элементами строковой константы в регулярном определении, например, следующее регулярное выражении задает строку символов следующим образом: "*строка, состоящая из нуля или более символов латинского алфавита (прописных и/или строчных) и также символа пробела, ограниченная одинарными кавычками, причем кавычки исключаются*": *(?<=^')[a-zA-Z,]*(?='$)*. При синтезе текста нужно "знать", как выделить строковую константу в синтезируемом тексте, поэтому этот элемент нужно задать, осуществляя синтез. При анализе достаточно регулярного определения.

Целое – в сети понятий, соответствующей синтаксическим ограничениям для конкретного вида информации, должна быть единственная вершина-экземпляр вершины "Целое" с таким же точно названием. Эта вершина задает ограничения на вид целочисленных констант.

Вещественное – в сети понятий, соответствующей синтаксическим ограничениям для конкретного вида информации, должна быть единственная вершина–экземпляр вершины "Вещественное" с таким же точно названием. Эта вершина задает ограничения на вид вещественных констант.

Синтаксис – в сети понятий, соответствующей синтаксическим ограничениям для конкретного вида информации, должна быть единственная вершина–экземпляр вершины "Синтаксис" с таким же точно названием. Эта вершина является корнем поддерева, задающего синтаксические ограничения конкретного вида информации.

Понятие из описания класса сетей понятий – это нетерминальная вершина, которой соответствует некоторое понятие из S_{nc}, синтаксически объемлющее другие понятия.

Определение – это терминальная вершина, описывающая строковый тип, которой соответствует лексическое или синтаксическое ограничение для понятия, представленного родительской вершиной. В сети понятий, описывающей синтаксические ограничения для конкретного вида информации, вершиной, соответствующей данной вершине будет являться терминальная вершина, значение которой задает в виде строки конкретное лексическое или синтаксическое ограничение для конкретного понятия из описания класса сетей понятий или вида лексемы.

В сети понятий, задающей синтаксические ограничения для конкретного класса семантических сетей, у всех вершин, прототипами которых являются вершины из сети понятий, представляющей модель синтаксических ограничений, с потомком "*определение*", может быть единственный потомок. Другими словами, для лексики и синтаксиса может быть определено единственное ограничение для каждого элемента.

Строка, представляющая синтаксическое определение, может содержать следующие метасимволы:

« » – пробел является символом разделителем и используется для наглядности. Наличие или отсутствие пробела не меняет смысла записи.

|| – две вертикальные черты. Являются метасимволом конкатенации синтаксических конструкций.

? – знак вопроса. Является метасимволом факультативности. Означает нуль или одно вхождение элемента, за которым он находится. Этим элементом может быть как понятие, так и элемент конкретного синтаксиса. В общем случае одно синтаксическое определение, включающее вхождения метасимвола факультативности,

эквивалентно 2^n синтаксическим определениям без этого метасимвола, где n – количество факультативных элементов в этом ограничении.

{;…} – метасимвол перечисления. В фигурные скобки заключается понятие из описания класса сетей понятий (S_{nc}), *Relation_EndSp* отношения, концом которого является данное понятие, есть "Множество". Другими словами, понятие, являющееся экземпляром данного понятия в S_{inst}, может входить в объемлющее понятие (также из S_{inst}) произвольное число раз. Первый символ после фигурной скобки рассматривается как символ-разделитель. Этим символом, помимо символа конкатенации, в тексте будут разделены повторяющиеся элементы. Если символ-разделитель состоит из нескольких литер, то такой символ заключается в двойные кавычки. То же самое можно сделать и с разделителем, записываемым одной литерой, например, {*";"*operator} вместо {;operator}. Строка после символа-разделителя до второй фигурной скобки вся рассматривается как имя некоторого понятия из описания класса сетей понятий S_{nc}.

(…|…|…) – альтернатива. Определяет возможность выбора одного из множества вариантов синтеза или анализа текста. Элементы альтернативы разделяются вертикальной чертой – |. Все эти элементы, так же, как и в случае перечисления, рассматриваются как имена понятий из описания класса сетей понятий S_{nc}. Строка, содержащая символ альтернативы, не может содержать других конструкций вне скобок, относящихся к определяемой альтернативе. Многоточие "…" не является метасимволом и означает, что в этой позиции должно находиться имя некоторого понятия из описания класса сетей понятий (S_{nc}).

"" – двойные кавычки. Используются для определения элементов конкретного синтаксиса, соответствующего описываемому синтаксическому ограничению.

\ – обратная косая черта. Используется для определения элементов конкретного синтаксиса языка, которые записываются теми же литерами, что и служебные символы.

Пусть, например, в качестве метаинформации выбрано описание языка Милан [Ершов, 1977]. *Описание класса сетей понятий* (S_{nc}), представляющее устройство языка Милан, показано на рис. 2. Здесь "Program MILAN" – начальное понятие, "id" – понятие, представляющее идентификатор. Символ "+" представляет спецификатор множественности отношения "множество", а символ "!" – "единственность".

Представление программы на языке Милан в виде семантической сети (информация S_{inst}) показано на рис. 3. На рисунке также представлено *текстовое представление* этой программы, соответствующее представлению в виде семантической сети понятий.

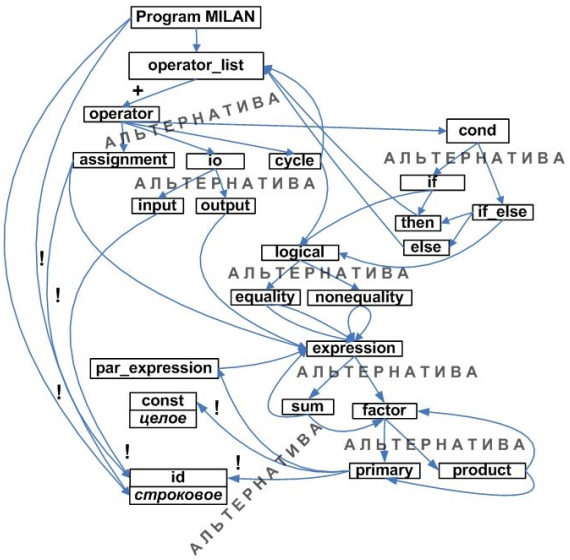

Рисунок 2 – Описание класса сетей понятий, представляющего устройство языка Милан

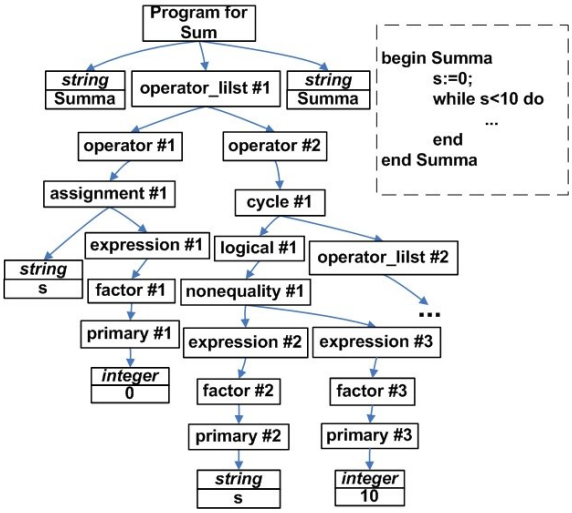

Рисунок 3 – Программа в терминах сети понятий, описывающей устройство языка Милан

Фрагмент описания синтаксических ограничений языка Милан представлен на рис. 4.

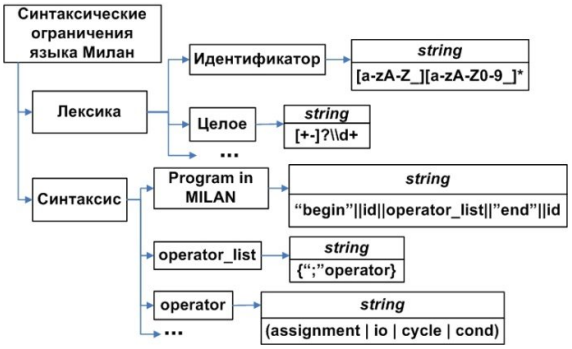

Рисунок 4 – Фрагмент описания синтаксических ограничений для языка Милан

3. Модель описания структурных проекций

В определении структурной проекции было сказано, что описание проекции μ представляет собой множество правил (соответствий) $\mu = \{Rule_i\}_{i=1}^{rulescount}$. В общем виде каждое правило $Rule_i$ можно представить следующим образом: $\alpha \rightarrow \beta$, где $\alpha \in Source\_Concepts$, где $Source\_Concepts$ – множество понятий из описания класса семантических сетей исходной информации SS_{nc};

$\beta = \{Target\_Concept\_Descr_i\}_{i=1}^{conceptscount} \cup \{Target\_Relation\_Descr_j\}_{j=0}^{relationscount}$, где

$\{Target\_Concept\_Descr_i\}_{i=1}^{conceptscount}$ – возможно пустое множество описаний порождаемых понятий в семантической сети целевой информации (TS_{inst}). $Target\_Concept\_Descr_i$ есть описание порождения понятия в TS_{inst}. $\{Target\_Relation\_Descr_j\}_{j=0}^{relationscount}$ – возможно пустое множество описаний порождаемых отношений в семантической сети целевой информации (TS_{inst}). $Target\_Relation\_Descr_j$ есть описание порождения отношения в TS_{inst}. При этом $\beta \neq \varnothing$ и для любых α_1, $\alpha_2 \in Source\_Concepts$, если правила $\alpha_1 \rightarrow \beta_1$, $\alpha_2 \rightarrow \beta_2$ входят в μ, то $\alpha_1 \neq \alpha_2$.

Описание создания понятия в TS_{inst} (*Target_Concept_Descr*) представляет собой шестерку: *Target_Concept_Descr* = <*Proto_Concept_Name*, *SConceptValue*, *isTProtoConceptName*, *Storable*, *Loadable*, *Concept_Value*>.

1. *Proto_Concept_Name* – имя понятия-прототипа из описания класса сетей понятий целевой информации (TS_{nc}), которое представляет собой непустую последовательность символов.

2. *SConceptValue* = <*isSConceptValue*, *isTerminal*>. Описывает выбор значения для создаваемого в TS_{inst} понятия на основе имени (значения) понятия из семантической сети исходной информации (SS_{inst}).

2.1. *isSConceptValue* ∈ {“true”, “false”}. Определяет, нужно ли создаваемому в TS_{inst} понятию присваивать в качестве значения имя (значение) понятия из SS_{inst}.

2.2. *isTerminal* ∈ {“true”, “false”}. Определяет, значение какого понятия из SS_{inst} нужно присвоить: экземпляра понятия, стоящего в левой части правила (“false”), или терминального понятия-потомка экземпляра понятия, стоящего в левой части правила (“true”).

3. *isTProtoConceptName* ∈ {“true”, “false”}. Определяет, нужно ли создаваемому в TS_{inst} понятию присваивать в качестве значения имя (значение) его понятия-прототипа из TS_{nc}.

4. *Storable* = <*isStorable*, *Alias*, *storeConcept*>. Описывает сохранение значений в хеш-таблицу.

4.1. *isStorable* ∈ {“true”, “false”}. Определяет, нужно ли значение, которое будет присвоено создаваемому в TS_{inst} понятию, поместить (сохранить) в хеш-таблицу.

4.2. *Alias* – строка символов, возможно пустая, представляющая псевдоним, под которым нужно сохранить значение понятия в хеш-таблицу.

4.3. *storeConcept* ∈ {“true”, “false”}. Определяет, нужно ли поместить (сохранить) в хеш-таблицу создаваемое в TS_{inst} понятие целиком или же только его значение (имя).

5. *Loadable* = <*isLoadable*, *Alias*>. Описывает извлечение значений из хеш-таблицы.

5.1. *isLoadable* ∈ {“true”, “false”}. Определяет, будет ли значение, которое необходимо присвоить создаваемому в TS_{inst} понятию, извлекаться из хеш-таблицы.

5.2. *Alias* – строка символов, возможно пустая, представляющая псевдоним, по которому нужно получить значение для понятия из хеш-таблицы.

6. *Concept_Value* = <*Concept_Value_Type*, *Concept_Value_Value*>. Описывает значение (имя), присваиваемое создаваемому в TS_{inst} понятию.

6.1. *Concept_Value_Type* ∈ {“Строковое”, “Целое”, “Вещественное”, “Логическое”, “Бинарные данные”, “Вычислимое целое”, “Вычислимое вещественное”, “Вычислимое строковое”} – тип значения.

6.2. *Concept_Value_Value* – собственно значение понятия – строка символов, интерпретируемая в зависимости от значения типа понятия. Значением понятия может быть строковая константа (*Concept_Value_Type* = “Строковое”), целое число из множества целых чисел (*Concept_Value_Type* = “Целое”), вещественное число из множества вещественных чисел (*Concept_Value_Type* = “Вещественное”), элемент множества {“true”, “false”} (*Concept_Value_Type* = “Логическое”), бинарные данные (*Concept_Value_Type* = “Бинарные данные”). Если тип значения понятия “Вычислимое целое” (*Concept_Value_Type* = “Вычислимое целое”), то значение понятия интерпретируется как выражение, значение которого требуется вычислить, и результат вычисления будет интерпретироваться как целое число. Переменные в этом выражении могут инициализироваться либо значением понятия из семантической сети понятий, представляющей исходную информацию SS_{inst}, либо значениями из хеш-таблицы. Если тип значения понятия “Вычислимое вещественное” (*Concept_Value_Type* = “Вычислимое вещественное”), то аналогично предыдущему случаю, но результат вычисления будет интерпретироваться как вещественное число. Если тип значения понятия “Вычислимое строковое” (*Concept_Value_Type* = “Вычислимое строковое”), то вычисленное значение будет строковой константой.

Описание порождения отношения в TS_{inst}

(*Target_Relation_Descr*) представляет собой тройку: *Target_Relation_Descr* = <*Relation_Begin_Concept*, *Relation_End_Concept*, *Number*>.

1. *Relation_Begin_Concept* = <*Begin_Concept_Name*, *isProtoConcepName*, *isLoadable*, *Alias*>. Описывает понятие-начало отношения.

1.1. *Begin_Concept_Name* – имя понятия-начала отношения, представляющее собой непустую последовательность символов.

1.2. *isProtoConceptName* ∈ {“true”, “false”}. Определяет, является ли это имя именем понятия из TS_{nc}.

1.3. *isLoadable* ∈ {“true”, “false”}. Определяет, будет ли понятие-начало отношения, извлекаться из хеш-таблицы.

1.4. *Alias* – строка символов, возможно пустая, представляющая псевдоним, по которому нужно получить понятие из хеш-таблицы.

2. *Relation_End_Concept* = <*End_Concept_Name*, *isProtoConceptName*, *isLoadable*, *Alias*>. Описывает понятие-конец отношения. Аналогично описанию *Relation_Begin_Concept*, только *End_Concept_Name* есть имя понятия-конца отношения, которое также представляет собой непустую последовательность символов.

3. *Number* – целое положительное число – порядковый номер отношения. Определяет номер, под которым создаваемое отношение будет добавлено к отношениям, в которых уже участвует понятие-начало.

Для формирования пользователем множества правил в терминах приведенной выше модели разработан язык для описания структурных проекций. Специфицированы его абстрактный и конкретный синтаксис, а также операционная семантика.

Заключение

Разработана модель для описания классов семантических сетей. Данная модель базируется на формализме представления семантических сетей, предложенном в рамках архитектуры ИРУО, которая в свою очередь основана на максимально абстрактных категориях “сущность” и “отношение” и предлагает схему организации понятий, предназначенную для описания информационных моделей различных уровней общности. Расширение модели позволяет специфицировать преобразования вида “текст – семантическая сеть” и “семантическая сеть – текст” одинаковым образом (используется один и тот же формализм, нотация, вид правил и т.д.). Это означает, в частности, что при выполнении эндогенных преобразований одна и та же спецификация используется для выполнения этих двух типов преобразований. При этом в обоих случаях никаких дополнительных описаний не требуется.

Разработаны модель и язык для описания структурных проекций, предназначенные для описания преобразований вида “семантическая сеть – семантическая сеть”. В отличие от многих других моделей и языков для описания спецификаций преобразования они не ориентированы на конкретное технологическое пространство и позволяют специфицировать преобразования в терминах модели для описания классов семантических сетей.

Работа выполнена при финансовой поддержке РФФИ, проект 10-07-00089-а.

Библиографический список

[Kurtev, 2002] Kurtev, I. Technological Spaces: An Initial Appraisal / I. Kurtev, J. Bezivin, M. Aksit // Int. Federated Conf. (DOA, ODBASE, CoopIS), Industrial track, Irvine, 2002.

[Bezivin, 2005] Bezivin, J. Model-based Technology Integration with the Technical Space Concept / J. Bezivin, I. Kurtev // In Proceedings of the Metainformatics Symposium, Springer-Verlag, 2005.

[Czarnecki, 2006] Czarnecki, K. Feature-based survey of model transformation approaches / K. Czarnecki, S. Helsen // IBM Systems Journal, special issue on Model-Driven Software Development. 45(3). – 2006. – Pp. 621 – 645.

[Wimmer, 2007] Wimmer, M. Towards Model Transformation Generation By-Example / M. Wimmer, M. Strommer, H. Kargl, G. Kramler // In Proceedings of the 40th Hawaii International Conference on Systems Science (HICSS-40), [Electronic resource], CD-ROM / Abstracts Proceedings, Big Island, HI, USA. – 2007. – 285 p.

[Орлов, 2006a] Орлов, В.А. Компьютерные банки знаний. Универсальный подход к решению проблемы редактирования информации / В.А. Орлов, А.С. Клещев // Информационные технологии. – 2006. – №5. – С. 25 – 31.

[Орлов, 2006b] Орлов, В.А. Компьютерные банки знаний. Модель процесса редактирования информационного наполнения / В.А. Орлов, А.С. Клещев // Информационные технологии. – 2006. – №7. – С. 11 – 16.

[Christiansen, 1996] Christiansen, T. PERL5 Regular Expression Description / T. Christiansen // [Electronic resource]. URL: http://www.perl.com/doc/FMTEYEWTK/regexps.html. – 1996. (дата обращения: 15.05.2010).

[Ершов, 1977] Ершов, А.П. О сущности трансляции / А.П. Ершов // Препринт, ВЦ Сибирское отделение АН СССР, Новосибирск. – 1977.

THE MODEL OF SEMANTIC NETWORK CLASSES AND THEIR TRANSFORMATIONS

Timchenko V.A.

Institute of Russian Academy of Sciences Institute of Automation and Control Processes Far Eastern Branch of RAS Vladivostok, Russia

rakot2k@mail.ru

The paper presents a model for semantic networks classes description, as well as a model for structure mappings description. The latter is model for semantic networks transformations definitions in terms of their classes. Both models are invariant to the different technological spaces. The model for semantic networks classes includes a formalism for describing a relationship between the semantic networks class and

the concrete syntax elements of the language for their textual representation.

INTRODUCTION

The problem of different types of information (data, knowledge, computer programs) transformation is still knotty and, at the same time, one of the crucial for researche. Different specialists in their professional fields deal with such problems. These fields include, for example, software engineering, knowledge engineering, ontologies engineering, development of the systems for the analysis and transformation of programs, models, etc.

Semantic networks have been successfully used for representation of a various types of data, knowledge, ontologies in different domains. Therefore many conceptual problems of transformation information appearing in different technological spaces [Kurtev, 2002] [Bezivin, 2005], as well as at their joint, can be formulated in terms of the semantic networks transformation. The problem of semantic networks transformation generally can be formulated as follows: on basis of initial semantic network and the transformation specification to generate the new semantic network that meets this specification.

At the same time, in different technological spaces the efficient models and methods as well as software systems for information transformation were developed independently. However, as practice and experience find out, the methods and formalisms applied in a technological space often cannot be directly "mapped" to another one [Czarnecki, 2006]. Therefore, if the problem of transformation arises at the joint of different technological spaces and this class of problems does not solved with some software system, it is necessary in this case to develop a new specialized transformation tool or to adapt an existing one.

In particular, the existing models for the definition of a meta-information (in fact classes of semantic networks) provide a level of abstraction and notation generally accepted in the technological spaces on which these models are oriented. For example, the MOF language and its dialects are oriented on the modelware technology space, EBNF and other formalisms for the grammars specification are oriented on the grammarware technological space [Wimmer, 2007]. Therefore, the definition in terms of such models the meta-information from different technological spaces is either impossible or very difficult.

The paper presents the model of semantic networks classes and the model for their transformations. Both models are invariant to the different technological spaces. These models allow to define both structural transformations and transformations from textual representation of information into structural representation (in the form of a semantic network) and vice versa.

MAIN PART

The first section introduces the basic terms used further and their definitions: *semantic networks class*, *semantic network instance*, *the specification of the of semantic networks transformation*, *the transformation of semantic networks*.

The second section presents the model for semantic networks classes description. The model is based on the formalism for semantic networks representation proposed in the IDEA framework. This model includes a formalism for describing a relationship between the semantic networks class and the concrete syntax elements of the language for their textual representation – the syntax restrictions. Each syntax restriction defines a nonterminal concept from a semantic networks class through others nonterminal or terminal concepts from the semantic networks class and the concrete syntax elements. The definition fixes the sequence of the concepts and elements of the concrete syntax. The lexeme corresponds to one of basic data types – integer, real and string or to identifier. All the restrictions on the type of CF language lexemes are specified with regular expressions using the perl5 notation.

The third section presents the model for semantic networks transformations definitions in terms of their classes. Each rule matches the concept from source semantic networks class and the description of target semantic network subnet structure generation. This subnet is an instance of a target semantic networks class subnet. The structure of the target semantic networks subnet in each rule is defined by the description the set of concepts and relationships generation. The techniques of the concepts (relations) generation in the target semantic network from their meta-concepts (meta-relations) are described.

CONCLUSION

The model for semantic networks classes description is developed. This model is based on the formalism for semantic networks representation proposed in the IDEA framework. The formalism is in turn based on the most abstract categories of "essence" and "relation' and provides a possibility for data models of different levels of generality definition. The extension of the model allows specifying "text-to-semantic network" and "semantic network-to-text" transformations in the same way (using the same formalism, notation, rules, etc.). It means in particular that if an endogenous transformation is performed the same specification is used for these two types of transformations. In both cases, no additional definitions are required.

The model and language for structure mappings description are developed. They are used for "semantic network-to-semantic network" transformations definition. Unlike many other models and languages for the transformation specifications definition, they are invariant to the different technological spaces and allow specifying the transformations in terms of the model for semantic networks classes description.

The paper was written with financial support from RFBR, project 10-07-00089-a.