Взаимное отображение канонической информационной модели и языка OWL 2♣

© С.А. Ступников, Н.А. Скворцов

Институт проблем информатики РАН, г. Москва
ssa@ipi.ac.ru, nskv@ipi.ac.ru

Аннотация

В статье рассматриваются основные аспекты взаимного отображения канонической информационной модели и отличительных конструкций языка OWL 2. Построенное отображение позволяет использовать для решения задач вывода в онтологиях канонической модели автоматические системы вывода для дескриптивных логик. Тем самым достигается верификация концептуального моделирования предметных посредников и интеграции в посредниках неоднородных информационных ресурсов.

1 Введение

Развитие разнообразных информационных моделей в современном мире, а также быстрое увеличение числа компонентов и сервисов, представленных в этих моделях, приводят к необходимости разработки и совершенствования методов и средств интеграции неоднородных информационных ресурсов.

Одной из перспективных технологий интеграции неоднородных ресурсов являются *предметные посредники* [1, 14] – специальный вид программного обеспечения, образующий промежуточный слой между пользователем (приложением) и ресурсами. При разработке посредников предпочтительным является *движимый приложениями* подход. Описание предметной области приложения в таком подходе образуется независимо от ресурсов (в терминах понятий, структур данных, функций, процессов), а затем релевантные приложению ресурсы отображаются в это описание. Преимуществами подхода являются масштабируемость по отношению к числу ресурсов, возможность достижения семантической интеграции ресурсов в контексте конкретного приложения и доказательной идентификации релевантных приложению ресурсов, а также повышение стабильности спецификации посредника в процессе эволюции ресурсов, релевантных приложению.

Труды 12й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL'2010, Казань, Россия, 2010

Создание предметного посредника состоит из двух фаз. На фазе *консолидации* происходит концептуальное моделирование предметной области посредника силами некоторого экспертного сообщества. Во время *операционной* фазы провайдеры информационных ресурсов регистрируют ресурсы в посреднике, т. е. представляют спецификации ресурсов в терминах концептуальной модели посредника.

Одно из центральных мест в технологии посредников занимает понятие *канонической информационной модели*, служащей общим языком, унифицирующим разнообразные модели ресурсов. В данной статье в качестве канонической модели рассматривается язык СИНТЕЗ [2], нацеленный на разработку предметных посредников для решения задач в средах неоднородных ресурсов. Применение технологии посредников с использованием языка СИНТЕЗ как канонической модели для решения научных задач рассматривается в работах [12, 13].

Обе фазы создания посредников существенным образом используют онтологические модели, методы и средства. Так, концептуальная модель посредника включает онтологию, описывающую понятия и отношения, характерные для предметной области. При регистрации ресурсов важным моментом является согласование (интеграция) онтологии посредника и онтологий ресурсов [3].

В процессе концептуального проектирования и интеграции онтологий возникает ряд задач, для решения которых необходимы методы формального определения и инструментальные средства верификации онтологий. К таким задачам относятся доказательство непротиворечивости онтологий и поглощения одного понятия другим, определение полной иерархии понятий и т. д. Автоматическое решение подобных задач возможно в рамках дескриптивных логик [4] при помощи специализированных инструментальных средств – систем вывода (например, Pellet [5]).

Необходимым шагом, обеспечивающим применение автоматических систем вывода в дескриптивных логиках для верификации концептуального моделирования и интеграции онтологий в предметных посредниках, является построение взаимного отображения канонической информационной модели (языка СИНТЕЗ) и онтологической модели, служащей входным языком для систем вывода.

Целью данной статьи является построение корректного взаимного отображения канонической модели и языка OWL 2 [6], расширяющее и уточняющее ранее проведенные исследования по отображению канонической модели и языка OWL [8].

Статья организована следующим образом. В разделе 2 кратко описаны конструкции языка СИНТЕЗ, использующиеся для моделирования онтологий. В разделе 3 на примерах рассмотрены основные детали взаимного отображения канонической модели и языка OWL 2. В разделе 4 рассмотрены интерпретации и модели онтологий языка СИНТЕЗ, а также сводимость задач вывода в языке СИНТЕЗ. В разделе 5 доказана корректность отображения канонической модели и языка OWL 2.

Предполагается, что читатель знаком с языком OWL 2 [6, 7, 9, 11].

2 Каноническая информационная модель – язык СИНТЕЗ

Язык СИНТЕЗ представляет собой комбинированную слабоструктурированную и объектную модель данных. Слабоструктурированные (самоописывающие) данные представлены *фреймами* [2]. Фреймы используются для описания любого объекта, включая объекты самого языка, такие как спецификации типов, классов, функций, утверждений. Фрейм можно рассматривать как множество атрибутов, называемых *слотами*. Каждый слот может иметь множество значений. Информация, характеризующая слот как целое, может быть представлена в *метаслоте*. Слоты разделяются знаком точка с запятой, сам фрейм заключается в фигурные скобки. Имя слота и множество его значений разделяются двоеточием, значения одного слота разделяются запятыми.

Единицей определения канонической модели является *модуль*. Каждый модуль задает обобщенное представление информационных ресурсов, либо является модулем спецификации предметной области или концептуального проекта информационной системы.

Типизированные данные должны соответствовать *абстрактным типам данных* (АТД), описывающих состояние и поведение своих экземпляров в терминах *атрибутов* и методов типа. В определения типов могут быть включены *инварианты* типов (ограничения, выраженные в виде замкнутой логической формулы). Помимо АТД, язык содержит также обширную коллекцию встроенных типов данных. Отношение подтипа над множеством абстрактных типов данных формирует решётку с *Taval* (общий супертип) в корне и типом *Tnone* (общий подтип) внизу решётки. Значение подтипа может использоваться всюду, где ожидается значение супертипа; подтип наследует элементы спецификации супертипа; допускается множественное наследование спецификаций подтипом.

Однородные совокупности объектов предметной области представляются в канонической модели *классами*. Явно разграничивая объявление типа и объявление коллекции (класса), язык СИНТЕЗ подчеркивает роль коллекции как представителя множества объектов и роль типа как спецификации структуры и поведения объектов, связанных с коллекцией. Определения классов также могут включать инварианты.

Коллекции могут связываться друг с другом отношением обобщения-специализации (класс–подкласс). Суперкласс включает в себя подкласс как множество; тип экземпляра суперкласса является супертипом типа экземпляра подкласса.

Атрибуты объектов в языке рассматриваются, в свою очередь, как объекты метаклассов ассоциаций, устанавливающие соответствие между набором объектов в области определения ассоциации и набором объектов в области значений ассоциации.

Для определения формул используется язык логических формул многосортного объектного исчисления – типизированный вариант логики предикатов первого порядка.

3 Взаимное отображение канонической информационной модели и языка OWL 2

Отображение онтологий. Онтология, представляемая в языке СИНТЕЗ онтологическим модулем (например, *Staff* – описание персонала компании), отображается в одноименную онтологию OWL.

СИНТЕЗ	OWL
{ Staff; in: module, ontology; … }	Ontology(Staff …)

Заметим, что конструкции OWL в данной статье представлены в *функциональном синтаксисе* [9], позволяющем компактно описывать структуру онтологий.

Отображение классов. Класс (например, *employee*) и тип его экземпляров (*Employee*) языка СИНТЕЗ представляются в OWL единой конструкцией класса.

СИНТЕЗ	OWL
{ employee; in: class; instance_section: Employee; … } { Employee; in: type; … }	Class(employee)

Отображение свойств объектов. Свойства объектов представляются в языке СИНТЕЗ при помощи метаклассов ассоциаций (например, *WorksOn*), задающих связь между объектами из области определения ассоциации (класс *employee*) и объектами их области значения ассоциации (класс *project*). Атрибут АТД (например, *worksOn* типа *Employee*) может быть объявлен принадлежащим некоторому метаклассу ассоциаций. Такие свойства представляются в OWL конструкцией *ObjectProperty* (если область значения ассоциации – коллекция объектов) или *DataProperty* (если область значения ассоциации – множество значений встроенного типа *real, string* и т. д., например, атрибут *salary* типа *Empolyee*). Ат-

рибуты, тип которых не является типом множества (например, *salary*), имеют в языке СИНТЕЗ кардинальность *1* по умолчанию. В OWL кардинальность атрибута указывается явно (в случае атрибута *salary* при помощи конструкции *DataExactCardinality*).

СИНТЕЗ	OWL

{ WorksOn;
in: association, metaclass;
instance_section: {
domain: employee;
range: project;
};
}
{ Employee; in: type;
worksOn: {set;
type_of_element: Project;};
metaslot in: WorksOn; end
 	
salary: integer;
}
	ObjectProperty(
	worksOn)
	ObjectPropertyDomain(
	worksOn employee)
	ObjectPropertyRange(
	worksOn project)
	
	DataProperty(salary)
	DataPropertyDomain(
	salary employee)
	DataPropertyRange(
	salary xsd:integer)
	SubClassOf(employee
	DataExactCardinality (
	1 salary))

Отображение отношений между классами. Отношение класс – подкласс в простых случаях (например, *manager* – подкласс *employee*) представляется в языке СИНТЕЗ в слоте *superclass* соответствующего подкласса. Более сложные отношения между классами (например, *manager* $\subseteq$ *areaManager* $\cup$ *topManager*) представляются инвариантами классов (*managerCompleteness*). В OWL отношения между классами представляются при помощи аксиом классов.

СИНТЕЗ	OWL

{ manager; in: class;
superclass: employee;
instance_section: {
managerCompleteness: {
in: invariant;
{ manager <=
union(areaManager,
topManager) }
};
}; }
	SubClassOf(manager
	employee)
	
	SubClassOf(manager
	ObjectUnionOf(
	areaManager
	topManager
	)
	)

Отображение фактов. Факты (утверждения о значениях свойств объектов) представляются в языке СИНТЕЗ константами объектных типов. В OWL факты представляются утверждениями о существовании индивида (*NamedIndividual*), о принадлежности индивида классу (*ClassAssertion*), о значениях свойств (*DataPropertyAssertion, ObjectPropertyAssertion*) и т. д.

СИНТЕЗ	OWL
{ JohnJohnson; in: employee; salary: 3200; boss: JackJackson; }	NamedIndividual(JohnJohnson) ClassAssertion(employee JohnJohnson) DataPropertyAssertion(salary JohnJohnson 3200) ObjectPropertyAssertion(boss JohnJohnson JackJackson)

Отображение самоограничений. Пример инварианта класса языка СИНТЕЗ (*topManager*), описывающего ситуацию, когда объект согласно некоторому свойству (*boss*) соотносится сам с собой (топ-менеджер является своим собственным начальником), приведен ниже. В OWL данная ситуация описывается конструкцией *ObjectHasSelf*.

СИНТЕЗ	OWL
all m/TopManager (is_in(topManager, m) ->> is_in(m.boss, m))	SubClassOf(topManager ObjectHasSelf(boss))

Здесь *all* обозначает квантор всеобщности, знак / – типизацию переменной типом, предикат *is_in* – принадлежность элемента множеству, знак –> – логическую импликацию.

Отображение ограничений на размер области и класс значений свойств. Ниже приводится пример инварианта класса (*employee*), утверждающий, что среди начальников (свойство *boss*) любого служащего должен быть, по крайней мере, один топ-менеджер. В OWL данная ситуация представляется конструкцией *ObjectMinCardinality*.

СИНТЕЗ	OWL
all e/Employee (is_in(employee, e) ->> cardinal({m/Manager | is_in(e.boss, m) & is_in(topManager, m)}) >= 1)	SubClassOf(Employee ObjectMinCardinality(1 boss topManager))

Здесь функция *cardinal* вычисляет мощность множества.

Отображение рефлексивных, иррефлексивных и асимметричных свойств. Для представления свойств специального вида в языке СИНТЕЗ определяются соответствующие метаклассы ассоциаций. Так, для представления рефлексивных свойств определяется метакласс ассоциаций *reflexive*:

```
{ reflexive; in: association, metaclass;
 instance_section: {
 reflexivity: { in: invariant;
 all x ( (is_in(this.domain, x) -> is_in(this, [x, x]))
 & (is_in(this.range, x) -> is_in(this, [x, x])) )
 } } }
```

Здесь *this* обозначает класс ассоциаций, принадлежащий метаклассу *reflexive*, *domain* обозначает область определения класса ассоциаций, *range* – область значения, *[a, b]* обозначает пару объектов, участвующих в ассоциации. Метакласс каждого рефлексивного свойства становится подклассом *reflexive* (например, *SameSalary*). В OWL рефлексивность отражается при помощи конструкции *ReflexiveObjectProperty*.

СИНТЕЗ	OWL
{ SameSalary; in: association, metaclass; superclass: reflexive; }	ReflexiveObjectProperty(sameSalary)

Аналогично отображаются иррефлексивные и асимметричные свойства.

Отображение непересекающихся свойств. Ниже рассмотрен пример инварианта класса (*employee*), описывающего непересекающиеся свойства (*hireDate* и *fireDate*). В OWL непересекающиеся свойства представляются конструкциями *DisjointObjectProperties* и *DisjointDataProperties*.

СИНТЕЗ	OWL
all e/Employee(is_in(employee, e) -> e.hireDate <> e.fireDate)	DisjointDataProperties(hireDate fireDate)

Отображение иерархии композиций свойств. Ниже рассмотрен пример композиции свойств (*locatedIn* и *partOf*), являющейся подсвойством некоторого свойства (*locatedIn*). Если город *c* находится в регионе *r* и регион *r* является частью региона *s*, то *c* находится в *s*. Иерархия композиций свойств выражается в OWL при помощи комбинации конструкций *SubPropertyOf* и *ObjectPropertyChain*.

СИНТЕЗ	OWL
all c/City, r/Region, s/Region (is_in(city, c) & is_in(region, r) & is_in(region, s) & is_in(c.locatedIn, r) & is_in(r.partOf, s) -> is_in(c.locatedIn, s)	SubPropertyOf(ObjectPropertyChain(locatedIn partOf) locatedIn)

Отображение ключей. Ключи представляются в языке СИНТЕЗ при помощи встроенных утверждений уникальности атрибутов (*unique*). В OWL ключи представляются при помощи конструкции *HasKey*.

СИНТЕЗ	OWL
{ Employee; in: type; empCode: integer; keys: { unique; { empCode } }; }	HasKey(employee empCode)

4 Интерпретации и модели онтологий языка СИНТЕЗ

В качестве семантических структур, интерпретирующих онтологии языка СИНТЕЗ, будут рассмотрены те же структуры, что используются в [7] для интерпретации онтологий OWL. Это позволит упростить доказательство корректности отображения языка СИНТЕЗ и OWL.

Так, интерпретация $I = \langle \Delta_I, \Delta_D, \bullet^C, \bullet^{OP}, \bullet^{DP}, \bullet^I, \bullet^{DT}, \bullet^{LT} \rangle$ представляет собой кортеж из следующих компонентов:

- Δ_I – объектный домен;
- Δ_D – домен типов данных;
- $\bullet^C$ – функция интерпретации классов; для каждого класса *Cl* выполнено $(Cl)^C \subseteq \Delta_I$;
- $\bullet^{OP}$ – функция интерпретации свойств объектов, область значения которых – коллекция объектов; для каждого свойства *op* выполнено $(op)^{OP} \subseteq \Delta_I \times \Delta_I$;
- $\bullet^{DP}$ – функция интерпретации свойств объектов, область значения которых – множество значений встроенного типа данных; для каждого свойства *dp* выполнено $(dp)^{DP} \subseteq \Delta_I \times \Delta_D$;
- $\bullet^I$ – функция интерпретации объектов; для каждого объекта *o* выполнено $(o)^I \in \Delta_I$;
- $\bullet^{DT}$ – функция интерпретации встроенных типов данных; для каждого типа *dt* выполнено $(dt)^{DT} \subseteq \Delta_D$;
- $\bullet^{LT}$ – функция интерпретации литералов; для каждого литерала *l* типа данных *dt* выполнено $(l)^{LT} \in (dt)^{DT}$.

Интерпретация *I удовлетворяет* онтологии *O* (является *моделью O*), если все конструкции онтологии удовлетворяют соответствующим условиям, приведенным в нижеследующих таблицах. Для упрощения приведены лишь конкретные конструкции, рассмотренные в примерах раздела 3.

Условия для свойств объектов. Когда класс (*employee*) связывается со своим типом экземпляров (*Employee*), на область определения, область значения и кардинальность свойств, составляющих тип, накладываются соответствующие ограничения. В случае, когда метакласс ассоциаций свойства является подклассом некоторого специального метакласса (например, метакласс *Same Salary* свойства *sameSalary* является подклассом метакласса *reflexive*), свойство должно удовлетворять всем инвариантам суперкласса (свойство *sameSalary* должно быть рефлексивным).

Конструкция СИНТЕЗ	Условие

{ employee; in: class;
instance_section:
Employee;
… }
{ Employee; in: type;
worksOn: {set;
type_of_element: Project;};
metaslot in: WorksOn; end
salary: integer;
}
{ WorksOn;
in: association, metaclass;
instance_section: {
domain: employee;
range: project;
};
}
	$\forall e, s$ (
	$(e, s) \in (salary)^{DP} \rightarrow$
	$e \in (employee)^C \wedge$
	$s \in (integer)^{DT}) \wedge$
	$\forall e, s1, s2$ (
	$(e, s1) \in (salary)^{DP} \wedge$
	$(e, s2) \in (salary)^{DP} \rightarrow$
	$s1 = s2$)
	
	$\forall e, p$ (
	$(e, p) \in (worksOn)^{OP} \rightarrow$
	$e \in (employee)^C \wedge$
	$s \in (project)^C$)

{ SameSalary;
in: association, metaclass;
superclass: reflexive;
instance_section: {
domain: employee;
range: employee;
};
}
{ Employee; in: type;
sameSalary: {set;
type_of_element:
Employee;};
metaslot in: SameSalary end
}
	$\forall e$ (
	$e \in (employee)^C \rightarrow$
	$(e, e) \in (sameSalary)^{OP}$)

Условия для фактов.

Конструкция СИНТЕЗ	Условие
{ JohnJohnson; in: employee; salary: 3200; boss: JackJackson; }	$(JohnJohnson)^I \in (employee)^C \wedge$ $((JohnJohnson)^I, (3200)^{LT}) \in$ $(salary)^{DP} \wedge$ $((JohnJohnson)^I, (JackJackson)^I)$ $\in (boss)^{OP}$

Условия для подклассов.

Конструкция СИНТЕЗ	Условие
{ manager; in: class; superclass: employee; … }	$(manager)^C \subseteq (employee)^C$

Условия для утверждений уникальности.

СИНТЕЗ	OWL
{ Employee; in: type; empCode: integer; keys: { unique; { empCode } }; }	$\forall x, y, k$ ($(x, k) \in (empCode)^{DP} \wedge$ $(y, k) \in (empCode)^{DP} \rightarrow$ $x = y$)

Условия для инвариантов.

Конструкция СИНТЕЗ	Условие
manager <= union(areaManager, topManager)	$(manager)^C \subseteq (areaManager)^C \cup (topManager)^C$
all m/TopManager (is_in(topManager, m) ->) is_in(m.boss, m))	$\forall m$ ($(m)^I \in$ $(topManager)^C \rightarrow$ $((m)^I, (m)^I) \in (boss)^{OP}$)
cardinal({m/Manager | is_in(e.boss, m) & is_in(topManager, m)}) >= 1	$\#\{m \mid (e, m) \in (boss)^{OP} \wedge$ $m \in (topManager)^C \} \geq 1$
e.hireDate <> e.fireDate	$\forall h, f$ ($(e, h) \in (hireDate)^{DP} \wedge$ $(e, f) \in (fireDate)^{DP} \rightarrow$ $h \neq f$)

Для доказательства корректности отображения канонической модели и OWL 2 необходимо также убедиться, что основные задачи вывода (непротиворечивость онтологии, поглощение понятий и т. д.), сводимые друг к другу в дескриптивных логиках, сводимы друг к другу и в канонической модели. Действительно, доказательства сводимости для дескриптивных логик легко преобразуются в доказательства сводимости для канонической модели. В качестве примера рассмотрим доказательство сводимости задачи поглощения к задаче отсутствия модели (unsatisfiability).

Рассмотрим онтологию O и задачу поглощения для классов $C_1 \subseteq C_2$ из O. По определению поглощения, $C_1 \subseteq C_2$ iff для любой модели $I = \langle \Delta_I, \Delta_D, \bullet^C, \bullet^{OP}, \bullet^{DP}, \bullet^I, \bullet^{DT}, \bullet^{LT} \rangle$ онтологии O выполнено $(C_1)^C \subseteq (C_2)^C$. Рассмотрим задачу отрицания поглощения: пусть существует такая модель I и такой объект a, что $a \in (C_1)^C$ и $a \notin (C_2)^C$. Рассмотрим онтологию O_1, полученную из O добавлением класса A и утверждения, что объект с именем id_a входит в A:

{ A; in: class;
superclass: C_1;
instance_section: {
inv: { in: invariant;
{ A <= differ(thing, C_2) }
};
}

{ id_a; in: A; }

Тогда очевидно, что интерпретация I, расширенная утверждением $(id_a)^I = a$, является моделью O_1, т.е. из отрицания поглощения $C_1 \subseteq C_2$ следует существование модели для O_1. Следовательно, из отсутствия модели для O_1 следует поглощение $C_1 \subseteq C_2$. Таким образом, задача поглощения сведена к задаче отсутствия модели.

Аналогичным образом для канонической модели адаптируются и другие сводимости.

5 Корректность взаимного отображения канонической модели и языка OWL 2

Обозначим чрез Σ: $S_{SYN} \times S_{OWL}$ взаимное отображение (являющееся формально отношением) множества S_{SYN} схем, выраженных в канонической модели и S_{OWL} – множества онтологий, выраженных в языке OWL 2 (примером такого отображения является отображение σ, определенное в разделе 3). Онтологии $s \in S_{SYN}$ и $o \in S_{OWL}$ отображаются друг в друга iff $\langle s, o \rangle \in \Sigma$.

Назовем отображение Σ *корректным*, если для любых двух онтологий $s \in S_{SYN}$ и $o \in S_{OWL}$ таких, что $\langle s, o \rangle \in \Sigma$ следующие условия эквивалентны:

- существует модель M_S для онтологии s;
- существует модель M_O для онтологии o.

Заметим, что корректность отображения σ означает эквивалентность задач вывода в канонической модели и в дескриптивной логике SROIQ [11], отвечающей языку OWL 2 DL.

Действительно, основные задачи вывода в SROIQ сводимы к задаче существования модели в SROIQ [11], существование модели в SROIQ эквивалентно существованию модели в языке СИНТЕЗ, а задачи вывода в языке СИНТЕЗ сводимы к задаче существования модели в языке СИНТЕЗ.

В разделе 4 показано, как онтологии языка СИНТЕЗ интерпретируются при помощи семантических структур, используемых при интерпретации языка OWL 2. Таким образом, если рассмотреть для онтологий $s \in S_{SYN}$ и $o \in S_{OWL}$, связанных отображением σ, одну и ту же интерпретацию I, то для доказательства корректности отображения σ остается показать лишь, что I удовлетворяет s iff I удовлетворяет o. Для этого нужно показать, что условия удовлетворения интерпретации онтологии канонической модели, рассмотренные в разделе 4, эквива-

лентны условиям удовлетворения интерпретации онтологии OWL [7]. В общем случае доказательство осуществляется индукцией по построению онтологии (по классам, свойствам, фактам, инвариантам и т. д.). В данном разделе эквивалентность условий продемонстрирована на примерах из раздела 3, иллюстрирующих основные закономерности доказательства. В левой части таблиц приводятся условия для конструкций СИНТЕЗ, рассмотренных в разделе 3. В правой части таблиц приводятся условия для соответствующих конструкций OWL, также рассмотренных в разделе 3. В условиях для OWL сохранены обозначения, используемые в [7].

Условия для свойств объектов.

Условия для СИНТЕЗ	Условия для OWL
$\forall e, s$ ($(e, s) \in (salary)^{DP} \rightarrow e \in (employee)^{C} \wedge s \in (integer)^{DT}) \wedge$ $\forall e, s1, s2$ ($(e, s1) \in (salary)^{DP} \wedge (e, s2) \in (salary)^{DP} \rightarrow s1 = s2$)	$\forall x, y$ ($(x, y) \in (salary)^{DP} \rightarrow x \in (employee)^{C}$) $\forall x, y$ ($(x, y) \in (salary)^{DP} \rightarrow y \in (xsd:integer)^{DT}$) $(employee)^{C} \subseteq \{ x \mid \#\{ y \mid (x , y) \in (salary)^{DP} \} = 1 \}$
$\forall e, p$ ($(e, p) \in (worksOn)^{OP} \rightarrow e \in (employee)^{C} \wedge s \in (project)^{C}$)	$\forall x, y$ ($(x, y) \in (worksOn)^{OP} \rightarrow e \in (employee)^{C}$) $\forall x, y$ ($(x, y) \in (worksOn)^{OP} \rightarrow s \in (project)^{C}$)

Условия для отношений между классами.

Условия для СИНТЕЗ	Условия для OWL
$(manager)^{C} \subseteq (employee)^{C} \wedge$ $(manager)^{C} \subseteq (areaManager)^{C} \cup (topManager)^{C}$	$(manager)^{C} \subseteq (employee)^{C}$ $(manager)^{C} \subseteq (areaManager)^{C} \cup (topManager)^{C}$

Условия для фактов.

Условия для СИНТЕЗ	Условия для OWL
$(JohnJohnson)^{I} \in (employee)^{C} \wedge ((JohnJohnson)^{I}, (3200)^{LT}) \in (salary)^{DP} \wedge ((JohnJohnson)^{I}, (JackJackson)^{I}) \in (boss)^{OP}$	$(JohnJohnson)^{I} \in (employee)^{C}$ $((JohnJohnson)^{I}, (3200)^{LT}) \in (salary)^{DP}$ $((JohnJohnson)^{I}, (JackJackson)^{I}) \in (boss)^{OP}$

Условия для самоограничений.

Условия для СИНТЕЗ	Условия для OWL
$\forall m ((m)^{I} \in (topManager)^{C} \rightarrow ((m)^{I}, (m)^{I}) \in (boss)^{OP}$)	$(topManager)^{C} \subseteq \{ x \mid (x , x) \in (boss)^{OP} \}$

Условия для ограничений на размер области и класс значений свойств.

Условия для СИНТЕЗ	Условия для OWL
$\forall$ e ($e \in (employee)^{C} \rightarrow \#\{m \mid (e, m) \in (boss)^{OP} \wedge m \in (topManager)^{C} \} \geq 1$)	$(employee)^{C} \subseteq \{ x \mid \#\{ y \mid (x , y) \in (boss)^{OP} \wedge y \in (topManager)^{C} \} \geq 1 \}$

Условия для рефлексивных свойств.

Условия для СИНТЕЗ	Условия для OWL
$\forall e$ ($e \in (employee)^{C} \rightarrow (e, e) \in (sameSalary)^{OP}$)	$\forall x$ ($x \in \Delta_I \rightarrow (x, x) \in (sameSalary)^{OP}$)

Условия для непересекающихся свойств.

Условия для СИНТЕЗ	Условия для OWL
$\forall e$ ($e \in (employee)^{C} \rightarrow \forall h, f$ ($(e, h) \in (hireDate)^{DP} \wedge (e, f) \in (fireDate)^{DP} \rightarrow h \neq f$))	$(hireDate)^{DP} \cap (fireDate)^{DP} = \varnothing$

Условия для иерархии композиций свойств.

Условия для СИНТЕЗ	Условия для OWL
$\forall c, r, s$ ($c \in (city)^{C} \wedge r \in (region)^{C} \wedge s \in (region)^{C} \wedge (c, r) \in (locatedIn)^{OP} \wedge (r, s) \in (partOf)^{OP} \rightarrow (c, s) \in (locatedIn)^{OP}$)	$\forall y_0, y_1, y_2$ ($(y_0, y_1) \in (locatedIn)^{OP} \wedge (y_1, y_2) \in (partOf)^{OP} \rightarrow (y_0 , y_2) \in (locatedIn)^{OP}$)

Условия для утверждений уникальности.

Условия для СИНТЕЗ	Условия для OWL
$\forall x, y, k$ ($(x, k) \in (empCode)^{DP} \wedge (y, k) \in (empCode)^{DP} \rightarrow x = y$)	$\forall x, y, w$ ($x \in (employee)^{C} \wedge y \in (employee)^{C} \wedge (x, k) \in (empCode)^{DP} \wedge (y, k) \in (empCode)^{DP} \rightarrow x = y$)

Нетрудно видеть, что конъюнкции условий для СИНТЕЗ и условий для OWL эквивалентны. В простейших случаях условия просто совпадают (например, отношения между классами) или совпадают с точностью до переменных.

Таким образом, построенное отображение σ корректно и позволяет использовать для решения задач вывода в онтологиях канонической модели автоматические системы вывода для дескриптивных логик [5].

Трансформация, отвечающая отображению подмножества канонической информационной модели в язык OWL 2, реализована на языке трансформаций моделей ATL (ATLAS Transformation Language) [10]. Трансформация позволяет автоматически порождать онтологию на языке OWL 2 из онтологии, выраженной в канонической информационной модели.

6 Заключение

Важной задачей при создании предметных посредников для интеграции неоднородных информационных ресурсов является верификация онтологического моделирования и интеграции онтологий. Онтологии представляются в предметных посредниках в канонической информационной модели,

служащей общим языком, унифицирующим разнообразные модели ресурсов.

В данной работе представлено взаимное отображение канонической модели (языка СИНТЕЗ) и онтологической модели, служащей входным языком для автоматических систем вывода в дескриптивных логиках – языка OWL 2. Построенное отображение позволяет использовать системы вывода для решения задач вывода в онтологиях канонической модели. Тем самым достигается верификация концептуального моделирования и интеграции онтологий.

Литература

[1] Kalinichenko L.A., Briukhov D.O., Martynov D.O., Skvortsov N.A., Stupnikov S.A. Mediation framework for enterprise information system infrastructures // The 9th Int. Conf. on Enterprise Information Systems (ICEIS). – 2007. – P. 246-251.

[2] Kalinichenko L.A., Stupnikov S.A., Martynov D.O. SYNTHESIS: a language for canonical information modeling and mediator definition for problem solving in heterogeneous information resource environments. – M.: IPI RAS, 2007. – 171 p.

[3] Briukhov D.O., Kalinichenko L.A., Skvortsov N.A. Information sources registration at a subject mediator as compositional development// Advances in Databases and Information Systems: Proc. of the 5th East European Conf. LNCS 2151. – Berlin-Heidelberg: Springer-Verlag, 2001. – P. 70-83.

[4] Horrocks I., Sattler U., Tobies S. Practical reasoning for very expressive description logics // Logic J. of the IGPL. – 2000. – V. 8, No 3. – P. 239-264.

[5] Pellet: OWL 2 reasoner for Java. – http:// clarkparsia. com/pellet/.

[6] Patel-Schneider P.F. OWL 2 Web ontology language new features and rationale. – 2009. – http:// www.w3.org/TR/owl2-new-features/.

[7] Horrocks I., Parsia B., Sattler U. OWL 2 Web ontology language direct semantics. – 2009. – http://www.w3.org/TR/owl2-direct-semantics/.

[8] Калиниченко Л.А., Скворцов Н.А.. Реверсивное онтологическое моделирование при унифицированном представлении различных онтологических моделей источников информации в предметном посреднике // Системы и средства информатики: Спец. вып. Формальные методы и модели в композиционных инфраструктурах распределенных информационных систем / Под ред. И.А. Соколова. – М.: ИПИ РАН, 2005. – С. 184-212.

[9] Bock C. et al. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. – 2009. – http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/.

[10] ATL Project. – http://www.eclipse.org/m2m/atl/.

[11] Horrocks I., Kutz O., Sattler U. The even more irresistible SROIQ // Proc. of the Tenth Int. Conf. Principles of Knowledge Representation and Reasoning. – Menlo Park, California: AAAI Press, 2006. – P. 57-67.

[12] Брюхов Д.О., Вовченко А. Е., Захаров В.Н., Желенкова О.П., Калиниченко Л.А., Мартынов Д.О., Скворцов Н.А., Ступников С.А. Архитектура промежуточного слоя предметных посредников для решения задач над множеством интегрируемых неоднородных распределенных информационных ресурсов в гибридной грид-инфраструктуре виртуальных обсерваторий // Информатика и ее применения. – М., 2008. – Т. 2, Вып. 1. – С. 2-34.

[13] Kalinichenko L.A., Stupnikov S.A., Vovchenko A.E. Mediation based semantic grid // Distributed computing and grid technologies in science and education: Proc. of the Int. Conf. – Dubna: JINR, 2010.

[14] Wiederhold G. Mediators in the architecture of future information systems // IEEE Computer. – 1992. – V. 25, No 3. – P. 38-49.

A mutual mapping of the canonical information model and OWL

S.A. Stupnikov, N.A. Skvortsov

Main points of a mutual mapping of the canonical information model and distinguishing features of OWL 2 language are presented. The mapping allows to use description logic automatic reasoners for inference problems solving in ontologies of the canonical model. Thus a verification of subject mediators conceptual modeling and heterogeneous information resources integration in mediators are achieved.

♣ Работа выполнена при финансовой поддержке РФФИ (проекты 08-07-00157, 10-07-00342, 10-07-00640) и Программы фундаментальных исследований Президиума РАН № 3 «Фундаментальные проблемы системного программирования», раздел 4 «Системы управления базами данных», проект «Исследование методов и средств промежуточного слоя предметных посредников, обеспечивающего решение задач над множеством неоднородных распределенных информационных ресурсов»