Каркас логических диалектов RIF: анализ и демонстрация применения[1]

Калиниченко Л. А.[I], Ступников С. А.[I]

[I] Институт проблем информатики РАН
Россия, г. Москва, 119333, ул. Вавилова, 44-2
leonidk@synth.ipi.ac.ru, ssa@ipi.ac.ru

Аннотация. Каркас логических диалектов RIF предназначен для описания синтаксиса и семантики логических диалектов RIF при помощи таких родовых конструкций, как сигнатуры, пространства символов, семантические структуры и др. Целью статьи является анализ каркаса RIF и демонстрация его применения для определения конкретных диалектов. В статье рассмотрены основные черты синтаксического и семантического каркасов, специализацией которых должны являться конкретные диалекты.

Ключевые слова: формат обмена правилами, логические диалекты, синтаксис и семантика правил

1 Принципы создания каркаса

Инициатива RIF (Rule Interchange Format) консорциума W3C ставит своей целью разработку пакета стандартов обмена правилами между различными системами на правилах в различных прикладных областях, включая Семантический Веб [RIF]. Достижение интероперабельности в среде существующих разнородных систем на правилах, коммерческих и научных, значительно различающихся не только синтаксисом, но и семантикой языков, является сложной задачей. Разработчиками данного стандарта RIF представляется расширяемым набором *диалектов* – языков с точно определенными синтаксисом и семантикой. Необходимость использования множества диалектов (а не одного единственного языка) вызвана разнообразием парадигм, на которых основаны языки правил, а также разнообразием их семантики.

1 Работа выполнена при финансовой поддержке РФФИ (проекты 08-07-00157-а, 10-07-00342-а, 10-07-00640-а) и Программы фундаментальных исследований Президиума РАН № 15 *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*.

Для упрощения создания новых диалектов и сокращения времени их разработки рабочая группа RIF определила каркас расширения RIF, называемый Каркасом логических диалектов (RIF Framework for Logic Dialects, RIF-FLD) [1, 2]. RIF-FLD представляет собой формализм, предназначенный для описания логических диалектов RIF, в том числе Базового логического диалекта RIF (RIF-BLD) и Диалекта-ядра RIF (RIF-Core). В каркасе RIF синтаксис и семантика описываются при помощи механизмов, обычно используемых в различных логических языках. Однако, эти механизмы редко используются совместно. Необходимость совместного использования в каркасе вызвана тем, что каркас должен быть достаточно широким для того, чтобы объединить несколько различных типов логических языков. Каркас не только дает точное определение этим механизмам, но и позволяет варьировать различные их детали. RIF проектировался таким образом, чтобы его возможные логические диалекты базировались на RIF-FLD и определялись как специализации FLD. Из этого следует, что диалекты RIF сравнимы синтаксически и семантически, т.е. существует возможность формально идентифицировать расширения, ограничения и общие подмножества различных диалектов. При этом *системы на правилах* получают возможность обмениваться множествами правил при помощи этих диалектов с сохранением семантики правил.

Мотивацией к использованию каркаса RIF является значительная трудность описания диалектов «с нуля». Так, описание относительно простого диалекта RIF-BLD занимает около 30 страниц плотного текста. С другой стороны, описание диалекта как специализации каркаса требует существенно меньших усилий: описание RIF-BLD как специализации занимает около пяти страниц.

Каркас RIF является достаточно общим и покрывает большинство логических языков, применяемых в областях баз данных, логического программирования и Семантического Веба. Однако, разработчики RIF-FLD подчеркивают, что диалекты, которые будут разрабатываться в будущем, могут стимулировать дальнейшее развитие каркаса.

Целью данной статьи является анализ RIF-FLD и демонстрация его применения для определения конкретных диалектов. В статье подробно рассмотрены основные компоненты RIF-FLD: синтаксический и семантический каркас. *Синтаксический каркас* (раздел 2) определяет родовой механизм описания допустимых в диалекте термов и формул, а также способ конкретизации этого механизма для порождения новых диалектов. *Семантический каркас* (раздел 3) определяет теоретико-модельные механизмы описания логического вывода в диалектах. XML каркас определяет основные принципы отображения абстрактного синтаксиса RIF-FLD в конкретный XML-синтаксис.

В качестве примера диалекта, определяемого как специализация каркаса, в данной статье рассматривается RIF-CASPD - диалект программирования множества ответов (раздел 4).

Наконец, в разделе 5 рассматривается вопрос конформности диалектов процессорам RIF – конкретным системам на правилах.

2 Синтаксический каркас

Главными чертами синтаксического каркаса являются его общность и расширяемость (для того, чтобы синтаксис различных логических диалектов мог быть определен как специализация каркаса), а также интерпретируемость в теоретико-модельных терминах: RIF-FLD разрабатывался как каркас для диалектов с теоретико-модельной семантикой.

Синтаксический каркас определяет виды термов и формул, допустимых в диалекте.

2.1 Термы

В каркасе определяются следующие виды термов.

Константы. Константа имеет вид *"literal"^^symspace,* где *literal* – последовательность Unicode-символов, *symspace* – идентификатор *пространства символов,* которому принадлежит константа. Например, *"1.2"^^xs:decimal* – синтаксически правильная константа, т.к. *1.2* является элементом пространства символов типа данных *xs:decimal* языка XML Schema (в дальнейшем, в некоторых случаях, для упрощения кавычки и идентификаторы пространств символов в константах будут опускаться). Аналогично, *"a+2"^^xs:decimal* не является синтаксически правильной константой (*a+2* не является элементом *xs:decimal*). Список пространств символов, которые должен включать каждый диалект, приводится в [3].

Строго определенная синтаксическая форма констант позволяет избежать неоднозначности при синтаксическом разборе выражений с константами. Классификация констант по пространствам символов помогает структурировать установление соответствий между константами, использующимися в различных языках на правилах, при построении сохраняющих семантику отображений языков.

Переменные. Переменные обозначаются идентификаторами, состоящими из букв и цифр и начинающимися со знака «?», например, *?X*. Выделение идентификаторов переменных специальным символом (в

данном случае «?») традиционно для языков на правилах для устранения неоднозначности при синтаксическом разборе выражений с переменными.

Позиционные термы. Если t, t_1, ... t_n – термы, тогда $t(t_1 \ ... \ t_n)$ – позиционный терм. Заметим, что в абстрактном синтаксисе запятые не используются для отделения аргументов предикатов и термов (аргументы отделяются пробелами). Понятие позиционного терма соответствует одному из правил образования терма в языке HiLog [4] и обобщает понятие терма из логики первого порядка. Действительно, в логике первого порядка $t(t_1 \ ... \ t_n)$ является термом только в том случае, когда t – функциональный символ валентности n (функциональные символы являются специальной частью алфавита логики). В позиционном терме t может быть произвольным термом и никак не ограничивается своей валентностью. И вообще, валентность изначально не приписывается термам, в том числе константам, функциональным и предикатным символам.

Переменные могут быть использованы в любом месте, в котором может быть использован терм. Например, позиционным термом является даже *?X("abc"^^xs:string ?W)(?Y ?Z(?V "33"^^xs:integer))*. Действительно, это позиционный терм вида $t(t_1 \ ... \ t_n)$, в котором t = *?X("abc"^^xs:string ?W)* – терм второго порядка, возвращающий функцию или предикат, которые применяются, в свою очередь, к набору аргументов *(?Y ?Z(?V "33"^^xs:integer))*.

Термы с именованными аргументами. Терм имеет вид $t(s_1 \rightarrow v_1 \ ... \ s_n \rightarrow v_n)$, где t, v_1, ..., v_n – термы, s_1, ..., s_n – идентификаторы аргументов. Так, терм *person(name→Bob age→33)* отличен от термов *person(Bob→33)* и *person(spouse→Bob age→33)*, но эквивалентен терму *person(age→33 name→Bob),* поскольку порядок аргументов неважен.

Позиционные термы и термы с именованными аргументами используются для представления двух важнейших синтаксических конструкций:

- атомарных предикатов;
- вызовов функций.

Атомарные предикаты являются утверждениями о принадлежности кортежа (состоящего из аргументов предиката) некоторому отношению (например, реляционной таблице). Так, предикаты *person(Bob 33)* и *person(name→Bob age→33)* являются утверждениями о принадлежности кортежа *⟨Bob, 33⟩* отношению *person*.

Наряду с обычными вызовами функций, такими как *rotate(figure→?Square degree→45*) и *rotate(?Square 45*) могут использоваться вызовы функций высших порядков, например *hobby(Tom)(chess)*. Функция *hobby* возвращает предикат, который утверждает, что некоторое занятие (например, шахматы) является (или не

является) увлечением индивидуума. Таким образом, терм *hobby(Tom)* возвращает предикат, который принимает на вход занятие, и возвращает истину в том случае, если это занятие является увлечением индивида *Tom*. Терм *hobby(Tom)(chess)* принимает истинное значение, если *Tom* увлекается шахматами.

Термы-списки. Замкнутый список имеет вид $List(t_1 \ldots t_n)$, где $n \geq 0$ и $t_1, \ldots t_n$ – термы. *List()* обозначает пустой список. *Открытый список* (список с хвостом) имеет вид $List(t_1 \ldots t_n \mid t)$, где t – произвольный терм (хвост), который может быть и списком. Например, *List(1, 2, 3)* – замкнутый список, *List(1, 2 | ?X)* – открытый список. Списки как структуры данных характерны для языков на правилах и вообще для декларативного программирования.

Терм-равенство. Терм имеет вид $t=s$, где t и s – термы. Равенство является одним из важнейших логических предикатов.

Фреймовые термы (фреймы). Данный вид термов заимствован из F-логики [5]. Фрейм имеет вид $t[p_1 \rightarrow v_1 \ldots p_n \rightarrow v_n]$, где $t, p_1, \ldots, p_n, v_1, \ldots, v_n$ – термы. Порядок аргументов, как и в случае термов с именованными аргументами, неважен. Однако, семантика фреймов значительно отличается от семантики термов с именованными аргументами. Так, во фрейме *bob[name→Bob age→33]* идентификатор *bob* обозначает объект, а *name→Bob* и *age→33* являются утверждениями о свойствах объекта. Напротив, терм *person(name→Bob age→33)* не является утверждением о свойствах объекта *person*. Здесь *person* является именем отношения и *name→Bob age→33* описывает конкретный кортеж этого отношения. Таким образом, *bob[spouse→Mary]* является утверждением о свойствах того же объекта *bob*, а утверждение *person(spouse→Mary)* не имеет никакого отношения к предыдущему утверждению *person(name→Bob age→33)*.

Комбинируя фреймы и позиционные термы или термы с именованными аргументами, можно описывать вызовы *методов* объектов, например: *Bob[raiseSalary(2000)→?newSalary]* или *Bob[raiseSalary(allowance→2000)→?newSalary]*. При этом у типа, которым типизирован объект *Bob*, предполагается наличие метода *raiseSalary,* получающего на вход прибавку к зарплате (*allowance*) и возвращающего величину новой зарплаты.

Классификационный терм. Различают два вида классификационных термов: терм членства объекта в классе $t\#s$, где t и s – термы (например, *John#Student*) и терм *отношения класс-подкласс* $t\#\#s$ (например, *Student##Person*). Два этих отношения характерны для языков

программирования и спецификации вообще, но, по-видимому, пришли в каркас RIF из F-логики вместе с фреймами. Первое отношение позволяет описывать факты членства конкретных объектов в классах, второе – иерархии классов.

Внешне определенные термы. Если *t* – константа, позиционный терм, терм с именованными аргументами, равенство, классификация или фрейм, то *External(t loc)* есть внешне определенный терм. Локатор *loc* должен единственным образом идентифицировать ресурс терма *t.* Например,

External("http://example.com/acme"^^rif:iri[
"http://example.com/mycompany/president"^^rif:iri(?Year)→?Pres]
<http://example.com/acme>)

может быть представлением внешнего метода *president* внешнего объекта *acme*, идентифицируемого адресом *http://example.com/acme.* Внешние термы обобщают идеи присоединения процедур (procedural attachment) и запросов к внешним ресурсам.

Агрегатные термы. Терм имеет вид $sym\{?V\ [?X_1 \ldots ?X_n] \mid t\}$, где *sym* – идентификатор агрегирующей функции, *t* – терм (который может содержать агрегатные термы). Переменные $?V, ?X_1, \ldots ?X_n$ должны входить в *t* свободно, все остальные переменные в *t* должны быть связаны. Выделяющая переменная *?V* считается связанной агрегатным термом, переменные $?X_1, \ldots ?X_n$ остаются свободными. Пример агрегатного терма выглядит следующим образом: *avg{?Sal [?Dept] | Exists ?Empl salary(?Empl ?Dept ?Sal)}*. Если подставить в этот терм конкретное значение отдела вместо переменной *?Dept,* то значением терма будет средняя зарплата по отделу.

Удаленный терм. Терм имеет вид *p@r*, где *p* – терм, *r* – константа, переменная, позиционный терм или терм с именованными аргументами. Удаленные термы предназначены для описания запросов к удаленным RIF-документам (см. раздел 2.2, *документные формулы*), называемым *удаленными модулями*. При этом *p* является запросом[1], *r* – ссылкой, идентифицирующей удаленный модуль. Следует отличать удаленные термы от внешних термов: семантика удаленных термов определяется RIF-FLD, поскольку они описывают запросы к RIF-документам; а внешние термы используются для запросов к внешним ресурсам, семантика которых неизвестна. Пример удаленного терма выглядит следующим образом: *?O[?N→"John" salary→?S]@http://acme.foo*. Данному запросу удовлетворяют объекты удаленного модуля, идентифицируемого ссылкой *http://acme.foo,* в структуре которых

[1] Термин «запрос» в данном случае является неформальным. Синтаксически, *p* является произвольным термом над удаленным RIF-документом.

присутствует атрибут *salary* и некоторый другой атрибут со значением «*John*».

Формульный терм. Пусть S – логическая связка или квантор, $t_1, \ldots t_n$ – термы, тогда $S(t_1, \ldots t_n)$ – формульный терм. Формульные термы соответствуют составным формулам, конструируемых из атомарных формул и других составных формул при помощи связок и кванторов. Некоторые связки для простоты могут записываться в инфиксной или префиксной форме (*a :- b, Naf a*). То, что формула является термом, является одним из важнейших обобщающих свойств каркаса RIF, отличающего его, например, от логики первого порядка, в которой термы и формулы четко разделены. В настоящее время пока не определено диалекта, допускающего смешивание термов и формул. Во всех известных диалектах - RIF-BLD, RIF-CASPD, RIF-CLPWD – термы и формулы разделяются так же, как в логике первого порядка.

Такое большое разнообразие термов вызвано необходимостью поддержки *реверсивности* (round-tripping) в RIF. При отображении набора правил системы S_1 в RIF, затем в систему S_2, затем снова в RIF и, наконец, обратно в S_1, мы должны получить не просто семантически эквивалентный набор правил, но практически точно такой же набор с точки зрения моделирования. Так, например, фрейм при реверсивном отображении не должен превратиться в отношение (позиционный терм или терм с именованными аргументами). Многообразие термов также облегчает отображение из/в RIF и делает его более естественным.

Фреймы, классификации, равенства (и другие сущности, обычно рассматриваемые как формулы) трактуются в RIF как термы для обеспечения некоторой степени *материализации (reification),* т.е. представления формул (которые являются утверждениями о фактах или убеждениях) как термов (объектов). Таким образом, допускаются диалекты, в которых могут определяться утверждения об утверждениях, и эти мета-утверждения могут быть обработаны при помощи правил.

2.2 Формулы

В каркасе определяется несколько видов формул, большинство которых являются адаптациями формул в известных логиках. RIF-FLD объединяет эти формулы в одну логическую систему.

Атомарные формулы. Терм является формулой. Как было указано в предыдущем разделе, формула (хотя и не всякая) в каркасе RIF также является термом (в отличие от логики первого порядка, в которой терм не является формулой и формула не является термом). Это размывает

различие между объектами и утверждениями об объектах и является основой для мета-рассуждений в диалектах RIF.

Конъюнкция и дизъюнкция. Обычные логические связки из логики первого порядка. В RIF они выглядят следующим образом: *And(f_1 ... f_n), Or(f_1 ... f_n)*.

Отрицание. RIF поддерживает как симметричное отрицание, обозначаемое *Neg* (удовлетворяющее свойству *Neg Neg f* ↔ *f*), так и *отрицание по умолчанию* (default negation), используемое в логическом программировании, обозначаемое *Naf*.

При помощи симметричного отрицания в диалектах могут представляться, например, такие виды отрицания, как классическое отрицание логики первого порядка, сильное (strong) отрицание [14] и явное (explicit) отрицание [14]. Классическое отрицание удовлетворяет закону исключенного третьего *Or(f Neg f),* а сильное и явное отрицания – нет. Сильное отрицание также характеризуется тем, что *f* и *Neg f* не могут одновременно обращаться в истину. Явное отрицание удовлетворяет лишь свойству симметричности и не запрещает одновременной истинности *f* и *Neg f*.

Отрицание *Naf* (образованное как сокращение от термина *negation as failure*) предназначено для выражения различных видов несимметричного отрицания – собственно отрицания по умолчанию Кларка [6], а также отрицания, основанного на хорошо обоснованной (well-founded) семантике [8] или семантике стабильных моделей (stable model) [7].

Импликация. Импликация имеет вид *f :- p,* происходит из логического программирования и не эквивалентна формуле *Or(f Neg p),* которой эквивалентна обычная импликация логики первого порядка $p \Rightarrow f$. Неформально, *f :- p* означает «*f* не менее истинна, чем *p*». Формальная семантика импликации определена в разделе 3.2, она совпадает с семантикой импликации логики первого порядка в том случае, если логика диалекта двузначна и симметричное отрицание является классическим.

Кванторы. Формулы со встроенными кванторами всеобщности и существования имеют вид соответственно *Forall* $?X_1$ ... $?X_n$ *(f)* и *Exists* $?X_1$ ... $?X_n$ *(f),* где *f* – формульный терм.

Групповые формулы. Формула имеет вид *Group(f_1 ... f_n),* где f_i – формула. Группы формул могут быть вложенными, и предназначены для удобства описания: например, группе формул можно присвоить идентификатор и метаданные, а затем ссылаться на группу в других документах.

Документные формулы (документы, RIF-документы). Формула имеет вид *Document(D_1 ... D_n G),* где *G* – опциональная групповая формула (называемая *ассоциированной* с документом), D_i – директивы. Документы и группы являются механизмами, обеспечивающими *идентификацию наборов правил* (одно из требований к RIF, мотивированных примерами использования – см. [13], раздел 4.2.2). RIF-документ формализует понятие «множества правил» и является структурной единицей RIF.

Директивы могут быть:

- *директивой диалекта Dialect(D)*, где *D* – имя определяемого диалекта (данная директива обеспечивает требование *идентификации диалекта* [3]);
- *базовой директивой Base(<uri>),* где *uri* – идентификатор ресурса. Все лишенные префикса символы, используемые в правилах, по умолчанию идентифицируются как принадлежащие ресурсу, указанному в базовой директиве. То есть базовая директива описывает синтаксическое сокращение, позволяющее превращать относительные идентификаторы ресурсов в абсолютные. Так в нижеследующем примере запись идентификатора индивида *<Yorick>* является сокращением полной записи *"http://www.shakespeare-literature.com/Hamlet/Yorick"^^rif:iri.* Расширение в абсолютный идентификатор ресурса производится с использованием пути *http://www.shakespeare-literature.com/Hamlet/,* указанного в базовой директиве;
- *префиксной директивой Prefix(p <v>)*, где *p* – имя префикса, *v* – идентификатор префикса. Префиксная директива является еще одной формой синтаксического сокращения идентификаторов ресурсов. Так, например, запись *ex:man* разворачивается в полную константу *"http://example.org/ontology#man"^^rif:iri* с использованием префиксной директивы *Prefix(ex <http://example.org/ontology#>)*;
- *директивой импорта Import(loc),* где *loc* – локатор импортируемого документа. Данная директива обеспечивает модульность структуры RIF-документов. Все правила, входящие в импортированные документы, можно считать частью импортирующего документа. Семантика импортируемого и импортирующего документа должна быть одинаковой (см. раздел 3.2), т.е. документы должны принадлежать одному диалекту. Для импорта документов других диалектов в каркасе существует директива вида *Import(loc p),* где *p* – *профиль импорта,* который должен описывать, какая сущность и с какой семантикой импортируется. Однако, в каркасе RIF определяется лишь семантика директивы импорта вида *Import(loc).* Семантика импорта *Import(loc p)* должна определяться в диалектах;
- *директивой удаленного модуля Module(n loc)*, где *loc* – локатор удаленного модуля, *n* – то имя, через которое на модуль ссылаются в документе. В отличие от импортируемых документов, семантика

удаленных модулей практически не ограничивается, она должна совпадать с семантикой документа, подключающего удаленный модуль, только на константах (см. раздел 3.2). Доступ к структурам удаленных модулей осуществляется при помощи удаленных термов (см. раздел 2.1).

В документе могут быть только одна директива диалекта и одна базовая директива. Примером использования формул является следующий RIF-документ:

```
Document(
   Base(<http://www.shakespeare-literature.com/Hamlet/>)
   Prefix(dc <http://http://purl.org/dc/terms/>)
   Prefix(ex <http://example.org/ontology#>)
   Group(
      Exists ?X (And(?X#ex:RottenThing
                  ex:partof(?X <http://www.denmark.dk>)))
      Forall ?X (Or(<tobe>(?X)  Naf <tobe>(?X)))
      Forall ?X (
       And(
        Exists ?B (And(ex:has(?X ?B) ?B#ex:business))
        Exists ?D (And(ex:has(?X ?D) ?D#ex:desire))
       ) :- ?X # ex:man
      )
      Group(<Yorick>#ex:poor <Hamlet>#ex:prince)
   )
)
```

Первая формула группы говорит о том, что *прогнило что-то в Датском государстве,* вторая – что для всего сущего верно *быть или не быть*, третья – что *у всех свои желанья и дела*. Вложенная группа содержит утверждения о том, что *Гамлет – принц*, а *Йорик – бедный*.

Заметим, что группы и документы существенно отличаются от остальных формул. Во-первых, они не являются термами (остальные формулы являются формульными термами). Во-вторых, группы и документы не могут быть подформулами других формульных термов, и документы не могут быть подформулами групп. Это связано с тем, что для формульных термов существуют аналоги среди формул известных логик, а документы и группы – это специальные механизмы,

обеспечивающие структурирование правил. Именно поэтому документные и групповые формулы называются еще *документами* и *группами* соответственно. Формулами документы и группы называются для единообразия, а также потому, что как и формульные термы, документы и группы интерпретируются при помощи семантических структур, т.е. документы и группы являются синтаксическими конструкциями, интерпретируемыми при помощи теоретико-модельной семантики.

2.3 Общая характеристика синтаксических средств каркаса

Каркас поддерживает широкий спектр синтаксических средств описания логических правил.

Поддерживаются термы и предикаты, характерные для логики первого порядка, предикаты отношений (для представления, например, реляционных таблиц).

Обеспечивается поддержка средств, характерных для F-логики: фреймы (утверждения об объектах), иерархия класс-подкласс, вызовы методов объектов.

Поддерживаются средства, характерные для HiLog: функции высоких порядков, смешивание символов индивидов, функций и предикатов, возможность использования переменных везде, где разрешается использование термов.

Поддерживается широкий спектр клауз логического программирования – от Хорновских до дизъюнктивных. Поддерживаются логические операции и кванторы, в частности, различные виды отрицания. Правила также могут принимать вид ограничений и фактов.

Поддерживается широкий набор встроенных типов данных (типы данных XML Schema), встроенных функций и предикатов над ними. Поддерживаются разнообразные функции агрегации.

Обеспечивается поддержка запросов к удаленным RIF-документам, присоединения процедур и запросов к внешним ресурсам.

Таким образом, каркас покрывает синтаксические возможности известных логических языков правил и даже превосходит их.

2.4 Сигнатуры

Одним из важнейших компонентов RIF-FLD как каркаса для определения языков (диалектов) является понятие *сигнатуры*. Сигнатуры определяют, какие термы и формулы являются *хорошо сформированными* (well-formed). Сигнатура является обобщением понятия сорта в логике

первого порядка [9][1]. С каждым символом языка (символом является любое выражение языка – переменная, константа, терм, формула, а также логические символы языка – кванторы, связки и т.д.) ассоциирована некоторая сигнатура, определяющая синтаксические контексты, в которых символ может появляться. Фактически, определения сигнатур можно считать аналогом грамматики языка. Сигнатуры являются одним из средств, с помощью которых диалект задается полностью (см. раздел 3.5).

Например, сигнатура, ассоциированная с *p* (не будем конкретизировать ее описание), допускает появление *p* в терме вида *f(p)*, но не допускает появление *p* в терме вида *p(a b)*. Сигнатура для *f*, с другой стороны, допускает *f(p)* и *f(p q)*, но не допускает *f(p q r)* и *f(f)*. Заметим, что *f(f)* является термом, но не является хорошо сформированным термом. Таким образом можно определять, какие символы являются предикатными, а какие – функциональными, где допустимо использование переменных, а где – нет.

Сигнатура представляет собой утверждение вида $n\{e_1, ..., e_n, ...\}$, где *n* – имя сигнатуры, $e_1, ..., e_n, ...$ – счетный набор *стрелочных выражений*. Каждое стрелочное выражение представляет собой контекст, в котором может употребляться терм (с которым ассоциирована сигнатура). Набор таких контекстов (стрелочных выражений) и определяет сигнатуру.

Позиционное стрелочное выражение представляет собой утверждение вида $(k_1 \ldots k_n) \Rightarrow k$, где k, k_i – имена сигнатур. Например, если *term* – имя сигнатуры, то $() \Rightarrow term$, $(term) \Rightarrow term$ – позиционные стрелочные выражения.

Стрелочное выражение с именованными аргументами представляет собой утверждение вида $(p_1 \rightarrow k_1 \ldots p_n \rightarrow k_n) \Rightarrow k$, где p_i – имена аргументов. Например, стрелочным выражением с именованными аргументами является $(arg1 \rightarrow term \ \ arg2 \rightarrow term) \Rightarrow term$.

Например, в диалекте RIF-BLD вводится сигнатура индивидов *individual{},* вообще не содержащая стрелочных выражений. Определяется, что этой сигнатуре принадлежат переменные и константы, не являющиеся предикатными и функциональными символами (в том числе константы типов XML Schema).

С использованием этой сигнатуры определяется сигнатура предикатных символов *p*, содержащая следующие стрелочные выражения: $() \Rightarrow atomic$, $(individual) \Rightarrow atomic$, $(individual\ individual) \Rightarrow atomic$ и т.д., а также стрелочные выражения для предикатов с именованными аргументами $(s_1 \rightarrow individual \ \ldots \ s_k \rightarrow individual) \Rightarrow atomic$. Данные

[1] В многосортной логике переменные, константы разделяются на сорта; сорт приписывается также функциональным символам. Тем самым каждому терму также приписывается сорт, и в качестве аргументов функциональных термов могут использоваться термы только определенных сортов. Аналог многосортности в языках программирования – типизация переменных.

стрелочные выражения можно применить для проверки, являются ли выражения с предикатными символами хорошо сформированными термами. Например, предположим, что символ *is_taught_by* является предикатным символом, т.е. принадлежит сигнатуре *p,* а *Turing* и *Einstein* – константы, принадлежащие сигнатуре *individual.* Тогда терм *is_taught_by(Turing Einstein)* является хорошо сформированным, т.к. удовлетворяет стрелочному выражению *(individual individual)* $\Rightarrow$ *atomic* сигнатуры *p.* Терм *is_taught_by(apprentice*$\rightarrow$*Turing teacher*$\rightarrow$*Einstein)* также является хорошо сформированным, т.к. удовлетворяет стрелочному выражению *(apprentice*$\rightarrow$*individual teacher*$\rightarrow$*individual)* $\Rightarrow$ *atomic.*

Имена сигнатур должны образовывать счетное частично упорядоченное множество, включающее следующие зарезервированные имена сигнатур: *atomic* – для – атомарных формул, *formula* – для формул, ∞*-connective* – для n-арных связок *And, Or* и т.д., *2-connective* – для бинарных связок, например *:-, 1-connective* – для унарных связок и кванторов, = – для равенств, # – для включения, ## – для классификации, $\rightarrow$ – для фреймов, *aggregate* – для функций агрегации, *remote* – для удаленных термов, *list* – для замкнутых списков, *openlist* – для открытых списков. Имена сигнатур должны быть уникальны.

Знак < обозначает порядок на сигнатурах; $a < b$ означает, что термы сигнатуры *a* могут использоваться везде, где используются термы сигнатуры *b*; $a \leq b$ означает, что $a = b$ либо $a < b$. Так, например, *atomic* < *formula,* т.е. атомарные формулы (сигнатура *atomic*) могут использоваться везде, где используются просто формулы (сигнатура *formula*).

Множество сигнатур диалекта должно быть *согласованным* (coherent) – сигнатуры не должны конфликтовать друг с другом. Формально, множество сигнатур согласовано, если выполняются следующие свойства:

- если $a < b$, то множество стрелочных выражений сигнатуры *b* является подмножеством стрелочных выражений сигнатуры *a*;
- *atomic* < *formula;*
- каждое выражение e_i в сигнатуре ∞*-connective*$(e_1 \ldots e_n \ldots)$ имеет вид *(formula* $\ldots_{n\ раз}$ *... formula)* $\Rightarrow$ *formula*, и сигнатура ∞*-connective* назначена связкам *And, Or*;
- каждое выражение e_i в сигнатуре $=(e_1 \ldots e_n \ldots)$ имеет вид *(k k)* $\Rightarrow$ *v* (аргументы должны быть сравнимы), и по крайней мере одно выражение должно иметь вид *(k k)* $\Rightarrow$ *atomic* (равенства являются атомарными формулами).

Аналогичные ограничения налагаются на другие встроенные сигнатуры [1].

Сигнатуры используются для определения контекста, в котором могут появляться символы, при помощи понятия *хорошей сформированности* (well-formedness). В разделах 2.1 и 2.2 рассматривались термы и формулы, однако не указывалось, какие из них являются хорошо сформированными. Для точного определения этого понятия необходимо предполагать, что каждому символу алфавита языка соответствует некоторая уникальная сигнатура.

Терм является *хорошо сформированным*, если удовлетворяет следующим условиям:

- Константа или переменная сигнатуры s есть хорошо сформированный терм сигнатуры s.
- Терм $t(t_1 \ ... \ t_n)$ – хорошо сформированный терм сигнатуры s iff t – хорошо сформированный терм сигнатуры, содержащей стрелочное выражение вида $(s_1 \ ... \ s_n) \Rightarrow s$, и каждый t_i есть хорошо сформированный терм сигнатуры $g_i \leq s_i$.

Аналогично определяются хорошо сформированные термы всех остальных видов [1].

Хорошо сформированная атомарная формула – это хорошо сформированный терм сигнатуры $s \leq atomic$. Заметим при этом, что термы равенства, включения, классификации, фреймы являются атомарными (*atomic* – одна из их сигнатур). *Хорошо сформированная формула* – это хорошо сформированный терм сигнатуры $s \leq formula$, групповая формула или документная формула.

Так, например, если p имеет сигнатуру $s\{(obj) \Rightarrow obj, (obj\ obj) \Rightarrow obj, (obj\ obj\ obj) \Rightarrow obj\}$ и *a, b, c* имеют сигнатуру *obj{ }*, тогда *p(p(a) p(a b c))* - хорошо сформированный терм сигнатуры *obj*. С другой стороны, *p(a b c a)* – это терм, не являющийся хорошо сформированным, т.к. сигнатура *p* не позволяет *p* иметь четыре аргумента.

Рассмотрим еще один пример. Пусть *John* и *Mary* – символы сигнатуры *obj{ }*, переменная *?P* имеет сигнатуру $h_2\{(obj\ obj) \Rightarrow obj\}$, и *closure* имеет сигнатуру $h_3\{(h_2) \Rightarrow p_2\}$, где p_2 – сигнатура $p_2\{(obj\ obj) \Rightarrow obj\}$. Тогда *?P(John Mary)* и *closure(?P)(John Mary)* - хорошо сформированные термы сигнатуры *obj*.

2.5 Синтаксис диалекта RIF как специализация RIF-FLD

Синтаксис диалекта (множество хорошо сформированных формул) может быть получен из синтаксического каркаса RIF специализацией следующих параметров:

- алфавита;

- сигнатур;
- видов термов, поддерживаемых диалектом;
- пространств символов, поддерживаемых диалектом;
- формул, поддерживаемых диалектом.

Алфавит. Алфавит RIF-FLD состоит из следующих непересекающихся подмножеств символов:

- счетное множество *Const* символов констант;
- счетное множество *Var* символов переменных;
- конечное множество символов логических связок, включающее *And, Or, Naf, Neg, :-* и *NEWCONNECTIVE,* являющейся *точкой расширения*. Точка расширения не является синтаксической конструкцией сама по себе, но может быть заменена некоторой конкретной синтаксической конструкцией. В диалектах *NEWCONNECTIVE* должна быть заменена на конечное множество новых связок (точка расширения в диалекте не может быть сохранена). Некоторые предопределенные связки могут быть отброшены. Семантика предопределенных связок не может быть изменена;
- счетное множество кванторов, состоящее из кванторов существования и всеобщности и точки расширения *NEWQUANTIFIER*, которая должна быть актуализирована по тем же правилам, что и *NEWCONNECTIVE;*
- счетное множество символов агрегатных функций, включающее *Min, Max, Count, Avg, Sum, Prod, Set, Bag* и точку расширения *NEWAGGRFUNC*;
- символов *=, #, ##, ->, External, Dialect, Base, Prefix, Import, and Module, List, OpenList, Group, Document*; вспомогательных символов *(,), [,], {, }, <, >, |, ?, @,* ^^ и
- точки рас*ширения NEWSYMBOL.*

Пространства символов. Множество символов констант диалекта RIF делится на подмножества – пространства символов, используемые для представления типов данных XML Schema; типов данных, определенных другими спецификациями W3C (например, *rdf:XMLLiteral*); других множеств констант. Синтаксис (а иногда и семантика) символов констант определяются соответствующим пространством символов. Каждая константа принадлежит ровно одному пространству символов. У каждого пространства символов есть уникальный идентификатор и лексическое пространство (непустое множество строк), определяющее «форму» символов.

Термы. Диалект может поддерживать все или некоторые виды термов, описанные в разделе 2.2. Некоторые диалекты могут не поддерживать термы с именованными аргументами или фреймы. Диалект может поддерживать дополнительные виды термов (например, типизирующие термы F-логики [5]). Это достигается актуализацией точки расширения *NEWTERM*.

Схемы внешне определенных термов. Каждый внешний терм диалекта должен быть конкретизацией некоторой *схемы внешне определенных термов.* Схемы определяют, какие именно внешние термы допустимы в диалекте. Схема имеет вид *(?X_1 ... ?X_n; t; loc)*, где *loc* – локатор внешнего ресурса, *t* – константа, позиционный терм, терм с именованными аргументами, равенство, классификация, фрейм; *?X_i* – различные переменные, имеющие вхождения в *t*. Внешний терм *External(u loc)* является конкретизацией схемы *(?X_1 ... ?X_n; t; loc),* если *u* получается из терма *t* заменой переменных *?X_i* на термы s_i.

Например, термы

External("http://foo.bar.com"^^rif:iri["foo"^^xs:string→
"123"^^xs:integer] <http://example.com/acme>)

и

External(pred:isTime("22:33:44"^^xs:time) <pred:isTime>)

являются конкретизациями схем

(?X ?Y; ?X["foo"^^xs:string->?Y]; <http://example.com/acme>)

и

(?V; pred:isTime(?V); <pred:isTime>)

соответственно.

Множество схем внешне определенных термов диалекта должно быть *согласованным* (coherent) – никакой терм *t* не может являться конкретизацией двух различных схем диалекта.

Сигнатуры. Сигнатуры определяют, какие термы диалекта являются хорошо сформированными а какие – нет. Точный способ назначения сигнатур символам (переменным, константам, связкам, кванторам) зависит от конкретного диалекта. Константе могут быть назначены несколько сигнатур, переменной – только одна (поскольку множество хорошо определенных термов незамкнуто относительно подстановок: например, терм *f(?X ?X)* может быть хорошо определенным, а *f(a a)* - нет).

Разработчики конкретного диалекта RIF могут выбирать, какие сигнатуры следует назначить каким символам, и тем самым полностью определить синтаксис диалекта.

В диалекте могут определяться новые сигнатуры, помимо встроенных. Например, в RIF-BLD определяется сигнатура *individual*. В диалекте также определяется частичный порядок на сигнатурах.

Диалекты могут специализировать сигнатуры =, #, ##, ->, например, сигнатура знака равенства может быть специализирована двумя следующими стрелочными выражениями: *=\{(individual individual) ⇒ atomic, (individual individual) ⇒ individual\}*.

Формулы. Диалект может поддерживать все виды формул, рассмотренные в разделах 2.1 и 2.2, а может налагать на формулы различные ограничения. Например, ограничения могут налагаться на головы или тела правил (в RIF-BLD допускаются только Хорновские правила), некоторые варианты использования кванторов могут быть запрещены (RIF-BLD запрещает использование квантора всеобщности в голове правила), может быть запрещено отрицание (как в RIF-BLD), и т.д. В диалекте могут определяться новые формулы при помощи актуализации точек расширения *NEWCONNECTIVE* и *NEWQUANTIFIER*. Например, могут быть введены классическая импликация или эквивалентность (двухсторонняя импликация).

3 Семантический каркас

Каркас определяет понятие семантических структур и моделей формул RIF-FLD. Семантика диалекта выводится из этих понятий при помощи уточнения следующих параметров:

- *Синтаксис*. Синтаксис диалекта может ограничивать виды допустимых термов. Например, если синтаксис диалекта исключает фреймы, тогда те части семантических структур, которые интерпретируют фреймы, становятся ненужными.
- *Истинностные значения*. Семантический каркас разрешает формулам принимать истинностные значения из некоторого частично упорядоченного множества *TV* с отношением порядка $<_t$. В диалекте должно быть выбрано конкретное множество истинностных значений. Так, большинство диалектов используют два значения $\mathbf{f} <_t \mathbf{t}$ (соответствующие *лжи* и *истине*), но диалекты логического программирования, основанные на хорошо обоснованной семантике, используют три значения $\mathbf{f} <_t \mathbf{u} <_t \mathbf{t}$ (**u** - undefined). Множество *TV* должно образовывать полную решетку (для каждого подмножества *TV* должна существовать точная верхняя грань *lub* и точная нижняя грань *glb*) и иметь выделенные элементы **f** и **t**, что для любого $e \in TV$ выполнено $\mathbf{f} \leq_t e$, $e \leq_t \mathbf{t}$. На *TV* должен быть определен оператор отрицания ~ такой, что для любого $e \in TV$ выполнено $\sim\sim e = e$ и $\sim\mathbf{t} = \mathbf{f}$.
- *Типы данных.* Тип данных есть пространство символов, имеющих фиксированную интерпретацию в любой семантической структуре.

С типом ассоциировано *пространство значений*, и отображение из лексического пространства в пространство значений. Лексическое пространство и пространство значений могут быть даже неизоморфными (например, различные константы *1.2^^xs:decimal* и *1.20^^xs:decimal* отображаются в одно и то же значение). RIF-FLD определяет набор типов данных (ядро) [3], которые поддерживаться каждым диалектом. В диалекте также могут определяться специфические типы данных. Семантические структуры всегда определяются по отношению к некоторому множеству *DTS* типов данных.

- *Логическое следование (logical entailment).* Логическое следование в RIF-FLD определяется относительно заранее не заданного множества предопределенных (intended) моделей. В диалекте должно быть указано, какие именно модели считаются предопределенными. Например, в некотором диалекте может быть задано, что все модели считаются предопределенными (что ведет к истинностному следованию логики первого порядка), в другом диалекте предопределенными считаются только минимальные модели, в третьем – только хорошо обоснованные или стабильные модели.

3.1 Семантические структуры

Целью семантических структур, называемых также интерпретациями, является определение множеств и функций, позволяющих сообщить истинностное значение (из множества *TV*) каждой хорошо определенной формуле языка. Семантическая структура, на которой формула обращается в *true*, называется *моделью* формулы.

Семантическая структура I – это кортеж вида $\langle TV, DTS, D, I_C, I_V, I_F, I_{NF}, I_{list}, I_{tail}, I_{frame}, I_{sub}, I_{isa}, I_{=}, I_{external}, I_{connective}, I_{truth}\rangle$. Здесь D – непустое множество, называемое *доменом I*. Остальные элементы структуры – тотальные интерпретирующие функции:

- *Константы.* I_C отображает константы в элементы D.
- *Переменные.* I_C отображает переменные в элементы D.
- *Позиционные термы.* I_F отображает D в тотальные функции $D^* \rightarrow D$ (D^* обозначает множество конечных последовательностей элементов из D).
- *Термы с именованными аргументами.* I_{NF} отображает D в тотальные функции $SetOfFiniteBags(ArgNames \times D) \rightarrow D$ (*SetOfFiniteBags* обозначает множество конечных мультимножеств, *ArgNames* – множество имен аргументов). Пара $\langle s, v\rangle \in ArgNames \times D$ обозначает пару имя аргумента-значение. Имена аргументов, как и пары

целиком могут повторяться: например, в результате подстановки символа *b* вместо переменных *A?, B?* в терм $p(a \to A?\ a \to B?)$ получается терм $p(a \to b\ a \to b)$. Заметим, что $p(a \to b\ a \to b)$ не эквивалентен $p(a \to b)$.

- *Списки*. Функции интерпретации списков имеют вид $I_{list}: D^* \to D$, $I_{tail}: D^+ \times D \to D$. Функция I_{list} инъективна и удовлетворяет соотношению $I_{tail}(a_1, ..., a_k, I_{list}(a_{k+1}, ..., a_{k+m})) = I_{list}(a_1, ..., a_k, a_{k+1}, ..., a_{k+m})$.
- *Фреймы*. I_{frame} отображает D в тотальные функции $SetOfFiniteBags(D \times D) \to D$. Как и в случае термов с именованными аргументами, имена аргументов и пары могут повторяться, но $o[a \to b\ a \to b]$ эквивалентен $p[a \to b]$.
- *Классификация, принадлежность классу, равенство.* Функции интерпретации классификации и равенства – это тотальные функции $D \times D \to D$.
- *Истинность формул*. Функция интерпретации истинности формул I_{truth} – это тотальная функция $D \to TV$.
- *Внешне определенные термы.* Функция $I_{external}$ – отображение согласованного множества схем внешне определенных термов в тотальные функции $D^* \to D$. Для каждой схемы $s = (X_1 \ldots ?X_n; t; loc)$, $I_{external}(s)$ – это тотальная функция $D^n \to D$. Для каждой схемы s предполагается, что функция $I_{external}(s)$ определена в некотором внешнем документе. Например, если s – схема встроенной функции или предиката RIF, то $I_{external}(s)$ определена в [3].
- *Связки, кванторы, агрегатные функции.* Функция интерпретации $I_{connective}$ ставит в соответствие каждой связке, квантору, агрегатной функции функцию $D^* \to D$.

Для семантической структуры I определяется одноименная *функция интерпретации термов* (для хорошо определенных термов):

- $I(k) = I_C(k)$ для $k \in Const$;
- $I(t[p_1 \to v_1 \ldots p_n \to v_n]) = I_{frame}(I(t))(\{\langle I(p_1) \to I(v_1)\rangle, \ldots, \langle I(p_n), I(v_n)\rangle\})$, где $\{\}$ обозначает мультимножество;
- $I(S(t_1 \ldots t_n)) = I_{connective}(S)(I(t_1) \ldots I(t_n))$, где S – связка, квантор, функция агрегации.

Для остальных видов термов функция I определяется аналогично с использованием соответствующих функций интерпретации.

Ограничения семантические структур, налагаемые сигнатурами. Для любой сигнатуры sg диалекта существует подмножество $D_{sg} \subseteq D$, называемое *доменом сигнатуры,* включающее все термы сигнатуры sg и удовлетворяющее следующим свойствам:

- если $sg < sg'$, тогда $D_{sg} \subseteq D_{sg'}$;
- $I_C(k) \in D_{sg}$ для константы k сигнатуры sg;

- $I_V(v) \in D_{sg}$ для переменной v сигнатуры sg;
- если s включает выражение $(s_1 \ldots s_n) \Rightarrow s$, тогда для всех $d \in D_{sg}$ функция $I_F(d)$ отображает $D_{s1} \times \ldots \times D_{sn}$ в D_s;
- если s включает выражение $(p_1 \rightarrow s_1 \ldots p_n \rightarrow s_n) \Rightarrow s$, тогда для всех $d \in D_{sg}$ функция $I_{NF}(d)$ отображает множество $\{\langle p_1, D_{s1}\rangle, \ldots, \langle p_n, D_{sn}\rangle\}$ в D_s;
- если сигнатура $\rightarrow$ включает выражения $(sg, s_1, r_1) \Rightarrow k$, ..., $(sg, s_n, r_n) \Rightarrow k$, тогда для всех $d \in D_{sg}$ функция $I_{frame}(d)$ отображает множество $\{\langle D_{s1}, D_{r1}\rangle, \ldots, \langle D_{sn}, D_{rn}\rangle\}$ в D_k;
- если сигнатура # включает выражение $(s\ r) \Rightarrow k$, тогда функция I_{isa} отображает $D_s \times D_r$ в D_k;
- если сигнатура ## включает выражение $(s\ s) \Rightarrow k$, тогда функция I_{sub} отображает $D_s \times D_s$ в D_k;
- если сигнатура = включает выражение $(s\ s) \Rightarrow k$, тогда функция $I_=$ отображает $D_s \times D_s$ в D_k.

Ограничения семантических структур, налагаемые типами данных. Пусть тип данных $dt \in DTS$ и L_{dt}: $LS_{dt} \rightarrow VS_{dt}$ – ассоциированное с ним отображение из лексического пространства в пространство значений. Тогда константы типа интерпретируются в соответствии с отображением L_{dt} : $VS_{dt} \subseteq D$ и для любой константы $lit \in LS_{dt}$ выполнено $I_C(lit) = L_{dt}(lit)$.

3.2 Интерпретация недокументных формул

В данном разделе рассматриваются формулы, не являющиеся документными (т.е. *недокументные* формулы) или удаленными (см. раздел 2.2).

Каждая семантическая структура I определяет истинностное значение $TVal_I(f)$ формулы f.

Истинностное означивание хорошо определенных формул, т.е. определение функции $TVal_I$, производится при помощи интерпретирующих функций семантической структуры: $TVal_I\ (t) = I_{truth}(I(t))$ если t – константа, переменная, позиционный терм, терм с именованными аргументами, равенство, классификация, включение, фрейм, формула со связкой или квантором или внешне определенная атомарная формула. При этом должны выполняться следующие ограничения семантических структур:

- *Ограничение равенства.* $TVal_I\ (x = y) = true$ если $I(x) = I(y)$ и $TVal_I\ (x = y) = false$ иначе.
- *Транзитивность отношения класс-подкласс*. Для всех $c_1, c_2, c_3 \in D$ выполнено $glb(TVal_I(c_1\ \#\#\ c_2), TVal_I(c_2\ \#\#\ c_3)) \leq TVal_I(c_1\ \#\#\ c_3)$.

- *Членство объекта во всех подклассах класса, членом которого он является.* Для всех *o, c, s* $\in D$ выполнено $glb(TVal_I(o \# c), TVal_I(c \#\# s)) \le TVal_I(o \#\# s)$.
- *Ограничение фреймов.* $TVal_I(o[a_1 \rightarrow v_1 \ ... \ a_k \rightarrow v_k]) = glb(TVal_I(o[a_1 \rightarrow v_1]), \ ... \ , TVal_I(o[a_k \rightarrow v_k]))$. Фрейм является утверждением о состоянии объекта. Данное ограничение выражает истинность утверждения о совокупном состоянии объекта (значении всех его атрибутов a_i) посредством истинности значений отдельных атрибутов. Например, если хотя бы одно утверждение $o[a_i \rightarrow v_i]$ о состоянии атрибута a_i обращается в ложь, то и наибольшая нижняя грань (*glb*) значений истинности всех утверждений о состоянии атрибутов обратится в ложь, а значит, обратится в ложь и совокупное утверждение о состоянии.
- *Ограничение конъюнкции.*
  $TVal_I(And(c_1 \ ... \ c_k]) = glb(TVal_I(c_1), \ ... \ , TVal_I(c_k))$.
- *Ограничение дизъюнкции.*
  $TVal_I(Or(c_1 \ ... \ c_k]) = lub(TVal_I(c_1), \ ... \ , TVal_I(c_k))$.
- *Ограничения отрицания.* $TVal_I(Neg \ Neg \ f) = TVal_I(f)$ *и*
  $TVal_I(Naf \ f) = \sim TVal_I(f)$. Данные ограничения описывают свойства симметричного и несимметричного отрицания соответственно. Для симметричного отрицания истинность двойного отрицания есть истинность самой формулы под отрицанием, а несимметричное отрицание сводится к отрицанию на множестве истинностных значений *TV* (см. начало раздела 3).
- *Ограничения кванторов.* $TVal_I(Exists \ ?v_1 \ ... \ ?v_n \ (f)) = lub(TVal_{I*}(f))$ и
  $TVal_I(Forall \ ?v_1 \ ... \ ?v_n \ (a)) = glb(TVal_{I*}(a))$, где верхняя и нижняя грани берутся по всем интерпретациям *I*,* совпадающих с *I* во всем, кроме, может быть, интерпретации переменных $?v_1 \ ... \ ?v_n$.
- *Ограничение импликации.* $TVal_I(head :- body) = true,$ если $TVal_I(head) \ge TVal_I(body)$ и $TVal_I(head :- body) < true$ иначе.

В диалектах, определяющих новые связки или кванторы, должны определяться также соответствующие ограничения на истинностную функцию.

Истинностное означивание групповых формул производится следующим образом: $TVal_I(Group(f_1 \ ... \ f_n)) = glb(TVal_I(f_1), \ ..., \ TVal_I(f_n))$, т.е. группа формул трактуется как конъюнкция.

3.2 Интерпретация документов

Документы (см. раздел 2.2) интерпретируются при помощи семантических мультиструктур, являющихся множествами семантических структур. Мультиструктуры предназначены для определения семантики

мультидокументов RIF – документов, импортирующих другие документы или содержащие ссылки на документы.

Семантическая мультиструктура I' есть множество семантических структур $\{J, K, I^{ik}, M^{ik}\}$, где *J, K* – семантические структуры, а I^{ik}, M^{ik} – семантические структуры, помеченные локаторами документов (т.е. пары локатор-структура).

Различные структуры используются для интерпретации различных конструкций: недокументных формул (*J*), документных формул (*K*), импортируемых документов (I^{ik}), удаленных модулей (M^{ik}).

Структуры *J, K,* I^{ik} могут отличаться только на константах пространства *rif:local* (специального пространства локальных констант документа [3]). Это означает, что недокументные формулы, документные формулы и импортируемые документы интерпретируются в соответствии с одной и той же семантикой, и отличаться могут лишь на константах, являющихся локальными для документов. Структуры M^{ik} должны совпадать с остальными структурами мультиструктуры только на константах, не принадлежащих *rif:local.* Это означает, что семантика удаленного модуля может быть совершенно отличной от семантики документа, подключающего удаленный модуль. Совпадать интерпретации должны лишь на константах, не являющихся локальными для документов.

Интерпретация удаленных термов. Пусть *d* – документ, $I'=\{J, K, I^{ik}, M^{ik}\}$ – семантическая мультиструктура, содержащая семантические структуры для всех документов, импортированных в *d*, а также для всех внешних модулей, подключенных к *d*. Пусть *f@r* – удаленный терм, входящий в *d* или в один из импортируемых документов или внешних модулей (обозначим этот документ *d'*). Пусть $L \in I'$ – некоторая структура. Если документ включает единственную директиву удаленного модуля (см. раздел 2.2) *Module(n* j_k*)* в *d',* причем *L(r) = L(n)*, тогда

$$L(f@r) = M^{ik}(f)$$

Если такой директивы нет или она не уникальна, то *L(f@r)* не определено и может быть любым элементом домена *L*. Истинностная функция определяется на удаленном терме следующим образом:

$$TVal_L(f@r) = I_{truth}(L(f@r))$$

Данное определение является весьма общим, однако на практике удаленные термы имеют смысл только тогда, когда они интерпретируются фиксированными хорошо определенными значениями. Предполагается, что в диалектах накладываются соответствующие ограничения. Примерами фиксированных интерпретаций являются типы данных и эрбрановские домены [1].

Истинностное означивание документов. Пусть *d* – документ; $d_1 \ldots d_n$ – все документы, импортируемые в *d*; $G, G_1 \ldots G_n$ – групповые формулы, ассоциированные с d_i. Пусть $I'=\{J, K, I^{ik}, M^{ik}\}$ – семантическая

мультиструктура, содержащая структуры, помеченные локаторами $i_1, \ldots, i_n$ документов $d_1 \ldots d_n$. Тогда

$$TVal_I(d) = glb(TVal_K(G), TVal_{Ii1}(G_1), \ldots, TVal_{Iin}(G_n))$$

Заметим, что структура K используется для интерпретации основного документа, а структуры I^{i1} – для интерпретации импортируемых документов G_i.

Семантическая структура или мультиструктура I называется *моделью* формулы f (записывается $I \models f$) iff $TVal_I(f) = true$.

3.4 Логическое следование

Одной из главных особенностей FLD как каркаса, поддерживающего множество различных семантик, является понятие логического следования, основанного на предопределенных (intended) моделях. Семантика следования различается в разных логических языках. Например, в логике первого порядка формула $p \leftarrow \neg p$ *влечет* (в смысле логического следования) p, но, например, не влечет q. В логическом языке с семантикой стабильных моделей формула $p \leftarrow \neg p$ является противоречивой, и из нее следует что угодно, в том числе и q. В логическом языке с хорошо обоснованной семантикой формула $p \leftarrow \neg p$ непротиворечива, но ничего интересного из нее не следует: ни p, ни q. Решение, которое используется в RIF-FLD для объединения различных семантик, предложено в работе [10], и основывается на понятии семантических структур. Если S – множество семантических структур, то формула f *S-влечет* формулу p iff для каждой семантической структуры s из S, если s – предопределенная модель для f, то s также является моделью p, т.е. p истинна по крайней мере на тех интерпретациях, на которых истинна f (p не менее истинна, чем f). Оказывается, что во многих логических языках понятие истинностного следования определяется именно таким или эквивалентным образом. Отличие состоит лишь в том, из каких именно семантических структур состоит множество S и какие структуры считаются предопределенными моделями.

Формально, формула $f \models$ *(влечет, entails)* g iff для любой семантической мультиструктуры I формулы f выполнено $TVal_I(f) \leq TVal_I(g)$.

4 Диалект RIF-CASPD как специализация каркаса

Диалект-ядро программирования множества ответов (RIF Core Answer Set Programming Dialect, RIF-CASPD) [11] разработан как язык для обмена правилами между системами, основанными на парадигме программирования множества ответов (Answer Set Programming, ASP) [12]. Парадигма ASP наиболее полезна при описании и решении NP-полных задач [13].

Синтаксически, RIF-CASPD является языком правил без функциональных символов, в котором головы правил могут содержать дизъюнкцию или быть пустыми, но не могут содержать импликацию. Диалект также содержит средства, типичные для RIF-диалектов: фреймы, типы данных XML-Schema, два вида отрицания: классическое и отрицание по умолчанию с ASP-семантикой.

Рассмотрим пример, иллюстрирующий синтаксические возможности диалекта.

Пример. Судоку.

```
Document(
 Prefix(su <http://example.com/sudoku#>)

 Group(
  su:coordinates(1 1)  su:coordinates(1 2)
  su:coordinates(2 1)  su:coordinates(2 2)
  su:position(1 1 1) su:position(1 2 2)

  Forall ?X ?Y (
   Or(su:position(1 ?X ?Y) su:position(2 ?X ?Y)
      su:position(3 ?X ?Y) su:position(4 ?X ?Y)
      su:position(5 ?X ?Y) su:position(6 ?X ?Y)
      su:position(7 ?X ?Y) su:position(8 ?X ?Y)
     su:position(9 ?X ?Y)
   )  :- su:coordinates(?X ?Y)
  )
  Forall ?X1 ?X2 ?Y ?N (
   :- And ( su:coordinates(?X1 ?Y) su:coordinates(?X2 ?Y)
            su:position(?N ?X1 ?Y) su:position(?N ?X2 ?Y)
       Naf ?X1 = ?X2 )
  )
))
```

Первое из правил, описывающее игру *судоку,* определяет, что в каждой из клеток поля могут находиться цифры от 1 до 9. Второе правило, являющееся ограничением, определяет, что одна и та же цифра не может появляться в нескольких клетках столбца. Данные привила, конечно, не являются полным описанием игры и приведены лишь в качестве примера. Приведенные факты описывают подмножество доски судоку (su:coordinates) и ее частичное заполнение (su:position).

4.2 Синтаксис диалекта

Синтаксис диалекта определяется как специализация синтаксиса каркаса (см. раздел 2.5).

Точки расширения. Диалект не содержит точек расширения.

Алфавит. Алфавит диалекта совпадает с алфавитом RIF-FLD.

Сигнатуры. Диалект содержит следующие сигнатуры, определяющие контексты, в которых могут появляться символы языка:

- Стандартные сигнатуры *individual{}* (для индивидов) и *atomic{}* (для атомарных формул).
- Сигнатура предикатов *p,* содержащая стрелочные выражения для позиционных предикатов *() ⇒ atomic, (individual) ⇒ atomic, (individual individual) ⇒ atomic* и т.д., а также стрелочные выражения для предикатов с именованными аргументами *($s_1 \rightarrow$individual ... $s_k \rightarrow$individual) ⇒ atomic*.
- Сигнатура равенства *={(individual individual) ⇒ atomic}*.
- Сигнатура фреймов *→{(individual individual individual) ⇒ atomic}*.
- Сигнатура принадлежности классу *#{(individual individual) ⇒ atomic}*.
- Сигнатура отношения класс-подкласс *##{(individual individual) ⇒ atomic}*.

Каждый символ константы имеет сигнатуру *individual, atomic,* и, возможно, *p*, т.е. может обозначать функцию, предикат, индивид или утверждение. Переменные и символы, соответствующие типам данных, имеют сигнатуру *individual.*

Термы. Диалект содержит следующие виды термов: константы, переменные, позиционные термы, термы с именованными аргументами, равенство, фреймы, включение в класс, класс-подкласс, внешние термы, формулы. Переменными в языке не разрешается пользоваться в контексте формул, предикатных и функциональных символов (такими символами могут быть только константы). Это означает, что не допускаются термы вида *?X(t_1 ... t_n)* или *?X($s_1 \rightarrow v_1$ ... $s_n \rightarrow v_n$)* – здесь переменная *?X* используется в контексте предикатного или функционального символа. Также переменная не является формулой, т.е. недопустимы формулы вида *And(?X f)*, *Forall ?X (?Y)* или *Neg(?X)*.

Пространства символов диалекта указаны в документе [3].

Формулы. В диалекте поддерживаются следующие виды формул:

- *Условия*. Условиями являются атомарные формулы, их отрицания, конъюнкции и дизъюнкции. Все условия могут находиться под квантором существования. Отрицания атомарных формул могут иметь вид *Neg f, Naf f, Naf Neg f*.
- *Правила*. Правилом диалекта является импликация *f :- p* или импликация под квантором всеобщности. Заключением правила может быть атомарная формула, атомарная формула с отрицанием, конъюнкция атомарных формул или дизъюнкция таких конъюнкций. Посылкой правила может быть только *условие*.
- *Ограничения* – формулы вида *:- p,* где *p* – условие. Ограничения могут находиться под квантором всеобщности.
- *Факты* – дизъюнкции (возможно под квантором всеобщности) атомарных формул или их отрицаний, не содержащие *Naf*.
- *Группы* могут содержать только правила, факты, ограничения и группы.
- *Документы* состоят из директив (см. раздел 2.2) и одной группы. Не допускаются директивы *Module*, директива диалект может быть только вида *Dialect("RIF-CASPD"),* т.е. импортироваться могут только документы диалекта RIF-CASPD.

4.3 Семантика диалекта

Семантика диалекта определяется как специализация семантического каркаса (см. раздел 3). Конкретизации в данном случае требуют следующие параметры: *истинностные значения* (множество *TV* состоит из двух элементов: **t** и **f**, **f** $<_t$ **t**), *типы данных* (диалект включает типы, определенные в разделе *Datatypes* документа [3]) и л*огическое следование,* определяемое предопределенными моделями диалекта.

Семантика ASP обычно определяется с использованием *эрбрановских интерпретаций*, поэтому предопределенные модели для документов диалекта выбираются среди эрбрановских семантических структур RIF-FLD вида $\langle TV, DTS, HD, I_C, I_V, I_F, I_{NF}, I_{list}, I_{tail}, I_{frame}, I_{sub}, I_{isa}, I_{=}, I_{external}, I_{connective}, I_{truth}\rangle$. В этих интерпретациях в качестве домена *D* используется специальный *эрбрановский домен HD* – множество всех хорошо сформированных базовых (т.е. не содержащих переменных) термов, факторизованное по отношению равенства.

Семантическая мультиструктура *I* диалекта RIF-CASPD является *предопределенной моделью* документа *G* тогда и только тогда, когда *I* является предопределенной моделью для *G** - множества базовых экземпляров *G* (множества всех таких документов, которые получены из

G заменой переменных на базовые термы, причем вхождения одной и той же переменной заменяются на один и тот же терм).

Мультиструктура *I* является *предопределенной моделью* базового документа *G* (не содержащего переменных) тогда и только тогда, когда *I* является *минимальной* моделью для фактора *G/I*.

Фактор *G/I* базового документа *G* получается из *G* заменой каждой атомарной формулы с отрицанием вида *Naf f* на ее истинностное значение и максимальным упрощением документа при помощи законов пропозициональной логики.

Модель *I* документа *G* называется *минимальной*, если не существует такой другой модели *I',* что для любой базовой атомарной формулы *f* или ее отрицания *Neg f* выполнено, что истинность *f* в *I'* влечет истинность *f* в *I* и ложность *f* в *I* влечет ложность *f* в *I'*.

Заметим, что документ диалекта может вообще не иметь предопределенных моделей. Простейшим примером является формула *p:- Naf p.* Модели также могут не существовать из-за противоречивости формул. Так, например, у формулы

```
Group(
  p
  Neg p
)
```

нет предопределенных моделей, поскольку формулы *p* и *Neg p* не могут одновременно обращаться в истину в семантических структурах RIF-CASPD.

Рассмотрим примеры предопределенных и непредопределенных моделей формулы

```
Forall ?X1 ?X2 ?Y ?N (
 :- And ( su:coordinates(?X1 ?Y) su:coordinates(?X2 ?Y)
          su:position(?N ?X1 ?Y) su:position(?N ?X2 ?Y)
          Naf ?X1 = ?X2 )
)
```

из спецификации судоку.

Среди базовых экземпляров данной формулы имеются следующие ограничения:

```
:- And ( su:coordinates(1 1) su:coordinates(1 1)
         su:position(1 1 1)  su:position(1 1 1)
         Naf 1 = 1 )
```

и

```
:- And ( su:coordinates(2 1) su:coordinates(1 1)
```

```
        su:position(1 2 1)  su:position(1 1 1)
        Naf 2 = 1 )
```

Рассмотрим структуру *I* с означиванием, обращающим в истину следующие предикаты: *su:coordinates(1 1), (su:coordinates(1 2), (su:coordinates(2 1), su:coordinates(2 2), su:position(1 1 1), su:position(1 2 1)*. Это означает, что доска квадратная 2x2, и единица стоит в клетках с координатами (1, 1) и (2, 1):

1	1

Такое решение судоку не является правильным – единица появляется дважды в одной строке. Действительно, утверждение *Naf 1=1* – истинно, *Naf 2=1* – ложно, и фактор второго из ограничений имеет вид:

```
:- And ( su:coordinates(2 1) su:coordinates(1 1)
         su:position(1 2 1)  su:position(1 1 1)
```

На *I* тело данного ограничения истинно, а значит *I* нарушает ограничение и не является предопределенной моделью.

С другой стороны, структура, на которой в истину обращаются предикаты *su:coordinates(1 1), su:coordinates(1 2), su:coordinates(2 1), su:coordinates(2 2), su:position(1 1 1), su:position(1 2 2))*, является предопределенной моделью:

1	

5 Конформность диалектов RIF конкретным процессорам

RIF не требует от конкретных процессоров реализации презентационного синтаксиса диалекта RIF. Конформность диалекта процессору описывается в терминах сохраняющих семантику взаимных трансформаций синтаксиса системы и XML-синтаксиса каркаса [1].

Пусть *T* – множество типов данных и пространств символов, включающее типы данных из [3]. Пусть *E* – согласованное множество схем внешне определенных термов, включающее встроенные схемы из [3]. Пусть *D* – диалект RIF. Тогда формула *f* есть $D_{T,E}$ – формула iff *f* – формула диалекта *D;* все типы данных и пространства символов, используемые в *f*, есть в *T*; все внешне определенные термы, используемые в *f*, являются конкретизациями схем из *E*.

Процессор RIF называется *конформным* $D_{T,E}$ *потребителем* iff он реализует сохраняющее семантику отображение *m* множества $D_{T,E}$ в язык

L процессора. Формально это означает, что для любой пары f, g из $D_{T,E}$, для которых определено $f \models_D g$, выполнено $f \models_D g \Leftrightarrow m(f) \models_L m(g)$. Здесь $\models_D$ означает истинностное следование в диалекте D.

Процессор RIF называется *конформным $D_{T,E}$ поставщиком* iff он реализует сохраняющее семантику отображение m языка L процессора в множество $D_{T,E}$. Формально это означает, что для любой пары f, g из L, для которых определено $f \models_D g$, выполнено $f \models_L g \Leftrightarrow m(f) \models_D m(g)$.

Допустимым документом диалекта D является XML-представление хорошо определенного документа диалекта D.

Литература

1. RIF Framework for Logic Dialects. W3C Proposed Recommendation / Eds. H. Boley, M. Kifer. – 2010. – http://www.w3.org/TR/2010/PR-rif-fld-20100511/
2. M. Kifer. Rule Interchange Format: the Framework // Web Reasoning and Rule Systems: Proc. of the Second International Conference, LNCS 5348. – Berlin-Heidelberg: Springer Verlag, 2008. – P. 1-11.
3. RIF Datatypes and Built-Ins 1.0. W3C Recommendation / Eds. A. Polleres, H. Boley, M. Kifer. – 2010. – http://www.w3.org/TR/2010/PR-rif-fld-20100511/
4. W. Chen, M. Kifer, D.S. Warren. HiLog: A Foundation for higher-order logic programming // Journal of Logic Programming. – 1993. - V. 15, N. 3. - P. 187-230.
5. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages // Journal of ACM. - 1995. – V. 42. – P. 741–843.
6. K. L. Clark. Negation as failure // Logic and Data Bases / Eds. H. Gallaire and J. Minker. - Plenum Press, 1978. – P. 292–322.
7. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming // Logic Programming: Proceedings of the Fifth Conference and Symposium. – 1988. - 1070–1080.
8. A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs // Journal of ACM. – 1991. – V. 38, № 3. - P. 620–650.
9. H. B. Enderton. A Mathematical Introduction to Logic. - Academic Press, 2001.
10. Y. Shoham. Nonmonotonic logics: meaning and utility // In Proc. 10th International Joint Conference on Artificial Intelligence. - Morgan Kaufmann, 1987. – P. 388–393.
11. RIF Core Answer Set Programming Dialect / Eds. S. Heymans, M. Kifer. – 2009. - http://ruleml.org/rif/RIF-CASPD.html
12. M. Gelfond and N. Leone. Logic programming and knowledge representation - The A-Prolog perspective // Artificial Intelligence. - 2002. – V. 138, № 1-2. – P. 3-38.
13. Л.А.Калиниченко, С.А.Ступников. Анализ мотивации, целей и подходов проекта унификации языков на правилах // Труды Второго симпозиума «Онтологическое моделирование: состояние, направления исследований и применения» (ассоциирован с XII Всероссийской научной конференцией RCDL'2010). – М.: ИПИ РАН, 2011.
14. J. J. Alferes, L. M. Pereira, and T. C. Przymusinski. Strong and Explicit Negation in Non-Monotonic Reasoning and Logic Programming // Proc. of the European

Workshop on Logics in Artificial Intelligence, LNCS 1126. – Springer-Verlag, 1996.

Abstract

RIF Framework for Logic Dialects: Analysis and a Case Study
L. A. Kalinichenko, S. A. Stupnikov

RIF Framework for Logic Dialects is intended for describing of syntax and semantics of RIF logic dialects through such generic constructions as signatures, symbol spaces, semantic structures and so on. The aim of the paper is to provide analysis of the RIF Framework and a case study of concrete dialect definition. The main features of syntactic and semantic frameworks are considered. Concrete dialects are assumed to be specializations of these frameworks.