Особенности алгебраического подхода к формированию онтологий♣

© Е.М. Бениаминов, В.А. Лапшин, Д.В. Перов

Российский государственный гуманитарный университет, г. Москва
mefrill@ yandex.ru

Аннотация

В работе описывается алгебраический подход к формированию онтологий. Традиционные методы описания онтологий основаны на языках исчислений предикатов. Алгебраический подход имеет свои особенности, которые обсуждаются в данной работе. Одной из таких особенностей является возможность динамического расширения пользователями синтаксиса языка формирования онтологии для наиболее близкого соответствия языка описания описываемой модели. В работе представлены принципы формирования открытого языка представления знаний на основе алгебраического подхода. Также в работе обсуждаются методы определения понятий онтологии на основе расширения и уточнения понятий, присутствующих в данной онтологии. Описываемый в работе подход реализован в проекте ЭЗОП (http://www.ezop–project.ru).

1 Введение

Большинство популярных инструментов построения онтологий использует в качестве основного формализма моделирования те или иные логики исчисления предикатов [2]. Так, в языке OWL [5], который на данный момент является «de facto» стандартным языком описания содержания онтологий, используется формализм логики предикатов первого порядка с элементами языка второго порядка (различными встроенными в язык ограничениями на вид предикатов онтологии). В данной работе описывается другой формализм, на основе которого можно формировать онтологии, а именно: алгебраический подход к формированию онтологий [9, 11].

В алгебраическом подходе онтология представляется в виде теории, высказывания которой формируются на основе множества имен типов и имен операций, называемого сигнатурой. Сигнатуры обычно формируются пользователями системы, предоставляющей сервис по формированию онтологий. Операции представляют в онтологии как отношения между объектами, так и средства построения новых понятий (типов) из уже существующих. Смысл операций задается соотношениями между ними, выраженными в виде равенств.

Для образования выражений на основе имен операций сигнатуры обычно вводится множество переменных, которые помечены типами этой сигнатуры. На основе имен операций и заданного множества переменных можно сформировать множество синтаксически правильных выражений этой сигнатуры или термов. Некоторые термы могут быть преобразованы друг в друга с помощью подстановок, правила которых задаются соотношениями–равенствами данной сигнатуры. Такие термы можно считать семантически эквивалентными. Если вместо переменных использовать константы (т. е. имена тех операций, которые не имеют параметров), то множество всех термов данной сигнатуры разбивается на непересекающиеся классы эквивалентности. Полученное множество классов эквивалентности является внутренним (машинным) представлением онтологии и может быть использовано для вычислений.

Каждый класс эквивалентности в описанной выше конструкции содержит бесконечное множество термов. Оперировать такими множествами невозможно, поэтому для каждого класса выбирается его представитель или дескриптор. Пусть, например, операции сигнатуры задают арифметические действия:

«сложение» +:INT,INT→INT

и

«вычитание» –:INT,INT→INT.

Тогда константа 5: →INT будет дескриптором класса, содержащего термы +(2,3), –(+(2,7),4) и т. п. Каждый терм здесь представляет арифметическое выражение, для вычисления которого надо найти дескриптор класса, которому принадлежит данный терм.

В алгебраическом подходе, как и в логическом, онтологии представляют собой теории, но в алгебраическом подходе пользователь может определять свои операции, которые из одних понятий (типов) и отображений между ними (предполагается, что операции удовлетворяют некоторым условиям, которые также задаются пользователем в виде равенств) строит новые понятия и отображения. Типы сигнатуры – это типы (классы) онтологии, а операции

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

могут представлять соотношения между объектами типов или быть конструкторами новых объектов. На операции (т. е. на соотношения между объектами онтологии) налагаются ограничения посредством соотношений в виде равенств. На основе такого описания можно производить вычисления (находить дескрипторы классов) стандартным способом, используя правила переписывания, задаваемые соотношениями–равенствами, или используя операции вычисления, специально заданные для некоторых сигнатур. Например, можно было бы для имен операций «сложение» и «вычитание» задать их семантику (вычисление дескриптора терма) обычным вычислением арифметического выражения, как это делается на калькуляторе.

Имена операций в сигнатуре синтаксически можно рассматривать как правила контекстно-свободной порождающей грамматики Хомского [6]. Например, операцию «сложения» +:INT,INT→INT можно задать как INT→+(INT,INT). Это правило контекстно-свободной грамматики можно записать в инфиксном виде. В этом случае получим правило INT→INT+INT. Здесь символ «+» выступает в качестве терминала контекстно-свободной грамматики, а имя типа INT – как нетерминальный символ.

Такие инфиксные записи имен операций со строками внутри будем называть шаблонами. Каждому имени операции можно поставить в соответствие один или более шаблонов и, обратно, каждому шаблону можно сопоставить имя операции. Таким образом, имена операций сигнатуры можно описывать шаблонами. Шаблоны задают язык онтологии, который более синтаксически близок к содержанию описываемой онтологии, чем имена операций, записанные в префиксном виде.

2 Иллюстративный пример

В качестве простой иллюстрации введенных выше понятий рассмотрим простой пример онтологии семейных отношений. Этот пример будет также полезен и для понимания дальнейшего изложения.

Множество типов онтологии семейных отношений состоит из следующих элементов: Человек, Мужчина и Женщина. Типы Мужчина и Женщина являются подтипами (подклассами) типа Человек. Отношение тип – подтип обозначим с помощью символа «<», т. е. имеем:

Мужчина < Человек,
Женщина < Человек.

Ввиду того, что отец и мать у каждого человека определяются однозначно, эти отношения можно представить в виду функциональных символов:

Отец : Человек → Мужчина,
Мать : Человек → Женщина.

Пусть X – переменная типа Человек, тогда определим параметрически родственные классы как подклассы класса Человек следующим образом:

Дедушка(X : Человек) < Мужчина,
Бабушка(X : Человек) < Женщина,
Брат-по-отцу(X : Человек) < Мужчина,
Родная-сестра(X : Человек) < Женщина.

После того, как введены сигнатурные символы операций и классов, можно определить соотношения равенства. Мы определим только некоторые из возможных соотношений, этого кажется вполне достаточным для иллюстрации:

Дедушка(X)={Отец(Отец(X)); Отец(Мать(X))},
Бабушка(X)={Мать(Отец(X)); Мать(Мать(X))},
Отец(Брат-по-отцу(X))=Отец(X),
Отец(Родная-сестра(X))=Отец(X),
Мать(Родная-сестра(X))=Мать(X).

Введенные соотношения очевидны и выражают известные факты относительно родственных отношений. Подклассы Дедушка и Бабушка задаются перечислением их элементов, а остальные родственные подклассы – условиями в виде равенств. Заметим, что в операциях используются подстановки типов вместо их подтипов. Например, операция Отец берет в качестве единственного параметра тип Человек, но в соотношениях ему этой операции передаются типы Мужчина и Женщина.

Введем теперь константы (объекты классов) как операции без параметров:

Иван : → Мужчина,
Василий : → Мужчина,
Владимир : → Мужчина,
Константин : → Мужчина,
Марья : → Женщина,
Татьяна : → Женщина,
Анастасия : → Женщина.

Можно теперь определить, кто чьим родственником является. Для этого также используем соотношения-равенства:

Отец(Иван) = Василий,
Мать(Татьяна) = Марья,
Мать(Анастасия) = Марья,
Брат-по-отцу(Владимир) = Константин,
Родная-сестра(Владимир) = Анастасия.

Полученное описание является онтологией родственных отношений некоторой семьи, в которой не все родственные связи известны.

Если теперь попытаться вычислить значение операции Дедушка(Владимир), то ввиду определенных выше соотношений будут вычисляться выражения:

Отец(Отец(Владимир))

и

Отец(Мать(Владимир)).

Но, так как выполняется Анастасия=Родная-сестра(Владимир), то в онтологии вычисляемого выражения Мать(Владимир), Мать(Анастасия) и Марья – синонимы, среди которых в качестве представителя (дескриптора) выделяется простейшее: Марья. Отсюда выражение для второго дедушки Отец(Мать(Владимир)) синонимично выражению Отец(Мария), которое и выдается как более короткое.

Конечно, определенные в примере соотношения достаточно примитивны. Но вполне можно определить онтологию булевой алгебры и использовать ее операции для задания логики более сложных соотношений. Булеву алгебру можно определить в от-

дельной онтологии и просто подгружать к онтологии семейных отношений как внешний модуль. Модульность онтологий, формируемых на основе алгебраического подхода, реализуется просто, это является одним из его преимуществ.

На практике в качестве базовых онтологий (онтологий ядра) сначала определяют различные математические понятия: множества и операции на них, числа и булевы типы. Только после этого вводят операции как элементы специального типа «Операция». Соотношения вида тип – подтип обычно также определяют с помощью специально введенной операции, не используя для этого структуру решетки на множестве типов сигнатуры алгебры, которая представляет данную онтологию.

Далее мы рассмотрим введенные выше понятия более формально.

3 Абстрактные алгебраические типы

В алгебраическом подходе для представления понятий используются так называемые многосортные алгебры [1, 8]. Для определения многосортной алгебры сначала задается ее сигнатура как пара $\Sigma=(S,OP)$, состоящая из множества S имен типов (сортов) и множества имен операций OP.

Обычно на множестве S не задается никаких отношений, что, в общем, не отражает того, что типы в онтологии в подавляющем большинстве случаев образуют иерархию. Поэтому часто на множестве S определяется отношение частичного порядка так, чтобы это множество образовывало решетку. Решеткой называется алгебраическая структура, заданная на частично-упорядоченном множестве таким образом, что выполняются следующие условия:

- для любых двух элементов $s_1, s_2 \in S$ имеется их непосредственный родительский элемент $s_p \in S$, представляющий собой наименьшую верхнюю грань этих элементов;
- для любых двух элементов $s_1, s_2 \in S$ имеется их непосредственных дочерний элемент $s_c \in S$, представляющий собой наибольшую нижнюю грань этих элементов.

Обычно используют т. н. полные решетки, т. е. такие решетки, у которых существуют наибольший и наименьший элементы для любого множества элементов. В онтологии наименьший тип (класс) обычно называют «ничто», а наибольший носит название «объект» (thing). Отношение иерархии формирует дерево, если для каждого типа разрешается иметь не более одного родителя (в этом случае наибольшей нижней гранью любых двух элементов является наименьший элемент), и образует направленный ациклический граф, если разрешено множественное наследование типов.

Имена операций имеют вид $\sigma:s_1\times s_2\times..\times s_n\rightarrow s$, т. е. вместе с именем операции указываются типы ее аргументов и тип результата.

Σ-алгебра для сигнатуры $\Sigma=(S,OP)$ – это семейство A множеств A_s, заданных для каждого сорта $s \in S$ и называемых носителями алгебры A, а также семейство отображений $\alpha_\sigma:A_{s1}\times A_{s2}\times\ldots\times A_{sn} \rightarrow A_s$, заданных для каждого имени операции σ в сигнатуре $\Sigma=(S,OP)$. Иначе говоря, на семействе множеств, представляющих в Σ-алгебре типы, имена которых заданы в множестве S, задаются отображения, имена и типы которых заданы в множестве Σ. Каждая такая Σ-алгебра представляет собой модель вычисления, заданную на именах типов и операций, перечисленных в сигнатуре.

Пусть X обозначает некоторое множество (множество переменных), элементы которого типизированы типами из множества S. Тогда термом сигнатуры называется выражение вида $\sigma(T_1, T_1,\ldots, T_n)$, где $\sigma \in \Sigma$ – это имя операции, а $T_1, T_1,\ldots, T_n$ – это либо переменные соответствующих типов, либо термы, сформированные из имен операций данной сигнатуры тем же способом. Например, пусть $+:INT\times INT\rightarrow INT$ – это операция из сигнатуры $\Sigma=(S,OP)$ и $a, b, c \in X$ – переменные. Тогда выражение +(a,b) образует терм. Вместо параметров можно поставить имя операции «+», т. к. тип ее результата совпадает с типами параметров. Например, выражение +(+(a,b),c) также образует терм. Множество термов, образованных на основе операций данной Σ-алгебры с переменными X, обозначается как $T_\Sigma(X)$. В качестве переменных можно использовать константы, образуемые из имен операций с местностью ноль, т. е. операций вида $\sigma:\rightarrow s$ с пустым списком параметров. Множество всевозможных термов, образованных из констант и имен операций для данной сигнатуры, обозначается T_Σ.

Множество термов $T_\Sigma(X)$ представляет собой Σ-алгебру, т. е. имеет алгебраическую структуру. Дей-

ствительно, каждому имени типа $s \in S$ в сигнатуре $\Sigma=(S,OP)$ можно сопоставить множество термов, чья сигнатура имени операции имеет тип s в качестве результата. Например, типу INT из примера выше будут соответствовать термы вида +(a,b), +(+(a,b),c), +(+(a,b),+(c,b)) и т. д. Семантика операций также прозрачна: каждому имени операции $\sigma:s_1\times s_2\times..\times s_n\rightarrow s$ и каждой n-ке термов t_1, t_1,…, t_n типов s_1, s_1,…, s_n, соответственно, дается в качестве значения терм типа s вида $\sigma(t_1, t_1,\ldots, t_n)$. Например, для операции *:INT×INT→INT термам +(a,b) и +(+(a,b),+(b,c)) дается в соответствие терм *(+(a,b),+(+(a,b),+(c,b))).

Для наложения ограничений на операции Σ-алгебры используются соотношения в виде равенств. Точнее, равенство – это тройка вида (X,T_1,T_2), где T_1 и T_2 – термы, образованные из операций Σ-алгебры и переменных из X. Например, ({a},+(a,b),+(b,a)) выражает свойство коммутативности операции +. Часто равенства записывают просто в виде $T_1= T_2$, а множество переменных определяется из контекста. Например, равенство ({a},+(a,b),+(b,a)) можно записать просто +(a,b)=+(b,a).

Множество термов T_Σ, где в качестве переменных используются константы (т. е. символы операций, список аргументов которых пуст. Например, константа 5 определяется как операция 5:→INT), образуют т. н. *алгебру замкнутых термов*. Множество T_Σ можно факторизовать по отношению эквивалентности, задаваемому соотношениями равенства. Полученное множество также образует алгебру[i] и называется инициальной Σ-алгеброй. Классы эквивалентности этой алгебры – это бесконечные множества термов. На практике используется конечное приближение классов эквивалентности инициальной Σ-алгебры, в котором каждый класс содержит только конечное множество термов. Такое приближение называется конечной аппроксимацией инициальной Σ-алгебры.

Каждому классу эквивалентности инициальной алгебры обычно дается в соответствие его представитель или «дескриптор». Чаще всего дескриптор представляет собой самый короткий терм данного класса эквивалентности среди построенных или специальное выражение, введенное пользователем. Вычислить терм означает найти дескриптор класса эквивалентности, которому этот терм принадлежит. Вычисление можно производить переписыванием термов, если рассматривать соотношения эквивалентности как задание правил переписывания. Конечно, в этом случае не гарантируется, что дескриптор класса эквивалентности данного терма будет находиться всегда, так как алгоритм переписывания термов не всегда успешно заканчивает свою работу. Тогда в качестве дескриптора можно использовать сам вычисляемый терм.

Нахождение дескриптора терма можно проводить на основе правил переписывания, но можно реализовать отдельную семантику для некоторых операций. Например, для операции *:INT×INT→INT можно ввести семантику простым процессорным вычислением произведения двух чисел. Вычисление производится по древовидному представлению терма путем обхода в глубину слева направо. Таким образом, для терма *(+(3,2),4) сначала будут вычислены термы *(+(3,2) и 4, а затем задействована внешняя процедура, которая вычислит результат сложения 5 и 4, и этот результат будет возвращен в качестве дескриптора терма *(+(3,2),4).

Алгебраический подход к спецификации достаточно хорошо изучен. Например, язык алгебраических спецификаций CASL [7] разработан специально созданной для этой цели группой CoFI [4]. Этот язык используется преимущественно для моделирования программных модулей, но известны работы, посвященные использованию языка CASL для моделирования онтологий [11].

4 Описание синтаксиса языка онтологии шаблонами

Термы представляющей онтологию конечной аппроксимации инициальной Σ-алгебры формируются на основе операций сигнатуры $\Sigma=(S,OP)$. Операции сигнатуры записываются в префиксном виде, но можно ввести инфиксную запись операций. Например, терм операции +:INT×INT→INT вида +(3,5) можно записать в виде 3+5. Каждая такая инфиксная запись представляет собой правило контекстно-свободной порождающей грамматики [6], в левой части которого стоит символ сорта результата операции. Инфиксную запись операции, в которой кроме параметров могут быть строки символов, будем называть *шаблоном*.

Шаблоны позволяют расширять синтаксис языка представления онтологии. Синтаксически шаблон представляет собой строку, в которой могут быть «дырки», которые имеют типы. Также шаблон имеет тип результата. Таким образом, каждый шаблон представляет операцию в сигнатуре Σ-алгебры, представляющей данную онтологию. Добавление синтаксических шаблонов расширяет множество имен операций Σ-алгебры онтологии и позволяет расширить синтаксис языка представления онтологии.

Шаблоны рассматриваются как правила контекстно-свободной грамматики, в которых имена типов выступают как нетерминальные символы, а строки между «дырками» могут рассматриваться как терминалы. Например, инфиксная запись операции +:INT×INT→INT – это правило INT→INT ’+’ INT, где INT – это нетерминальный символ, а ’+’ представляет собой терминал грамматики.

Для разрешения синтаксических неоднозначностей иногда приходится вводить в сигнатуру «синтаксические» типы, которые необходимы только

для того, чтобы управлять процессом синтаксического анализа. Например, можно задать грамматику арифметических выражений следующим образом. Сигнатура Σ=(S,OP) состоит из типов INT, PLUS и MUL с операциями +:MUL×PLUS→PLUS и *:INT×MUL→MUL. Также сделаем тип PLUS подтипом типа INT, а тип MUL – подтипом типа PLUS. Каждое такое отношение тип – подтип генерирует правило вида тип → подтип. Добавим также тип NUM для выделения констант и сделаем его подтипом типа MUL. Таким образом, для данной сигнатуры можно определить следующую грамматику шаблонов:

PLUS → MUL ’+’ PLUS
MUL → NUM ’*’ MUL
INT → PLUS
PLUS → MUL
MUL → NUM

Предположим теперь, что в сигнатуре Σ=(S,OP) заданы константы 1:→NUM, 2:→NUM и 3:→NUM. Тогда выражение 1+2*3 будет преобразовано в терм +(1,*(2,3)).

5 Определение шаблонов в системе ЭЗОП

В системе ЭЗОП [3] реализованы описанные выше принципы моделирования онтологий с помощью алгебраического подхода. ЭЗОП также поддерживает введение новых операций и типов в сигнатуру, представляющую описание мира моделируемой онтологии. Операции вводятся с помощью шаблонов. Рассмотрим примеры описания онтологии в системе ЭЗОП.

Сначала рассмотрим простой пример таксономии помещений учебного учреждения. В этом примере не вводится шаблонов, задаются только типы и их иерархия.

[помещения учебного процесса –] область.

Подобласти помещения учебного процесса:
[учебное помещение],
[служебное помещение].

Подобласти учебное помещение:
[аудитория],
[учебный класс],
[помещение лаборатории].

Подобласти служебное помещение:
[помещение деканата],
[помещение кафедры],
помещение лаборатории.

Подобласти помещение деканата:
[помещение учебной части],
[помещение бухгалтерии].

В ЭЗОП каждый тип является элементом типа «область», верно и обратное. Поэтому выражение вида «**А – область**» задает **А** как тип сигнатуры. В примере выше задаются базовый тип «**помещения учебного процесса**» и различные его подтипы. В ЭЗОП отношение тип – подтип задается не с помощью рассмотренного выше механизма определения решетки типов сигнатуры, а с помощью специально определенной для этих целей операции. Иначе говоря, типы сигнатуры рассматриваются как простое множество без какой-либо алгебраической структуры, а отношение тип – подтип моделируется посредством введения в сигнатуру соответствующей операции. Каждый раз, когда в тексте онтологии встречается выражение «[А] – область», в сигнатуру онтологии добавляется тип А, а также в конечную аппроксимацию инициальной алгебры онтологии добавляется терм вида «**элемент(А,область)**». Операция «**элемент**» используется для моделирования отношения «**элемент области**».

Для задания отношения тип – подтип используется шаблон вида:

Подобласти @A : @Type-list.

Здесь символ **@** используется для выделения имен типов в шаблоне. Вместо типа **Type-list** можно вставлять различные конструкции, задаваемые шаблонами с типом результата **Type-list**. В этом шаблоне не только вводятся подтипы типа А, но также имена типов в списке добавляются в сигнатуру онтологии, если они еще не были объявлены как типы. Чтобы выделить имя такого еще необъявленного типа, используются квадратные скобки.

Другой пример представляет собой простейшую онтологию равномерного движения.

Путь, скорость, время: real.
Путь = скорость*время.
Время = путь/скорость.
Скорость = путь/время.
Введем шаблон «@Тело равномерно движется со скоростью @V»
с переменными: «Тело: new; V: real_выражение»
и переменной результата «x: команда»;
Пояснения: [Вводится объект @Тело, равномерно движущийся со скоростью @V]
Условие применения шаблона:
[]
Действие шаблона:
[x=пустая команда;
тело - объект понятия «равномерное движение»;
тело’s скорость = V].

Первая строка текста онтологии задает переменные **путь**, **скорость** и **время**. В алгебраических спецификациях переменные рассматриваются как константы (операции с пустым списком параметров), которые могут быть использованы в соотношениях равенства. Таким образом, следующие три строки задают соотношения равенства, в которых с левой стороны стоят термы из переменных, а с правой – термы, обозначающие имена арифметических операций.

Следующая синтаксическая конструкция вводит новый шаблон. Введение нового шаблона производится с помощью специального шаблона:

Введем шаблон «текст вводимого шаблона» с переменными «текст объявления переменных» и переменной результата «текст объявления переменной результата»
Условие применения шаблона: ограничения на применение шаблона
Действие шаблона: семантика шаблона

В данном случае вводится шаблон для описания онтологий равномерного движения. В этом шаблоне имеются две «дырки»: **Тело** и **V**. Скорость задается вещественным числом, а **Тело** задает каждый раз при использовании данного шаблона новую константу (тип **Тело** объявлен как **new**). Тип результата шаблона – это команда, что означает выполнение некоторых действий при нахождении дескриптора класса эквивалентности терма, задаваемого шаблоном. Действия задаются в разделе **Действие шаблона**. В нашем случае константа, введенная на месте «дырки» Тело, задается как объект типа «равномерное движение» (онтологии также рассматриваются как типы сигнатуры). В этом случае значения переменных **путь**, **скорость** и **время** будут привязаны к новому объекту.

Онтологию равномерного движения можно использовать в других онтологиях. Рассмотрим, например, учебную задачу на равномерное движение.
Пешеход равномерно движется со скоростью 5.
Пешеход's время =2.
Велосипедист равномерно движется со скоростью 30.
Велосипедист's время = пешеход's время.

Когда в новой онтологии используется шаблон, введенный в другой онтологии, то в новую онтологию загружается та онтология, в которой этот шаблон был определен, и вводятся соотношения, связывающие понятия новой и старой онтологий в соответствии с действием используемого шаблона. В данном случае используется шаблон «**@Тело равномерно движется со скоростью @V**» из онтологии равномерного движения, поэтому подгружаются все понятия и соотношения из этой онтологии. Конструкция вида «**X's время = число**» задает значение времени для объекта X. В системе не реализована морфология, поэтому используется англоязычная конструкция.

Этот пример иллюстрирует одно из преимуществ алгебраического подхода: модульность определений. Использование языковых шаблонов позволяет автоматически загружать онтологии, шаблоны которых используются в данной онтологии.

К онтологии задачи на равномерное движение уже можно задавать вопросы. Например, можно задать вопрос:

Велосипедист's путь?

На что система даст ответ: **60**.

6 Сравнение алгебраического и логического подходов формирования онтологий

Основное отличие алгебраического и логического подходов к формированию онтологий от состоит в том, что типы теорий, которыми представляются онтологии в этих подходах, различны. В алгебраическом подходе используются теории с выделенным отношением равенства.

Теории с выделенным отношением равенства используются в логике и базах данных, где представляют их алгебраические модели. В соответствии с общим алгебраическим подходом онтологии, в которой определяются сигнатура и аксиомы теории, соответствует алгебра, которая строится как множество всех правильно построенных выражений в сигнатуре онтологии, профакторизованное по отношению логической эквивалентности этих выражений, следующей из аксиом-равенств, введенных в онтологии. В логике обычно интересуются только выражениями, принимающими истинностные значения (формулами). Основные вопросы логики – истинна ли данная формула. В базах данных основной вопрос: найти область истинности данной формулы (запроса) в заданном состоянии базы данных.

Базе данных с заданным состоянием соответствует полная теория. Онтологиям в общем случае соответствуют неполные теории, но и при работе с ними пользователю хочется, чтобы система умела отвечать на вопросы, чему равно данное выражение или при каких значения переменных истинна данная формула. Алгебраический подход к онтологиям переносит методы, разработанные в алгебраических подходах к типам данных и базам данных, для моделирования процессов построения ответов на данные вопросы с учетом неполноты теорий, соответствующих онтологиям.

В алгебраическом подходе к спецификациям хорошо разработана теория совместного использования различных модулей спецификаций. Модули естественным образом реализуются как набор определенных типов и операций, который может быть легко загружен и использован в другой спецификации при необходимости. При этом, конечно, необходимо, чтобы используемая спецификация могла быть интегрирована в исходную на основе неких общих определений (языка ядра системы).

В теории алгебраических спецификаций разработан специальный формализм, позволяющий объединять спецификации друг с другом. Англоязычный термин для этого формализма – Institution [9, 10]. Institution можно использовать для интеграции онтологий, которые формируются на основе алгебраического подхода. Теория Institution хорошо разработана и используется во многих системах алгебраических спецификаций. Например, основания языка алгебраических спецификаций CASL [7] определяются с помощью формализма Institution.

Литература

[1] Бениаминов Е.М. Алгебраические методы в теории баз данных и представлении знаний. – М.: Научный мир, 2003.

[2] Лапшин В.А. Онтологии в компьютерных системах. – М.: Научный мир, 2010.
[3] Сайт проекта ЭЗОП. – http://www.ezop-project.ru.
[4] Страница группы CoFI. – http://www.informatik.uni-bremen.de/cofi/wiki/ index.php/CoFI.
[5] Страница описания языка OWL. – http://www.w3.org/TR/owl-features.
[6] Хопкрофт Дж.Э., Мотвани Р., Ульман Дж.Д. Введение в теорию автоматов, языков и вычислений, 2-е изд. – М.: Изд-во «Вильямс», 2002.
[7] Bidoit M., Mosses P. CASL User manual. Introduction to using the common algebraic specification language. – Series: Lecture Notes in Computer Science, 2004. – V. 2900.
[8] Goguen J., Malcolm G. Algebraic semantics of imperative programs. – MIT Press, 1996.
[9] Goguen J. Data, schema, ontology, and logic integration // Proc. CombLog'04 Workshop, Lisbon, Portugal, July 2004.
[10] Goguen J., Burstall R. Institutions: abstract model theory for specification and programming // J. of the ACM. – 1992. – V. 39, No 1.
[11] Luttich K., Mossakowski T. Specification of ontologies in CASL // A. Varzi, L. Vieu eds., Formal Ontology in Information Systems. – Proc. of the Third Int. Conf. (FOIS-2004): IOS Press, Amsterdam, 2004. – V. 114. – P. 140-150.

Specifics of an algebraic approach to ontology description

E.M. Beniaminov, V.A. Lapshin, D.V. Perov

In the text algebraic approach to ontology forming is described. The traditional methods of ontology forming are based on some modifications of predicate calculus' language. The algebraic approach has it own specifics that are discussed in the work. One of such specifics is the possibility to extend dynamically the language of ontology forming by user defined syntax. In the text principles of open language implementation for ontology forming are described. Also, approaches to ontology concepts definition basing on concept extension and precision are discussed. The described approach is implemented in project EZOP (http://www.ezop-project.ru).

♣ Работа выполнена при финансовой поддержке РФФИ (проект 09-07-00079)

[i] Ввиду того, что эквивалентные термы обязательно имеют один и тот же тип, можно сопоставить каждому типу сигнатуры множество классов эквивалентности термов этого типа. Нетрудно показать, что отображение, которое мы определили выше для термов над множеством переменных, работает и здесь и корректно преобразует разные классы эквивалентности друг в друга