УДК 004.8 (575.2)(04)

ОНТОЛОГИИ И АЛГЕБРАИЧЕСКИЕ СПЕЦИФИКАЦИИ

В.А. Лапшин

Приводятся различные аспекты рассмотрения онтологий как алгебраических спецификаций. Алгебраический подход для формирования спецификаций разработан достаточно хорошо, поэтому представляется полезным использовать накопленный опыт для построения онтологий.

Ключевые слова: онтологии; алгебраические спецификации; открытые языки представления спецификаций; CASL; institution.

Введение. Большинство популярных инструментов построения онтологий используют в качестве основного формализма моделирования те или иные логики исчисления предикатов [3]. Так, в языке OWL [3, 6], 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 подход, на основе которого можно формировать онтологии – представление онтологии в виде алгебраической спецификации [9, 11].

Алгебраическая спецификация представляется в виде теории, высказывания которой формируются на основе множества имен типов и имен операций, называемый сигнатурой. Операции выражают в спецификации как отношения между объектами, так и средства построения новых 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 операций без параметров), то множество всех термов данной сигнатуры разбивается на непересекающиеся классы эквивалентности. Полученное множество классов эквивалентности является внутренним (машинным) представлением спецификации и может быть использовано для вычислений.

Каждый класс эквивалентности в описанной выше конструкции может содержать бесконечное множество термов. Оперировать такими множествами на машинном уровне 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.

Имена операций в сигнатуре синтаксически можно рассматривать как правила контекстно-свободной порождающей грамматики Хомского [4]. 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.

Абстрактные алгебраические типы. В алгебраическом подходе для представления он-

тологий используются так называемые многосортные алгебры [1, 8]. Для определения многосортной алгебры сначала задается ее сигнатура как пара Σ=(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 непосредственный дочерний элемент $s_c \in S$, представляющий собой наибольшую нижнюю грань этих элементов.

Обычно используют так называемые 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 Σ=(S,OP) – это семейство A множеств As, заданных для каждого сорта $s \in S$ и называемых носителями алгебры A, а также семейство отображений $\alpha\sigma$:As1×As2×…×Asn →As, заданных для каждого имени операции σ в сигнатуре Σ=(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,…, T_n – это либо переменные соответствующих типов, либо термы, сформированные из имен операций данной сигнатуры тем же способом. Например, пусть +:INT×INT→INT – это операция из сигнатуры Σ=(S,OP) и a, b, c ∈ X – переменные. Тогда выражение +(a,b) образует терм. Вместо параметров можно поставить имя операции “+”, так как 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 σ:→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. Действительно, каждому имени типа $s \in S$ в сигнатуре Σ=(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), образующие так называемую *алгебру замкнутых термов*. Множество T_Σ можно факторизовать по отношению эквивалентности, задаваемому соотношениями равенства. Полученное множество также образует алгебру 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*. Обычно 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.

Вычисление терма для некоторых операций можно также реализовать специальным образом. 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 [6] разработан специально созданной для этой цели группой CoFI [5]. Этот язык используется преимущественно для моделирования программных модулей, но известны работы, посвященные использованию языка CASL для моделирования онтологий [11].

*Описание синтаксиса языка онтологии шаблонами.*Термы представляющей онтологию конечной аппроксимации инициальной Σ-алгебры формируются на основе операций сигнатуры Σ=(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 [4], 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 – это операция 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)).

Заключение. В настоящей работе представлен метод представления онтологий посредством алгебраического подхода. Онтология рассматривается как алгебраическая спецификация, в которой определяются сигнатура и аксиомы алгебраической теории, представляющей данную онтологию. Для онтологии на основе ее алгебраической теории строится алгебра, получаемая как множество всех правильно построенных выражений в сигнатуре онтологии, профакторизованное по отношению эквивалентности, следующей из аксиом-равенств, введенных в онтологии. Эта алгебра может быть использована в качестве машинного представления онтологии, позволяющего производить формальные манипуляции над онтологией: вести диалог с онтологией, передавать машинное представление в последовательной форме, объединять онтологии друг с другом и т.д.

Данный подход реализован в системе ЭЗОП [2] (http://www.ezop-project.ru), представляющей собой Веб-сервер для коллективного построения онтологий. В системе реализовано расширение синтаксиса языка представления знаний с помощью описанного в статье механизма. Онтологии в системе представляются как алгебраические спецификации, на основе которых строится внутреннее представление, как это описано в данной работе. Система предоставляет сервис для осуществления диалога с описываемой онтологией, а также возможность импорта онтологий из других форматов представления онтологии. Дальнейшие перспективы разработки связаны с реализацией расширения языка представления онтологии на лексическом уровне.

Литература

1. *Бениаминов Е.М.* Алгебраические методы в теории баз данных и представлении знаний. М.: Научный мир, 2003.
2. *Бениаминов Е.М., Болдина Д.М.* Система представления и обработки знаний ЭЗОП // Материалы конференции Диалог'20001. Прикладные проблемы, 2001.
3. *Лапшин В.А.* Онтологии в компьютерных системах. М.: Научный мир, 2010.
4. *Лапшин В.А.* Лекции по математической лингвистике. М.: Научный мир, 2010.
5. Страница группы CoFI. http://www.informatik.uni-bremen.de/cofi/wiki/index.php/CoFI.
6. Страница описания языка OWL. http://www.w3.org/TR/owl-features.
7. *Bidoit M., Mosses P.* CASL User Manual. Introduction to Using the Common Algebraic Specification Language. Series: Lecture Notes in Computer Science, Vol. 2900, 2004.
8. *Goguen J., Malcolm G.* Algebraic semantics of imperative programs. MIT Press, 1996.
9. *Goguen J.* Data, Schema, Ontology, and Logic Integration // Proceedings CombLog'04 Workshop, Carnielli W., Dionisio M., Mateus P. Lisbon, Portugal, July 2004.
10. *Goguen J., Burstall R.* Institutions: abstract model theory for specification and programming // Journal of the ACM. 39. 1. 1992.
11. *Luttich K., Mossakowski T.* Specification of Ontologies in CASL // A. Varzi, L. Vieu eds., Formal Ontology in Information Systems – Proceedings of the Third International Conference (FOIS-2004), Vol. 114, pp. 140–150. IOS Press, Amsterdam, 2004.