НАУКОЁМКИЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. Переславль-Залесский, 2009

У. Н. Тихонова

Новый метод определения проблемно-ориентированных языков

Научный руководитель: к. ф.-м. н. Ф. А. Новиков

Аннотация. В статье описан метод определения проблемно-ориентированных языков с помощью интерпретируемых автоматов. Этот метод позволяет формально определить проблемно-ориентированный язык в виде трех составляющих: абстрактного синтаксиса, конкретного синтаксиса и семантики. Программная реализация этого метода, машина автоматного программирования, осуществляет интерпретацию автоматов формального определения языка, и таким образом выполняет разбор и интерпретацию программ на данном языке.

1. Введение

Создание программного обеспечения — это сложный процесс. Наиболее критичной частью этого процесса является непосредственно программирование приложения, так как программистам приходится работать с двумя предметными областями: предметной областью целевой задачи и используемым языком программирования. Программист осуществляет отображение одной предметной области в другую, поэтому от него требуется знание обеих предметных областей. Программирование в смысле такого отображения может быть упрощено с помощью проблемно-ориентированных языков (domain-specific languages, DSL). Проблемно-ориентированные языки позволяют решать целевую задачу в терминах этой задачи, а не в терминах вычислительной машины [1] и в идеале могут использоваться непосредственно специалистами в данной области. Такой подход облегчает разработку программ и повышает их качество.

Чтобы достичь всех преимуществ использования проблемно-ориентированных языков, требуется языковый инструментарий, позволяющий легко создавать и модифицировать проблемно-ориентированные языки и инструменты для работы с ними, а также предоставляющий возможность совмещать в разработке программы использование нескольких языков. Ключевым моментом при работе с языковым инструментарием является легкость его использования, так как главной задачей является практическая простота.

Создание языкового инструментария — это достаточно широкая тема. Конкретная задача, решаемая в данной работе, — это исследование метода определения проблемно-ориентированных языков и рассмотрение возможных языковых инструментов, которые могут быть получены при этом автоматически. В дальнейшем на основе данного подхода может быть разработан языковый инструментарий.

2. Постановка задачи

Проблемно-ориентированный язык, как и язык программирования общего назначения, является средством решения задач. Он определяется синтаксисом и семантикой. Далее будем подразделять синтаксис на абстрактный и конкретный. Абстрактный синтаксис описывает класс допустимых языком программ, в смысле абстрактного описания решения некоторых задач на уровне понятий предметной области и их взаимосвязей. Конкретный синтаксис определяет представление для этого класса программ, в смысле конкретного описания решения некоторых задач, например, в виде текста, диаграмм или сценариев работы в графическом интерфейсе. Семантика определяет отображение программы в вычислительную модель. Следовательно, описание языка программирования должно включать в себя определение всех этих трех составляющих. А интерпретация описания проблемно-ориентированного языка позволит использовать теоретическую спецификацию языка как его практическую реализацию. Таким образом, необходимо решить следующие задачи:

- определить методы для описания абстрактного синтаксиса, конкретного синтаксиса и семантики проблемно-ориентированного языка;
- проанализировать выразительные средства этих методов;
- выполнить программную реализацию метода, которая позволит интерпретировать описание проблемно-ориентированного языка.

3. Методы исследования

Обычно на практике для определения языка используются формальные грамматики, то есть язык описывается в виде синтаксиса, а семантика описывается неформально и скрывается в реализующий язык транслятор. При этом значимое с точки зрения авторов разделение синтаксиса на абстрактный и конкретный не проводится. Такой подход влечет немало неудобств, как для использования языка, так и для его поддержки.

Одной из технологий, разработанных с целью устранить этот недостаток, является языковый инструментарий Meta Programming System (MPS) [2]. В его основе лежит постулат о том, что программа на языке предметной области — это любое точно определенное решение некоторой задачи, а не набор инструкций для компьютера. Общепринятое в практике программирования текстовое представление программы является лишь одним из множества представлений этого решения, и далеко не самым удобным.

Другим методом определения языков программирования, включающим в себя не только формальное описание конкретного синтаксиса языка, но и формальное определение его абстрактного синтаксиса и семантики, является Венский метод [3]. Именно в этом методе впервые было проведено разделение синтаксиса на абстрактный и конкретный и показана важность абстрактного синтаксиса для определения семантики. Для описания абстрактного синтаксиса в Венском методе используется абстрактная грамматика, а для определения семантики языка — интерпретирующие автоматы.

На данный момент автоматы широко применяются как для описания и проектирования алгоритмов, так и для их реализации [4]. Программирование с использованием автоматов, или автоматное программирование, основано на работе виртуальной машины, интерпретирующей автоматы. Согласно этому подходу, определение семантики языка с помощью автоматов является одновременно интерпретатором данного языка. Кроме того, автоматное программирование позволяет рассматривать определение конкретного синтаксиса языка в виде диаграммы состояний и переходов как синтаксический анализатор языка [5].

В данной работе рассматривается метод определения проблемно-ориентированных языков с помощью системы автоматов, интерпретируемых виртуальной машиной автоматного программирования.

Системы интерпретируемых автоматов описывают операционную семантику и конкретный синтаксис языка. Для описания структуры языка, его абстрактного синтаксиса, используется техника моделирования, предлагаемая стандартом OMG [6] — метамоделирование. Метамоделирование активно применяется при разработке объектно-ориентированных технологий и стандартов. Одним из примеров таких разработок является унифицированный язык моделирования UML [7]. Определение абстрактного синтаксиса (метамодели) в виде диаграммы классов в нотации UML позволяет использовать все концепции объектно-ориентированного моделирования, создавать гибкие и переносимые метамодели.

Метамодель проблемно-ориентированного языка определяет представление конкретной программы, которое используют автоматы. Семантика языка задается системой автоматов, которые реализуют интерпретацию программы как экземпляра метамодели. Конкретный синтаксис задается с помощью системы автоматов, реализующих разбор программы. При этом прототипом такой системы автоматов является структура языка, в том смысле, что предложены методы сведения метамодели языка к системе автоматов. Автоматы интерпретируются виртуальной автоматной машиной, поэтому такая спецификация языка и есть программа, его реализующая. Полученный таким образом языковый процессор не использует грамматического описания языка, а основан на связывании распознающих и интерпретирующих автоматов непосредственно с абстрактной структурой языка.

4. Результаты

Предлагаемый метод опробован на примере описания спецификации языка СЛОН (СЛежение и Обработка Наблюдений) [8] системы ЭРА (Эфемеридные Расчеты Астрономии) [9] — прикладной проблемно-ориентированной системы программирования, предназначенной для решения разнообразных задач астрономии. Ниже рассмотрены основные идеи предлагаемого метода и примеры его применения для описания языка СЛОН, а также некоторые аспекты программной реализации.

4.1. Определение абстрактного синтаксиса с помощью диаграммы классов

Для описания абстрактного синтаксиса с помощью диаграмм классов в предлагаемом методе определения проблемно-ориентированных языков используются следующие конструкции UML:

- классы — для представления понятий определяемого языка;
- атрибуты классов — для представления свойств понятий;
- обобщение — для классификации понятий;
- композиция и (реже) ассоциация — для представления отношений между понятиями.

Аналогично тому, как формальная грамматика некоторого языка определяет синтаксическую структуру программы на этом языке, модель абстрактного синтаксиса (метамодель) определяет абстрактную структуру программы и составляющих ее частей. При этом выразительные средства диаграммы классов UML могут заменить все выразительные средства формальных грамматик следующим образом.

(1) Множество нетерминалов заменяется множеством классов.
(2) Множество терминалов разделяется на «семантически значимые» токены (например, имена и значения переменных, знаки операций) и «разделители» (скобки, запятые и т.д.). Семантически значимые токены заменяются значениями атрибутов соответствующих классов (например, идентификатор переменной — это атрибут класса Переменная), а разделители в метамодели не указываются.
(3) Множество правил заменяется отношениями между классами.
(4) В качестве начального нетерминала (аксиомы грамматики) выступает класс, определяющий абстрактную программу.
(5) Отношение выводимости заменяется отношением конкретизации (будем говорить, что объект a и класс A связаны отношением конкретизации, если объект a является экземпляром класса A).

Таким образом, выразительные средства диаграммы классов UML не слабее выразительных средств контекстно-свободных формальных грамматик. На рис. 1 приведен фрагмент метамодели языка СЛОН и

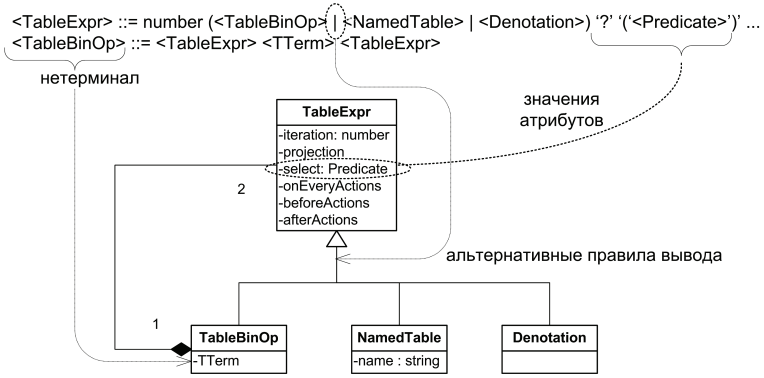

Рис. 1. Фрагмент грамматики и метамодели языка СЛОН: табличное выражение

указаны соответствия между выразительными средствами формальной грамматики и выразительными средствами диаграммы классов.

Важно заметить, что терминалы: „?“, „(“, „)“, — присутствующие в правиле формальной грамматики, не нашли своего выражения в метамодели, потому что они не являются семантически значимыми.

4.2. Определение конкретного синтаксиса с помощью распознающих автоматов

В классической теории трансляции для определения конкретного синтаксиса языка используются формальные грамматики. Общеизвестно, что леворекурсивные (автоматные) грамматики эквивалентны конечным автоматам–распознавателям и регулярным выражениям. Однако использование автоматов для грамматического описания не ограничиваются только леворекурсивными грамматиками. Действительно, одним из способов представления грамматических правил вывода являются диаграммы переходов (синтаксические диаграммы Вирта [10]). Если исключить в них появление нетерминалов на переходах и допустить произвольные эффекты на переходах, то можно рассматривать систему таких диаграмм как автоматную программу, которая является синтаксическим анализатором языка. Этот же подход используется авторами статьи [5] для создания системы

автоматического завершения ввода на основе парадигмы автоматного программирования. В предлагаемом методе для описания конкретного синтаксиса проблемно-ориентированного языка используется система конечных автоматов, определенных следующим образом.

(1) Входным алфавитом системы автоматов является множество терминалов языка.
(2) Для описания взаимодействия автоматов используется нотация диаграммы состояний UML: составное состояние — это переход к соответствующему автомату.

При этом под терминалом понимается элемент нотации языка. С практической точки зрения терминал — это событие, посылаемое лексическим анализатором текста или графическим редактором, или диалоговым окном — любым источником событий. Такой подход позволяет использовать для представления программы текст, или диаграмму, или последовательность нажатий кнопок в графическом интерфейсе. Прототипы распознающих автоматных программ могут быть получены из метамодели языка с помощью следующих правил сведения.

(1) Автомат–распознаватель определяется для каждой сущности (каждого класса) метамодели.
(2) Отношение обобщения сводится к ветвлению в автомате (набору альтернативных сторожевых условий).
(3) Отношение ассоциации сводится к циклу или к последовательности составных состояний в автомате в зависимости от кратности полюсов ассоциации.

Полученные прототипы автоматных программ преобразуются в распознающие конкретный синтаксис автоматные программы путем внесения необходимых изменений в структуру прототипов и задания элементов нотации (терминалов) языка на переходах. На рис. 2 приведен автомат, распознающий фрагмент языка СЛОН — табличные выражения. Соответствующий фрагмент метамодели языка приведен на рис. 1.

Определенная таким образом спецификация конкретного синтаксиса языка одновременно реализует задачу распознавания для языка (так как есть виртуальная машина, интерпретирующая автоматы спецификации).

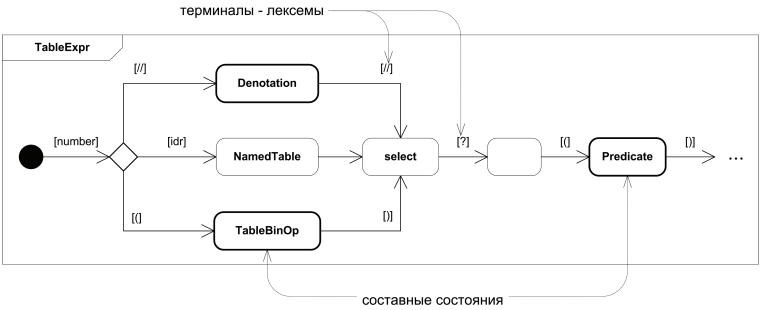

Рис. 2. Автомат, распознающий табличные выражения языка СЛОН

4.3. Определение семантики с помощью интерпретирующих автоматов

Операционный подход к определению семантики языка предполагает описание алгоритма интерпретации программы в терминах некоторой абстрактной машины. Будем строить алгоритм интерпретации программы на проблемно-ориентированном языке в виде системы конечных автоматов. При этом интерпретируемая программа (вход алгоритма) — это дерево, являющееся экземпляром абстрактного синтаксиса (метамодели) языка. Для описания алгоритма выполнения проблемно-ориентированной программы в предлагаемом методе используется следующая модель (детерминированного конечного) автомата.

(1) Автомат выполняется, используя входные события и предикаты.
(2) Результатом выполнения автомата являются некоторые эффекты (выходные воздействия) и точки выхода (exit points).
(3) Для взаимодействия между автоматами используются составные состояния и точки выхода.

Согласно парадигме автоматного программирования, автомат управляет некоторым объектом управления. Здесь объектом управления является интерпретируемая программа — экземпляр метамодели проблемно-ориентированного языка. Таким образом, предикаты вычисляются, и эффекты выполняются для экземпляра метамодели. Точка выхода — это состояние, в котором может завершить свое выполнение вложенный в составное состояние автомат. По сути, точка

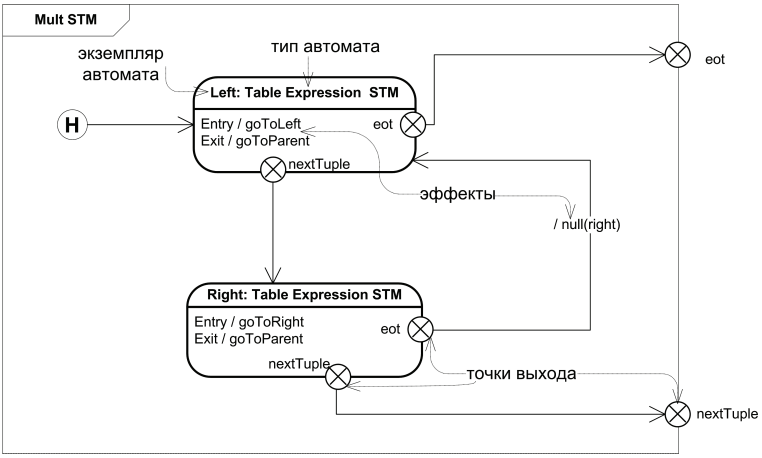

Рис. 3. Семантика операции умножения таблиц

выхода автомата — это возвращаемое им значение. Именно с помощью такой модели системы взаимодействующих автоматов удалось описать семантику наиболее сложного фрагмента проблемно-ориентированного языка СЛОН — табличных выражений. Язык СЛОН реализует таблично ориентированное программирование [9], которое рассматривает таблицу с одной стороны как массив данных, с другой стороны как программу, определяющую последовательность присваивания значений переменным — величинам предметной области. Как массив данных таблица может храниться во (внешней) памяти. Как программа таблица — это итератор, перебирающий кортежи таблицы, чтобы выполнять с ними некоторые действия. Табличные выражения реализуют механизм алгебры таблиц, позволяющий строить из имеющихся таблиц произвольные таблицы. Например, операция умножения таблиц (в метамодели языка она представлена классом бинарных табличных операций TableBinOp, см. рис. 1) позволяет последовательно объединить все кортежи двух таблицы. Если рассматривать таблицу как программу (или как цикл), то умножению таблиц соответствует вложенность циклов (рис. 3).

Кроме того, для описания семантики табличных операций языка СЛОН используется следующая модель выполнения (интерпретации) системы автоматов:

- структура автомата (его состояния и переходы между ними) определяет класс автоматов;
- при выполнении (интерпретации) автомата создается соответствующий экземпляр класса автоматов;
- в процессе выполнения алгоритма, описанного с помощью системы интерпретирующих автоматов, может создаваться столько экземпляров автоматов, сколько потребуется.

Такая модель выполнения системы автоматов позволяет использовать рекуррентное определение семантики с сохранением локальных данных каждого автомата (смыслового элемента этого определения).

4.4. Программная реализация метода с помощью машины автоматного программирования

Виртуальная машина автоматного программирования позволяет рассматривать определение проблемно-ориентированного языка как реализующую его программу. Интерпретация распознающих автоматов, описывающих конкретный синтаксис, решает задачу распознавания для языка. Интерпретация интерпретирующих автоматов, определяющих семантику, реализует интерпретатор языка. Машина автоматного программирования реализована с помощью применения метода раскрутки (bootstrapping) к предлагаемому методу: автоматное программирование — это проблемно-ориентированный язык. Абстрактный синтаксис этого языка определен в виде метамодели автоматной программы. Семантика выполнения автоматной программы описана в виде автоматной программы (рис. 4). Конкретный синтаксис автоматной программы задан с помощью автоматной программы, распознающей конкретное представление автоматной программы.

Реализованная согласно этой спецификации машина автоматного программирования была применена для интерпретации этой же спецификации, что доказывает действенность автоматного подхода для определения проблемно-ориентированных языков.

5. Выводы

С помощью рассмотренного метода определения проблемно-ориентированных языков удалось описать наиболее сложный с точки зрения семантики фрагмент языка СЛОН: табличные выражения. Машина автоматного программирования позволяет получить языковый процессор этого фрагмента практически без дополнительных трудозатрат.

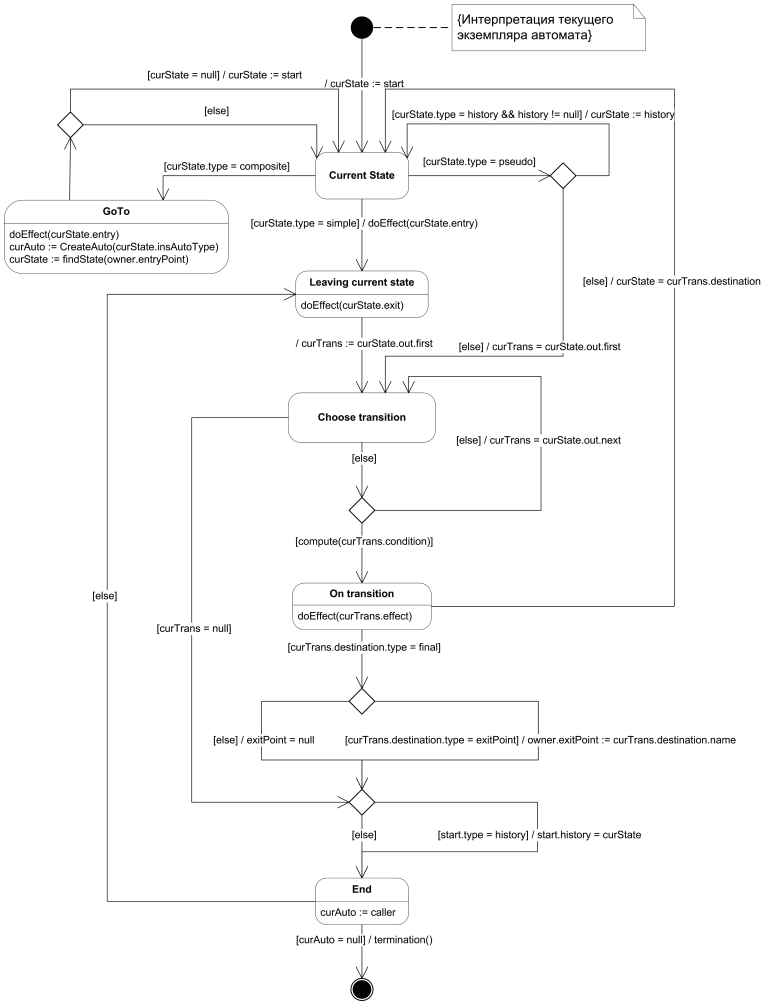

Рис. 4. Алгоритм интерпретации (семантика) автоматной программы

В дальнейшем планируется расширить машину автоматного программирования событийной моделью и поддержкой параллельных процессов, что в том числе позволит создавать нетекстовые редакторы (описания конкретного синтаксиса) проблемно-ориентированных

языков. Кроме того, планируется провести анализ выразительных средств рассмотренного метода в сравнении с более широким (чем контекстно-свободные грамматики) классом формальных грамматик и с различными методами определения семантики языка. Интерпретация описания языка в дальнейшем может быть дополнена кодогенерацией, компиляцией и возможностью корректно обрабатывать синтаксически неправильные программы.

Список литературы

[1] Martin Fowler Language Workbenches: The Killer-App for Domain Specific Languages?, June 2005, http://martinfowler.com/articles.html. ↑1

[2] Sergey Dmitriev Language Oriented Programming: The Next Programming Paradigm, February 2005, http://www.onboard.jetbrains.com. ↑3

[3] Оллонгрен А. Определение языков программирования интерпретирующими автоматами. — М.: Мир, 1977. — 288 с. ↑3

[4] Шалыто А. А. Сайт по автоматному программированию и мотивации к творчеству. — СПбГУ ИТМО Кафедра «Технологии программирования», http://is.ifmo.ru. ↑3

[5] Гуров В. С., Мазин М. А., Шалыто А. А. Автоматическое завершение ввода условий в диаграммах состояний, 2008, Режим доступа к статье: http://is.ifmo.ru/works/_2008-02-28_auto_stop.pdf. ↑3, 4.2

[6] The Object Management Group, http://www.omg.org. ↑3

[7] Буч Г., Якобсон А., Рамбо Дж. UML. — 2-е изд. — СПб.: Питер, 2006. — 736 с. ↑3

[8] Krasinsky G. A., Novikov F. A., Skripnichenko V. I. Problem Oriented Language for Ephemeris Astronomy and its Realization in System ERA. — Vol. 45: Cel. Mech., 1989. — 219-229 с. ↑4

[9] Новиков Ф. А. Архитектура системы ЭРА - табличный подход к обработке данных. — Л.: ИПА РАН, 1990. ↑4, 4.3

[10] Йенсен К., Вирт Н. Паскаль. — М.: Финансы и статистика, 1982. ↑4.2

U. N. Tikhonova. *The New Method of Definition of Domain-Specific Languages* // Proceedings of Junior research and development conference of Ailamazyan Pereslavl university. — Pereslavl, 2009. — p. 183–194. (*in Russian*).

Abstract. The method of domain-specific languages definition by interpreted automata is described. This method enables one to define domain-specific language in the form of three parts: abstract syntax, concrete syntax and semantics. The program implementation of this method, virtual machine of automata-based programming, interprets automata of language specification and realizes interpretation and parsing of programs in the language thereby.