УДК 004.434

АВТОМАТНЫЙ МЕТОД ОПРЕДЕЛЕНИЯ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ (Часть 1)

Ф. А. Новиков,
канд. физ.-мат. наук, заведующий лабораторией астрономического программирования
Институт прикладной астрономии РАН
У. Н. Тихонова,
аспирант
Санкт-Петербургский государственный политехнический университет

Описывается новый метод определения синтаксиса и семантики проблемно-ориентированных языков с помощью диаграмм классов и диаграмм автоматов.

Ключевые слова *— проблемно-ориентированный язык, абстрактный синтаксис, метамодель, автоматное программирование.*

Введение

Создание прикладного программного обеспечения — это сложный процесс. Наиболее критичной его частью является непосредственно программирование приложения, так как на этом этапе осуществляется мысленное отображение понятий прикладной предметной области в используемый язык программирования, что требует знания как языка предметной области, так и языка программирования. Прикладное программирование может быть упрощено с помощью проблемно-ориентированных языков (языки предметной области, domain-specific languages — DSL), которые позволяют формулировать решение целевой задачи в терминах предметной области этой задачи, а не в терминах вычислительной машины. Такой подход облегчает разработку программ и повышает их качество.

Парадигма прикладного программирования, основанная на использовании проблемно-ориентированных языков, известна уже давно [1]. Основным препятствием для ее широкого применения является сложность создания новых проблемно-ориентированных языков и всего сопутствующего им программного обеспечения. Одной из технологий, разработанных в последнее время с целью устранить этот недостаток, является языковый инструментарий Meta Programming System (MPS) [2]. В его основе лежит постулат о том, что программа на проблемно-ориентированном языке — это любое точно определенное решение некоторой задачи, а не набор инструкций для компьютера. Программа, как абстрактное решение задачи, может иметь различные представления. Общепринятое в практике программирования текстовое представление является лишь одним из множества возможных представлений такого решения. Причем далеко не всегда текстовое представление является самым удобным.

Отделение абстрактного решения задачи от реализующей его программы возможно при разделении синтаксиса языка программирования на абстрактный и конкретный. Абстрактный синтаксис описывает структуру допустимых языком решений (класс абстрактных программ). Конкретный синтаксис определяет представление программ этого класса. Семантика языка определяет способ выполнения абстрактной программы, т. е. ее отображение в целевую вычислительную модель.

Первым методом определения языков программирования, в котором последовательно проведено разделение абстрактного и конкретного синтаксисов, является Венский метод [3]. Именно в Венском методе показано, что при определении семантики языка необходимо и достаточно опираться на абстрактный синтаксис языка. Кроме того, в Венском методе предложен оригинальный подход к определению семантики языка, основанный на интерпретирующих автоматах.

Традиционным способом описания синтаксиса языка являются формальные грамматики. Правила формальной грамматики некоторого

языка могут быть сведены к автомату, распознающему этот язык [4]. Этот подход позволяет определить конкретный синтаксис, в том числе и нетекстовый, с помощью системы автоматов [5].

В данной статье предлагается автоматный метод определения проблемно-ориентированных языков, отталкивающийся от перечисленных идей и состоящий в совместном определении абстрактного синтаксиса, конкретного синтаксиса и операционной семантики.

Статья состоит из пяти разделов. В первом определяются назначение и область применения предлагаемого метода. Во втором и третьем разделах описывается способ определения структурных составляющих проблемно-ориентированных языков. В четвертом разделе описано применение автоматного программирования для определения конкретного синтаксиса. В пятом разделе показано, как определить операционную семантику системой взаимодействующих автоматов.

Назначение и область применения метода

Чтобы достичь практических преимуществ решения прикладных задач с помощью проблемно-ориентированных языков, необходимо облегчить создание новых языков, модификацию уже существующих, поддержку и развитие диалектов. В предлагаемом методе эта возможность достигается за счет полного формального определения языка, исчерпывающим образом определяющего интерпретацию как вычислительной машиной (исполнителем языка), так и человеком (пользователем языка), что также немаловажно для распространения и использования нового языка.

Автоматный метод определения проблемно-ориентированных языков позволяет задать язык в виде трех составляющих: метамодели языка (включающей абстрактный синтаксис), конкретного синтаксиса (или набора конкретных синтаксисов) и операционной семантики. При этом мы используем в качестве средства описания всех трех составляющих стандартный унифицированный язык моделирования **UML [6]: абстрактный синтаксис и метамодель языка описываются с помощью диаграммы классов, конкретный синтаксис и семантика — с помощью специализированных диаграмм автомата. Новые языки можно определять, используя уже разработанные модели. Таким образом, предлагаемый метод поощряет повторное использование моделей.**

Мы разделяем взгляды С. Дмитриева [2] и М. Фаулера [1] на понятие программы: программой является абстрактная структура, а для ее редактирования, хранения и выполнения могут использоваться различные представления — в виде текста, диаграмм, таблиц, формул, звуков и др. При таком подходе любую прикладную программу можно рассматривать как языковой процессор, обрабатывающий свои входные данные как предложения некоторого специального входного языка. Именно поэтому центральное место в нашем методе определения языков как классов программ занимает абстрактный синтаксис в составе метамодели языка.

Охарактеризуем более точно область применения метода. Мы рассматриваем язык, в том числе проблемно-ориентированный, с самой общей точки зрения — как знаковую систему для передачи информации. Знаки могут иметь различное представление: буквы некоторого алфавита, или графические фигуры на диаграмме, или элементы управления (кнопки) на форме графического интерфейса, или звуковые сигналы, или что угодно. Более того, мы считаем, что один знак может иметь несколько различных представлений, и даже в рамках одного языка. Множество знаков предполагается конечным и заранее заданным так, что знаки однозначно различимы в любом представлении. Таким образом, в автоматном методе предполагается зафиксированным конечный алфавит *имен* абстрактных знаков.

Зафиксировав природу знаков, необходимо определить, каковы рассматриваемые знаковые системы, т. е. как можно комбинировать абстрактные знаки или, другими словами, какие абстрактные структуры допустимы в определяемых языках. Классический подход, ориентированный на формальные языки, представленные текстами, или на естественные языки, представленные живой речью, фактически рассматривает один тип структур — **конечные последовательности** знаков (линейно упорядоченные мультимножества). Это, разумеется, самый распространенный тип структур, более того, конечные структуры любой другой природы можно представить (закодировать) последовательностями. Однако такое «вытягивание» нелинейной структуры в линейное представление может быть не вполне удобным в некоторых случаях, навязывая искусственный линейный порядок, которого в абстрактной структуре на самом деле нет, и маскируя нелинейную структуру в линейном коде. В автоматном методе мы допускаем не только линейные структуры, но и ориентированные деревья, и (более общий случай) ориентированные графы с одним источником. Таким образом, предлагаемый метод имеет весьма широкую область применения.

Автоматный метод состоит из четырех шагов:

1) определение абстрактного синтаксиса как иерархической композиции конструкций языка;

2) определение метамодели как абстрактного синтаксиса, дополненного системой неиерархических отношений между конструкциями языка;

3) определение конкретного синтаксиса как распознавателя, конструирующего абстрактную программу по ее представлению;

4) определение операционной семантики как интерпретатора абстрактных программ.

Следует подчеркнуть, что автоматный метод определения проблемно-ориентированных языков позволяет использовать такое формальное определение языка как его программную реализацию.

Определение абстрактного синтаксиса диаграммой классов

В предлагаемом автоматном методе определения языков в центр ставится описание абстрактной структуры предложений языка, которое обычно называют абстрактным синтаксисом языка. При использовании классического метода определения языков с помощью порождающих формальных грамматик абстрактный синтаксис обычно не описывается отдельно, но извлекается из правил вывода формальной грамматики. А именно, структурные единицы обозначаются нетерминальными символами и определяются правилами вывода. В результате применения правил вывода происходит последовательное уточнение, разворачивание определения начальной структурной единицы языка — его аксиомы. Различные последовательности таких уточнений порождают различные предложения (строго говоря, деревья вывода). Предложение на языке является частным случаем применения синтаксических правил языка, а значит экземпляром структуры, определяемой правилами абстрактного синтаксиса. При этом экземпляр структуры содержит только абстрактные знаки, тогда как в предложении на языке присутствуют еще и терминальные символы, используемые для обозначения структуры (скобки, запятые и т. д.). Очевидно, что в абстрактном синтаксисе, описывающем структуру предложений языка, эти терминальные символы не определяются. Они входят в определение конкретного синтаксиса языка [3].

Заметим, что в классическом подходе все предложения языка являются экземплярами структуры, но не все экземпляры структуры являются предложениями языка. Дело в том, что язык определяется не только правилами абстрактного синтаксиса, но и другими правилами, в частности, контекстными условиями, задание которых вызывает известные трудности при использовании формальных грамматик. В автоматном методе контекстные условия задаются неиерархическими отношениями метамодели, рассматриваемыми далее.

В предлагаемом автоматном методе для описания абстрактного синтаксиса правила порождения не используются. Мы описываем структуру языка явным образом, а именно абстрактный синтаксис языка определяем с помощью диаграмм классов (**A, B и C на рис. 1 обозначают некоторые** конструкции, в частности абстрактные знаки).

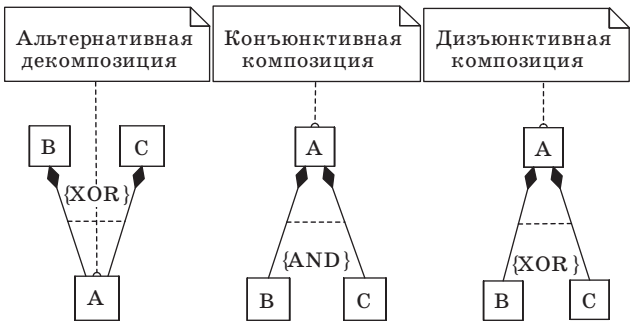
• Абстрактные знаки языка и их комбинации — конструкции языка — описываются в виде классов. При использовании порождающих формальных грамматик им соответствуют терминалы и нетерминалы.

• Определение конструкции языка через ее составляющие производится с помощью отношения композиции. При этом мы понимаем под композицией классов отношение типа «часть-целое» с наложенным на него ограничением *альтернативной декомпозиции*: в каждом экземпляре отношения композиции экземпляр конструкции A включается либо в экземпляр конструкции B, либо в экземпляр конструкции **C, но не в оба вместе** (см. рис. 1, слева). На языке порождающих формальных грамматик альтернативная декомпозиция соответствует паре правил B ::= A и **C ::= A, где в каждом конкретном случае порождение** может пойти только по одному пути.

• Обязательные составляющие некоторой конструкции, включенные в нее по отношению композиции, при реализации в совокупности образуют одно целое. Такую композицию мы называем *конъюнктивной*: в конструкцию A включаются и конструкция B, и конструкция C (см. рис. 1, в центре). На языке порождающих формальных грамматик конъюнктивной композиции соответствует правило **A ::= BC. Заметим, что конъюнктивная** композиция в нашем варианте задания абстрактного синтаксиса, в отличие от задания правилами грамматики, не предписывает упорядоченности составных частей.

• Альтернативное определение конструкции языка может осуществляться с помощью *дизъюнктивной композиции*: в конструкцию **A включается** конструкция B или конструкция C, но не обе вместе (см. рис. 1, справа). На языке порождающих формальных грамматик это соответствует правилу A ::= B | C.

На применение введенных правил композиции накладываются следующие ограничения. Во-первых, абстрактные знаки являются атомарными, они не могут быть результатами композиции, только аргументами. Во-вторых, полученная структура не должна содержать бесконечной рекурсии по отношению композиции. То есть во всяком цикле по композиции должно присут-

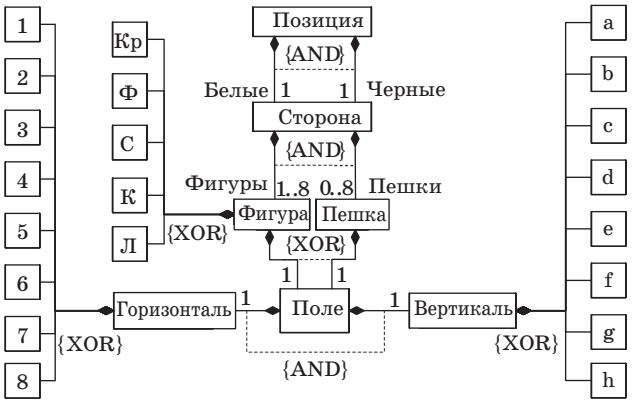


■ ***Рис. 1.*** *Правила композиции*

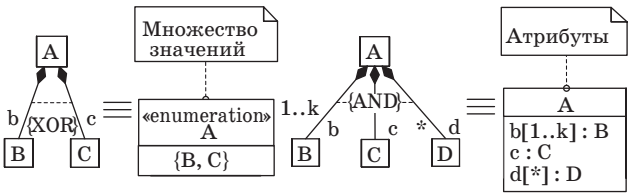
ствовать, по крайней мере, одно вхождение дизъюнктивной композиции, позволяющее завершить рекурсию [соответствующий пример см. ниже, (рис. 7)]. **Нетрудно показать, что введенные правила** позволяют описывать языки, реализациями (экземплярами) абстрактного синтаксиса которых являются корневые деревья.

Кроме перечисленных выше элементов диаграммы классов **UML, мы используем свойства** отношения композиции: кратность, роль и ограничения полюса [6]. Эти выразительные средства диаграммы классов имеют аналоги в известных методах определения абстрактного синтаксиса языка. Роль полюса отношения композиции соответствует селектору [3, 7]. С помощью кратности полюса можно определять необязательные элементы отношений (например, с помощью кратности вида 0..1) и так называемые серийные компоненты [7]. Ограничения позволяют наложить некоторые контекстные условия.

Таких сравнительно ограниченных средств оказывается достаточно для описания абстрактных синтаксисов самых разнообразных проблемно-ориентированных языков. Например, нелинейный язык описания шахматных позиций может быть описан структурой, представленной на рис. 2.



■ ***Рис. 2.*** *Абстрактный синтаксис языка описания шахматных позиций*



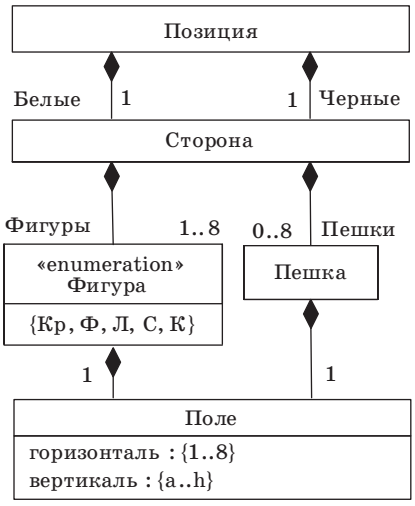
■ ***Рис. 3.*** *Использование атрибутов и множеств значений*

Чтобы сделать нотацию более лаконичной и выразительной, мы используем средства диаграмм классов, соответствующие атрибутам и множествам значений (перечислимым типам).

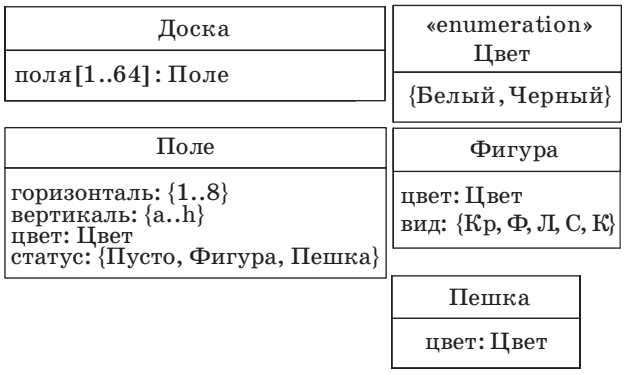
• *Множество значений* соответствует дизъюнктивной композиции (рис. 3, слева). Для описания множеств значений используем стандартный стереотип «enumeration» [6] и альтернативные текстовые обозначения перечислением {B, C} или указанием отрезка {**B..C}, если имена знаков** естественно упорядочены.

• *Атрибутом класса* является конструкция, включаемая в него по конъюнктивной композиции (см. рис. 3, справа). При этом если указано имя роли, то оно переходит в имя атрибута. А если указана кратность полюса, то атрибут является массивом. Важно подчеркнуть, что мы допускаем в качестве кратности неопределенное число, обозначаемое, по правилам UML, символом *, что позволяет описывать итеративные конструкции явно, не прибегая к рекурсии, как это приходится делать в порождающих грамматиках.

Применяя эти соглашения к некоторым композиционным отношениям абстрактного синтаксиса языка описания шахматных позиций (см. рис. 2), получим структуру, представленную на рис. 4. Заметим, что при этом уже нет нужды пи-



■ ***Рис. 4.*** *Сокращенный абстрактный синтаксис языка описания шахматных позиций*



■ ***Рис. 5.*** *Альтернативный абстрактный синтаксис языка описания шахматных позиций*

сать явно ограничения {XOR} и {AND}, поскольку они однозначно восстанавливаются из контекста.

Таким образом, в автоматном методе абстрактный синтаксис описывается в виде множества (композиционных) комбинаций абстрактных знаков языка, т. е. в виде множества понятий языка. При этом используются следующие конструкции диаграмм классов:

1) классы, возможно с атрибутами и стереотипами, для обозначения понятий языка;

2) отношение конъюнктивной композиции или атрибуты классов для определения понятий языка через свои составляющие;

3) отношение дизъюнктивной композиции или перечислимые типы для альтернативных определений понятий языка;

4) кратность полюса отношения композиции или массив, в том числе и с неопределенным количеством элементов, для повторяющихся конструкций языка.

Немаловажно то, что один язык можно описать многими разными способами. Например, абстрактный синтаксис на рис. 2 вдохновлен традиционной «человекочитаемой» шахматной нотацией. Для компьютерного использования, возможно, была бы удобнее структура, показанная на рис. 5.

Заметим, что хотя на рис. 5 передан абстрактный синтаксис, т. е. набор отношений композиции, обозначения композиций в виде стрелок с закрашенными ромбиками отсутствуют — вместо них применяются атрибуты и перечислимые типы.

Определение отношений метамодели языка

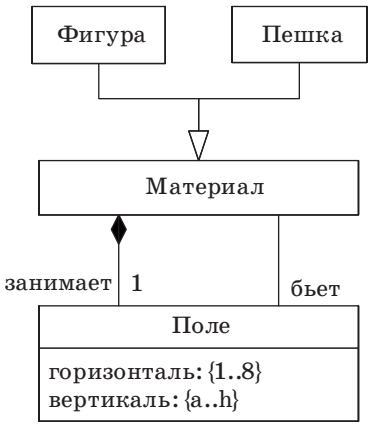
Абстрактный синтаксис языка определяет строго иерархическую композицию понятий языка. Поэтому абстрактная программа как единица языка, т. е. как экземпляр структуры, предписанной абстрактным синтаксисом, является ориентированным деревом. На практике же программа обычно является ориентированным графом: наряду с отношениями иерархической композиции используются и другие отношения. Например перекрестные ссылки от выражений, использующих идентификаторы, к определяющим вхождениям этих идентификаторов. Неиерархические, ссылочные отношения важны с семантической точки зрения и известны в литературе как статическая семантика, или семантика отношений [7]. В автоматном методе для их описания мы используем ассоциации на диаграмме классов [6].

Не менее важной, чем композиционная иерархия, является классификация понятий языка, т. е. определение отношений вида «А есть В». Для выражения отношения классификации понятий языка как нельзя лучше подходит отношение обобщения на диаграмме классов [6]. Так же, как дизъюнктивная композиция, обобщение может использоваться для описания альтернативного определения понятий языка (общее понятие определяется через свои частные случаи). Однако между этими двумя способами альтернативного определения есть два важных отличия. В случае использования обобщения подклассы наследуют все свойства суперкласса (в том числе участие в отношениях композиции и ассоциации). В случае использования дизъюнктивной композиции этого не происходит. С другой стороны, дизъюнктивный композит обеспечивает доступ ко всем своим непосредственным составляющим, а суперкласс не предоставляет информацию о своих подклассах.

Абстрактный синтаксис, дополненный отношениями ассоциации, обобщения и ограничениями, называется *метамоделью*. Таким образом, программа как частный случай применения абстрактного синтаксиса языка и указания отношений между используемыми понятиями, т. е. как экземпляр метамодели, является ориентированным графом с единственным источником — экземпляром аксиомы языка.

Например, в языке описания шахматных позиций можно определить отношения вида «фигура бьет поле» и «пешка бьет поле» вдобавок к уже определенному в абстрактном синтаксисе отношению «фигура или пешка занимает поле». Эти отношения являются общими для классов Фигура и Пешка, поэтому их можно обобщить с помощью класса Материал (рис. 6).

Выше мы упоминали, что абстрактный синтаксис языка может быть получен из правил вывода формальной грамматики. Определим более формально правила преобразования формальной грамматики языка в описание абстрактного синтаксиса языка с помощью автоматного метода.



■ ***Рис. 6.*** *Отношения в метамодели языка описания шахматных позиций*

• Множество нетерминалов заменяется множеством классов.

• Правила вывода заменяются конъюнктивной композицией классов.

• Альтернативные правила вывода заменяются дизъюнктивной композицией классов или обобщением.

• Альтернативные правила вывода вида $A ::= a_1 | a_2 ... | a_n$, где $a_1, a_2, ..., a_n$ — терминалы, заменяются множествами значений вида $\{a_1, a_2, ..., a_n\}$.

• Рекурсивные правила вывода заменяются композицией с неопределенной кратностью.

Рассмотрим в качестве еще одного примера мини-язык, предназначенный для выполнения теоретико-множественных операций с подмножествами некоторого множества элементов. Для простоты изложения элементы обозначим строчными латинскими буквами, а множества — прописными латинскими буквами. Программа в этом языке является последовательностью предложений, каждое из которых — это определение множества либо перечислением элементов, либо с помощью операций объединения и пересечения, примененных к ранее определенным множествам. Приведем формальную грамматику языка в традиционной нотации Бэкуса—Наура [8] с указанием как значимых, так и не значимых (с точки зрения абстрактного синтаксиса) терминальных символов. Для удобства дальнейших ссылок правила перенумерованы.

1. **Программа ::= Предложение.** | **Предложение ; Программа**

2. **Предложение ::= Имя = Выражение**

3. **Выражение ::= Слагаемое** | **Слагаемое ∪ Выражение**

4. **Слагаемое ::= Множитель** | **Множитель ∩ Слагаемое**

5. **Множитель ::= (Выражение)** | **Имя** | **{ Задание множества }**

6. **Имя ::=** A | B | ... | X | Y | Z

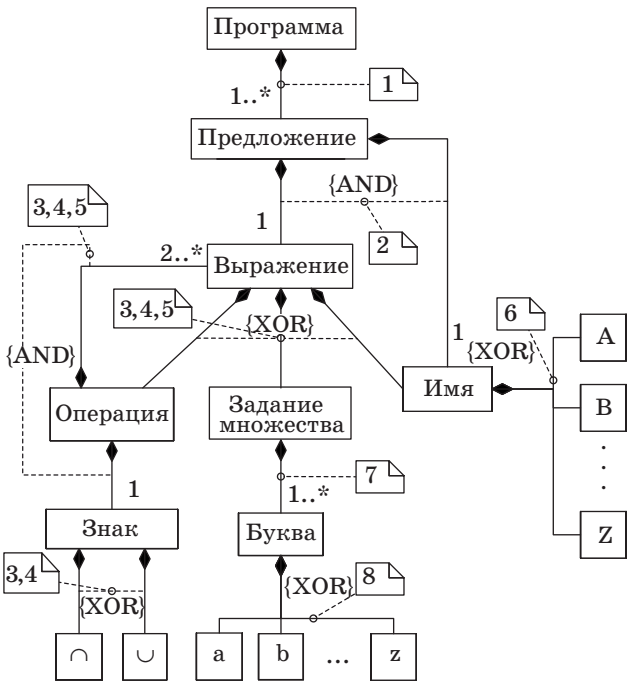
7. **Задание множества ::= Буква** | **Буква , Задание множества**

8. **Буква ::=** a | b | ... | x | y | z

Здесь нетерминалы выделены полужирным шрифтом. Терминалами языка являются знаки = { } . , ; ∪ ∩ ( ) и латинские буквы a b ... x y z и A B ... X Y Z.

Определение соответствующего абстрактного синтаксиса мини-языка множеств с помощью автоматного метода представлено на рис. 7. В отличие от приведенной формальной грамматики, абстрактный синтаксис не описывает приоритет операций. Это обусловлено тем, что экземпляр абстрактного синтаксиса, дерево программы, строится уже с учетом приоритета операций. То есть соглашения о приоритете операций относятся к конкретному синтаксису языка. Поэтому мы вводим конструкцию Операция и определяем конструкцию Выражение как дизъюнктивную композицию конструкций Операция, Имя и Задание множества. Для наглядности указано с помощью примечаний, из какого правила грамматики получилось каждое отношение абстрактного синтаксиса.

Однако, как уже было сказано, правил абстрактного синтаксиса недостаточно для полного задания языка. Например, в мини-языке множеств подразумеваются два контекстных условия, которые невозможно описать средствами контекстно-свободной грамматики:

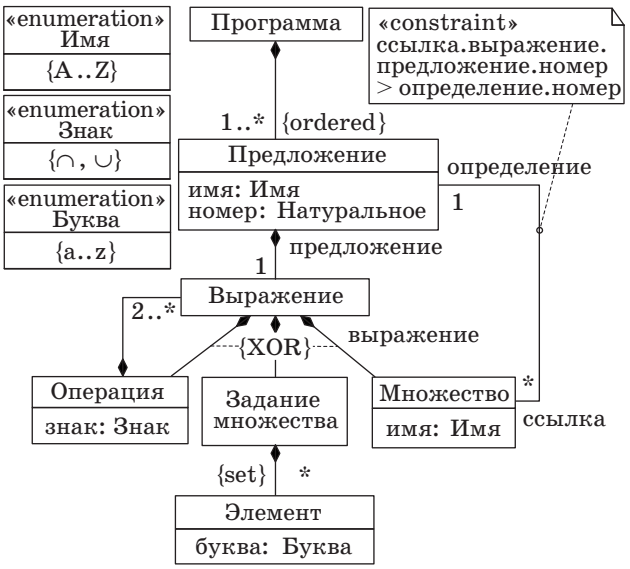


■ ***Рис. 7.*** *Абстрактный синтаксис мини-языка множеств*

1) все буквы в определении множества перечислением элементов должны быть различны;

2) всякому вхождению имени множества в правую часть равенства должно предшествовать вхождение этого имени в левую часть.

Покажем, как эти контекстные условия выражаются в метамодели средствами UML. Для этого в метамодель необходимо ввести дополнительные атрибуты, отношения и ограничения. Заметим, что преобразование формальной грамматики в абстрактный синтаксис, согласно указанным выше правилам, может быть произведено (почти) формально, автоматически и без участия человека. Но задание неформальных контекстных условий не может быть выполнено автоматически, это требует человеческой изобретательности. Первое контекстное условие можно задать очень просто: достаточно воспользоваться стандартным ограничением полюса ассоциации {set}, которое появилось в UML 2. Второе контекстное условие несколько сложнее, и возможны различные варианты его задания. Мы использовали нумерацию предложений программы с помощью дополнительного атрибута номер (рис. 8) и наложения ограничения на значения этого атрибута в ассоциации, связывающей использующие и определяющие вхождения имени. Также для наглядности введены дополнительные классы Элемент и Множество.



■ ***Рис. 8.*** *Метамодель мини-языка множеств*

Таким образом, автоматный метод позволяет полностью и формально задать определяемый язык. Общая структура языка задается абстрактным синтаксисом, причем, при наличии порождающей грамматики это может быть сделано автоматически. Контекстные условия и перекрестные ссылки задаются вручную с помощью развитых средств диаграмм классов UML.

Окончание следует.

Литература

1. **Fowler M.** Language Workbenches: The Killer-App for Domain Specific Languages? // Martin Fowler's Personal Page, 2005. http://www.martinfowler.com/articles/languageWorkbench.html (дата обращения: 13.07.2009).
2. **Dmitriev S.** Language Oriented Programming: The Next Programming Paradigm // Electronic magazine onBoard. 2005. http://www.onboard.jetbrains.com/is1/articles/04/10/lop/index.html (дата обращения: 13.07.2009).
3. **Оллонгерен А.** Определение языков программирования интерпретирующими автоматами. — М.: Мир, 1977. — 288 с.
4. **Гуров В. С., Мазин М. А., Шалыто А. А.** Автоматическое завершение ввода условий в диаграммах состояний // Информационно-управляющие системы 2008. № 1. С. 24–33.
5. **Поликарпова Н. И., Шалыто А. А.** Автоматное программирование. — СПб.: Питер, 2008. — 176 с.
6. **Буч Г., Рамбо Д., Джекобсон А.** Язык UML. Руководство пользователя. — М.: ДМК пресс, 2007. — 496 с.
7. **Ершов А. П., Грушецкий В. В.** Метод описания алгоритмических языков, ориентированный на реализацию / ВЦ Сибирского отд-ния АН СССР. — Новосибирск, 1977. № 74. — 40 с.
8. **Братчиков И. Л.** Синтаксис языков программирования. — М.: Наука, 1975. — 232 с.