Проблемы управления. 2006. № 1. С. 32-43.

УДК 681.3

РАЗНЕСЕННЫЙ ГРАММАТИЧЕСКИЙ РАЗБОР

В.С. Выхованец

Институт проблем управления им. В.А.Трапезникова, г. Москва

Рассматривается контекстная технология программирования, основанная на создании специализированного языка для решения заданного класса прикладных задач путем понятийного анализа предметной области и отражения ее понятийной структуры в понятиях создаваемого языка. Описывается метод разнесенного грамматического разбора, который необходим для эффективной реализации компилятора системы контекстного программирования.

ВВЕДЕНИЕ

Традиционно грамматический разбор текста делится на фазы лексического и синтаксического анализа. В фазе лексического анализа входной текст, рассматриваемый как поток знаков, разбивается на лексемы. Лексические анализаторы реализуются двумя способами: либо в виде подпрограммы, вызываемой синтаксическим анализатором для получения очередной лексемой, либо в виде модуля, осуществляющего полный просмотр текста (проход), результатом которого является входной текст, разбитый на лексемы и рассматриваемый как поток лексем [1].

В свою очередь синтаксический анализ предназначен для получения структуры текста, где под структурой понимается дерево грамматического разбора. В настоящее время используется либо LL(1)-анализ и его вариант – рекурсивный спуск, либо LR(1)-анализ и его варианты. Рекурсивный спуск чаще применяется при ручном, а LR(1) – при автоматизированном построении синтаксических анализаторов [2].

В настоящей статье рассмотрен грамматический разбор в контекстной технологии программирования [3], который не сводятся к известным методам, и отличает контекстную технологию от других технологий. Статья построена следующим образом. В первом разделе кратко описана контекстная технология программирования в объеме, необходимом для понимания метода разнесенного грамматического разбора. Во втором разделе описываются основные механизмы разнесенного разбора – контекстное сопоставление и структурное распознавание. Третий раздел посвящен учету при грамматическом разборе приоритета понятий и предложений. В четвертом разделе обсуждаются неоднозначности,

которые могут возникнуть в программе, и их влияния на эффективность разнесенного грамматического разбора.

КОНТЕКСТНАЯ ТЕХНОЛОГИЯ

В контекстной технологии программирования формализация знаний о предметной области осуществляется в виде понятийной модели. Построение понятийной модели осуществляется путем выявления и выражения понятийной структуры предметной области в объеме, достаточном для решения некоторого класса задач. В отличие от других технологий, например структурной и объектно ориентированной, которые основаны соответственно на структурном и объектном анализе, контекстная технология программирования строится на более устойчивых формах отражения результатов декомпозиции и определяется на основе понятийного анализа предметной области.

Средствами контекстной технологии создается специализированный язык, отражающий понятийную структуру предметной области в аспекте решаемых задач. Такой подход основывается на допущении, что уже в процессе изучения предметной области формируется система понятий, наиболее приспособленная для постановки и решения прикладных задач. Выявленная таким образом система понятий ложится в основу создаваемого языка программирования.

Понятийный анализ

Понятийный анализ как методика построения понятийной структуры основан на абстрагировании понятий и осуществляется путем представления и сравнения сущностей предметной области с целью выявления их общих и отличительных признаков, фиксации необходимых понятий и определения отношений между ними.

В рассматриваемом контексте *предметной областью* называется выделенный фрагмент реальной действительности, представляемый совокупностью некоторых сущностей, которые могут быть уникально идентифицированы и описаны.

Под простым *понятием* (концептом) понимается множество сущностей, объединенных на основе общности признаков (атрибутов, свойств). Понятие задается именем, интенсионалом и экстенсионалом.

Интенсионал или содержание понятия представляется как множество его взаимосвязанных признаков, позволяющих отличать сущности, принадлежащие понятию от других сущностей предметной области. Иными словами, интенсионал определяет смысл, который вкладывается в понятие. В свою очередь *экстенсионал* или объем понятия определяется как множество сущностей, принадлежащих этому понятию.

Сущности, составляющие экстенсионал понятия различаются с помощью признаков. *Признак* является специализированным понятием, которое характеризуется именем, семантической ролью и множеством допустимых значений. Множество допустимых значений признака образует его экстенсионал или *домен*, а интенсионал признака вырождается в его *семантическую роль*. Таким образом, интенсионал признака – это минимальная семантическая единица, на основе которой строится понятийная структура предметной области, а экстенсионал признака – множество минимальных (неделимых) синтаксических единиц, понимаемые в широком смысле.

В предельном случае, когда у понятия единичный объем, имеем некоторую именованную сущность. Следовательно, имена могут использоваться не только для обозначения понятий, но и для обозначения единичных сущностей. Более того, каждая единичная сущность может быть представлена как отдельное понятие. Сущности, на которые можно ссылаться с помощью имени называются *терминальными понятиями* или денотатами.

Под *понятийной структурой* понимается совокупность понятий и связей между ними. Из четырех известных видов абстракции понятий: ассоциация, обобщение, типизация и агрегация [4], в понятийной структуре отображаются только фундаментальные абстракции – обобщение и агрегация. Последнее связано с тем, что абстракция типизации является частным случаем обобщения, а абстракция ассоциации эквивалентна агрегации по способу своего формирования [5]. Таким образом, понятийная структура предметной области может быть описана множеством понятий, на котором определены отображения обобщения и агрегации.

Обобщение понятий – это форма порождения нового понятия на основе одного или нескольких подобных, когда порождаемое понятие сохраняет общие признаки исходных понятий, являющихся его *конкретизацией*, но игнорирует более тонкие различия. *Агрегация* используется в тех случаях, когда вновь порождаемое сложное понятие включает исходные *специализированные* понятия в качестве своих составных частей.

Рис. 1

Пример 1. Рассмотрим предельно простую предметную область, которую условно назовем «Учащийся». На рис. 1 показана ее понятийная структура, представленная в объеме, достаточном для решения некоторого класса задач, где пунктирные линии задают отношения обобщения, сплошные – отношения агрегации, а множество признаков задано списком понятий в квадратных скобках. ♦

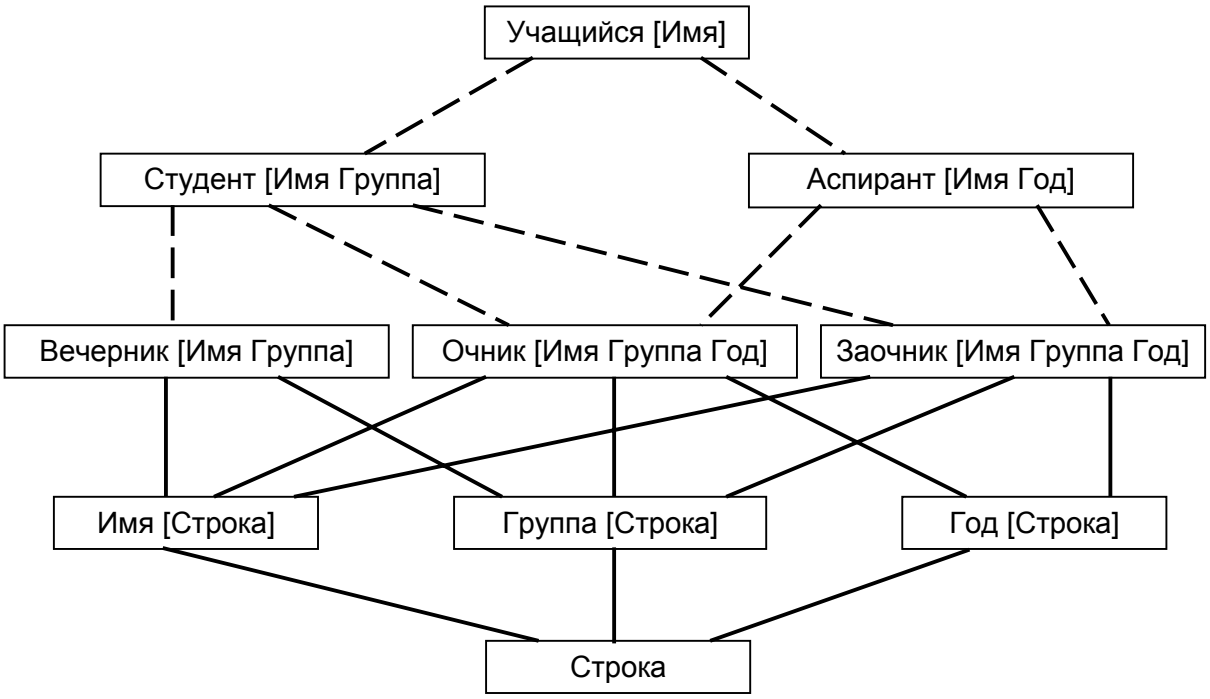

Рис. 1. Пример понятийной структуры

Понятийная модель

Программой в контекстной технологии является понятийная модель, дополненная описанием решения одной или нескольких прикладных задач:

Программа = Понятийная модель + Решение задачи (Ситуация).

В понятийной модели выделяют описания понятийной структуры, синтаксиса понятий, семантики понятий и процесса компиляции:

Понятийная модель = Структура + Синтаксис + Семантика + Компиляция.

Для обеспечения семантической полноты модели понятийная структура и синтаксис понятий описываются на декларируемом в контекстной технологии метаязыке, а описание семантики понятий и решаемых задач выполняется на определяемом специализированном языке, задаваемом понятийной структурой и описанием синтаксиса.

Для проверки контекстных условий и привязки понятийной модели к целевой вычислительной платформе используется описание процесса компиляции, которое осуществляется на определяемом языке с использованием некоторого множества базовых примитивов – предопределенных сущностей вычислительной платформы и сервисов системы контекстного программирования, имеющих аппаратно-зависимую или виртуальную реализацию.

Понятийная модель и решение задачи подвергаются грамматическому разбору и компиляции в код целевой платформы, таким образом, что откомпилированные ранее

предложения исполняются при грамматическом разборе и компиляции следующих предложений.

Метаязык

Рис. 2

На рис. 2 приведена порождающая грамматика метаязыка контекстной технологии, описанная с точностью до обозначенных курсивом нетерминальных (*notion*, *aspect*) и терминальных (*string*) понятий определяемого языка, а также пробельных знаков, разделяющих нетерминальные понятия метаязыка. В квадратных скобках обозначены части продукций грамматики, которые могут быть опущены. Приведенный метаязык является детализацией метаязыка из [3] и представлен в виде, необходимом для описания методов грамматического разбора.

1	program	→	declaration [situation] | declaration situation program
2	declaration	→	essence | essence declaration
3	essence	→	generalization *notion* [aggregation] [description]
4	generalization	→	'(' [notions] ')'
5	aggregation	→	'[' notions ']'
6	notions	→	notion [compilation] | notion [compilation] notions
7	description	→	sentence | sentence description
8	sentence	→	syntax semantics
9	syntax	→	context |
10			context lexeme |
11			context lexeme parsing |
12			lexeme parsing |
13			lexeme
14	context	→	notion | notion context
15	lexeme	→	term | pattern
16	term	→	'' *string* ''
17	pattern	→	'"' *string* '"'
18	parsing	→	item | item parsing
19	item	→	notion | lexeme | compilation
20	semantics	→	imperative | imperative semantics
21	imperative	→	[*aspect*] '{' text '}'
22	text	→	phrase | phrase text
23	phrase	→	string | [aspect] '{' text '}'
24	compilation	→	'[' text ']'
25	situation	→	'<' text '>'

Рис. 2. Грамматика метаязыка

Текст программы program состоит из последовательности деклараций declaration и ситуаций situation (строка 1). В декларативной части определяется язык, на котором в ситуационной части описывается решение некоторой прикладной задачи.

Декларация состоит из описаний сущностей предметной области essence (строка 2). Каждой сущности присваивается имя *notion* нетерминального понятия определяемого языка, а само понятие задается как находящееся в отношении обобщения generalization и агрегации aggregation с системой других, ранее определенных понятий notions (строки 3-6).

Помимо отношений обобщения и агрегации, необходимых для выражения понятийной структуры предметной области, каждое понятие содержит свое описание description, включающее множество предложений sentence (строка 7). Предложения используются для задания способов выражения понятий в тексте программы и состоят из определений синтаксиса syntax и семантики semantics (строка 8).

Синтаксис предложений задается последовательностью элементов item (строка 19): имен понятий notion, лексем lexeme и текстов компиляции compilation. Лексема является терминальным понятием определяемого языка. Для выражения лексем могут использоваться как термы term, так и множества термов, задаваемые на языке регулярных выражений в виде терминальных шаблонов pattern (строки 16, 17).

Здесь под *термом* понимается элементарная синтаксическая единица, состоящую из последовательности знаков терминального алфавита, а *лексема* определяется как элементарная семантическая единица, представленная одним или несколькими термами.

Допустимое множество именованных семантик semantics задается в виде императивов imperatives с именами *aspect* (строки 20, 21). Для описания императива imperative, процесса компиляции compilation и ситуации situation используется конструкция text (строки 22, 23), являющаяся текстом, построенным по правилам определяемого в понятийной модели специализированного языка.

Таким образом, под *текстом* понимается последовательность лексем string, предназначенная для выражения некоторого сложного смысла. Текст, в отличие от лексем предполагает свое деление на смысловые части, в то время как лексема такого деления не допускает.

Рис. 3

Пример 2. На рис. 3 задана одна из возможных понятийных моделей для предметной области из Примера 1. В рассматриваемой модели используется не определенное в модели понятие Строка, для которого определена операция присваивания, а %n является метасимволом (алиасом) и обозначает n-ый по порядку элемент из описания синтаксиса предложения (понятие или лексему).

В строках 15, 17 и 18, где в текстах компиляции задана проверка контекстных условий [Группа == ''], [Год == 0] и [%3 < 5], подразумевается использование не описанных в рассматриваемой модели понятий Логическое и Целое.

Синтаксис предложений позволяет выделить в анализируемом тексте выражаемые понятия, а семантика каждого предложения, заданная неименованным императивом, обеспечивает сохранение значений признаков сущностей, встречающихся в тексте.

Для рассматриваемой понятийной модели в тексте ситуационной части программы могут быть, например, следующие строки: ‘Иван заочник ДЗ’, ‘Петр аспирант 1 года’, ‘Сергей учащийся’, причем первые две строки служат для выражения не только понятий Заочник и Аспирант соответственно, но и понятия Учащийся, так как являются конкретизациями последнего. ♦

```
1   () Имя [Строка]
2       ″[А-Я][а-я]+″ {Строка = %1}
3   () Год [Строка]
4       ″[1-5]″ {Строка = %1}
5   () Группа [Строка]
6       ″[А-Я][1-9]″ {Строка = %1}
7   () Вечерник [Имя Группа]
8       Имя 'вечерник' Группа {Имя = %1, Группа = %3}
9   () Заочник [Имя Группа Год]
10      Имя 'заочник' Группа {Имя = %1, Группа = %3, Год = 0}
11      Имя 'заочник' Год ‘года’ {Имя = %1, Группа = '', Год = %3}
12  () Очник [Имя Группа Год]
13      Имя 'очник' Группа {Имя = %1, Группа = %3, Год = 0}
14      Имя 'очник' Год 'года' {Имя = %1, Группа = '', Год = %3}
15  (Вечерник [Год == 0] Очник [Год == 0] Заочник [Год == 0]) Студент [Имя Группа]
16      Имя 'студент' Группа {Имя = %1, Группа = %3}
17  (Очник [Группа == ''] Заочник [Группа == '']) Аспирант [Имя Год]
18      Имя 'аспирант' Год [%3 < 5] 'года' {Имя = %1, Год = %3}
19  (Студент Аспирант) Учащийся [Имя]
20      Имя 'учащийся' {Имя = %1}
```

Рис. 3. Пример понятийной модели

СОПОСТАВЛЕНИЕ И РАСПОЗНАВАНИЕ

При *контекстной интерпретации*, лежащей в основе контекстной технологии программирования, определение семантического значения терма осуществляется по его

месту в тексте. В общем случае одна и та же последовательность терминальных знаков может быть разбита на термы различным образом. Более того, один и тот же терм может быть сопоставлен различным лексемам, т.е. одна и та же последовательность терминальных знаков может служить носителем различных смыслов. Верно и обратное, различные последовательности терминальных знаков могут использоваться для выражения одного и того же смысла, но в различных контекстах.

Отсюда следует, что при контекстной интерпретации разделение анализа текста на фазы лексического и синтаксического анализа не представляется возможным и необходимо использовать метод грамматического разбора, учитывающий эту особенность. Для этого применим метод грамматического разбора, названный *разнесенный* и заключающийся в разделении определения применимости предложения sentence при анализе текста на две части – на контекстное сопоставление предложений, осуществляемое при просмотре текста программы назад, и структурное сопоставление, выполняемое при просмотре вперед.

Структура предложения

Разделим структурно каждое предложение понятийной модели (рис. 2, строки 9-13), описывающее некоторое понятие *notion* на четыре области: область контекста context, лексему lexeme, область разбора parsing и результат *notion*.

Контекстом предложения назовем последовательность понятий, с которого начинается описание синтаксиса предложения. Под *лексемой* предложения будем понимать ее первую лексему. Область *разбора* определим как часть описания синтаксиса, непосредственно следующая за первой лексемой. И, наконец, *результат* предложения – это понятие *notion*, определяемое предложением полностью (когда больше нет предложений с тем же результатом) или частично (если такие предложения имеются).

Контекстное сопоставление

Рис. 4

Пусть имеется структура данных, которую назовем *текущим контекстом* и реализуем в виде стека контекста (рис. 4). Текущий контекст содержит последовательность понятий, которые уже распознаны, а соответствующие им терминальные знаки интерпретированы (сопоставлены лексемам предложений) и извлечены из входного потока. Следовательно, в каждый момент времени грамматический анализатор имеет некоторое *текущее состояние*, определяемое состоянием стека контекста и текущей позицией во входном потоке.

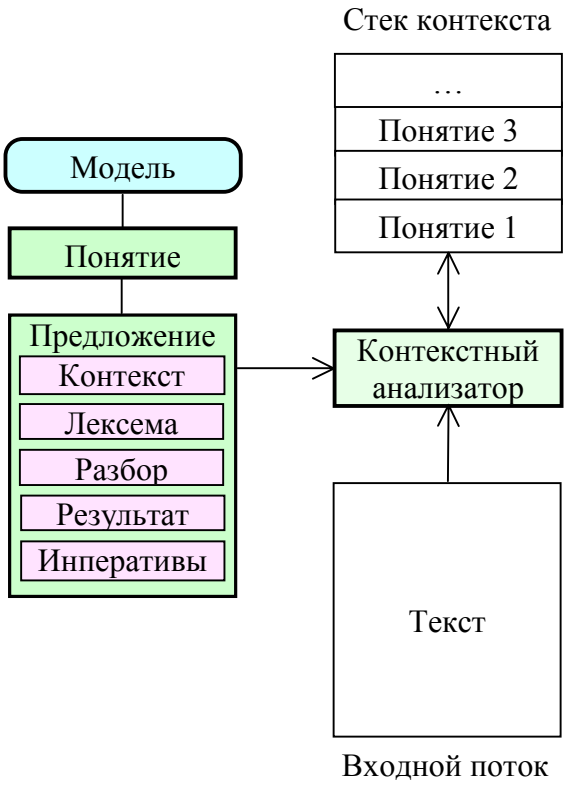

Рис. 4. Механизм контекстного анализа

Поиск предложений, применимых в текущем состоянии анализатора, осуществим путем сравнения контекста предложений модели с текущим контекстом. Предложения, имеющие контекст, сопоставимый с текущим, могут использоваться в текущем состоянии анализатора. Таким образом, под *сопоставимостью* понимается соответствие понятий на вершине стека контекста понятиям из контекста предложения с учетом возможной их конкретизации или специализации.

Однако из сопоставимых предложений следует выбрать те, лексема которых сравнима с термом в текущей позиции входного потока. В итоге, из всего множества предложений дальнейшему анализу следует подвергать те предложения, которые применимы в текущем состоянии анализатора. Тем самым осуществляется *контекстное сопоставление* предложений путем просмотра текста назад и определение применимости сопоставленных предложений путем сравнения терма в текущей позиции входного потока с лексемой предложения. Заметим, что грамматический разбор текста, подлежащего просмотру назад, уже выполнен и представлен в виде текущего контекста.

Структурное распознавание

После выявления предложения, применимого в текущем контексте, наступает этап разбора его оставшейся части, которая еще не сопоставлена входному потоку. Для этого

грамматический анализатор запоминает свое состояние, очищает текущий контекст и выполняет просмотр вперед, начиная с текущей позиции входного потока.

Если элементом разбора при просмотре вперед является лексема из области разбора, то выполняется ее сравнение с термом входного потока. При успешном сравнении терм, соответствующий лексеме, извлекается из входного потока. В случае неудачи, состояние анализатора восстанавливается, а анализируемое предложение считается нераспознанным.

Если требуется распознать понятие из области разбора, то для анализа выбираются те предложения, которые сопоставимы с текущим состоянием анализатора и имеют это понятие в качестве своего результата. При удачном распознавании анализатор переходит к следующему элементу области разбора. В случае неудачи, анализируемое предложение считается нераспознанным, состояние анализатора восстанавливается и он переходит к разбору следующего применимого предложения.

Если все элементы области разбора сопоставлены входному потоку, то предложение считается *структурно распознанным*, выполняется восстановление контекста, в котором контекст предложения заменяется на понятие-результат, а получившиеся при этом состояние входного потока становится текущим. Тем самым из входного потока извлекается распознанная часть текста, учтенная в текущем контексте.

ПРИОРИТЕТ ПОНЯТИЙ И ПРЕДЛОЖЕНИЙ

Для упорядочивания грамматического разбора установим приоритетность предложений и понятий исходя из естественных представлений о порядке описания предметной области.

Приоритет предложений

Для реализации приоритетности одних предложений над другими всем им припишем приоритет в соответствии с их порядком в понятийной модели. Этот приоритет, определяемый содержательным описанием предметной области, назовем *естественным* или абсолютным приоритетом предложений. При проверке применимости из двух сопоставимых предложений выбирается то, которое определено позже (далее по тексту), т.е. имеет меньший приоритет. Поясним роль приоритетов предложений в процессе грамматического разбора на примере.

Пример 3. Рассмотрим понятийную модель, описывающую булевы выражения (Рис. 5).

Рис. 5

```
() Boolean
    ″[A-Za-z][A-Za-z0-9]*″ {}
    ″false|true″ {}
    '(' Boolean ')' {}
    'not' Boolean {}
    Boolean 'and' Boolean {}
    Boolean 'or' Boolean {}
    Boolean 'imp' Boolean {},
```

Рис. 5. Понятийная модель «Булевы выражения»

В модели определено одно понятие Boolean, а предложения расположены в порядке естественного приоритета операций булевой алгебры, т.е. переменные имеют наибольший приоритет (первое предложение), в то время как операция импликации imp имеет наименьший приоритет (последнее предложение).

При грамматическом разборе текста 'true or x and y' с пустым текущим контекстом сопоставимыми являются первые четыре предложения. Однако применимы только первое и второе. Так как разбору в первую очередь подлежит второе предложение, то терм 'true' интерпретируется как логическая константа.

Если поменять местами первое и второе предложения, то терм 'true' будет интерпретирован как переменная с именем true. В последнем случае, если такой переменной не окажется, предложение помечается как неприменимое, а разбору подлежит следующее предложение, расположенное выше. В этом случае терм 'true' будет сопоставлен логической константе. ♦

Приоритет понятий

При наличии в понятийной модели отношений обобщения понятий порядок отбора предложений при грамматическом разборе несколько видоизменяется, так как в этом случае необходимо учитывать возможность представления одного понятия другим, являющимся его конкретизацией.

При структурном распознавании внутри группы предложений с одним и тем же понятием-результатом приоритеты предложений, как и ранее, задаются их естественным приоритетом. Однако приоритет самого результата определим относительным местом этого понятия в понятийной структуре. Из двух предложений, выражающих одинаковые понятия-результаты (сравнимые с учетом отношений обобщения), выбирается то, которое имеет меньший приоритет своего результата. Таким образом, при структурном распознавании предпочтение отдается конкретизации понятия по отношению к его

обобщению. Тем самым вводятся и учитываются *относительные* приоритеты понятий, задаваемые понятийной структурой модели.

Заметим, что при контекстном сопоставлении следует также учитывать возможность выражения одного понятия другим, являющимся его конкретизацией. Поясним роль приоритетов понятий на примере.

Рис. 6

Пример 4. Преобразуем понятийную модель из Примера 3, детализируя ее понятийную структуру (рис. 6).

```
1    () Variable
2        ″[A-Za-z][A-Za-z0-9]*″ {}
3    () Constant
4        ″false|true″ {}
5    (Variable) Logic
6        Variable {}
7        Integer {}
8        '(' Boolean ')' {}
9    (Constant Logic) Negation
10       'not' Logic {}
11   (Negation) Conjunction
12       Negation 'and' Negation {}
13   (Conjunction) Disjunction
14       Conjunction 'or' Conjunction {}
15   (Disjunction) Boolean
16       Disjunction 'imp' Disjunction {}
```

Рис. 6. Детализация понятийной модели «Булевы выражения»

Продолжим грамматический разбор текста 'true or x and y' при состоянии входного потока, равном 'or x and y' и текущем контексте Constant. В этом случае анализируемыми являются предложения понятий Constant, Negation, Conjunction, Disjunction и Boolean (расположены в порядке их относительных приоритетов) ввиду того, что Constant является конкретизацией этих понятий. В этом случае с текущим контекстом сопоставимы предложения 16, 14, 12, 10, 8 и 4. Однако применимо только предложение 14, так как его лексема 'or' может быть выражена термом 'or' входного потока.

После контекстного сопоставления наступает фаза структурного распознавания единственного отобранного предложения Conjunction 'or' Conjunction, которое применимо в текущем состоянии анализатора. Теперь из входного потока при очищенном контексте следует извлечь терм, выражающий понятие Conjunction или одну из его конкретизаций:

Variable, Logic, Constant или Negation (перечислены в порядке относительных приоритетов).

Находим, что в рассматриваемом случае сопоставимыми являются предложения 2, 4, 8 и 10, а применимо только предложение 2. Для применения предложения 2 из входного потока извлекается терм 'x', а текущий контекст становится равным результату предложения – понятию Variable.

Продолжаем распознавание до тех пор, пока текущий контекст анализатора сопоставим с понятием Conjunction. Очевидно, при текущем состоянии входного потока 'and y' применимо предложение 12, производящее в стеке контекста искомое понятие Conjunction.

Рис. 7

В итоге получаем схему грамматического разбора, приведенную на рис. 7. ♦

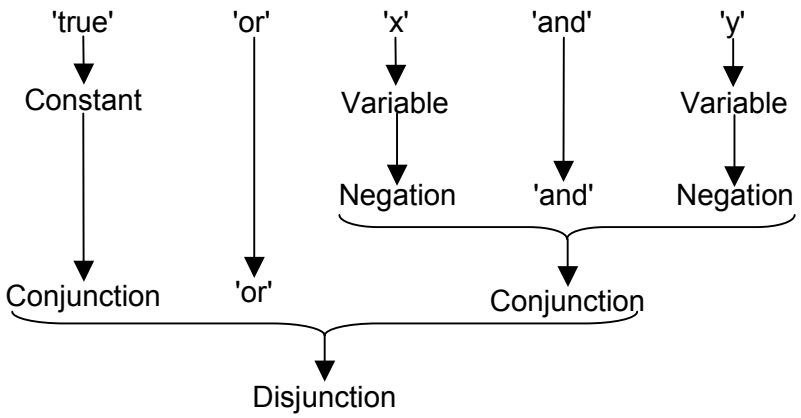

Рис. 7. Схема грамматического разбора

Конверторы

Заметим, что метаязык контекстной технологии позволяет определять предложения, не имеющие лексем, т.е. состоящих только из одного контекста: sentence → context. Такие предложения будем называть *конверторами*. Конвертор устанавливает эквивалентность последовательности понятий некоторому понятию

При наличии в понятийной модели конверторов контекстное сопоставление и структурное распознавание следует осуществлять с учетом возможного преобразования одного или нескольких понятий в другое понятие с помощью соответствующих конверторов. Приоритет таких преобразований определяется естественным приоритетом использующихся для этого предложений.

Пример 5. В понятийной модели на рис. 6 имеются два конвертора, выраженных предложениями 6 и 7. Они устанавливаю синонимию понятия Variable и понятия Integer понятию Logic, однако обратное неверно, т.к. соответствующие конверторы в модели не

определены. Назначение перечисленных конверторов – преобразование пропозиционной переменной и целого числа в булево значение.

Например, благодаря конвертору 7 при анализе текста 'true or 2 and y' целая константа 2 может быть интерпретирована как истинное булево значение, а переменная y, представленная адресом ячейки памяти, преобразована конвертором 6 в свое значение. ♦

НЕОДНОЗНАЧНЫЕ ГРАММАТИКИ

При разнесенном грамматическом разборе каждое предложение сопоставляется входному потоку не полностью и не сразу. Область контекста предложения не подлежит сопоставлению, она есть результат грамматического разбора уже принятого ранее текста. В свою очередь лексема предложения служит для выбора предложения, применимого в текущем состоянии анализатора. Следовательно, действительно выполняемому грамматическому разбору подлежит последняя часть предложения (область разбора), если, конечно, она присутствует в описании синтаксиса.

Метод разнесенного грамматического разбора используется для повышения эффективность контекстного анализа. Последнее необходимо ввиду возможности анализа текста, имеющего неоднозначности (описываемого неоднозначными грамматиками). Признаком неоднозначности при грамматическом разборе является наличие одном и том же текущем состоянии анализатора нескольких применимых предложений.

При выявлении неоднозначности анализатор перед началом разбора сохраняет свое состояние (создает точку отката назад), после чего продолжает грамматический разбор. Если в результате анализа ни одно предложение не применимо, восстанавливается состояние, сохраненное последним (производится откат назад), и разбору подвергается следующее предложение. Если такового не окажется, делается вывод об ошибке в анализируемом тексте.

Описанная процедура реализуется рекурсивным вызовом анализатора всякий раз, когда при контекстном сопоставлении предложений обнаруживается неоднозначность. Учитывая то, что понятийная модель призвана описать некоторую предметную область непротиворечивым образом, то доля неоднозначности в описании предметной области не должна быть высока. Более того, неоднозначность понятийной модели становится выразительным средством в контекстной технологии и используется для повышения уровня специализированного языка.

Как бы то ни было, при реализации грамматического разбора предусматривается некоторое критическое количество точек отката назад. При достижении последнего

делается вывод о недостаточной проработке понятийной модели или об ошибке в анализируемом тексте.

ЗАКЛЮЧЕНИЕ

Контекстная технология программирования основана на понятийном анализе и позволяет объективировать (формализовать) описание предметной области в наиболее устойчивых формах, лежащих в основе мышления, а именно, в понятиях и умозаключениях, где понятия представляются понятийной структурой предметной области, а умозаключения – предложениями понятийной модели.

Метаязык контекстной технологии имеет средства для выражения произвольных взаимосвязей между понятиями, не обязательно сводимых к известным видам абстракции. По своей сути каждое предложение понятийной модели является декларацией некоторого отношения между понятием-результатом и понятиями (терминальными и нетерминальными), включенными в описание синтаксиса предложений. В частности, на примерах показан способ выражения синонимии понятий.

Известные методы грамматического разбора не могут быть использованы в контекстной технологии в виду того, что на вид контекстно-свободных грамматик, описывающих тексты программ на специализированном языке, создаваемом в процессе программирования, не накладывается никаких ограничений

В разработанном методе разнесенного грамматического разбора определение применимости предложений модели разделяется на две части – на контекстное сопоставление, осуществляемое при просмотре анализируемого текста назад, и структурное сопоставление, выполняемое при просмотре вперед.

Разнесенный грамматический разбор позволяет выполнять анализ текста, порожденного в том числе и неоднозначными грамматиками. Эффективность метода в этом случае напрямую зависит от полноты и непротиворечивостью понятийной модели. Ввиду того, что целью создания понятийной модели является адекватное решаемым задачам описание некоторой предметной области, следует ожидать небольшого числа таких неоднозначностей. С другой стороны, введение в понятийную модель неоднозначностей может служить улучшению выразительных качеств специализированного языка, создаваемого в процессе проектирования информационной системы.

Проблемы управления. 2006. № 1. С. 32-43.

ЛИТЕРАТУРА

1. Хантер Р. Основные концепции компиляторов. – М.: Вильямс, 2002

2. Ахо А., Сети Р., Ульман Д. Компиляторы. Принципы, технологии, инструменты. – М.: Вильямс, 2001.

3. Выхованец В. С., Иосенкин В. Я. Понятийный анализ и контекстная технология программирования // Проблемы управления. – 2005. – № 4. – С. 2-11.

4. Макетирование, проектирование и реализация диалоговых информационных систем / Под ред. Е.И. Ломако. – М.: Финансы и статистика, 1993.

5. Гаскаров Д.В. Интеллектуальные информационные системы. – М.: Высш. шк., 2003.