Проблемы бионики
1998. Вып. 48. С. 150—158

ХТУРЭ

УДК 519.687

В.А. ГРЕБИННИК, В.Н. ШЕВЕЛИН

ПРОГРАММНО-ТЕХНОЛОГИЧЕСКИЙ КОМПЛЕКС ДЛЯ ПОСТРОЕНИЯ ЭКСПЕРТНЫХ СИСТЕМ РАЗЛИЧНОГО НАЗНАЧЕНИЯ

Одним из наиболее приоритетных направлений в разработке программного обеспечения, относимого к классу «программ с элементами искусственного интеллекта», было и есть создание экспертных систем, ориентированных на работу в самых различных предметных областях. Обычно работа с ЭС представляет собой формирование запросов к ЭС в форме, доступной для восприятия ее блоком анализа запросов. Разработчики ЭС стремятся максимально приблизить язык запросов к естественному, точнее к тому его подмножеству, которое соответствует данной предметной области. Тем более, что такие подмножества отличаются достаточно высокой степенью формализации, что существенно упрощает анализ. Следует отметить, что независимо от конкретной предметной ориентации ЭС, начальные этапы анализа запросов, составленных на языке, близком к естественному, имеют много общего, так как основная задача на данном этапе — расшифровать структуру запроса и определить значения ключевых элементов языковой конструкции, посредством которой выражен запрос. Однако зачастую при разработке ЭС все ее компоненты, включая и блок анализа запросов, пишутся с нуля. Поэтому весьма важной и актуальной является задача создания программно-технологического комплекса (ТК) проектирования и реализации ЭС. Такие технологические комплексы существуют и называются генераторами ЭС [1]. Настройка генератора ЭС на конкретную ЭС осуществляется базой знаний (БЗ). При этом используется только одна модель представления знаний из логической, сетевой или фреймовой моделей [2]. Процесс создания ЭС с помощью таких генераторов весьма эффективен, но сама ЭС в большинстве случаёв получается неэффективной, негибкой для модификации, а ее БЗ — трудной для построения. Такая ситуация возникает, на наш взгляд, из-за практически полного устранения процедурных элементов из процесса проектирования. Мы полагаем, что введение процедурных фрагментов в проектирование ЭС позволит повысить эффективность спроектированных ЭС, их гибкость, а использование нескольких моделей представления знаний будет содействовать этому процессу.

ТК проектирования и реализации ЭС (ТКПиРЭС) должен строиться из программных “заготовок” - компонентов, которые можно соединять между собой, настраивать изменением каких-либо декларативных и процедурных

элементов. Спроектированная с помощью ТК экспертная система должна обязательно транслироваться, и полученный модуль есть уже готовая к использованию ЭС. Продуктивность такого подхода подтверждена экспериментально, например существует система проектирования сложных программных систем Delphi фирмы Borland Int. [3].

Основными компонентами ТКПиРЭС являются механизмы анализа языковых конструкций, баз знаний, логического вывода, всевозможных интерфейсов. Возможно их соединение в любой комбинации. Мы структурировали знания на лингвистические, предметной области (области экспертизы), знания принятия решений. Каждой модели знаний соответствует своя компонента в ТКПиРЭС.

Как мы отметили выше, одной из основных компонент является "языковый процессор" (ЯП). Эта компонента состоит из других компонент и строится по тому же принципу, что и ТКПиРЭС. ЯП путем настройки становится транслятором некоторого конкретного языка. Повышению гибкости и эффективности процесса проектирования, способствует то, что ЯП не является жесткой программной заготовкой, настраиваемой только декларативными элементами, а позволяет использовать фрагменты процедурного подхода, жестко ограниченные структурными рамками, то есть процедуры являются компактными и локальными. Таким образом, разработка универсальной анализирующей системы, которую можно было бы настраивать на конкретный предметный язык, позволит сократить объем работы, сопутствующей разработке ЭС.

До сих пор средства построения систем, ориентированных на анализ языковых конструкций, ограничивались в основном классом программ, именуемых «компиляторами компиляторов». Наибольшую известность получила программа Jacc, которая распространена в вузах как практическое пособие по изучении курсов «Проектирование трансляторов» и родственных ему. Она сочетает в себе простоту в использовании для человека, знакомого с программированием, и предоставляет средства, достаточные для описания несложных языков с высокой степенью формализации, таких как языки программирования. Однако для языков менее формализованных, с достаточно емкими базовыми словарями средств Jacc'а для разрешения конфликтов явно не хватает, в основном из-за трудностей в учете контекстной информации, важность которой возрастает со снижением формализации анализируемого языка. Кроме того, высокой остается доля работ, связанных с описанием семантики языка — система Jacc не предлагает для этих целей никаких средств, кроме кодирования ее на языке С. Поэтому необходимо выработать основные концепции такой организации ЯП, кото-

рая позволила бы с минимальными затратами создавать системы, анализирующие сложные слабо формализованные языки.

В классическом варианте построения ЯП анализ языковой конструкции делится на два этапа — лексический и синтаксический анализ [4]. Целесообразность такого подхода не вызывает сомнений, так как только лексический анализ не способен выделить необходимый для адекватного восприятия смысла языковой конструкции (ЯК) объем информации, а для проведения синтаксического анализа ЯК, принадлежащей сложному языку с большим количеством словоформ, нужна поистине гигантская база знаний.

Назначение лексического анализатора — выделить во входной языковой конструкции известные словоформы (лексемы) и установить значения некоторых атрибутов лексем. В качестве основы лексического анализатора требуется выбрать систему, которая будет сочетать в себе ряд качеств, определяющих простоту и эффективность ее использования:

- простоту расширения словаря лексем как за счет добавления одиночных лексем, так и за счет слияния имеющихся словарей лексем;
- возможность полной детерминизации процесса анализа до уровня синонимов, что позволит применять высокоэффективные беступиковые алгоритмы анализа;
- простота и компактность хранения словаря лексем;
- возможность оптимизации набора управляющих правил (продукций) с целью уменьшения необходимых объемов памяти и повышения скорости анализа.

Наиболее перспективен механизм лексического анализа — базовые сети переходов — при ближайшем рассмотрении могут интерпретироваться как реализация при помощи конечных автоматов (КА) расширенных регулярных выражений (РРВ). Учитывая простоту лексического анализа при применении алгоритма, моделирующего КА и наличие математического аппарата, позволяющего проводить слияние КА, детерминизировать и оптимизировать их, а также широкие возможности расширенных регулярных выражений при описании лексики, в качестве основы описания лексики используются РВ, а ее анализ базируется на КА. Кроме того, нами разрабатывается аппарат, основанный на ассоциативной привязке множества переходов к конкретной лексеме, что позволит использовать для описываемых языков словари очень большого объема.

Имеется возможность выделения части семантики на этапе лексического анализа, что может быть полезно для получения некоторой дополнительной информации о конкретном содержании лексемы (например, вычислении атрибутов). Для этого необходимо создание расширяемой библиотеки процедур с унифицированным интерфейсом, причем при описании лек-

сики любому переходу между состояниями КА может быть поставлена в соответствие любая из библиотечных процедур. Такой механизм лексического анализа будем именовать семантическим конечным распознавателем.

Для описания лексики языка необходимо выработать язык записи расширенных регулярных выражений, который включал бы в себя помимо средств описания РРВ некоторые дополнения, позволяющие управлять процессом формирования словаря лексем для данного языка, а также правилами вычисления атрибутов лексем. Кроме того, ориентируясь на концепцию дружественности интерфейса, необходимо предоставить пользователю, конструирующему ЯП, более наглядные средства для составления описаний лексем. Их целесообразно выполнить в виде надстроек, позволяющих формировать описание лексем, к примеру, путем заполнения полей в специальных диалоговых окнах, что избавит пользователя от необходимости вручную включать в файл описания лексики ключевые директивы и разделители. Более подробно требования к инструментальным средствам конструктора ЯП, касающиеся формирования словаря лексем, рассмотрим в соответствующем разделе.

Второй этап анализа языковой конструкции — синтаксический анализ. Синтаксический анализатор определяет структуру языковой конструкции, то есть наличие и тип смысловых связей между отдельными лексемами. Основные требования к синтаксическому анализатору схожи с требованиями к лексическому:

- желательно применение полностью детерминированных методов разбора, что позволит применять высокопроизводительные беступиковые алгоритмы анализа;
- возможность модификации синтаксических БЗ путем добавления единичных правил и слияния с существующими БЗ;
- простота и наглядность представления правил о синтаксисе.

Синтаксический анализ базируется на системе формального вывода, называемой грамматикой. Полный вывод в грамматике — это получение цепочек терминальных символов, исходя из аксиомы, с помощью продукций. Наиболее продуктивные методы вывода найдены для контекстно-свободных (КС) — грамматик. При конструировании грамматики, описывающей некоторый язык, с каждой продукцией может быть связан некий семантический эквивалент. Во первых, нетерминальный символ является формальным отражением некоторой сущности реального мира, которой оперирует язык и грамматика. Во вторых, с продукцией может быть связана некоторая процедура, выполняющая семантические преобразования. Процедура может быть выражена специальным действия, записываемым в фи-

гурных скобках, и который становится одним из символов цепочки α в (3). В этом случае грамматика записывается как

$$G=(VT,VN,VD,P,S), \quad (1)$$

где VD — алфавит символов действия;
P — множество продукций вида $A\rightarrow\alpha$, где

$$\alpha\in(VT\cup VN\cup VD)^* \quad (2)$$

Грамматика (1) называется транслирующей грамматикой. Символы действия, являясь процедурами, должны иметь параметры-аргументы процедуры. В качестве таких аргументов используются атрибуты — специальные символы, приписываемые символу действия. Эти атрибуты получают свои значения от других атрибутов, приписанных к другим символам из цепочки α в (2). Так мы получаем атрибутные транслирующие грамматики, где каждому символу могут быть приписаны специальные переменные-атрибуты, обменивающиеся между собой значениями.

Будем считать, что результатом синтаксического анализа является синтаксическое атрибутное дерево языковой конструкции. Значение атрибутам присваивается путем обхода этого дерева по некоторым правилам, приписанным к каждой продукции. То есть продукция типа (5) состоит из собственно продукции и приписанных к ней правил присваивания значений атрибутам ее символов.

Часто атрибуты, которые получают свои значения снизу-вверх по синтаксическому дереву, называют синтезируемыми, а сверху вниз — наследуемыми. Имеются две основные стратегии вывода в грамматике: свертка и развертка соответственно — $w\Rightarrow^*S$, $S\Rightarrow^* w$. При развертке мы имеем значения наследуемых атрибутов на каждом шаге вывода, а при свертке — значения синтезируемых атрибутов.

Из существующего множества механизмов анализа после некоторого количества опытов по ряду причин была выбрана контекстно-свободная атрибутная транслирующая грамматика с расширенными средствами устранения недетерминизма. Для эффективного ведения анализа применяется метод разбора сверткой (снизу вверх). Применение этого метода позволит обойтись при анализе только синтезируемыми атрибутами, что само по себе снизит трудоемкость анализа из-за отсутствия необходимости дополнительного просмотра дерева вывода с целью вычислить значения некоторых атрибутов, а кроме того, при наличии средств однозначного выбора на каждом шаге применяемой продукции (расширенных средств устранения недетерминизма), позволит полностью детерминизировать процесс анализа.

Под расширенными средствами устранения недетерминизма подразумевается совокупность нестандартных средств, позволяющих обойти неизбежно возникающие при синтаксическом анализе достаточно сложного языка случаи недетерминизма. К ним относятся: дополнительные условия проверки применимости каждой продукции на основе атрибутной информации; средства управления стратегией вывода (назначения приоритетов продукциям) в зависимости от контекстной информации; применение специального вида нетерминальных символов, именуемых условными символами.

Подробнее о сущности каждого из этих средств. Первые представляют собой ничто иное как условия, накладываемые на атрибуты символов, стоящих в правой части продукции. Данные условия вычисляются всякий раз, когда правая часть продукции полностью соответствует анализируемой строке. Невыполнение любого из таких условий запрещает применение данной продукции.

Например, мы определяем известное понятие многочлена продукцией

$$\text{<МНОГОЧЛЕН>} \rightarrow \text{<ОДНОЧЛЕН><ОДНОЧЛЕН>} \mid \text{<ОДНОЧЛЕН><МНОГОЧЛЕН>}, \qquad (3)$$

где $\text{<ОДНОЧЛЕН>} \rightarrow \text{<ЗНАК><СПИСОК СОСТАВЛЯЮЩИХ>}$

$$\text{<ЗНАК>} \rightarrow + \mid - \mid \varepsilon \qquad (4)$$

Продукция (3) должна существовать только в том случае, если выбранная компонента правой части первой альтернативы ОДНОЧЛЕН и МНОГОЧЛЕН из правой части второй альтернативы как строки символов начинаются с + или - . Чтобы удовлетворить это требование, сделаем продукции атрибутивными :

$$\begin{aligned} &\text{<МНОГОЧЛЕН>}\, a \rightarrow \text{<ОДНОЧЛЕН>}a1\ \text{<ОДНОЧЛЕН>}\, a2 \\ &\quad \text{<МНОГОЧЛЕН>}a \rightarrow \text{<ОДНОЧЛЕН>}a1\ \ \text{<МНОГОЧЛЕН>}a2 \qquad (5) \\ &\text{<ОДНОЧЛЕН>}a \rightarrow \text{<ЗНАК>}a1\ \text{<СПИСОК СОСТАВЛЯЮЩИХ>}a2 \end{aligned}$$

Припишем теперь каждой продукции из (5) правила присваивания значения атрибутов и правила существования продукций (там где они есть). При этом мы считаем, что значениями атрибутов являются строки из символов, представляющих собой запись компонентов одночлена и многочлена.

1. $\text{<МНОГОЧЛЕН>}\, a \rightarrow \text{<ОДНОЧЛЕН>}a1\ \text{<ОДНОЧЛЕН>}\, a2$

   f: if $a2(1) =$ ' + ' v ' - ' then f:= true else f := false

   $a \leftarrow conc$ (a1,a2)

2. $\text{<МНОГОЧЛЕН>}a \rightarrow \text{<ОДНОЧЛЕН>}\, a1$

<МНОГОЧЛЕН>$a2$
f: if $a2(1)$ = ' + ' v ' - ' then f:= true else f := false (6)
$a \leftarrow$ conc ($a1$,$a2$)
3. <ОДНОЧЛЕН>$a \rightarrow$ <ЗНАК> $a1$ <СПИСОК СОСТАВЛЯЮЩИХ>$a2$
$a \leftarrow$ conc ($a1$,$a2$)

Продукции 1, 2 из (9) имеют два правила. Первое устанавливает условия существования продукции (если f = true), второе присваивает значение атрибуту a как результат конкатенации двух строк, являющихся значениями атрибутов a1 и a2 соответственно.

Средства управления стратегией вывода присутствуют и в системе Jacc в виде назначения приоритетов правилам вывода, однако там они задаются константно, то есть значения этих приоритетов невозможно изменить в процессе анализа. Такой подход достаточен, если необходимо всего лишь установить приоритет выполнения арифметических операций, но для некоторых приложений он непригоден. Предлагается рассчитывать приоритеты продукций как функции от текущего контекста, что достигается введением дополнительных процедурных элементов. Естественно, что рассчитываться значения этих функций будут только для продукций, между которыми возник конфликт на очередном шаге вывода.

Условные символы представляют собой элемент анализа более высокой степени абстрактности, чем просто нетерминалы. Условный символ представляет собой шаблон для нетерминалов определенного класса и включается в дерево вывода в случае, если на данном шаге вывода нет достаточного количества информации для выбора конкретного нетерминала из этого класса. По сути дела он является «заглушкой» в дереве вывода, конкретное значение которой вычисляется при получении нужной контекстной информации (во время применения какой-либо другой продукции, в теле которой имеется соответствующее присваивание). Применение условных символов во многих случаях позволит избежать перебора в процессе анализа из-за недостаточно определенного контекста на конкретном шаге вывода.

Рассмотрим пример введения условных символов в продукции грамматики. В естественном языке имеются предложения типа

ПОДЛЕЖАЩЕЕ СКАЗУЕМОЕ ДОПОЛНЕНИЕ,

такие как

"Водитель ведет автобус". (7)

В некоторой грамматике мы без труда построим продукции :
<ПРЕДЛОЖЕНИЕ>→
<ПОДЛЕЖАЩЕЕ><СКАЗУЕМОЕ><ДОПОЛНЕНИЕ>|
<ДОПОЛНЕНИЕ><СКАЗУЕМОЕ><ПОДЛЕЖАЩЕЕ>

ЮДЛЕЖАЩЕЕ> → <СУЩЕСТВИТЕЛЬНОЕ> *t*1
<ДОПОЛНЕНИЕ> → <СУЩЕСТВИТЕЛЬНОЕ> *t*2 (8)

(11) должно иметь значение "Именительный падеж", *t*1 — любой падеж, оме именительного. Вторая альтернатива продукции 1 в (8) выбрана с етом

"Автобус ведет водитель" (9)

С помощью продукций (8) мы не сможем сделать анализ предложений) и (9), так как знания о падежах не четкие:

Водитель — именительный или винительный падеж, Автобус — имеительный или винительный падеж.

Для устранения недетерминизма мы должны в продукцию (1) из (8) вести некоторое решающее правило. Но и этого мало, если мы будем вести нализ сверткой. Продукции (2), (3) из (8) задают нам необратимую грамматику. И на этом шаге анализа введенное нами правило в продукции 1 ало чем может помочь. Здесь мы вводим условный символ *X*:

ПРЕДЛОЖЕНИЕ> → *X t*1 <СКАЗУЕМОЕ> q *X t*2
X t → <СУЩЕСТВИТЕЛЬНОЕ>t1 (10)

Снабдив (10) правилами будем иметь :

<ПРЕДЛОЖЕНИЕ> → *X t*1 <СКАЗУЕМОЕ>q *X t*2

f — для глагола "ведет" при одинаковых падежах *t*1 и *t*2 (именительный, винительный) в качестве подлежащего выбираем живое существо или автоматическое средство управления; в качестве дополнения — транспортное средство.

2. *X t* → <СУЩЕСТВИТЕЛЬНОЕ> *t*1 (11)
t → *t*1

В результате введения условного символа нам удалось избежать недетерминизма, но за счет ведения правила, оперирующего семантическими значениями языковых конструкций.

Таким образом, продукция применяемой для синтаксического анализа грамматики будет иметь вид:

<НЕТЕРМИНАЛ > → α,

где α ∈(VT∪VN∪VD∪Vусл)* и каждый символ может иметь атрибуты. С продукцией связаны правила :

- правила получения значений атрибутами;
- отношения между значениями атрибутов (дополнительные условия применимости продукции);
- правила получения значений условными символами}.

В левой части продукции расположен определяемый продукцией нетерминальный символ, правая часть разделена на три фрагмента: тело продукции (шаблон, определяющий принципиальную возможность применения данной продукции — последовательность терминальных, нетерминальных и условных символов, которая заменяется в строке вывода определяемым нетерминалом), транслирующую часть (символы действия представляют собой имена стандартных или определенных пользователем процедур, выполняемых при применении данной продукции) и пространство расчета атрибутов (здесь производится вычисление значений атрибутов, дополнительных условий, значений условных символов).

На значениях атрибутов остановимся подробнее. В качестве единственного формата данных для их хранения используется строковый тип, так как при помощи последовательности символов можно записать любую информацию. Требуемая гибкость при работе с атрибутами достигается использованием пополняемой пользователем библиотеки функций обработки строк.

В результате применяемая нами грамматика может быть описана указанием алфавитов терминальных, нетерминальных и условных символов, используемых в ее продукциях, перечнем продукций, описывающих правила синтаксиса языка и необязательным фрагментом управления стратегией вывода, определяющим приоритеты продукций в зависимости от контекстной информации.

Список литературы: 1. *Перспективы* развития вычислительной техники: Справ. пособие / Под ред. Ю.М.Смирнова. Кн.2. Интеллектуализация ЭВМ: Е.С.Кузин, А.И.Ройтман, И.Б. Фоминых, Г.Хахалин. М.: Высш. шк. 1989. 159 с. 2. *Представление* и использование знаний / Х.Уэно, Т.Кояма, Т.Окамото и др. М.: Мир, 1989. 220 с. 3. *Конопка Рэй.* Искусство создания мощных программных компонент в Delphi/ Пер. с англ., К: НИПФ"Диасофт", 1996. 4. *Ахо А.В., Ульман Д.Д.* Теория синтаксического анализа, перевода и компиляции.: В 2-х т. / Пер. с англ. М.: Мир, 1978.

Поступила в редколлегию 23.10.97