МЕТОДЫ ОБРАБОТКИ ИНФОРМАЦИИ
(Вычислительные системы)
1978 год Выпуск 74

УДК 519.688

АВТОМАТИЧЕСКИЙ СИНТЕЗ АЛГОРИТМОВ ПО ДИНАМИЧЕСКИМ ВХОДНЫМ ДАННЫМ

Ю.Г.Косарев, Н.А.Чуканова

В настоящее время накоплен опыт применения R-технологии программирования для задач построения трансляторов [1]. По подходу эти задачи можно отнести к дедуктивным: задана грамматика (синтаксически - управляемый алгоритм) и порождаются возможные в данной грамматике конструкции, либо проверяется соответствие некоторой конструкции заданной грамматике.

Представляет интерес рассмотреть другой класс задач, основанных на индуктивном подходе, когда в качестве исходной информации задается не грамматика, а ее возможные порождения, по которым необходимо создать грамматику.

При индуктивном подходе обычно приходится иметь дело с большими объемами входных данных (порождений), поэтому весьма важно автоматизировать процесс синтеза грамматик.

Задача автоматизации синтеза R-грамматик типа дерева по статическим входным данным, когда все возможные порождения известны и число их фиксировано, рассмотрена в [2-4].

В данной работе обсуждается решение этой задачи с динамическими входными данными, когда по мере анализа новых порождений требуется переопределение ранее построенной грамматики.

При решении этой задачи важно обеспечить простоту переопределения грамматики и достаточную компактность, чтобы при большом числе порождений не выходить за рамки оперативной памяти. В этом случае также открывается возможность использования в процессе синтеза самой синтезируемой грамматики.

Схема синтеза в этом случае состоит из n последовательных шагов (по числу исходных порождений), на каждом из которых ал-

горитм синтеза АС переопределяет грамматику G_{i-1}, построенную на предыдущем шаге

$$AC(G_{i-1}, p_i, Y(S)) \Rightarrow G_i \qquad (i = 1, \ldots, n),$$

где p_i - очередная исходная последовательность языка; Y(S) - условия, при которых правилам грамматики присваиваются синтермы из заданного списка S.

I. Приведем некоторые понятия R-языка, которые потребуются далее.

Т е р м ы для исследуемых задач - это символы алфавита языка, цифры, знаки препинания и пустой символ.

С и н т е р м ы - некоторые выделенные подмножества термов, например,

буква::= А|В|...|Я|а ... я ,
цифра::= 0|I|...|9|.

С е м а н т и к а - действия, которые необходимо выполнить для преобразования входной информации.

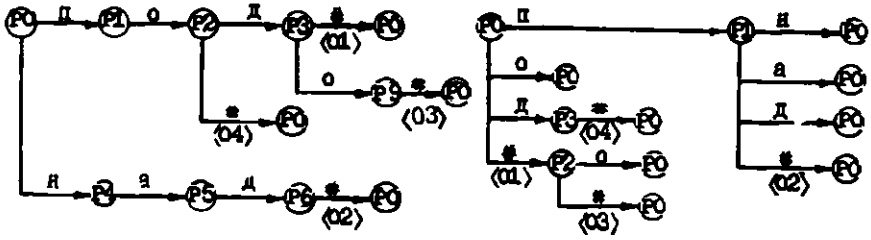
П р а в и л о R - я з ы к а - это четверка вида ⟨терм или синтерм⟩ ⟨семантика⟩ ⟨имя правила-преемника по совпадению⟩ ⟨неявно задаваемое имя правила-преемника по несовпадению⟩.

К о м п л е к с п р а в и л - упорядоченное множество правил. В комплексе объединяются правила-преемники по несовпадению. Каждый комплекс имеет имя. Имя первого правила комплекса совпадает с именем комплекса. Имена остальных правил комплекса задаются неявно (позицией, занимаемой правилом в упорядоченном множестве правил, входящих в комплекс). Последнее правило комплекса выделяется специальным признаком.

Г р а ф и ч е с к а я ф о р м а п р е д с т а в л е н и я R - я з ы к а - это ориентированный граф с нагруженными дугами. Вершины графа соответствуют комплексам правил, имя которого указанно в вершине, дуги графа - правилам. Правилам, входящим в один комплекс, соответствуют дуги, исходящие из одной вершины. Каждая дуга помечена термом либо синтермом и нагружена семантикой.

А н а л и т и ч е с к а я ф о р м а п р е д с т а в л е н и я R - я з ы к а - R -таблица.

2. Автоматический синтез алгоритмов, описанных на R-языке (R-грамматик), по динамическим входным данным встречает определенные трудности. R-грамматики строятся комплексами, в которые вхо-



Имя комплекса	Символ	Семантика	Имя преемника по совпадению
Р0	п		Р1
	ε		Р4
Р1	о		Р2
	ε		Р0
Р2	д		Р3
	ε		Р10
Р3	*	⟨01⟩	Р0
	ε		Р8
Р4	я		Р5
	ε		Р0
Р5	а		Р6
	ε		Р0
Р6	д		Р7
	ε		Р0
Р7	*	⟨02⟩	Р0
	ε		Р0
Р8	о		Р9
	ε		Р0
Р9	*	⟨03⟩	Р0
	ε		Р0
Р10	*	⟨04⟩	Р0

а)

Имя комплекса	Символ	Семантика	Имя преемника по несовпадению
Р0	п		Р1
	о		Р0
	д		Р3
	*	⟨01⟩	Р2
Р1	я		Р0
	а		Р0
	д		Р0
	*	⟨02⟩	Р0
Р2	о		Р0
	*	⟨03⟩	Р0
Р3	*	⟨04⟩	Р0

б)

Рис.1. Пример графической и аналитической форм представления в R-языке (а) и RD-языке (б) списка слов: под * ⟨01⟩ над * ⟨02⟩ подо * ⟨03⟩ по * ⟨04⟩.

дят правила-преемники по несовпадению, соответствующие разным входным конструкциям. Поэтому состав и число правил в комплексах определяются лишь после анализа в с е х входных конструкций (порождений).

Обеспечить средствами R-языка динамическое достраивание R-грамматик без их перестройки можно лишь путем некоторых ухищрений. Например, после каждого правила ставить ε-оператор R-языка (оператор безусловного перехода) [I], что равносильно введению правил, в которых явно задаются имена обоих правил преемников. Это, естественно, приводит к значительному увеличению объема грамматик. Правила комплекса оказываются записанными не подряд и объединяются с помощью относительных адресов, что, в свою очередь, увеличивает время выбора нужного правила.

Нами предлагается некоторая модификация R-языка – RD-язык описания грамматик (RD-грамматики), позволяющий избежать указанных недостатков. RD-язык отличается от R-языка следующим:

- имя правила-преемника по совпадению задается неявно;
- имя правила-преемника по несовпадению задается явно;
- в комплексе объединены правила-преемники по с о в п а д е н и ю.

По сути, изменение свелось к перемене местами правил-преемников по совпадению и несовпадению. Это, однако, привело к изменению графической и аналитической форм описания грамматик. Как можно видеть из примера (рис.I), RD-грамматика типа дерева строится в динамике без привлечения дополнительных операторов и содержит вдвое меньше правил.

3. Рассмотрим задачу автоматизации синтеза алгоритмов на RD-языке. В данной работе рассматривается частный случай решения этой задачи, определяемый следующими особенностями синтезируемого (АХ) и синтезирующего (АС) алгоритмов:

АХ алгоритм имеет структуру дерева и строится из одного типа правил RD-языка. Все правила имеют фиксированную структуру и одинаковый формат при аналитической форме представления. Структурно правила состоят из четырех полей: А0 для имени (адреса) правила; А1 для символа (терма или синтерма); А2 для имени семантики; А3 для имени (адреса) правила преемника по несовпадению. Наборы используемых семантик и синтермов ограничены предварительно задаваемыми списками.

АС алгоритм строит АХ алгоритм по следующим исходным данным:

- список (Р) последовательностей языка, по которому должна быть построена RD-грамматика. Каждая последовательность $p_i \in P$ может сопровождаться дополнительной информацией $d_i \in D$, где D-фиксированное множество служебных слов либо семантик;

- список (S) семантик;

- алгоритм (АУ) на R- или RD-языке, который для каждой цепочки r правил $r_1 \ldots r_k$, возникающей при анализе синтезируемой RD-грамматикой очередной входной последовательности языка p_i, указывает список пар (r_i, s_j), где r_i - имя правила из цепочки r, s_j - имя семантики из S.

К процессу синтеза алгоритма АХ, кроме работы по схеме [I], предъявляются некоторые дополнительные требования:

- возможность включения алгоритма синтеза АС в более общий алгоритм;

- возможность организации рекурсивного режима работы, когда множество выходных цепочек, порождаемых синтезируемым алгоритмом, рассматривается как входное множество Р и т.д. При этом остальные исходные данные могут либо оставаться постоянными, либо для каждой рекурсии должны быть заданы свои исходные данные;

- возможность задания множества Р в виде RD-алгоритма;

- возможность (в тех случаях, когда это допустимо) задания вместо алгоритма АУ множества (r) цепочек правил и соответствующих им семантик в некотором стандартном виде (в том же, что и для множества Р), позволяющем предварительно синтезировать алгоритм АУ с помощью того же синтезирующего алгоритма.

Процесс синтеза алгоритма можно представить в виде трех уровней (рис.2): I-й уровень - решение конкретной задачи; 2-й уровень - автоматический синтез алгоритма I-го уровня. Он порождает класс конкретных алгоритмов в зависимости от входных данных; 3-й уровень - синтез алгоритма 2-го уровня, который сводится к компиляции, редактированию связей в соответствии с управляющими параметрами, отражающими указанные выше требования к процессу синтеза.

В процессе работы создаются библиотеки алгоритмов I-го и 2-го уровней и библиотека процедур 3-го уровня, пополнение которых сокращает число обращений к верхним уровням.

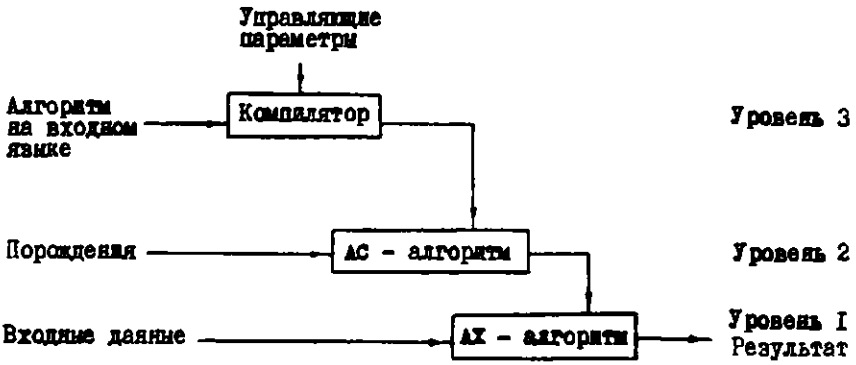


Рис. 2

4. Синтезирующий алгоритм (АС) написан на языке R-TRAN [I] и состоит из двух подграмматик:

RSYNTAX, которая на каждом шаге построения AX-алгоритма строит RD-грамматику для синтаксического анализа (т.е. правила с пустыми семантиками);

RSEMAN, которая приписывает правилам семантики, используя задаваемый пользователем алгоритм AYS .

В алгоритме АС используются следующие процедурные операции:

-P#COMP(T.A1,RR): ИДЕР1 ⌴ ИДЕР2 - сравнение символов. Содержимое поля A1 доступной строки RD-таблицы T сравнивается с содержимым регистра RR. При совпадении - переход на комплекс правил с именем ИДЕР1, иначе - на ИДЕР2.

-P#PACK (T.A,RR) - упаковка символов (A=A1), семантик (A=A2) адреса (A=A3). Содержимое регистра RR записывается в поле A RD-таблицы T (соответствует операции T.A←RR R-языка).

-P#KIS (T.A) - вычеркивание символов, семантик и адреса. Содержимое поля A RD-таблицы T обнуляется (соответствует операция T.A←∅'∅'X R-языка).

-NEXT(T) - установление очередной строки RD-таблицы по совпадению (соответствует стандартной процедуре R#TGO(T) R-языка).

-NNEXT(T): ИДЕР1 ⌴ ИДЕР2 - установление очередной строки RD-таблицы T по несовпадению. Содержимое поля A3 RD-таблицы T (адрес правила преемника по несовпадению) сравнивается с нулем. При T.A3≠0 - переход на комплекс правил с именем ИДЕР1, иначе - на ИДЕР2.

- FREE(T) - установление свободной строки RD-таблицы T.

- NFREE(T) - установление строки RD-таблицы по адресу из T.A3.

5. Проиллюстрируем синтез алгоритмов в виде RD-грамматик на примере решения некоторых задач прикладной лингвистики.

ЗАДАЧА I. По обучающей выборке-списку парадигм вида ⟨каноническая форма⟩: ⟨словоформа⟩,..., ⟨словоформа⟩

- необходимо разбить парадигмы на типы по способам образования словоформ из канонической формы;

- поставить в соответствие каждой словоформе номер типа словоизменительной парадигмы, определяющей множество преобразований канонической формы;

- поставить в соответствие графеме словоформы код, отражающий ее грамматический класс (существительное, глагол и т.д.) и тип (род, число, падеж - для существительных и т.п.) (в данной работе вопросы омонимии пока не рассматриваются);

- синтезировать алгоритм.

С и н т е з и р у е м ы й а л г о р и т м (алгоритм I-го уровня) предназначен для выполнения следующих функций:

- отнесение словоформы текста к типу словоизменительной парадигмы;

- перекодирование графемы с учетом ее грамматического класса и типа;

- проверка правильности написания словоформы.

Синтезирующий алгоритм (рис.3) использует специальную семантику SEM1, функции которой состоят в следующем.

Поступившая на вход каноническая форма записывается в RD - таблицу канонических форм (RDKF), начиная с первого символа. В поле семантик каждого символа записывается код поступившей канонической формы. Длина канонической формы в символах запоминается. Специальная семантика SEM2 осуществляет сравнение символов словоформы с RDKF.

При первом несовпадении входного символа с символом RDKF осуществляется вход в RD-таблицу окончаний (RDO). По этой таблице словоформа анализируется до конца. Возможны следующие случаи:

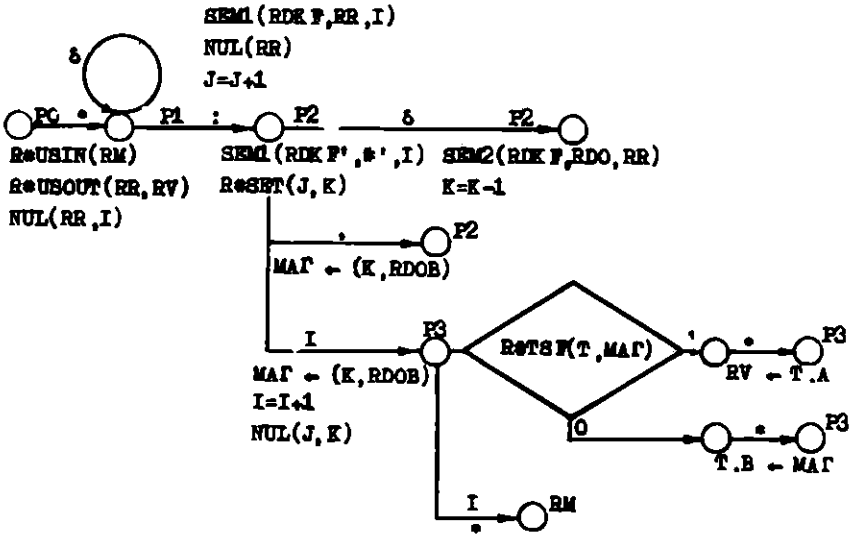


Рис. 3

I) окончание отсутствует в RDO. Поступившее окончание упаковывается в RDO, окончанию присваивается некоторый код, правило образования словоформы и код окончания записываются в таблицу правил;

2) окончание имеется в RDO. Правило образования и код окончания записываются в таблицу правил.

Для каждой парадигмы заполняется строка таблицы правил. Затем строки сравниваются и парадигмы с одинаковыми правилами образования относятся к одному типу. Номер типа присваивается канонической форме и всем словоформам.

Следующая каноническая форма поступает на вход RDKF .В случае совпадения входных символов с символами RDKF коды в поле семантик стираются. При первом же несовпадении символов доступной объявляется любая свободная строка RDKF , конец слова упаковывается, а в поле семантик записывается код этой канонической формы.

ЗАДАЧА 2. По обучающей выборке – списку канонических форм с соответствующим типом словоизменительной парадигмы

– построить формальные правила, описывающие типы словоизменительных парадигм;

– синтезировать алгоритм.

С и н т е з и р у е м ы й а л г о р и т м предназначен для классификации канонических форм по типам словоизменительных парадигм и восстановления парадигмы из канонической формы.

Синтезируемый алгоритм (рис.4) выполняет следующие действия:

– каноническая форма переписывается в некоторый регистр RV;

– номер типа парадигмы, к которой относится каноническая форма, запоминается в счетчике I;

– семантика ReINVR(RR, RV) генерирует содержимое регистра RV в обратном порядке и записывается в регистр RR;

– регистр RR определяется как входной.

При работе специальной семантики ЯИЗ (рис.5) возможны следующие случаи:

1) RD–таблица пуста. В этом случае содержимое RR последовательно переписывается в RD–таблицу, а номер типа из счетчика I – в поле A2;

2) в RD–таблице имеется запись. Первый символ из RR подается на вход RD таблицы и сравнивается с содержимым поля A1. В случае совпадения сравниваются типы. При совпадении типов доступной объявляется следующая строка RD–таблицы, иначе номер типа в поле A2 стирается и доступной объявляется любая свободная строка таблицы. Номер строки записывается в поле A3, а содержимое RR переписывается в RD–таблицу, начиная со свободной строки.

Если поле A3 непусто, то осуществляется переход по адресу из A3 и процесс анализа продолжается.

Запись в поле A2, отличная от нуля, означает, что выделена последовательность, которая отличает данный тип парадигмы от всех остальных.

ЗАДАЧА 3. По обучающей выборке – тексту, состоящему из правильно построенных предложений,

– построить формальные правила сопряжения последовательности словоформ в тексте;

– синтезировать алгоритм.

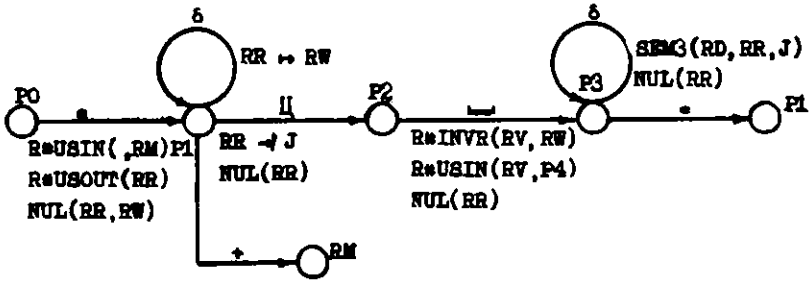

δ
RR ↦ RW
δ
SEM3(RD,RR,J)
NUL(RR)
P0
a
P2
P3
P1
R*USIN(,RM)P1
R*USOUT(RR)
NUL(RR,RW)
RR ≠ J
NUL(RR)
R*INVR(RV,RW)
R*USIN(RV,P4)
NUL(RR)
RM

Рис. 4

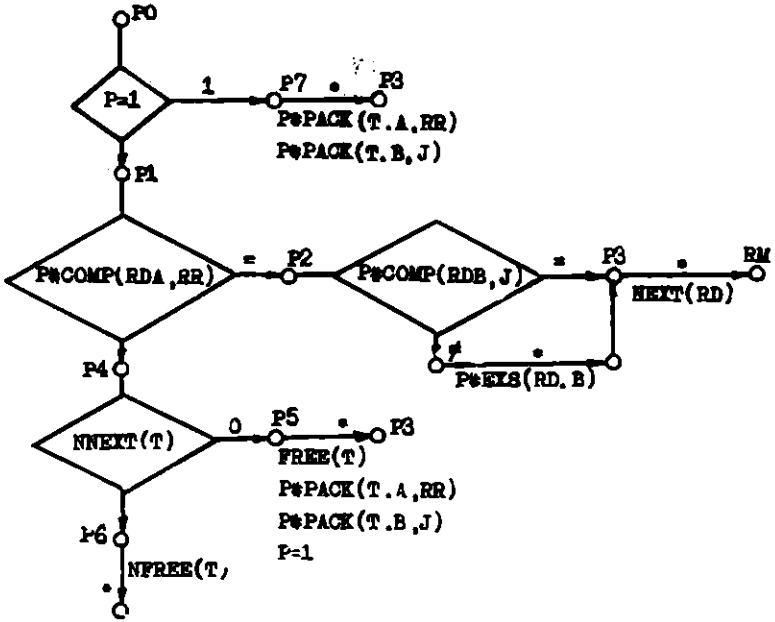

P0
P=1
1
P7
P3
P*PACK(T.A,RR)
P*PACK(T.B,J)
P1
P*COMP(RDA,RR)
=
P2
P*COMP(RDB,J)
=
P3
RM
NEXT(RD)
≠
P*EXS(RD.B)
P4
NNEXT(T)
0
P5
P3
FREE(T)
P*PACK(T.A,RR)
P*PACK(T.B,J)
P=1
P6
NFREE(T,

Рис. 5

С и н т е з и р у е м ы й а л г о р и т м предназначен для выполнения следующих функций:

- перекодирование словоформ входного текста с учетом их грамматического класса и типа;
- проверка правильности написания словоформ;
- проверка правильности сопряжения словоформ в предложении.

Синтезирующий алгоритм использует ранее синтезированный алгоритм задачи I. После перекодирования словоформ текста с учетом их грамматического класса и типа строится RD-таблица кодов.

6. Рассмотренные примеры показывают, что и в рамках наложенных нами ограничений, существует возможность решения определенных классов практических задач. Этот круг задач может быть расширен путем смягчения ограничений. Например, представляет интерес синтез грамматик более общего типа, чем типа дерева. Здесь могут быть два основных пути: ввести в RD-грамматиках ε-операторы (условного перехода) либо синтезировать R-грамматики, допуская некоторую их перестройку. Например, использовать ε-операторы не после каждого правила, а употреблять их только при появлении нового правила в комплексе. В этом случае ε-оператор ставится на место последнего правила в комплексе, которое отправляется (вместе с новым правилом) на очередное свободное место.

Для многих задач, в том числе и для рассмотренных выше, может оказаться важным введение в синтезируемый алгоритм синтермов. Это может сделать алгоритм более компактным и быстродействующим и, что более существенно, обобщить алгоритм на более широкий круг порождений. Поясним последнее. В R-языке правила из одного комплекса с различными термами, T1,... ,Tk, но с одинаковыми семантиками и правилами-преемниками могут заменяться одним правилом с соответствующим синтермом C::=T1| ... |Tk. Если зафиксировать заранее некоторое множество синтермов C1,...,Cм, то возникает задача найти для термов T1,...,Tk синтерм Cj, наиболее полно их покрывающий. Степень полноты покрытия задается числом q, характеризующим отношение числа термов из данного комплекса, входящих в синтерм, к общему числу термов в данном синтерме. В RD-грамматиках правила, которые могут объединяться в одно с помощью введения синтерма, находятся в разных комплексах, поэтому при динамическом построении введение синтерма связано с перестройкой RD-таблицы.

Литература

1. ГЛУШКОВ В.М., ЭЛЬБИЦКИЙ И.В. Технология программирования и проблемы ее автоматизации. – "Управляющие системы и машины", 1976, № 6, с.75-93.

2. ХАБАРОВ В.В., КОСАРЕВ Ю.Г. Об эффективности автоматического кодирования и исправления ошибок при подготовке данных. –В кн.: Вычислительные системы. Вып.62. Ассоциативное кодирование.Новосибирск, 1975, с.106-118.

3. КОСАРЕВ Ю.Г., ХАБАРОВ В.В. Применение В-метаязыка для кодирования, контроля и коррекции текстовой информации. –В кн.: Теория и практика системного программирования, Киев, 1975, с.131-139.

4. КОСАРЕВ Ю.Г., ХАБАРОВ В.В., ДУБОВСКАЯ Н.И., ШВЕДЧИКОВ С.И. Автоматизация составления средств кодирования, контроля и коррекции текстовой информации. –В кн.: Теория и практика системного программирования, Киев, 1976, с. 139-150.

Поступила в ред.-изд.отд.
24 апреля 1978 года