Ю. Г. КОСАРЕВ, зав лабораторией,
Институт математики СО АН СССР,
Н. А. ЧУЖАНОВА, асп.,
Новосибирский университет

АВТОМАТИЧЕСКИЙ СИНТЕЗ АЛГОРИТМОВ КЛАССИФИКАЦИИ СЛОВОФОРМ ПО ТИПАМ СЛОВОИЗМЕНИТЕЛЬНЫХ ПАРАДИГМ

Решение задач прикладной лингвистики приобретает все большее значение не только для самой лингвистики, но и для многих важных сфер применения, среди которых достаточно назвать общение человека и машины на естественном языке и автоматизацию редакционно-издательских работ. В связи с этим в прикладной лингвистике возникают довольно сложные задачи, связанные с анализом текстов и созданием вспомогательных средств в виде различного рода словарей для кодирования, контроля и исправления ошибок.

Решение этих задач, как правило, приводит к весьма трудоемким процессам: подготовке для ввода в ЭВМ больших объемов информации и сложному, трудно доступному для лингвистов программированию. Отсюда возникает проблема создания таких методов, которые позволили бы снизить трудоемкость указанных процессов. С этой целью далее предлагается метод автоматиче-

ского синтеза алгоритмов на примере решения задач обнаружения типов словоизменительных парадигм и закономерностей отнесения словоформ к типу.

Метод основывается на применении R-технологии [1, *75—93*], видоизмененной применительно к задачам лингвистики. В этом отношении данная статья является дальнейшим развитием работ [3—5], в которых было продемонстрировано удобство и эффективность применения *R*-языка для контроля, кодирования и коррекции текстов, а также для автоматического составления необходимых для этого словарей.

Описание структуры данных в *R*-языке является одновременно и алгоритмом синтаксического анализа. Именно это свойство *R*-языка используется как в [3—5], так и в данной работе. Структура данных в исследуемых задачах автоматического синтеза и сам процесс синтеза алгоритма сложнее, чем в [3—5]. Построение синтезируемого алгоритма ведется динамически в ходе анализа. Алгоритм при этом наращивается по мере поступления исходных данных без существенных изменений уже построенной части алгоритма.

1. Введем некоторые понятия *R*-языка.

Термы для исследуемых задач — это символы алфавита языка, цифры, знаки препинания и пустой символ.

Синтермы — некоторые выделенные подмножества термов, например: буква = А|Б|В|...|Я|а|б|...|я; цифра = 0|1|2|...|9.

Семантики — действия, которые необходимо выполнить для преобразования входной информации.

Правило R-языка — это четверка вида

<терм или синтерм> <семантика> <имя правила-преемника по совпадению> <неявно задаваемое имя правила-преемника по несовпадению>.

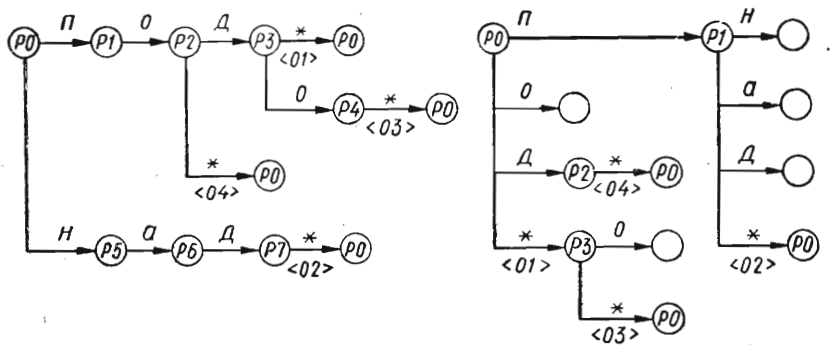
Комплекс правил — упорядоченное множество правил. В комплексе объединяются правила, каждое из которых (кроме первого) является правилом-преемником по несовпадению. Каждый комплекс имеет имя. Имя первого правила комплекса совпадает с именем комплекса. Имена остальных правил комплекса задаются неявно позицией, занимаемой правилом в упорядоченном множестве правил, входящих в комплекс. Последнее правило комплекса выделяется специальным признаком.

Графическая форма представления R-языка — это ориентированный граф с нагруженными дугами. Вершины графа соответствуют комплексам правил, имя которых указано 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.

Аналитическая форма представления R-языка — *R*-таблица (рис. 1).

Построение алгоритмов в виде *R*-таблиц встречает определенные трудности. *R*-таблицы строятся комплексами, в которые

входят правила, соответствующие разным исходным последовательностям. Поэтому состав и число правил в комплексах определяются лишь после анализа всех исходных последовательностей.



Имя комплекса	Условие перехода	Семантика	Код преемника
Р0	п		Р1
			Р4
Р1	о		Р2
Р2	д		Р3
			Р10
Р3	*	<01>	Р0
			Р8
Р4	н		Р5
Р5	а		Р6
Р6	д		Р7
Р7	*	<02>	Р0
Р8	о		Р9
Р9	*	<03>	Р0
Р10	*	<04>	Р0

а)

Имя комплекса	Условие перехода	Семантика	Код преемника
Р0	п		Р1
	о		
	д		Р3
	*	<01>	Р2
Р1	н		
	а		
	д		
	*	<02>	Р0
Р2	о		
	*	<03>	Р0
Р3	*	<04>	Р0

б)

Рис. 1. Графическая и аналитическая формы представления в *R* (а) и *RD* (б) языках.

Нами предлагается некоторая модификация *R*-языка — *RD*-язык, удобный для динамического синтеза алгоритмов. *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-преемника по *совпадению*;

— имеются правила с пустым именем правила-преемника по

несовпадению, которые соответствуют последнему правилу комплекса в *R*-языке.

По сути, изменение свелось к перемене местами правил-преемников по совпадению и несовпадению. Это привело к изменению графической и аналитической форм представления языков (рис. 1).

На рис. 1 показан пример составления *R*- и *RD*-таблиц по мере поступления слов: *под, над, подо, по.* Как можно видеть из данного примера, при таком динамическом составлении *R*-таблицы для сохранения уже составленной части приходится почти удваивать объем таблиц. *RD*-таблица в этом случае получается более компактной.

В исследуемых задачах входной информацией являются некоторые исходные символьные последовательности. В *R*-таблицу вся исходная символьная последовательность упаковывается сразу, и, что особенно важно, она полностью определяет состав и размеры соответствующего комплекса.

2. Объектом для автоматического синтеза алгоритмов служат следующие задачи, возникшие в практике работы отдела структурно-математической лингвистики Института языковедения АН УССР.

Задача 1. По обучающей выборке — списку парадигм вида <каноническая форма> : <словоформа>, ..., <словоформа>

— 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.

Синтезируемый алгоритм 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.

Входной информацией служит словоформа текста. Алгоритм представляется в виде *RD*-таблицы канонических форм, использующей семантики обращения к *RD*-таблице окончаний и таблице правил. Каждому типу парадигмы соответствует строка таблицы правил, столбцы которой состоят из двух частей: первая часть определяет количество отнимаемых от канонической формы букв, вторая — код добавляемого окончания.

Синтез алгоритма осуществляется в соответствии с *R*-графом (рис. 2) и состоит в следующем. Поступившая на вход каноническая форма записывается в *RD*-таблицу канонических форм (*RDKF*), начиная с первого символа. В поле семантик каждого

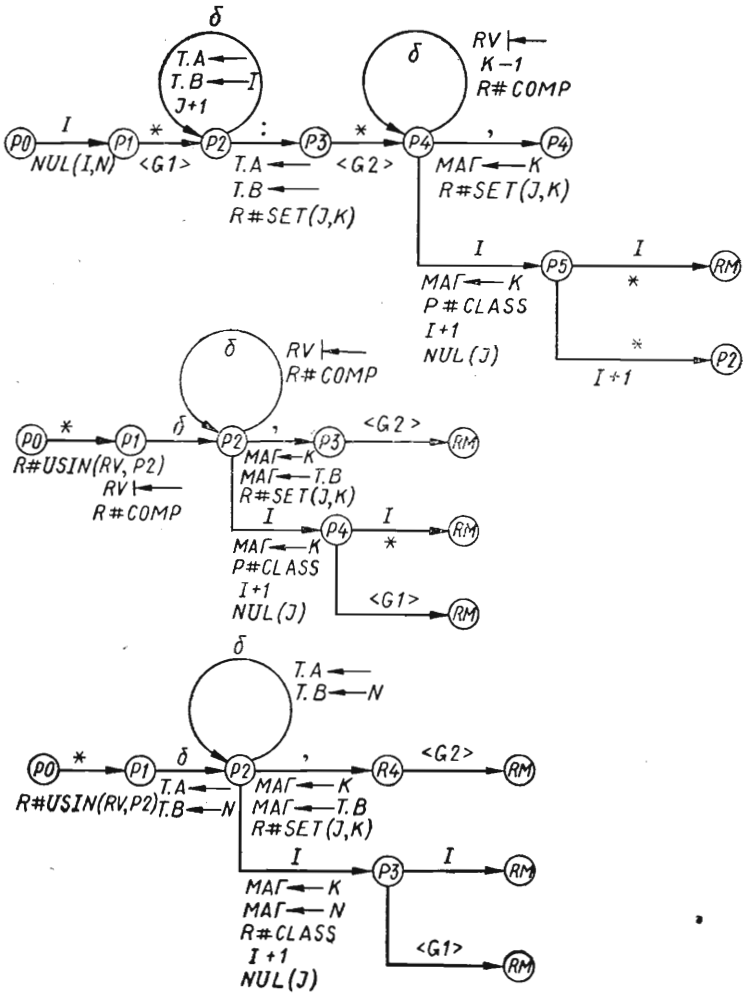


Рис. 2. R-граф алгоритма задачи 1.

символа записывается код поступившей канонической формы. Длина канонической формы в символах запоминается. Затем на *RDKF* посылаются все возможные преобразования канонической формы. 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: 1) 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 словоформы и код окончания записываются в таблицу правил.

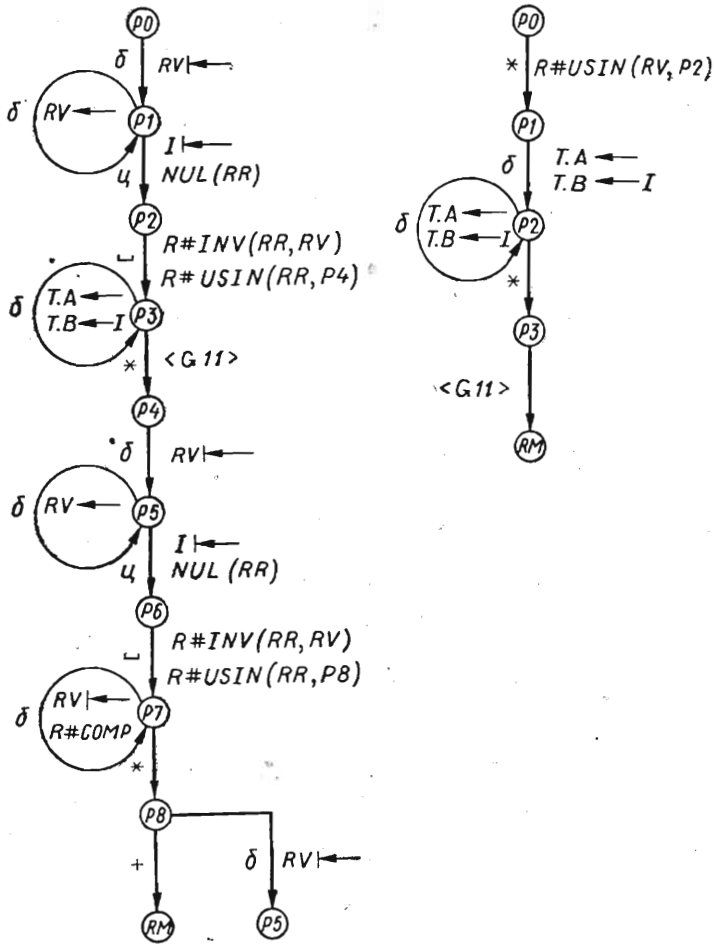


Рис. 3. R-граф алгоритма задачи 2.

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 случаях 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.

Процедурные операции

Наименование	Функции	Тип
NUL (R1, ..., I, K)	обнулить содержимое указанных параметров	стандартная
R#USIN (RR, PN)	определить входную строку	стандартная
R#SET (I, K)	переслать содержимое счетчика *K* в счетчик *I*	стандартная
R#INV (RR, RV)	генерировать содержимое *RV* в обратном порядке и записать в *RR*	специальная
R#COMP	записать символ в *R*-таблицу	специальная
R#CLASS	присвоить код	специальная

Синтезируемый алгоритм предназначен для классификации канонических форм по типам словоизменительных парадигм и восстановления парадигмы из канонической формы.

Входной информацией является каноническая форма. Алгоритм представляется в виде RD-таблицы канонических форм, упакованных с конца. После выявления в новой канонической форме цепочки символов, полностью определяющих принадлежность к типу, ей присваивается этот тип и восстанавливаются словоформы парадигмы.

Синтез алгоритма осуществляется в соответствии с *R*-графом (рис. 3) и состоит в следующем:

— каноническая форма переписывается в некоторый регистр *RV*;

— номер типа парадигмы, к которой относится каноническая форма, запоминается в счетчике 1;

— семантика *RD#NV (RR, RV)* генерирует содержимое регистра *RV* в обратном порядке и записывает в регистр *RR*;

— регистр *RR* определяется как входной.

Возможны следующие случаи: 1) *RD*-таблица пуста. В этом случае содержимое *RR* посимвольно переписывается в *RD*-таблицу, а номер типа из 1 — в поле семантик; 2) в *RD*-таблице имеются записи. Первый символ из регистра *RR* подается на

вход RD-таблицы и сравнивается с полем условий. В случае совпадения сравниваются типы. Если типы совпали, то доступной объявляется следующая строка *RD*-таблицы. В противном случае номер типа в поле семантик стирается и доступной объявляется следующая строка *RD*-таблицы.

Если входной символ не совпал с символом *RD*-таблицы, то доступной объявляется любая свободная строка *RD*-таблицы, номер этой строки записывается в код преемника, а содержимое *RR* переписывается в *RD*-таблицу со свободной строки.

Если в поле преемника по несовпадению уже есть запись, то осуществляется переход на строку, записанную в поле преемника, и процесс анализа продолжается.

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

При синтезе использовались процедурные операции (см. табл.), операции присваивания, пересылки, копирования [2], магазинная память с именем *МАГ*, табличная память *T*, счетчики *I, N, J, K*, регистровые памяти *RV* и *RR*.

3. Предложенные алгоритмы фактически позволяют решать не только задачи парадигматики, но и классы задач, структура данных которых может быть представлена в виде A_iO_i или O_iZ_i, где A_i — начальные части последовательностей; O_i — средние части; Z^i — конечные части последовательностей (для словоформ это соответственно префиксоиды, основы и суффиксоиды).

Таким образом, лингвисты и другие специалисты, не владеющие программированием, с помощью указанных методов могут решать следующие классы задач:

— для последовательностей вида A_iO_i или O_iZ_i на основе автоматического анализа их формальных признаков строить разбиение на классы. В качестве классов могут выступать части речи, типы словоизменительной парадигмы и т. п.;

— определять формальные признаки отнесения к классам;

— перекодировать словоформы текста с учетом их грамматических свойств.

Предложенные алгоритмы могут использоваться для поиска закономерностей и автоматического синтеза алгоритмов анализа более сложных структур данных, например, для решения задач синтагматики. Для этого тексты перекодируются и приводятся к виду, описанному выше.

Задание для ЭВМ пишется на естественном языке, что весьма важно для неспециалистов по программированию.

Предложенный в данной работе подход позволяет создать для некоторого класса лингвистических задач способ общения с ЭВМ, который иногда называют «программирование без программирования».

Список литературы. 1. *Глушков В. М., Вельбицкий И. В.* Технология программирования и проблемы ее автоматизации. — Управляющие системы и машины, 1976, № 6. 2. ДЗ RTK ОС ЕС ЭВМ. Руководство программиста, отчет ИК АН УССР. Киев, 1977. 3. *Косарев Ю. Г., Хабаров В. В.* Применение *R*-метаязыка для кодирования, контроля и коррекции текстовой информации. — В кн.: Теория и практика системного программирования. Киев, 1976, 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. 5. *Хабаров В. В., Косарев Ю. Г.* Об эффективности автоматического кодирования и исправления ошибок при подготовке данных. — Вычислительные системы. Новосибирск, 1975, вып. 62, с. 106—118.