дополнительные возбуждения нервной системы на периферии. Непрерывные возбуждения концов анализаторов сенсорных систем обусловливаются аномалией в обмене веществ — дополнительным производством кофеина у подагриков.

Таким образом, мозг человека без дополнительной стимуляции, т. е. в норме, реализует лишь ничтожную долю своих потенциальных возможностей. Естественные стимуляторы: дополнительное производство кофеина, возбуждающие периферию ЦНС и психотизмы генераторы спонтанного самовозбуждения — приводят в действие дремлющие при обычных условиях механизмы мозга, заставляя их работать на полную мощность, реализуя при этом полностью наследственную одаренность [3].

СПИСОК ЛИТЕРАТУРЫ

1. Чудаков В. Н. Классификация нормальных и аномальных форм интеллектуальной деятельности на основании квантово-волновой теории когерентной модели мозга. *Сообщение 1.*— В кн.: Проблемы бионики. Вып. 18. Харьков, 1976.
2. Чудаков В. Н. Классификация нормальных и аномальных форм интеллектуальной деятельности на основании квантово-волновой теории когерентной модели мозга. *Сообщение 2.*— В кн.: Проблемы бионики. Вып. 19, Харьков, 1977.
3. Чудаков В. Н. Классификация нормальных и аномальных форм интеллектуальной деятельности на основании квантово-волновой теории когерентной модели мозга. *Сообщение 3.*— См. статью в настоящем сборнике.
4. Клинический архив гениальности и одаренности.— «Практическая медицина», Л., 1925, т. I, вып. 4. 167 с.
5. Клинический архив гениальности и одаренности.— «Практическая медицина», Л., 1925, т. I, вып. 1. 241 с.
6. Ломброзо Ц. Гениальность и помешательство. Одесса, Спб, 1865. 340 с.
7. Труды IV МОКИИ. Естественный и искусственный интеллект (концептуальный подход). Тбилиси, 1976. 250 с.
8. Проблемы управления интеллектуальной деятельностью. Под ред. Чавчанидзе В. В. Тбилиси, «Мецниереба», 1974. 360 с.

Поступила 23 июня 1976 г.

УДК 62.506.2

Э. М. БУЗНИЦКАЯ

ОБ ОДНОМ ЯЗЫКЕ ФОРМАЛЬНОГО ОПИСАНИЯ АЛГОРИТМОВ МОРФОЛОГИЧЕСКОЙ ОБРАБОТКИ ЕДИНИЦ ТЕКСТА

Среди алгоритмических языков, к сожалению, нет такого, который способен удовлетворительно служить целям инженерной лингвистики, в силу чего методы программирования лингвистических задач носят, как правило, случайный характер. В связи с этим возникает проблема создания методов программирования, рассчитанных на более, чем один исходный язык для записи алгоритмов, например, для ряда (или даже для всех) языков, употребляемых в данной области.

Средством, позволяющим справиться с многообразием алгоритмических языков, может оказаться некоторый метаязык; специально созданная для этой цели система записи, при которой система программирования в целом становится двухступенчатой. Обращение к двухступенчатой системе и, следовательно, к некоторому промежуточному языку характерно для работ О. С. Кулагиной [1], Г. С. Цейтина [2], А. В. Кузнецова [3].

Главная идея, положенная в основу построения предлагаемого в настоящей статье метаязыка записи алгоритмов морфологической обработки единиц текста естественного языка, состоит в следующем. На первом этапе каждый алгоритм оформляется с помощью небольшого запаса простых правил. Каждое из них пишется словами и имеет фиксированную формулировку, в которую нужно поставить конкретные значения определенных параметров. Изменяя их, можно приспособить правило к работе с определенными признаками определенного слова или с определенным правилом. Запись таких правил не требует никаких знаний о работе машины и не зависит от типа выбранной ЭВМ и последующей реализации алгоритма.

Словарь функций

№ п/п	Вид функции	Интерпретация
1	$P_4(x, i) = P_1(x, P_2(x) - i + 1, 1)$	Выделение любой буквы, стоящей на i-м месте от конца в слове x
2	$P_5(x, y) = P(x, P_1(P_2(x) - P_2(y) + 1, P_2(y)) = y)$	Сравнение цепочки y с концом слова x
3	$P_6(x, \{x_1, x_2, \ldots, x_N\}) = P(x = x_1 \vee \ldots \vee x = x_N)$ N — количество слов в словаре	Сравнение слова x со словарем
4	$P_7(x, i, \alpha) = P(P_1(x, P_2(x) - i + 1, 1) = \alpha)$	Сравнение буквы, стоящей на i-м месте от конца в слове x с буквой α
5	$P_8(x, i, \alpha_{гл}) = P_7(x, i, а) \vee P_7(x, i, е) \vee$ $\vee P_7(x, i, и) \vee P_7(x, i, о) \vee P_7(x, i, у) \vee$ $\vee P_7(x, i, э) \vee P_7(x, i, ю) \vee P_7(x, i, ы) \vee$ $\vee P_7(x, i, я)$	Сравнение буквы, стоящей на i-м месте от конца в слове x с гласной буквой

Продолжение

№ п/п	Вид функции	Интерпретация
6	$P_9(x, i, \alpha_{\text{согл}}) = P_7(x, i, \text{б}) \vee P_7(x, i, \text{в}) \vee \vee P_7(x, i, \text{г}) \vee P_7(x, i, \text{д}) \vee P_7(x, i, \text{ж}) \vee \vee P_7(x, i, \text{з}) \vee P_7(x, i, \text{к}) \vee P_7(x, i, \text{л}) \vee \vee P_7(x, i, \text{м}) \vee P_7(x, i, \text{н}) \vee P_7(x, i, \text{п}) \vee \vee P_7(x, i, \text{р}) \vee P_7(x, i, \text{с}) \vee P_7(x, i, m) \vee \vee P_7(x, i, \text{ф}) \vee P_7(x, i, \text{х}) \vee P_7(x, i, \text{ц}) \vee \vee P_7(x, i, \text{ч}) \vee P_7(x, i, \text{ш}) \vee P_7(x, i, \text{щ})$	Сравнение буквы, стоящей на i-м месте от конца в слове x с согласной
7	$P_{10}(x, i, \alpha_{\text{шип}}) = P_7(x, i, \text{ж}) \vee P_7(x, i, \text{ч}) \vee \vee P_7(x, i, \text{ш}) \vee P_7(x, i, \text{щ})$	Сравнение буквы, стоящей на i-м месте от конца в слове x с шипящими согласными
8	$P_{11}(x, i, \alpha_{\text{г, к, х}}) = P_7(x, i, \text{г}) \vee P_7(x, i, \text{к}) \vee \vee P_7(x, i, \text{х})$	Сравнение буквы, стоящей на i-м месте от конца в слове x с «г», «к», «х»
9	$P_{12}(x, l, q, y) = P(P_1(x, l, q) = y)$	Сравнение цепочки длины q, начинающейся с символа с номером l в слове x, с заданной цепочкой y
10	$P_{13}(x, \sigma) = P(\sigma = 1)$	Проверка ударности окончания
11	$P_{14}(x, q) = P_1(x, 1, P_2(x) - q)$	«Стирание» последних q букв в слове x
12	$P_{15}(x, \alpha) = P_3(P_{14}(x, 2), \alpha, P_1(x, P_2(x), 1)$	Замена предпоследней буквы в слове x на α
13	$P_{16}(x, \alpha) = P_3(P_{14}(x, 2), P_1(x, P_2(x), 1)$ $(\alpha = o, e)$	«Стирание» в слове x буквы α, стоящей на предпоследнем месте
14	$P_{17}(x, \alpha) = P_3(P_1(x, 1, P_2(x) - 1), \alpha, P_1(x, P_2(x), 1)$	Введение буквы α перед последней буквой
15	$P_{18}(x, i, q, y) = P(P_1(x, P_2, (x) - i + 1, q) = y)$	Сравнение цепочки длины q, начинающейся с символа с номером i от конца слова x с заданной цепочкой y

На втором этапе происходит переход от исходной записи к метаязыку. В качестве метаязыка разработан один из вариантов прикладного исчисления предикатов, язык, который позволяет описывать основные лингвистические правила и, в принципе, осуществлять логическую переработку записанной информации. Синтаксис разработанного языка укладывается в узкое исчисление предикатов, что делает предлагаемый язык подобным любому другому языку, построенному на базе исчисления

предикатов 4, 5. Следовательно, все наблюдения о соотношении этого языка с естественным имеют общий характер. В то же время определенность словаря (словарь предлагаемого языка составлен на материале морфологических описаний системы прилагательных русского языка из 6, т. е. наличие точно заданного конечного перечня постоянных предикатов и переменных областей позволяет придать рассматриваемым синтаксическим проблемам конкретную и до конца точную формулировку.

Алфавит ЯМО

1. Предметные области переменных:
 1) два булевских числа false и true;
 2) 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 — цифры;
 3) а, б, . . ., я, —, ␣, (,) — морфологический алфавит;
 4) <система формул>.
2. Логические связки: und, $\vee$, $\rightarrow$, $\sim$;
 und — . . .; и . . .; $\vee$ . . .; или . . .; $\rightarrow$ . . .; только если . . .; $\sim$ — . . .; если и только если . . .; $|$ — неверно, что
3. Операции отношения: $=$, $>$, $<$;

Правила образования формул ЯМО

1. U и B — формулы, ∇ — связка und, $\vee$, $\rightarrow$ или $\sim$, $U \nabla B$ — формула.
2. U — формула, $|U$ — формула.
3. $P_i(—, \ldots, —)$ — n-местный предикат, $x_1, x_2, \ldots, x_n$ — переменные, $P_i(x_1, x_2, \ldots, x_n)$ — атомарная формула, i — некоторое натуральное число, соответствующее лексическому номеру предиката.
4. $f_i(—, \ldots, —)$ — n-местная функция, $x_1, x_2, \ldots, x_n$ — переменные, $f_i(x_1, x_2, \ldots, x_n)$ — элементарное функциональное выражение с приданными переменными.

Уточним некоторые понятия. В первую очередь нас интересует, при каких условиях можно сказать, что нам дано эффективное определение функции через некоторые другие. Исходим из следующего. Задание (определение) функции есть лингвистический объект — правильная фаза некоторого языка. Можно считать эти выражения именами функций или формами, определяющими функцию. Форма F, определяющая функцию φ, должна выражать некоторое условие, наложенное на переменный объект f и выполняющееся, если значением f является φ такое, что φ есть решение формы F относительно переменной f. Это решение должно быть единственным для того, чтобы форма F действительно определяла функцию φ (в этом случае переменную f можно считать именем функции φ «внутри» формы F). Форма F, определяющая функцию, должна иметь вид равенства. Воспользуемся соглашением, что место расположения указывает, значением какого аргумента является заданное значение. Заметим также, что мы имеем

единый алфавит переменных и единую область их изменения (алфавит ЯМО). Таким образом, проведем различие между постановкой задачи об определении некоторой функции и решением такой задачи. Задача об определении функции поставлена, если написана произвольная форма, ее определяющая; задача решена, если найдена другая форма, определяющая ту же функцию и принадлежащая классу форм, в котором имеется единый (для всех форм этого класса) алгоритм вычисления значений функции φ для каждого элемента x_i, принадлежащего исследуемому множеству.

Указанные рассуждения используют при описании строения *каждого алгоритма обработки имен прилагательных* следующим образом. Выбирают ряд функций и предикатов, образующих множество базисных функций для указанного круга задач. Любая небазисная функция определяется как композиция функций, причем начало композиции образуется с помощью нескольких или всех базисных функций. Тогда логическая схема любого аргумента может быть составлена из набора операторов, где каждый является процедурой-функцией, способ задания которых описан выше.

Специфика форм алгоритмов во многом связана с операциями над цепочками. Этой специфике непосредственно отвечают операции, относящиеся к «преобразованиям» и сравнениям, поэтому именно они в большой степени определяют вид системы базисных функций. Многообразие возможностей, связанных с операциями этих типов, вынуждает отбирать лишь те, которые обеспечат системе максимальную универсальность.

При описании используемого языка введены следующие обозначения для характеристики слова:

$x_i \in S$ — элемент входного множества слов, $\alpha \in A$ — буква алфавита A, m_i — количество букв в слове, α_j — j-я буква в слове $x\,(j = 1, 2, \ldots, m)$, нумерация идет от начала слова, $\overline{\alpha}_i$ — j-я — буква от конца в слове x, $y\,(x)$ — цепочка букв в слове x, l — номер буквы, с которой начинается цепочка $y\,(x)$, q — количество букв в цепочке $y\,(x)$, $y1\,(x)$ — часть слова x слева от цепочки $y\,(x)$, $y2\,(x)$ — часть слова x справа от цепочки $y\,(x)$, q_1 — количество букв в цепочке $y1$, q_2 — количество букв в цепочке $y2$, δ — признак ударности/безударности флексии, булева переменная, принимающая значение 1 при ударной флексии и 0 — в противном случае; в случае нулевой флексии δ — номер ударной гласной от конца слова, y — любая цепочка как конечная упорядоченная последовательность элементов множества S.

При этом справедливы следующие соотношения: $q_1 + q_2 + q = m$; $l = q_1 + 1$; $\alpha_j = y$ при $j = l$, $q = 1$; $j = m + 1 - i$. Тогда система базисных функций в предлагаемом языке примет такой вид:

1) $P_1\,(x, l, q)$ — выделение цепочки y длины q, начинающейся с символа с номером l в слове x. Введенная функция позволяет определить цепочку, а также обращаться к частям слова $y1$ и $y2$;

2) $P_2(x)$ — подсчет количества букв в слове x;

3) $P_3(y_1, \ldots, y_n)$ — операция конкатенации.

Когда появляется необходимость воспользоваться некоторой небазисной функцией, ей должно быть дано определение и названы аргументы этой функции.

Используем для построения небазисной функции φ операцию суперпозиции функций, которая заключается в подстановке одних функций вместо аргументов других. При этом надо помнить, что функция φ эффективно вычислима над базисом P_1, P_2, P_3, если операция суперпозиции применяется только к базисным функциям.

Исследование группы задач, связанных с созданием действующих моделей обработки прилагательных, и проводимых в ходе разработки алгоритмов указывает на целесообразность выделения системы небазисных функций, используемых в различных задачах и составляющих словарь, приведенный в таблице. Кроме словаря, каждый алгоритм может быть снабжен также собственным списком небазисных функций.

Любую функцию можно оформить в виде некоторой процедуры на одном или нескольких интересующих нас алгоритмических языках. Эта функция образует набор необходимых операторов для решения любой из указанного класса задачи. Логическая схема алгоритма определяет порядок работы операторов в зависимости от значения входящих в них логических условий. Последовательное выполнение нескольких операторов обозначается как их произведение, причем сомножитель, стоящий справа, действует после сомножителя, стоящего слева. От каждого оператора начинается определенная стрелка, которая оканчивается у какого-либо из членов логической схемы. Знак $\uparrow$ означает начало стрелки, знак $\downarrow$ — ее конец. Одинаковыми номерами помечают начало и конец одной и той же стрелки. Работа алгоритма начинается с того, что срабатывает самый левый член схемы. После этого определяется, какой член схемы должен работать следом за ним. Здесь возможны два случая. Если проверяемое условие выполнено, то должен работать оператор, соседний справа. Если оно нарушено, должен работать тот оператор, к которому ведет стрелка, начинающаяся после данного оператора. Работа алгоритма оканчивается тогда, когда последний из работающих операторов получит указание о прекращении работы.

Кроме операторов типа процедура-функция обычным образом вводятся операторы проверки логических условий, которые обозначаются $P(a = b)$, где в скобках записано условие проверки; оператор присвоения $[P := \kappa]$ (приписывает P значение k); оператор пересылки $[x \Rightarrow Q]$ (пересылает x в массив Q) и, наконец, оператор остановки работы алгоритма. Перечисленные типы операторов позволяют синтезировать алгоритмы обработки имен прилагательных русского языка.

СПИСОК ЛИТЕРАТУРЫ

1. Кулагина О. С. Об операторном описании алгоритмов перевода и автоматизации процесса их программирования.— В кн.: Проблемы кибернетики. Вып. 2. М., «Физматгиз», 1959, с. 289—302.
2. Цейтин Г. С. О промежуточном этапе при переводе с естественного языка на язык исчисления предикатов.— В кн.: Тезизы КОИМПАЧТ, М., «Наука», 1961, с. 163—165.
3. Кузнецов А. В. Логические контуры алгоритма перевода со стандартизированного русского языка на информационно-логический.— В кн.: Тезисы КОИМПАЧТ, М., 1961, с. 201—206.
4. Beth E. W. The relationship between formalized languages and natural languages.— In: Form and strategy in science. Dortech, 1964, p. 211—219.
5. Cappell S. A programmed formalized for a fragment of English.— «National Bureau of Standards Report», 1964, № 8171, p. 1358—1391.
6. Виноградов В. В. Русский язык (грамматическое учение о слове). М., «Высшая школа», 1972. 614 с.

Поступила 7 октября 1976 г.

УДК 62.506.2

Н. К. СЕРГЕЕВА

ОЦЕНКА ПОГРЕШНОСТИ ВЫЧИСЛЕНИЯ ПАРАМЕТРОВ НОРМАЛИЗАЦИИ ВРАЩЕНИЙ ПЛОСКИХ ИЗОБРАЖЕНИЙ

Решая некоторые задачи, связанные с обработкой и распознаванием изображений, необходимо нормализовать изображения, подвергнутые преобразованию вращения. Кроме того, очень важно точно вычислить параметры нормализации, которые используются на последующих этапах обработки изображений в качестве существенных признаков.

В работе [2] предложен ряд алгоритмов для отстройки от вращений плоских изображений. В целях исследования возможностей технической реализации предложенных алгоритмов представляет интерес оценка погрешности вычисления параметров нормализации в условиях присутствия возмущающих воздействий.

Данная работа посвящена исследованию указанного вопроса для изображений, заданных совокупностью точечных объектов, когда функционалы, определяющие параметры нормализации вращений, имеют вид

$$\Phi <B> = \sum_{i=1}^{n} \delta_i^m \operatorname{arctg} \frac{Y_i}{X_i}. \qquad (1)$$

Здесь δ_i^m — символ Кронекера, определяемый условием

$$\delta_i^m = \begin{cases} 1, & i = m \\ 0, & i \neq m, \end{cases} \quad \sum_{i=1}^{n} \delta_i^m = 1,$$