УДК 004.318

А.В. Палагин, Н.Г. Петренко

Методологические основы разработки лингвистического процессора для обработки лингвистических корпусов текстов сверхбольших объемов. I

Разработаны методологические основы построения быстродействующих аппаратных лингвистических процессоров с использованием метода многоуровневых проекций разработки структурной организации упомянутых процессоров с учетом онтологического подхода. Предложена функциональная схема процессора в целом и его подсистем. Рассмотрена задача оптимизации процессора.

The methodological bases of high-speed hardware linguistic processors construction, using the method of multi-level projections development of the structural organization of the mention above processors with ontological approach are elaborated. The processor functional diagram as a whole its subsystems is proposed. The task of processor optimization is considered.

Розроблено методологічні основи побудови швидкодіючих апаратних лінгвістичних процесорів з використанням методу багаторівневих проекцій розробки структурної організації згаданих процесів з урахуванням онтологічного підходу. Запропоновано функціональну схему процесора в цілому та його підсистеми. Розглянуто задачу оптимізації процесора.

Введение. Обработка речевой или текстовой информации обеспечивается лингвистическим процессором или на уровне «языкового» сознания человека, или в компьютерной системе (КС). В КС он – основная компонента, реализующая распознавание и понимание входной естественно-языковой информации, извлечение из нее первичных знаний в виде простых онтологических концептов с их последующим формально-логическим представлением. Полученной информационной структуры достаточно для реализации (знание-ориентированных) процедур для решения прикладных задач, принятия решений и др.

Постановка задачи

Одной из важных задач на пути разработки общей теории компьютерной обработки предметных знаний, представленных в естественно-языковой форме, есть построение эффективных лингвистических процессоров. Эта задача особенно актуальна для приложений обработки лингвистических корпусов текстов (ЛКТ) сверхбольших объемов (и в реальном времени). Это связано с тем, что современные персональные компьютеры программным способом выполняют лингвистический анализ одного слова входного текста средней длины примерно за одну миллисекунду (диаграмма зависимости времени обработки входного слова от его длины приведена на рис. 1), а такой анализ занимает значительную часть компьютерного времени в общей лингвистической обработке. При этом время обработки входного текста даже относительно небольшого объема занимает от нескольких до десятков минут. В итоге для приложений, работающих в реальном времени, часть информации будет потеряна.

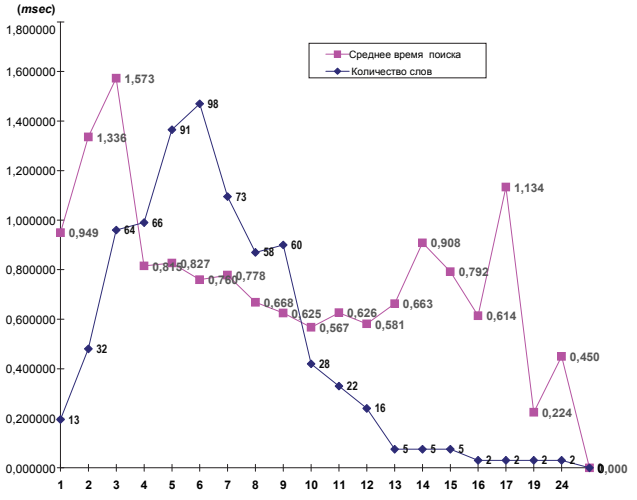


Рис. 1

Поэтому задача существенного (на два порядка и более) повышения быстродействия лингвистического анализа актуальна.

Отметим, что такое повышение быстродействия может быть достигнуто путем дополнительных аппаратурных затрат как стандартной, так и специализированной разработки. Аппаратные средства (АС) *первого* типа – это продукт известных фирм, доступный на рынке, к кото-

рым прилагается система автоматизированного проектирования (САПР). Несомненный лидер таких АС на рынке – платы с установленными на них программируемыми логическими интегральными схемами (ПЛИС) (в которых есть сверхбыстродействующая память) и быстродействующая память большого объема [1]. АС *второго* типа – специализированная разработка, для них необходимо спроектировать архитектурно-структурную организацию процессора, электрическую схему или граф-схемы алгоритмов, специальное программное обеспечение управления ими и драйверы совмещения с операционной системой компьютера. Для реализации лингвистического процессора оба эти варианта АС имеют свои преимущества и недостатки. Для АС первого типа преимуществом служит их доступность на рынке, их вычислительная мощность постоянно повышается разработчиками, к ним уже прилагается программное обеспечение, а проект аппаратного лингвистического процессора (АЛП) может быть разработан за время от двух месяцев. Недостаток этих АС – низкий процент использования установленного на плате оборудования.

К преимуществам АС второго типа следует отнести повышение быстродействия на один–два порядка в сравнении с АС первого типа, что есть главным критерием при разработке АЛП. А к недостаткам – необходимость коллектива разработчиков (системотехников и программистов), и время разработки проекта оценивается от одного года.

Повышение быстродействия реализации алгоритма лингвистического анализа для обоих типов АС достигается путем перевода операторов алгоритмического и программного уровней (реализация лингвистического анализа программным способом) на нижние уровни интерпретации [2]: для АС первого типа – на микропрограммный уровень, для АС второго типа – на микропрограммный и частично на физический уровни.

В [2] приведены дополнительные доводы целесообразности реализации лингвистического процессора (ЛП) в целом и морфологического – в частности, аппаратными средствами. Например, аппаратная реализация дает возможность параллельной обработки всех слов одного предложения одновременно. При этом упрощаются алгоритмы синтаксического и семантического анализа.

В статье рассмотрены следующие компоненты методологических основ разработки АЛП[1]:

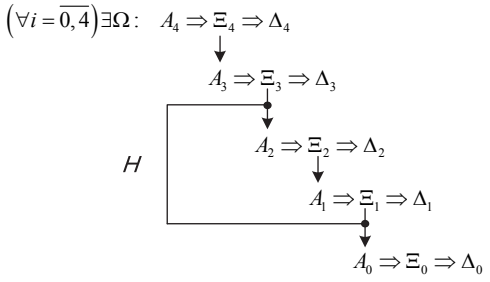
- онтологический подход к построению аппаратных средств лингвистического анализа естественно-языковых объектов (ЕЯО);
- разработка функциональной схемы АЛП;
- разработка подсистем АЛП;
- задача оптимального синтеза АЛП;
- структурная организация и проектирование аппаратного морфологического процессора (АМП);
- структурная организация АМП для обработки ЛКТ разного объема;
- оценки сложности структурной реализации АМП.

Онтологический подход к построению аппаратных средств лингвистического анализа естественно-языковых объектов

Метод многоуровневых проекций проектирования АЛП. Метод многоуровневых проекций (ММП) построения аппаратных лингвистических процессоров основан на применении онтологического подхода к проектированию архитектурной и информационной компонент онтологоуправляемой информационной системы (ОУИС). При этом схема логико-информационной модели [2] преобразуется в схему онтологической модели АЛП (рис. 2), где τ_0 – физический уровень; τ_1 – микропрограммный уровень; τ_2 – программный уровень; τ_3 – алгоритмический уровень; τ_4 – предметный уровень.

В модели (1) операторы алгоритмического уровня посредством отображения H переводятся на *нулевой*, или физический уровень интерпретации, чем реализуются свойства онтологического управления и реконфигурируемости.

[1] Из перечисленных семи компонент методологических основ в данной статье будут рассмотрены первые четыре компоненты, а остальные – в следующей публикации.



(1)

Рис. 2

Модель (1) можно описать сетью конечных автоматов:

$$M=\left\{X,\left\{M_n\right\},S,Y,\left\{\Psi_i\right\},\varphi,\mu\right\},$$

где X – входной алфавит сети автоматов; $\{M_n\}$ – множество структурных автоматов; S – множество связей между автоматами M_n; Y – выходной алфавит сети автоматов; $\{\Psi_i\}$ – множество функций переходов в $\{M_n\}$; φ – начальное состояние сети; μ – конечное состояние сети.

Учитывая современные технические характеристики микроэлектронной базы, можно реализовать все четыре этапа лингвистического анализа на аппаратно-микропрограммном уровне в виде сети комбинационных автоматов с памятью. При этом выполняется анализ входной естественно-языковой (ЕЯ) информации не пословно, а по предложениям. Предложение – минимальная предикативная синтаксическая единица, для которой может быть выполнен полный синтаксический и поверхностно-семантический анализ.

Суть МПП состоит в следующем.

На рис. 3 представлена структурная модель АЛП.

Входные данные модели – это множество текстов $\{T_k\}$, выходные в общем случае – это смысл обработанного текста или, по крайней мере, его семантическая структура. Внутри модели реализуется отображение $\widetilde{\Psi}: X\to Y$, которое поэтапно переводит текстовое представление входной информации (текст $\to$ графематическая структура текста $\to$ морфологическая $\to$ синтаксическая $\to$ $\to$ семантическая структура) в соответствующие структуры сети операционных автоматов. При этом микропрограммные автоматы естественным образом настраиваются на управление функционированием соответствующих операционных автоматов.

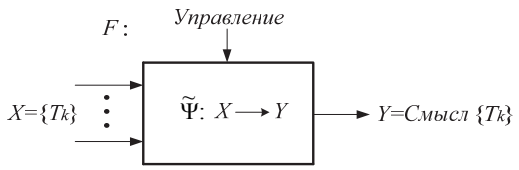


Рис. 3

Следует отметить существенную особенность реализации АЛП с применением ПЛИС-технологии, для которой существует возможность реконфигурации структуры АЛП на физическом уровне как инструменте настройки (перестройки) на обработку ЛКТ заданной предметной области (ПдО) или решение задачи предметно-проблемной ориентации аппаратных средств. При этом на логических схемах ПЛИС построены так называемые адаптивные логические сети (АЛС) с памятью [1].

Модель аппаратных средств реализации АЛП. *Адаптивная логическая сеть* – это дискретный преобразователь кодов типа асинхронного комбинационного автомата, заданный направленным графом, вершины которого – логические функции, а ребра – связи между ними типа *выход–вход*. АЛС предназначена для решения многопараметрических задач, исходная информация для которых представляется полями данных (алфавитными символами ЕЯ), сгруппированными по функциональным признакам. В основу организации АЛС положены требования динамической реконфигурируемости, иерархичности, многоуровневости и параллельной обработки данных. Адаптация структуры АЛС на класс задач с учетом производительности, регулярности структуры, организации синхронных и асинхронных процессов обработки информации основывается на многоуровневом представлении внешних и внутренних процессов. Относительно топологии системы АЛС представляют собой матрицу универсальных логических элементов (ЛЭ), которые группируются в функциональные узлы и блоки (ФБ), местоположение которых закреплено, при этом изменение их функционирования происходит в зависимости от класса задач и от их назначения. В институте кибернетики НАН Украины накоплен опыт разработки АЛС на основе ПЛИС-технологий [1].

Далее рассмотрим синтез дискретного преобразователя без памяти, представляющего многоуровневую комбинационную схему в элементном базисе ПЛИС – АЛС [1].

Пусть задано полное множество n-мерных двоичных векторов $G = \{g\}$ ($\mathrm{Card}\{G\} = 2^n$). Задано множество n-мерных двоичных векторов $D \subset G$, которое есть обучающей выборкой для алгоритма классификации. Для произвольного входного множества n-мерных двоичных векторов $X = \{x\}$, ($X \subset G$) необходимо реализовать следующую математическую модель:

$$Y = \begin{cases} 1, \forall x \in D \\ 0, \text{ в противном случае.} \end{cases}$$

Данная математическая модель может быть описана булевой сетью в виде треугольной матрицы. Структурно булева сеть представляется многоуровневой комбинационной схемой, а вершинам сети соответствуют ЛЭ.

Под уровнем понимаем линейку ЛЭ, каждый из которых настраивается на реализацию произвольной булевой функции и реализует отображение h-размерных двоичных векторов в $(h-1)$-размерные двоичные векторы. В пределах одного уровня ($i = \mathrm{const}$) тип логической функции задается для любого i-го ЛЭ независимо. Таким образом, j-й уровень треугольной матрицы (ТМ) представляется комбинационной схемой, имеющей l-разрядный вход и $(l-1)$-разрядный выход $\left(l = \overline{2,n}\right)$. Трапециедальная матрица (ТМ) реализует отображение $\Im$: $X \Rightarrow Y$, которое предполагает корректное отображение множества F. В общем случае задача синтеза структуры ТМ сводится к определению типов логических функций f_{ij} для всех элементов сети. Для определения множества логических функций $F = \{f_{ij}\}$ используем полиномы для описания булевой сети [1].

При кодировании значений булевой функции и ее аргументов перейдем к другому кодированию, используя значения единица и минус единица. Таким образом, множество переменных $X = \{x_1, x_2, \ldots, x_n\}$ для булевой функции f от n переменных будет представлено множеством $E = \left\{e_1, e_2, ..., e_n\right\}$, где $e_i = (-1)^{x_i}$, а множество значений $Y = \left\{y_1, y_2, ..., y_{2^n-1}\right\}$, где $y_j = \{0,1\}$ множеством $V = \left\{v_0, v_1, ..., v_{2^n-1}\right\}$, где $v_j = (-1)^{y_j}$.

Для любой булевой функции f от n переменных, принимающих значения из множества $\{1,-1\}$, существует эквивалентный полином $P_{f(n)}$ с коэффициентами из множества действительных чисел $f(X) = P_{f(n)}(X)$.

Коэффициенты полинома для функции f можно записать посредством матрицы Адамара:

$$A = \frac{1}{2^n} H_n V_n,$$

где $A = \left\{a_0, a_1, ..., a_{2^n-1}\right\}$ – множество коэффициентов полинома; H_n – матрица Адамара размерности 2^n; V_n – множество значений булевой функции.

Матрица Адамара n-го порядка H_n представляет собой квадратную матрицу размерности n, содержащую два типа элементов $\{1,-1\}$. Матрицу Адамара можно построить для любого значения n:

$$H_1 = \|1\|; \quad H_2 = \begin{Vmatrix} +1+1 \\ +1-1 \end{Vmatrix} \quad H_n = \begin{Vmatrix} +H_{n-1} + H_{n-1} \\ +H_{n-1} - H_{n-1} \end{Vmatrix}.$$

Полином от одной переменной представляется выражением $P_{f(1)} = a_0 + a_1 e_1$.

Соответственно, полином от n переменных: $P_{f(n)} = a_{f(\mathrm{n}\text{-}1)} + a_n e_n P_{f(n\text{-}1)}$.

Функциональные ограничения на алгоритмы адаптации. Класс устройств с реконфигурируемой структурой (УРС) предполагает возможность настройки структуры на реализацию заданной функции. УРС, представленный многоуровневой комбинационной логической схемой и реализующий полный набор булевых функций F_w от n переменных $\left(\mathrm{Card}\left\{F_W\right\} = 2^{2^n}\right)$, называется универсальным. Если же УРС реализует только подмножество F_u булевых функций ($F_u \subset F_w$), то говорят о многофункциональном устройстве, так как появляется подмножество F_r логических функций ($F_r = F_w \setminus F_u$), которое не может быть реализовано данной структурой. Мощность подмножества F_u зависит от структурной организации УРС. Подобная проблема «*XOR*»

существует для однослойных персептронов в виде подмножества линейно-разделимых функций.

Рассмотрим подход к определению функциональных ограничений, т.е. возможности настройки УРС на реализацию подмножества логических функций F_u. В качестве УРС рассматривается реконфигурируемая структура типа *треугольная матрица* с сотовой структурой связи, содержащая множество $L = \{L_{ij}\}$ универсальных логических элементов, каждый из которых реализует логическую функцию

$$Y_{ij} = f_{ij}\left(Y_{i,(j-1)}, Y_{(i+1),(j-1)}\right),$$

где f_{ij} – тип булевой функции ($f_{ij} \in F_w$); $j = 1 + (n-1)$ – номер уровня ТМ; $i = 1 + (n-j)$ – порядковый номер ЛЭ на j-м уровне ТМ.

Функциональные ограничения определяют возможность настройки ТМ на реализацию подмножества логических функций F_u.

Для определения функциональных ограничений воспользуемся полиномиальным представлением булевых функций:

$$P_{f(n)} = P_{f(n-1)} + a_n x_n P_{f(n-1)},$$

где $P_{f(n)}$, $P_{f(n\text{-}1)}$ – полиномы, описывающие булеву функцию от n и $(n-1)$ переменных соответственно; $x_i = (-1)^{e_i}$, где e_i – значение i-й булевой переменной; a_i – коэффициент полинома.

Исходя из структуры ТМ с сотовой структурой связи, произвольный ЛЭ описывается полиномом:

$$P_j^i = a_{j,0}^i + a_{j,1}^i P_{j-1}^i + a_{j,2}^i P_{j-1}^{i+1} + a_{j,3}^i P_{j-1}^i P_{j-1}^{i+1}, \quad (2)$$

где $P_{j-1}^i P_{j-1}^{i+1}$ – полиномы, представляющие собой переменные, поступающие на входы ЛЭ; $a_{j,0}^i, a_{j,1}^i, a_{j,2}^i, a_{j,3}^i$ – коэффициенты полинома P_j^i.

В обобщенном виде система уравнений для произвольного значения n определяется следующими принципами.

Произвольный входной вектор x в общем виде может быть представлен как $x = 2d + \gamma$, где $d = 1 + s$ – индекс номера группы ($s = 2$ – количество групп, на которые разбивается полное множество значений аргументов j-го уровня); $d = \mathrm{Int}\left[\dfrac{x}{2^j}\right]$; $\gamma = 0 + (s-1)$ – индекс элемента в группе, $\gamma = s_{\mathrm{mod}(2^j)}$.

Любой ЛЭ описывается полиномом $P_j^i = f\left[P_{j-1}^i, P_{j-1}^{i+1}\right]$, а с учетом указанных параметров

$$P_j^i\left(2^j d + \gamma\right) = f\left[P_{j-1}^{2i-1}(\gamma); P_{j-1}^{2i}\left(\mathrm{Int}\left[\left(2^j + d + \gamma\right)/2\right]\right)\right].$$

Выражение (2) примет вид

$$P_j^i(2d+\gamma) = a_{j,0}^i + a_{j,1}^i P_{j-1}^{2i-1}(\gamma) + a_{j,2}^i P_{j-1}^{2i}\left(\mathrm{Int}\left[\left(2^j + d + \gamma\right)/2\right]\right) + a_{j,3}^i P_{j-1}^{2i-1}(\gamma) P_{j-1}^{2i}\left(\mathrm{Int}\left[\left(2^j + d + \gamma\right)/2\right]\right).$$

Тогда

$$\forall\gamma = 2p\left(p \in N, p = 0 + \left[\left(2^j - 2\right)/2\right]\gamma = 0 + \left[2^j - 2\right], N \text{ – множество натуральных чисел}\right),$$

получим общие соотношения:

$$\frac{P_j^i(\gamma) - P_j^i(\gamma+1)}{P_j^i\left(2^j+\gamma\right) - P_j^i\left(2^j+\gamma+1\right)} = \frac{a_{j,1}^1 + a_{j,3}^1 P_{j-1}^{i+1}[\gamma/2]}{a_{j,1}^1 + a_{j,3}^1 P_{j-1}^{i+1}\left[\left(2^j+\gamma\right)/2\right]};$$

$$\frac{P_j^i(\gamma) - P_j^i\left(2^j+\gamma\right)}{P_j^i(\gamma+1) - P_j^i\left(2^j+\gamma+1\right)} = \frac{a_{j,2}^i + a_{j,3}^i P_{j-1}^{i+1}[\gamma]}{a_{j,2}^i + a_{j,3}^i P_{j-1}^{i+1}(\gamma+1)}.$$

Поскольку полиномы представляют булевы функции, принимающие значения –1 и +1, то эти функции могут быть равны или противоположны по знаку.

Полученные функциональные выражения позволяют без предварительного определения типов логических функций для всех составляющих структуру элементов определить возможность настройки структуры ТМ с сотовой структурой связи на реализацию заданной системы булевых функций.

Оценка сложности реализации алгоритма лингвистического анализа. Далее предложен формализованный подход к оценке сложности иерархической системы обработки информации, основанный на понятиях абсолютной и относительной оценок сложности операторов, составляющих языки различных уровней иерархии.

Определение 1. Словом оператора назовем конечную строку букв из заданного алфавита, отмечающего данный оператор при представлении его в памяти компьютерной системы.

Определение 2. Длина слова – число букв в нем.

Определение 3. Длину описания алгоритма определим как сумму длин слов операторов языка, в котором представлен алгоритм.

Если множеству операторов языка $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$ поставить в соответствие множество их информационно-кодовых представлений $U = \{u_1, u_2, \ldots, u_q\}$, где $u_i \in U\left(i = \overline{1,q}\right)$ – кодовое слово, отмечающее оператор $\lambda_i \in \Lambda(i = \overline{1,q})$, то сложность алгоритма A в соответствии с определением 3 равна

$$Q_A = \sum_{i=1}^{q} r_i \left|u_i\right|, \qquad (3)$$

где $\left|u_i\right|$ – разрядность кодового слова u_i, r_i – число операторов i-го типа в алгоритме A, $\sum_{i=1}^{q} r_i = N_A$ – длина алгоритма, т.е. число шагов, на каждом из которых реализуется один из операторов $\lambda_i \in \Lambda(i = \overline{1,q})$.

Если аналогично представлять операторы каждого уровня иерархической системы, то выражение (3) примет вид

$$Q_A = \sum_{j=1}^{h} Q_j = \sum_{j=0}^{h} \sum_{i=1}^{q_i} r_{ij} \left|u_{ij}\right|, \qquad (4)$$

где $Q_j (j = \overline{0,h})$ – сложность алгоритма, представленного на j-м уровне; $j = 0$ соответствует уровню микроопераций; $\underset{j=\overline{1,h}}{\forall} \sum_{i=1}^{q_i} r_{ij} = N_j$ – длина алгоритма j-го уровня.

Такой подход всегда подразумевает оценку *относительной* сложности алгоритмов и операторов. Действительно, один и тот же объект (процесс, функция) может быть определен алгоритмически в терминах операторов любой сложности. При этом очевидно, чем выше сложность операторов языка, тем меньше длина описания данного алгоритма в терминах языка этого уровня. Отсюда

Утверждение 1. С увеличением сложности операторов языка сложность алгоритма, представленного в этом языке, уменьшается.

Определение 4. Относительной сложностью алгоритма $\left(\tilde{Q}_A\right)$ будет отношение сложности алгоритма (Q_A) к сложности операторов (Q_Λ) языка, в котором он представлен:

$$\tilde{Q}_A = \frac{Q_A}{Q_\Lambda}. \qquad (5)$$

В терминах теории информации выражение (5) можно интерпретировать так: количество информации, необходимое для задания описания какого-либо процесса, обратно пропорционально мощности его алфавита.

Воспользовавшись выражением (5), для иерархической системы можно записать следующее отношение, определяющее относительную сложность алгоритма, представленного в операторах l-го языкового уровня:

$$\tilde{Q}_A(l) = \frac{\sum_{i=1}^{q_l} r_{il} \left|u_{il}\right|}{\sum_{j=0}^{l-1} \sum_{i=1}^{q_i} r_{ij} \left|u_{ij}\right|}, \quad l = \overline{1,h}. \qquad (6)$$

Знаменатель в выражении (6) характеризует сложность реализации операторов l-го уровня и, таким образом, есть ничем иным, как мерой сложности вычислений. Итак, выражение (6) устанавливает взаимосвязь между сложностью алгоритма и сложностью вычислений.

Наряду с понятием относительной сложности можно определить понятие *абсолютной* сложности. Его целесообразно применять в первую очередь для элементарных операторов, которые, в силу их определения, нельзя представить через какие-либо другие операторы. При этом сложность элементарного оператора может быть сколь угодно велика. Примером может служить таблично реализованный элементарный оператор вычисления функций одной и более переменных. Абсолютную сложность элементарного оператора можно оценивать с помощью энтропийной меры, которую можно трактовать как емкость памяти (в битах), необходимую для хранения гипотетической таблицы, с помощью

которой данный элементарный оператор реализуется как оператор отображения: $F : X \to Y$, где X и Y – множества значений аргумента и функции соответственно.

Если под элементарным оператором понимать микрооперацию, то абсолютная сложность микрокоманды будет $Q_l = \sum_{i=1}^{l} Z_i(F)$, где l – мощность микрокоманды; $Z_i(F)$ – абсолютная сложность i-й микрооперации.

В общем случае абсолютная сложность оператора определяется числом операндов, разрядностью операндов и основанием системы счисления. Отметим, что, в соответствии с принятой концепцией, иерархическая система программируемых автоматов позволяет представлять алгоритмы на любом уровне иерархии. В связи с этим для данного класса автоматов понятия сложности алгоритмов и сложности вычислений взаимосвязаны.

Задача оценки сложности алгоритма, таким образом, сводится к задаче оценки сложности языковых операторов и их оптимальной иерархии в системе.

Разработка функциональной схемы аппаратно-лингвистического процессора. Лингвистический процессор (аппаратный или программмный) интерпретирует текстовую информацию (некоторый ЕЯО – документ, статья, монография или ЛКТ) в соответствии с этапами лингвистического анализа: графематического, морфологического, синтаксического и семантического (точнее, поверхностно-семантического). Результат работы АЛП – информационная структура, предназначенная для проведения глубинно-семантического анализа в подсистеме экстралингвистической обработки, задача которого – извлечение и формирование структуры понятий, т.е. автоматическое извлечение из ЕЯО знаний, реализация их прагматической интерпретации в терминах прикладной задачи или соответствующая реакция, присущая человеку.

На рис. 4 представлена функциональная схема такого АЛП, которая включает в себя соответствующие подсистемы реализации этапов лингвистического анализа и языково-онтологическую картину мира (ЯОКМ), использование которой – одно из основных отличий АЛП от классического лингвистического процессора, а ее ЯОКМ-проектирование рассмотрено в [2].

Приняты следующие сокращения:

АСС – общепринятые аббревиатуры, сокращения и специальные символы; БСОС – блок сравнения основы; БСОК – блок сравнения окончания; ЛЕ – лексема; О –объект; Д – действие; ХО – характеристика объекта; ХД – характеристика действия.

Исходная информация для АЛП – это ЛКТ заданной ПдО. Лингвистический корпус текстов включает в себя конечное множество текстов $\{T_k\}, k = \overline{1, K}, K$ – количество текстов в ЛКТ, последовательно поступающее на вход подсистемы графематического анализа.

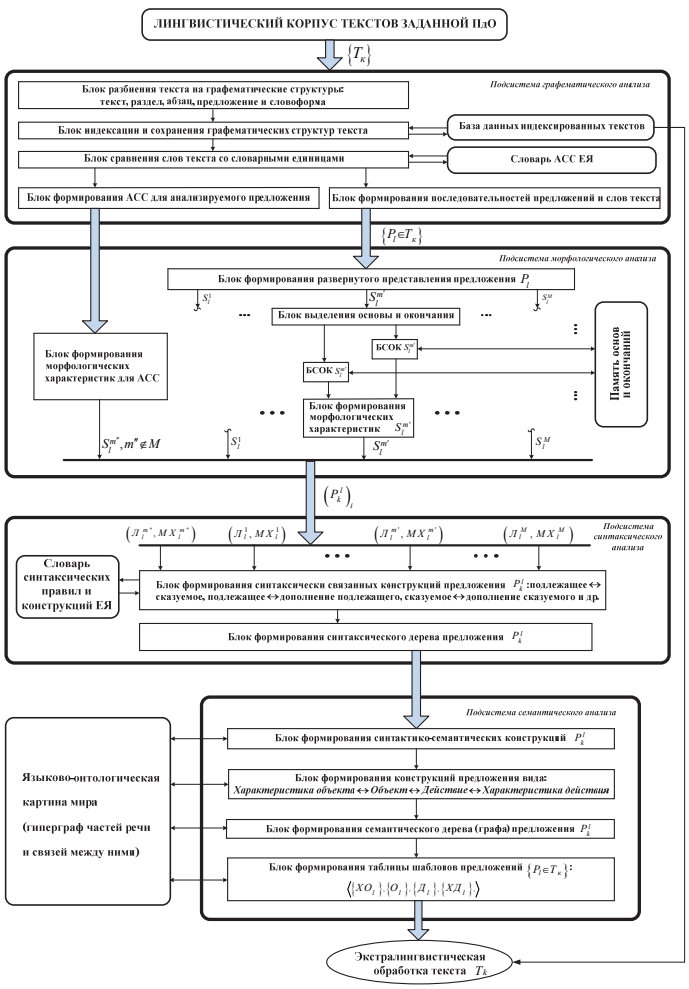


Рис. 4

В процессе выполнения общего алгоритма лингвистического анализа текст $\{T_k\}$ поэтапно преобразуется в графематическую, морфологическую, синтаксическую и семантическую структуры, каждая из которых имеет свою модель представления и свои средства обработки. Рассмотрим процесс лингвистического анализа в подсистемах АЛП.

Разработка подсистем АЛП

Подсистема графематического анализа

На рис. 5 представлена блок-схема графематического анализа (где Ψ – процедура отображения граф-схемы алгоритма реализации соответствующей подсистемы лингвистического анализа в соответствующую сеть комбинационных автоматов с памятью (*TnM*) в терминах САПР ПЛИС, ГфА – графематический анализ, а на рис. 6 показана диаграмма состояний (или граф-схема алгоритма) с описанием исполняемых процедур и анализируемыми условиями:

$p11$ – начало работы АЛП. Запись в буфер-формирователь предложения принятой словоформы;

$p12$ – передача из буфера-формирователя предложения сформированного предложения на вход процедуры $p2$;

$p21$ – индексация словоформ S_l^m и предложения P_k^l в целом;

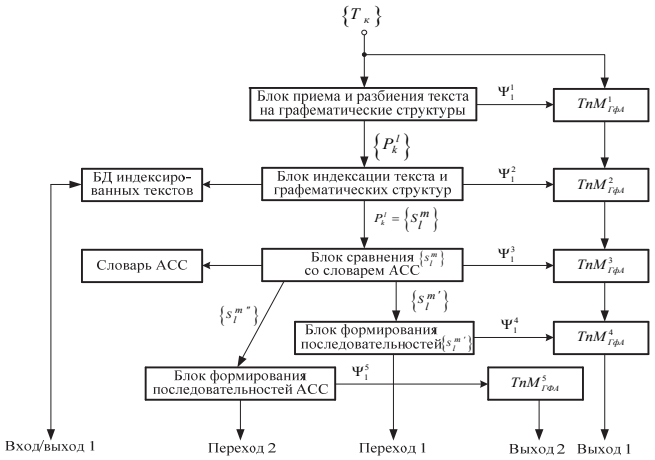


Рис. 5

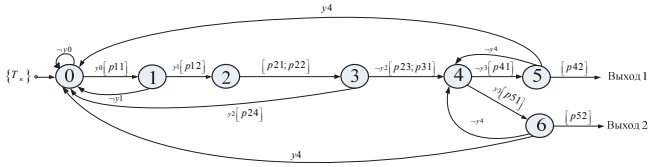


Рис. 6

$p22$ – запись предложения P_k^l в базу данных индексированных текстов;

$p23$ – передача из буфера-формирователя предложения P_k^l на вход процедуры $p31$;

$p24$ – индексация T_k и его запись в базу данных индексированных текстов;

$p31$ – сравнение S_l^m со словарем АСС;

$p41$ – формирование последовательности словоформ $S_l^{m'}$;

$p42$ – передача сформированной последовательности словоформ $S_l^{m'}$ на Выход 1 (рис. 5);

$p51$ – формирование последовательности словоформ $S_l^{m''}$;

$p52$ – передача сформированной последовательности словоформ $S_l^{m''}$ на «Выход 2» (рис. 6).

$y0$ – условие начала приема словоформы;

$y1$ – условие окончания приема словоформы;

$y2$ – условие окончания приема текста T_k;

$y3$ – условие сравнения анализируемой словоформы с АСС;

$y4$ – условие окончания сравнения словоформы S_l^m с АСС.

Поступивший на обработку текст $\{T_k\}$ в блоке разбиения текста на графематические структуры разбивается и структурируется на множество разделов, абзацев, предложений и словоформ. Далее структурные единицы текста индексируются и сохраняются в базе данных индексированных текстов, информация в которой используется в процессе выполнения алгоритма всего лингвистического анализа, а также при экстралингвистической обработке. Затем для каждого предложения текста в блоке сравнения слов текста со словарными единицами выделяются АСС естественного языка. Другими словами, каждое предложение разбивается на две части, причем для первой из них необходимо вычислить морфологические характеристики (и она поступает на обработку в блок формирования последовательностей предложений и слов текста), а для второй они извлекаются из соответствующего словаря (и она поступает в блок формирования АСС). На последнем этапе графематического анализа формируются две параллельные последовательности фрагментов предложений, посту-

пающих в подсистему морфологического анализа (МА), первая из них – в блок формирования морфологических характеристик (БФМХ) для АСС, а вторая – в блок формирования развернутого представления предложения P_k^l. Собственно вторая последовательность есть входной информацией, требующей обработки в подсистеме МА.

Подсистема морфологического анализа

На рис. 7 и 8 представлена блок-схема подсистемы МА – диаграмма состояний с описанием исполняемых процедур и анализируемыми условиями.

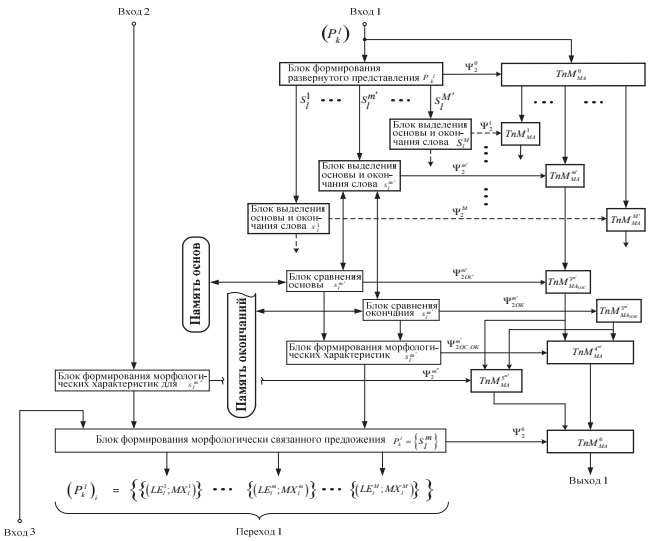


Рис. 7

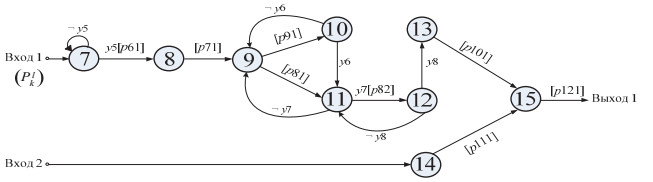


Рис. 8

$p61$ – первая процедура для словоформ P_k^l, не являющихся АСС, формирует развернутое представление предложения P_k^l;

$p71$ – совместно с процедурами $P8$ и $P9$ формирует раздельное представление основы и окончания словоформы $S_l^{m'}$. Выполняет предварительные установки информационных и управляющих регистров;

$p81$ – выполняет сравнение основы словоформы $S_l^{m'}$ с содержимым памяти основ;

$p82$ – формирует список основ–омонимов;

$p91$ – выполняет сравнение окончания словоформы $S_l^{m'}$ с содержимым памяти окончаний;

$p101$ – формирует лексемы и морфологические характеристики $S_l^{m'}$, в том числе и омонимов;

$p111$ – формирует морфологические характеристики $S_l^{m''}$;

$p121$ – формирует морфологически связанные предложения P_k^l, раздельные для всех неоднозначных словоформ;

$y5$ – условие начала приема последовательности словоформ предложения P_k^l;

$y6$ – условие сравнения окончания словоформы $S_l^{m'}$;

$y7$ – условие сравнения основы словоформы $S_l^{m'}$;

$y8$ – условие составления полного списка основ–омонимов.

В аппаратном морфологическом процессоре (АМП) для обработки каждой словоформы $S_l^{m'}$ предложения P_k^l выделен отдельный аппаратный блок (его структура и функционирование рассмотрены далее), в котором из словоформы выделяются основа и окончание, причем принципы такого выделения отличаются от традиционных [3] и описаны в [4]. Совокупность таких блоков, параллельно обрабатывающих все словоформы предложения P_k^l, составляют одну из основных компонент подсистемы морфологического анализа. Максимальное количество блоков определяется на основе статистических характеристик заданного ЛКТ, в частности, параметра максимального количества вхождений словоформ в предложения.

Выделенные основа и окончание словоформы $S_l^{m'}$ поступают соответственно в БСОС и БСОК, в которых по ассоциативному принципу формируется адрес фрагмента ячеек памяти основ и окончаний, в котором хранится морфологическая структура словоформы $S_l^{m'}$. Подробный алгоритм морфологической обработки одной словоформы и его аппаратная реализация описаны далее.

Аналогично формируются морфологические характеристики для всех словоформ $\left\{S_l^{m'}\right\}$ предложения P_k^l. Отметим, что указанные морфологические характеристики учитывают особенности только текстов научно-технического и делового стилей.

В подсистеме МА выполняется параллельный анализ словоформ $\left\{S_l^{m'}\right\}$, принадлежащих предложению $P_k^l \in T_k$, где $m' = \overline{1, M}$, M – количество словоформ в предложении P_k^l, $l = \overline{1, L}, L$ – количество предложений в тексте T_k. На выходе подсистемы в блоках формирования морфологических характеристик формируется выходная информация $\left\{\text{ЛЕ}_l^{m'}, MX_l^{m'}\right\}$, к которой присоединяется морфологическая информация о присутствующих в анализируемом предложении АСС $S_l^{m''}$, $m'' \notin M$. Причем, последнее множество может быть и пустым. На выходе подсистемы МА формируется морфологическая структура вида $\left\{\left(\text{ЛЕ}_l^1, MX_l^1\right), \ldots, \left(\text{ЛЕ}_l^m, MX_l^m\right), \ldots, \left(\text{ЛЕ}_l^M, MX_l^M\right)\right\}$ –входная информация для подсистемы синтаксического анализа.

Подсистема синтаксического анализа. На рис. 9 показанна блок-схема синтаксического анализа (где СнА – синтаксический анализ), а на рис. 10 – диаграмма состояний с описанием исполняемых процедур и анализируемыми условиями.

*p*131 – прием от подсистемы МА последовательности предложений $(P_k^l)_i$, где $i = \overline{1, I}$, I – максимальное количество неоднозначных предложений;

*p*132 – предсинтаксический анализ предложения P_k^l (разрешение грамматической и лексической омонимии, связывание слов в предложении синтаксическими связями);

*p*133 – сегментация предложения на синтаксические группы и подгруппы;

*p*141 – поочередное связывание групп в предложении. При этом выполняется поочередная обработка неоднозначных предложений. По окончании обработки *i*-го предложения на Выход 2 подсистемы выдается сообщение о загрузке (*i* + 1)-го предложения;

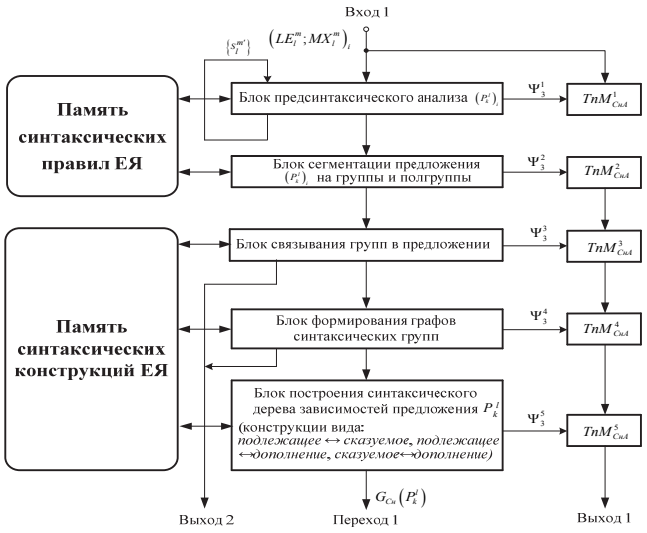


Рис. 9

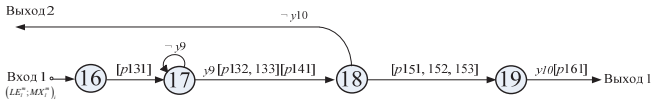


Рис. 10

*p*151 – формирование узлов графов групп и подгрупп;

*p*152 – построение ориентированных графов групп и подгрупп;

*p*153 – преобразование графов в синтаксические деревья зависимостей (конструкции вида: *подлежащее* ↔ *сказуемое*, *подлежащее* ↔ *дополнение*, *сказуемое* ↔ *дополнение* и др.).

*p*161 – построение общего синтаксического дерева предложения P_k^l;

*y*9 – условие отсутствия неоднозначных словоформ в предложении $(P_k^l)_i$;

*y*10 – условие перебора всех неоднозначных предложений.

На вход подсистемы СнА поступает морфологическая структура предложения P_k^l, которая есть линейным объектом, на выходе формируется синтаксическое дерево зависимостей – нелинейный объект, сложность которого существенно выше в сравнении с морфологической структурой.

Под синтаксической структурой предложения P_k^l понимается размеченное дерево зависимо-

стей, такое, что: множество его узлов образуют имена лексем, входящих в P_k^l; каждая дуга обозначена именем соответствующего синтаксического отношения, специфичного для заданного ЕЯ. Например, для русского и украинского языков выделяют более 50 синтаксических отношений. В свою очередь все синтаксические отношения делятся на типы – актантные, атрибутивные, сочинительные и служебные. Их полный список приведен в [3].

Алгоритм синтаксического анализа разбивается на два блока: предсинтаксический анализ и собственно синтаксический анализ.

Основное назначение блока предсинтаксического анализа состоит в разрешении лексической и грамматической омонимии словоформ по линейному контексту. Кроме того, этот блок для некоторых предложений устанавливает конкретные синтаксические связи между теми словами, лексические и синтаксические свойства и относительное линейное расположение которых позволяют однозначно сделать вывод о наличии между ними таких связей.

Исходное предложение разбивается на синтаксические группы, которые, в свою очередь, могут состоять из подгрупп. Выделяют группы подлежащего, сказуемого и др. Далее алгоритм выполняет связывание построенных групп в один граф синтаксических отношений.

Таким образом, в результате применения правил предсинтаксического анализа морфологическая структура предложения преобразуется в некоторый промежуточный объект, в котором, сравнительно с исходной структурой, существенно сокращена лексико-грамматическая омонимия и проведены некоторые синтаксические связи.

В подсистеме СнА одними из основных компонент служат словари синтаксических правил и конструкций ЕЯ. Посредством правил СнА формируется набор гипотез о возможных синтаксических связях между составляющими элементами P_k^l. Основным критерием, положенным в основу правил, формирующих синтаксические гипотезы, есть критерий максимальной согласованности связанных лексем по всем типам приписанной им комбинаторной информации – как морфологической, так и лексикографической. Информация первого типа извлекается из морфологической структуры, в которой она была выработана в процессе морфологического анализа; лексикографическая информация (включая синтаксическую, семантическую, сочетаемостную) извлекается из соответствующих статей комбинаторного словаря синтаксических правил и конструкций ЕЯ.

После окончания формирования множества синтаксических гипотез, алгоритм синтаксического анализа приступает к его оптимизации, устраняя из этого множества ложные гипотезы на основе некоторых универсальных и локальных требований к правильной синтаксической структуре предложения.

Синтаксические правила – сложные формальные объекты, состоящие из двух основных зон – зоны условий и зоны действий.

Условия в правилах представляют собой логические выражения, в записи которых могут быть использованы как элементарные, так и составные предикаты, с помощью которых можно описать наличие (или отсутствие) у слова определенных характеристик, всевозможные согласования между выделенными словами фразы, необходимый (или невозможный) линейный или древесный контекст выделенных слов фразы и т.д.

Зона действий в правилах – это перечень инструкций, последовательное исполнение которых осуществляет требуемое правилом преобразование рассматриваемого объекта. В подавляющем большинстве правил синтаксического анализа объектом преобразования выступает морфологическая структура.

Далее описан алгоритм синтаксического анализа [3].

Построение набора гипотетических синтаксических отношений для предложения P_k^l. В результате получим ориентированный граф гипотетических синтаксических отношений между словами предложения, состоящий из N узлов, где N – число слов анализируемого предложения.

Определение вершины синтаксической структуры.

Необходимо из графа выделить дерево – синтаксическую структуру исходного предложения. Этот процесс происходит значительно быстрее, если сразу удается определить вершину дерева зависимостей. Поэтому после построения графа начинает работать блок, задача которого – выявление слов – кандидатов на роль вершины дерева.

Описание свойств слов, которые могут быть вершинами в синтаксической структуре исходного предложения, задано в виде специального правила. Правило просматривает все омонимы всех слов предложения и помечает те из них, которые по своим характеристикам могут быть вершиной. Кроме того, с помощью приписывания некоторого веса кандидаты на роль вершины подаются в порядке убывания вероятности того, что это слово будет выбрано вершиной синтаксической структуры.

Применение шаблонов синтаксических конструкций и проверка содержимого построенного дерева.

Применение правил предпочтения. Если граф, оставаясь связным, деревом не считается, то происходит обращение к правилам предпочтения, позволяющим в отдельных случаях путем сравнения гипотетических синтаксических хозяев и слуг каждого слова оказать предпочтение одним гипотетическим связям, уничтожив другие.

Просмотр альтернативных деревьев. Описанный алгоритм приводит к построению по крайней мере одной правильной синтаксической структуры для обрабатываемого предложения.

Используя информацию, хранящуюся в словаре, блок формирования синтаксически связанных конструкций интерпретирует построенную синтаксическую структуру и формирует связки типа *подлежащее* ↔ *сказуемое*, *подлежащее* ↔ *дополнение* (как правило, это прилагательное, согласованное с существительным в роде, числе и падеже), *сказуемое* ↔ *дополнение* и др.

Подсистема семантического анализа. строит последовательно семантические структуры для каждого предложения входного текста. Семантическая структура состоит из семантических узлов и семантических отношений. Семантический узел – это такой объект текстовой семантики, у которого заполнены все валентности как эксплицитно выраженные в тексте, так и имплицитные – те, которые извлекаются из ЯОКМ. Применение ЯОКМ в подсистеме семантического анализа есть основным отличием от классических реализаций семантического анализа. Оно содержит кроме традиционной семантической информации (содержащейся в словарях и тезаурусах заданного ЕЯ) многоуровневые проекции как обобщенных, так и конкретизированных шаблонов представлений семантических конструкций простых предложений.

Из определения семантического узла также следует, что в процессе семантического анализа выделяются семантические узлы и их атрибуты, входящие в эти узлы. Как и на всех этапах анализа, семантические узлы образуются из слов исходного предложения. Главный источник гипотез о составе семантического узла дает синтаксический анализ. Многие синтаксические группы могут перейти в семантические узлы, другие должны превратиться в атрибуты узлов. Другим источником формирования семантических узлов служит ЯОКМ.

Кроме слов, семантические узлы могут включать в себя: знаки препинания, устойчивые обороты, устойчивые словосочетания, жесткие синтаксические группы и др.

Каждому узлу приписано множество атрибутов, таких, как набор графематических слов, из которых состоит данный узел; номер семантически главного слова в узле; грамматическая интерпретация узла (внешняя синтаксическая характеристика); номер фрагмента, которому принадлежит узел; предлог, который в синтаксисе управлял этим узлом; ссылка на словарную статью в ЯОКМ и др. (*www.aot.ru*).

Общая схема работы алгоритма может быть представлена следующей последовательностью шагов.

1. Инициализация семантических узлов.
2. Построение множества словарных интерпретаций узлов.
3. Построение групп времени.
4. Построение узлов из словосочетаний в кавычках.
5. Построение узлов типа «друг друга».
6. Построение устойчивых словосочетаний.
7. Построение лексических функций–параметров.

8. Установление отношений между локативными узлами.

9. Построение графа гипотетических связей.

10. Построение множественных актантов.

11. Проверка семантических узлов по семантическим характеристикам.

12. Проверка проективности построенной семантической конструкции.

13. Построение отношений по умолчанию.

14. Построение межфразных связей.

15. Построение анафорических связей.

На рис. 11 представлена блок-схема семантического анализа, а на рис. 12 – диаграмма состояний с описанием исполняемых процедур и анализируемыми условиями.

*p*171 – выполняет преобразование синтаксического дерева зависимостей предложения P_k^l в синтактико-семантическую структуру. При этом выполняются пп. 1 и 2 общего алгоритма семантического анализа, приведенного выше;

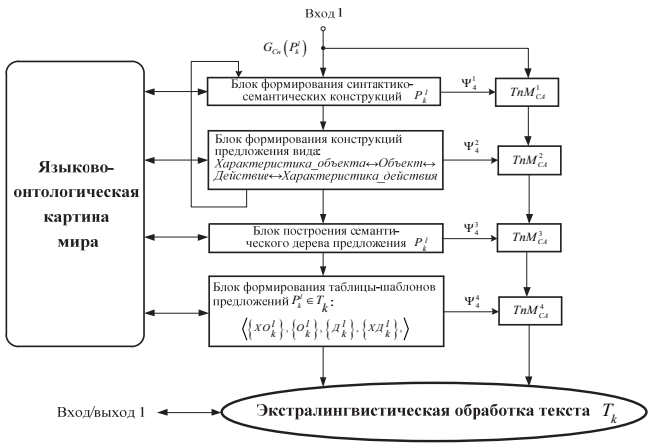


Рис. 11

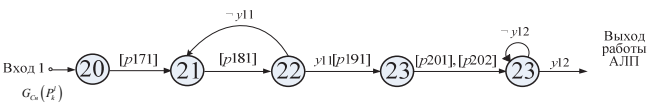


Рис. 12

*p*181 – формирует шаблон–конструкцию для навигации по ЯОКМ и итерационно находит наиболее подходящий вариант. При этом процедура может обратиться за информацией, полученной на любой стадии обработки. Выполняются пп. 3–8 общего алгоритма;

*p*191 – выполняет построение семантического дерева предложения P_k^l. При этом к сформированному предыдущей процедурой шаблону добавляются смысловые оттенки. Выполняются пп. 9–12 общего алгоритма;

*p*201 – к сформированному шаблону предложения добавляются недостающие связи. Выполняет окончательное формирование таблицы шаблонов текста T_k;

*p*202 – процедура завершения работы АЛП. По запросу блока экстралингвистической обработки передает содержимое таблицы шаблонов предложений текста T_k в целом.

Условия на диаграммах состояний интерпретируются так:

*y*11 – условие поиска подходящего варианта шаблона–конструкции для предложения $(P_k^l)_i$;

*y*12 – условие завершения работы АПЛ.

Как было показано, на вход подсистемы *семантического анализа* поступает синтаксическое дерево зависимостей предложения P_k^l.

Ее первые два блока, взаимодействуя с ЯОКМ, осуществляют перевод синтаксических конструкций предложения P_k^l в семантические в соответствии с обобщенным шаблоном: *Характеристика объекта* ↔ *Объект* ↔ *Действие* ↔ *Характеристика действия*, описанном ранее.

Далее, в блоке формирования семантического дерева выполняется разрешение синтаксической омонимии и устранение анафорических связей.

Алгоритм устранения анафорических связей выполняет замещение анафор соответствующими лексемами [5].

Такая конкретизация обобщенного шаблона уже достаточна для построения семантического графа предложения P_k^l. В таком графе вершины представляют собой сущности (объекты), а дуги – отношения (связи) между ними. Имена вершин и дуг обычно совпадают с именами соответствующих сущностей и отношений, используемых в естественном языке. Дуга и две связанные с ней вершины представляют минимальную значимую информацию – факт наличия связи определенного типа между соответствующими объектами.

В заключение работы АЛП будет сформирована таблица шаблонов предложений текста $\{T_k\}$, в которой хранится информационная структура для всего текста в целом, и которая

есть входной информацией для подсистемы экстралингвистической обработки текста $\{T_k\}$. В ней выполняется формально-логическое представление (перевод) предложений и текста в целом в подходящей формальной теории первого порядка, например, сначала в модифицированные концептуальные графы и затем в логику предикатов первого порядка.

Задача оптимального синтеза АЛП. Введение дополнительных аппаратных затрат с целью существенного выигрыша в быстродействии реализации алгоритмов лингвистического анализа требует рассмотрения вопроса оптимизации соотношения *Аппаратные затраты* (Q) ↔ *Время выполнения* (T). Такую оптимизацию можно выполнить на основе известного метода Парето.

Целевая функция в аналитическом виде находится одним из приближенных методов, например, линейной или нелинейной интерполяции или экстраполяции, по нескольким опорным точкам (структурные реализации алгоритма), которые получают путем предварительного формирования вариантов реализаций алгоритма. Из множества этих точек, где каждой r-й точке ($r = 1 \div m$) соответствует реализация с параметрами $<T_r, Q_r>$, формируется множество Парето на плоскости $T - Q$ с учетом упорядочения:

$$T_1 \le T_2 \le ... \le T_r \le ... \le T_m;$$
$$Q_1 \ge Q_2 \ge ... \ge Q_r \ge ... \ge Q_m.$$

В общем случае задача выбора (квази) оптимального варианта реализации алгоритма сводится к выбору некоторых граничных значений T_0, Q_0 (предельных параметров T_r, Q_r) и минимизации функционала:

$$F = \min_r \frac{c_r Q_r + b_r T_r}{c_r Q_0 + b_r T_0}, \qquad (7)$$

где $c_r = (c_1, c_2, ..., c_m), b_r = (b_1, b_2, ..., b_m)$ – нормативные коэффициенты (которые могут быть определены, например методом экспертных оценок), такие, что $\sum_{r=1}^{m} c_r = 1$, $\sum_{r=1}^{m} b_r = 1$ и $c_r \cdot b_r = k$, где k – коэффициент обратной пропорциональности. Суть этого коэффициента состоит в том, что чем меньше время T_r реализации алгоритма, тем больше необходимо затрат оборудования Q_r и наоборот.

Если функционал (7) найден и найдено минимальное значение $c = \min_r \frac{c_r Q_r + b_r T_r}{c_r Q_0 + b_r T_0}$, то $\forall c' \ge c$ получим неравенство $c'(c_r Q_0 + b_r T_0) \ge c_r Q_r + b_r T_r$.

Последнее неравенство означает, что какой бы ни был параметр c', реализация не будет превосходить значения нормативной суммы $c_r Q_0 + b_r T_0$.

Заключение. Анализ особенностей компьютерной обработки ЛКТ сверхбольших объемов показал, что для приложений, используемых в режиме реального времени, программной реализации лингвистического (и особенно морфологического) анализа недостаточно, так как часть информации может быть не обработана. Поэтому задача построения аппаратных лингвистических процессоров актуальна, и ее решение позволит: во-первых – сократить сроки предоставления пользователю оперативной информации (без потери части ее и снижения ее актуальности) для принятия решений; во-вторых – качественно повысить уровень лингвистических исследований с учетом большего количества параметров обработки.

Предложен метод многоуровневых проекций разработки структурной организации АЛП с учетом онтологического подхода, суть которого состоит в поэтапном отображении алгоритмов обработки информации в сети операционных автоматов АЛП и позволяет повысить в два и более раз производительность лингвистического анализа текстов больших объемов. Предложенная модель АС реализации АЛП в виде асинхронных комбинационных автоматов с памятью позволяет выполнять параллельную обработку словоформ предложения и допускает реконфигурацию структуры АЛП на физическом уровне как инструменте настройки (перестройки) на обработку ЛКТ заданной ПдО или решение задачи предметно-проблемной ориентации аппаратных средств. Выполнена оценка сложности реализации алгоритма лингвистического анализа.

Окончание на стр. 76

Разработана функциональная схема параллельного АЛП, включающая в себя подсистемы графематического, морфологического, синтаксического и поверхностно-семантического анализа. Для каждой из них созданы граф-схемы алгоритмов и соответствующие процедуры. Отличительные особенности АЛП – активное взаимодействие с информационной структурой языково-онтологической картины мира, что повышает степень точности распознавания семантических конструкций предложений текста, и средства реконфигурации, позволяющие реализовать механизмы настройки на обработку ЛКТ различных ПдО и разного объема и есть инструментом предметно-проблемной ориентации лингвистического процессора.

1. *Палагин А.В.*, *Опанасенко В.Н.* Реконфигурируемые вычислительные системы. – К.: Просвіта, 2006. – 280 с.
2. *Палагин А.В.*, *Крывый С.Л.*, *Петренко Н.Г.* Онтологические методы и средства обработки предметных знаний. – Луганск: Изд-во ВНУ им. В. Даля, 2012. – 324 с.
3. *Лингвистический* процессор для сложных информационных систем / Ю.Д. Апресян, И.М. Богуславский, Л.Л. Иомдин и др. – М.: Наука, 1992. – 256 с.
4. *Розробка* методів та засобів онтолого-лінгвістичного аналізу природномовних об'єктів / М.Г. Петренко, О.В. Палагін, В.Ю. Величко та ін. – К., 2009. – 38 с. – (Препр. / НАН України, Ін-т кібернетики ім. В.М. Глушкова; 2009-2).
5. *Про* один підхід до аналізу та розуміння природномовних об'єктів / О.В. Палагін, С.Ю. Світла, М.Г. Петренко та ін. // Комп'ютерні засоби, мережі та системи. – 2008. – № 7. – С. 128–137.

Поступила 17.06.2013
Тел. для справок: +38 044 526-3348 (Киев)