УДК 519.8

Г.Е. Цейтлин, В.А. Иовчев, А.А. Мусихин

МЕНТАЛЬНЫЕ АСПЕКТЫ МЕТОДОВ СИМВОЛЬНОЙ МУЛЬТИОБРАБОТКИ

Используя средства формализации и инструментарий алгебры алгоритмики, в данной статье спроектированы и реализованы нетривиальные алгоритмы символьной мультиобработки: сортировка, поиск и синтаксический анализ. Намечены перспективы для дальнейших исследований.

Введение

К числу актуальных и интенсивно развивающихся областей компьютерной науки относится алгебраическая алгоритмика (АА) [1]. В последние годы за рубежом была предложена концепция Ментального Программирования (International Programming, IP), которая неразрывно связана с формализацией программирования [2]. Данная концепция основана на моделировании программных систем и их реализации. В Украине исследования в данной области восходят к фундаментальным работам В. М. Глушкова по теории систем алгоритмических алгебр (САА) [3]. Дальнейшее развитие теории САА нашло свое воплощение в алгебре алгоритмики <АА> – в новом интенсивно-развивающемся направлении украинской алгебро-кибернетической школы [4].

Дальнейшее рассмотрение концепций IP и САА будет производиться в их сравнительном анализе. Современное состояние исследований САА связано с решением ряда важных теоретических и практических задач: построения сверхвысокого уровня языков проектирования и средств автоматизации процесса синтеза (сборки) программ. В свою очередь, IP представляет собой расширяемую среду программирования и метапрограммирования на основе так называемого "активного исходного кода" (active source), для которого характерно оригинальное поведение в части редактирования, отображения, отладки и контроля версий [5]. Основные принципы, на которых базируется IP, можно сформулировать таким образом:

- общие и предметно-ориентированные абстракции;
- биология программирования;
- экология программирования.

Следует заметить, что вышеприведенные принципы соответствуют базовым составляющим <АА>.

Абстракции представляются в виде взаимосвязанных форм проектирования алгоритмов:

- аналитическая – позволяет описывать алгоритмы в виде формул соответствующих алгебр, что является удобной для трансформации, а именно для их улучшения по выбранным критериям (памяти, скорости и т.д.);
- естественно-лингвистическая – позволяет проектировать алгоритмы на разных входных языках;
- визуальная – граф-схемная;

Биологическая компонента в <АА> – теория клонов, используется для описания, построения и исследования семейств алгебр. При этом сигнатура операций каждой алгебры из рассматриваемого семейства является функционально полной системой [5]. Для получения клона выбранной алгебры необходимо расширить ее сигнатуру путем добавления в нее операции суперпозиции. Следует отметить, что фундаментом для построения различных алгебр из некоторого семейства, служит решение проблемы функциональной полноты сигнатуры операций для клона, определяющего данное семейство. При использовании теории клонов можно описать разные языки, принадлежащие выбранному семейству. Примером служит язык САА и его модификации.

Экологическая компонента в <АА> представима в виде интегрированных ин-

© Г.Е. Цейтлин, В.А. Иовчев, А.А. Мусихин, 2008

струментальных средств разработки и синтеза алгоритмов и программ в объектных средах: МУЛЬТИПРОЦЕССИСТ, САА – машина, ДСП – конструктор, Трансформатор.

Метод многоуровневого структурного проектирования программ (МСПП) объединяет алгебраический и грамматический подходы по конструированию программ. Следует заметить, что основное назначение языков проектирования схем алгоритмов - представление алгоритмов в форме, ориентированной на человека и облегчающей процесс производства ПО. Для достижения цели возможно объединение разных технических средств преобразования и синтеза программ [6, 7].

Инструментальные средства МСПП появились еще в 80-х годах при разработке автоматизированной системы МУЛЬТИПРОЦЕССИСТ [6]. Данная система проектировалась путем «самораскрутки». Таким образом, компоненты системы, которые не относятся к синтезированным, получены за счет разработанных компонент системы. К числу средств инструментальной поддержки МСПП относятся языки проектирования в терминах САА-схем многоуровневых формализованных проектов классов алгоритмов и программ, структур данных и памяти. Схемы допускают лингвистические, аналитические и визуальные представления в рамках интегрированной технологии проектирования классов алгоритмов и программ. Сама технология базируется на соответствующих алгебрах алгоритмов. Математическим базисом данного языка является аппарат САА и его модификации [3]. Метод МСПП реализован в виде САА-машины, которая является интерпретатором САА-схем, предназначенных для отладки и верификации [7]. ДСП-конструктор – диалоговый конструктор синтаксически правильных САА-схем [8, 9, 10]. Трансформатор – инструмент для преобразования аналитических спецификаций алгоритмов с использованием соотношений и тождеств соответствующей алгебры [10].

Таким образом, в <АА> нашли свое воплощение такие компоненты IP: абстракция, биология, экология.

Изложение материала данной работы подчинено такой структуре:

- в разделе 1 изложена сущность аппарата ГСП, сочетающего понятия САА с проектированием алгоритмов в терминах грамматического вывода;
- подход к проектированию средствами САА продемонстрирован на примере синтаксического анализатора функционирующего по методу Кока–Янгера–Касами изложено в разделе 2;
- в разделе 3 охарактеризована реализация представленных проектов в последовательных и параллельных средах;
- в заключении изложены основные результаты выполненного исследования и намечены его ближайшие перспективы.

1. Грамматики структурного проектирования, П-программа

Грамматики структурного проектирования (ГСП) [6, 7] были ориентированы на решение таких важных проблем как:

- синтаксиса – описание синтаксически правильных цепочек;
- семантики – описание не только синтаксически, но и семантически правильных цепочек языка;
- прагматики – нацеленные на практическое применение аппарата ГСП для разработки ряда конкретного ПО.

Под ГСП будем понимать некую формальную систему $G = (T, N, R, P, U)$, составляющими которой являются следующие компоненты:

- T – терминальный алфавит;
- N – нетерминальный алфавит, которому принадлежат метапеременные логические, операторные, объектные, АТД, АТП, характеризующие разные уровни проектирования;
- U – механизм управления выводом;
- R – принадлежит нетерминальному алфавиту, это аксиома которая идентифицирует проектированный класс;
- P – совокупность помеченных постановок логического, операторного, объектного типов, детализирующие АТП,

АТД и используются при проектировании алгоритмов и программ.

Рассматриваемые терминальный алфавит, состоящий из совокупности базисных условий, операторов, объектов, абстрактный тип данных (АТД), абстрактный тип памяти (АТП), определяющие степень конкретизации проектированного класса алгоритмов и программ. А так же совокупность разделителей – символов операций сигнатуры САА-М, скобок, ограничителей и др.

Проектируемый класс программ $L(G) = \{X \mid R => X\}$ подмножество $F(T)$, которые выводятся из аксиомы в ГСП, создает ассоциированный с классом язык, который порожден данной грамматикой. С помощью механизма U последовательного, параллельного или комбинированного управления выводом в ГСП, реализуются контекстные зависимости по памяти и данным между проектируемыми программными модулями.

Следует заметить, что аппарат ГСП положено в основу метода МСПП и его инструментария – системы МУЛЬТИПРОЦЕССИСТ.

В нашем случае особый интерес представляют матричные ГСП (П-программа) с последовательным, параллельным или комбинированным применением подстановок в обобщенных матричных продукциях. При этом подстановки, которые применяются последовательно, записываются в строку и разделяются точкой с запятой, а параллельно – в столбец.

В качестве примера ГСП приведем П-программу, ориентированную на сортировку и поиск.

$$m_0: \left\| R - > \text{ПРС(МАС)} \times \text{СБОРКА(Ф)} \right\|,$$

$$m_1: \left\| \begin{array}{l} \text{МАС} - > \text{МАС}_1, \text{МАС} \\ \text{ПРС} - > \text{A}_1(t) \vee \text{ПРС}; t -> \text{МАС}_1 \\ \Phi - > \Phi_1, \Phi \end{array} \right\|,$$

$$m_2: \left\| \begin{array}{l} \text{МАС} -> \text{МАС}_{i+1}, \text{МАС} \\ \text{A}_i\ (\text{МАС}_i) \vee \text{ПРС} ->\text{A}_i\ (\text{МАС}_i) \vee \text{A}_{i+1}(t) \vee \text{ПРС} \\ t -> \text{МАС}_{i+1} \\ \Phi -> \Phi_{i+1}, \Phi \\ \text{МАС} -> \text{МАС}_{i+1} \end{array} \right\|,$$

$$m_3: \left\| \begin{array}{l} \text{A}_i\ (\text{МАС}_\text{i}) \vee \text{ПРС} - >\text{A}_i\ (\text{МАС}_i) \vee \text{A}_{i+1}(t) \\ t - > \text{МАС}_{i+1} \\ \Phi - > \Phi_{i+1} \end{array} \right\|.$$

Структуры данных обрабатываются ветками параллельной регулярной схемы (ПРС) и оператором СБОРКА. Матрицей подстановок m_1 формируется первая ветка. Потом матрицей m_2 рекурсивно формируются следующие ветки ПРС (при $i = 1, \ldots, n - 1$); количество веток n определяется в зависимости от имеющихся ресурсов. Процесс завершается применением матрицы m_3 (при $i = n - 1$).

Каждую из параллельных веток A_i ($МАС_i$) можно дальше интерпретировать через соответствующие матричные продукции для реализации процессов левосторонней, правосторонней или комбинированной обработки подмассива $МАС_i$. Подключая к продукции ГСП П-программы матричные продукции m_3, m_4 получаем ГСП сортировка методом ЧЕЛНОК:

$$m_3: \left\| \begin{array}{l} \text{A}_i\ (\text{МАС}_i) - > \text{ЧЕЛНОК}>< (\text{МАС}_i), \\ \text{МАС}_i - > \text{M}_i, \Phi_i - > L_i \end{array} \right\|,$$

$$m_4: \left\| \begin{array}{l} \text{СБОРКА}(L_1, \ldots, L_k) - > \text{СУМ}(L_1, \ldots, L_k) \end{array} \right\|,$$

где ЧЕЛНОК>< – это асинхронная встречная сортировка данным методом.

Таким образом различная интерпретация компонент, входящих в правые части продукций ГСП П-программы, позволяет получить матричные ГСП. В результате, порождаются те или иные классы алгоритмов, ориентированные на распределенную асинхронную мультиобработку.

Используемая при проектировании ГСП П-программа является ядром матричных ГСП, порождающие различные классы многослойных алгоритмов и программ. Суть многослойной обработки состоит в распределении массива входных данных на попарно непересекающиеся подмассивы для одновременной обработки по параллельным ветвях с последующей сборкой полученных промежуточных результатов. При этом используется параметрическая запись продукций, обеспечивающая самонастройку ГСП [7].

Известным фактом является то, что алгоритмы сортировки неразрывно связаны с соответствующими алгоритмами поиска. В дополнении к фундаментальным работам Дональда Кнута по сортировке и поиску [11]: существует возможность по любому алгоритму сортировки, построить соответствующий алгоритм поиска и наоборот. Известны следующие метаправила вывода в ГСП: свертка (укрупнение), развертка (детализация), переинтерпретация (последовательное выполнение свертки и развертки) и трансформация (преобразование схем алгоритмов с целью их улучшения по выбранным критериям).

Таким образом, механизм переинтерпретации позволяет преобразовать сортировку в поиск [12].

На примере вышеизложенной матричной ГСП (сортировка методом ЧЕЛНОК) получаем ГСП (поиск методом ЧЕЛНОК):

$$m_3: \left| \begin{array}{l} A_i\,(МАС_i) -> ПОИСК\,(МАС_i), \\ МАС_i -> М_i,\ \Phi_i -> L_i \end{array} \right|,$$

где ПОИСК – это асинхронный встречный поиск методом ЧЕЛНОК.

Методы сортировки и поиска обеспечивают построение простейших баз данных, к их числу можно отнести, например, базу данных успеваемости студентов.

2. Синтаксический анализ методом Кока–Янгера–Касами

Проблема синтаксического анализа состоит в установлении правильности обрабатываемых символьных цепочек – их принадлежность к соответствующему входному языку (задача контроля) и определение синтаксической структуры анализируемых цепочек (задача анализа). Для решения проблемы синтаксического анализа представлен известный метод Кока–Янгера–Касами (Cocke–Younger–Kasami), использующий контекстно-свободные грамматики, представленные в бинарной форме, для описания формальных языков. В качестве результата получаем ответ на вопрос, выводима или нет анализируемая терминальная цепочка символов в данной грамматике [11, 1].

Представим модификацию матричного метода синтаксического анализа (МАТРИЦА), основанного на представлении матриц контроля и анализа в плоской треугольной форме, с которой связан диагональный способ их вычислений.

Пусть $G = (T, N, @, P)$ – бесконтекстная грамматика в бинарной форме (T – терминальный алфавит; N – нетерминальный алфавит; $@$ – аксиома; P – множество правил) и $x = s_1 s_2 \ldots s_n \in F(T)$ – непустая терминальная цепочка из языка L. Под плоской треугольной матрицей контроля цепочки x по грамматике G понимается матрица $R = ||R_{ij}||$, где $0 <= I < n$, $i < j <= n$, элементами которой являются подмножества $R_{ij} \in N$, такие, что $\phi \in R_{ij}$ когда $\phi => s_{i+1} s_{i+2} \ldots s_j$.. Иными словами, матрица содержит все нетерминальные символы, порождающие цепочку x. Если $@ \in R_{0n}$, то контролируемая цепочка x является правильной, $x \in L(G)$. Такая матрица, кроме информации, связанной с распознаванием, содержит много дополнительных сведений о грамматической структуре данного предложения (цепочки). Пусть $d = j - i$ – длина подцепочки $x' = s_{i+1} s_{i+2} \ldots s_j$, соответствующей элементу R_{ij} матрицы контроля R цепочки x по грамматике G. Рассмотрим индуктивную процедуру вычисления элементов матрицы R по ее диагоналям на рис. 1.

Процедура 1.

1. При $d=1$, $R_{ij}=\{\phi \mid \phi \rightarrow s_j \in P\} \in N$, где $i = 0, 1, \ldots, n - 1$; $j = i+1$; вычисляются элементы первой диагонали матрицы R, которые расположены над главной диагональю $R_{tt} = 0$, для любого $t = 0, 1, \ldots, n$.

2. Пусть уже вычислены элементы диагоналей d' ($1 <= d' < d$). Элемент R_{ij}, принадлежит диагонали d, так, что $j - i = d$. Элемент $R_{ij} = \{\phi \mid \phi \rightarrow \phi_1 \phi_2 \in P, \phi_1 \in R_{ik}, \phi_2 \in R_{kj}\} \in N$, где $i=1, 2, \ldots, n-d$; $i+1 <= k <= j-1$, причем элементы R_{ik}, R_{kj} принадлежат ранее вычисленным диагоналям.

Проиллюстрируем сущность рассмотренной процедуры (1) при построении треугольной матрицы.

Пример 1.

Рассмотрим грамматику G в бинарной форме с правилами:

$C \rightarrow AC \mid AB;$
$A \rightarrow a$
$B \rightarrow BB \mid b.$

Пусть $x = aabbb$ входная цепочка, в качестве аксиомы выступает C. Треугольная матрица R имеет следующий вид:

–	A	–	C	C	C
–	–	A	C	C	C
–	–	–	B	B	B
–	–	–	–	B	B
–	–	–	–	–	B
–	–	–	–	–	–

Первая диагональ матрицы R построена по цепочке *aabbb* с применением заключительных продукций: $A \rightarrow a$; $B \rightarrow b$, заданной грамматики G. Затем последовательно вычисляются остальные диагонали матрицы согласно процедуре 1 с использованием бинарных продукций данной грамматики. Например, элемент $R_{03} = C$, так как для продукции $C \rightarrow AC$ и $k = 1$ имеем $A \in R_{01}$ и $C \in R_{13}$. Аналогично, $R_{13} = C$, на основании продукции $C \rightarrow AB$ и $k = 2$, ввиду $A \in R_{12}$ и $B \in R_{23}$.. В результате, $x = aabbb \in L(G)$, так как $C \in R_{05}$.

Матрица контроля используется в качестве основы для построения матрицы анализа.

Анализ предложения в языке $L(G)$ описание того, как предложение порождается G. С матрицей контроля R цепочки $x = s_1 s_2 \ldots s_n \in T$ по грамматике G в бинарной форме ассоциирована *треугольная матрица анализа* P данной цепочки, такая, что $P = ||P_{ij}||$, где $0 <= i <= n$, $i < j <= n$, причем элементы P_{ij} удовлетворяют следующим соотношениям:

$P_{0n}=@$, если $@ \in R_{0n}$;
$\phi \in P_{ij}$, если $\phi \rightarrow s_{i+1} s_{i+2} \ldots s_j$,
$@ \rightarrow s_1 s_2 \ldots s_i \phi s_{j+1} \ldots s_n$.

Треугольная матрица анализа P строится по соответствующей матрице контроля R посредством индуктивной процедуры, применяемой в направлении сверху вниз согласно стратегии развертки.

Процедура 2.

Формируем элемент $P_{0n}=@$, если $@ \in R_{0n}$, т.е. анализируемая цепочка $x \in L(G)$ в соответствии с уже построенной матрицей контроля R.

Пусть элемент P_{ij} определен. Для элементов P_{ik}, P_{kj} $(i+1 <= k <= j-)$ выполняются соотношения $\phi_1 \in P_{ik}$, $\phi_2 \in P_{kj}$, если $\phi_1 \in R_{ik}$, $\phi_2 \in R_{kj}$ и существует нетерминал $\phi \in P_{ij}$ и продукции вида $\phi \rightarrow \phi_1 \phi_2$, где R_{ik}, R_{kj} –соответствующие элементы матрицы контроля R.

Проиллюстрируем сущность рассмотренной процедуры при построении треугольной матрицы.

–	A	–	–	–	C
–	–	A	–	–	C
–	–	–	B	B	B
–	–	–	–	B	B
–	–	–	–	–	B
–	–	–	–	–	–

Пример 2.

Построим матрицу анализа цепочки $x = aabbb$ по грамматики G и матрице контроля R (из примера 1). Применяя к матрице контроля R процедуру 2, получим матрицу анализа P данной цепочки. По матрице анализа P может быть построен левый вывод $C \rightarrow AC \rightarrow aC \rightarrow aAB \rightarrow aaB \rightarrow aaBB \rightarrow aabB \rightarrow aabBB \rightarrow aabbB \rightarrow aabbb$ цепочки x в грамматике G.

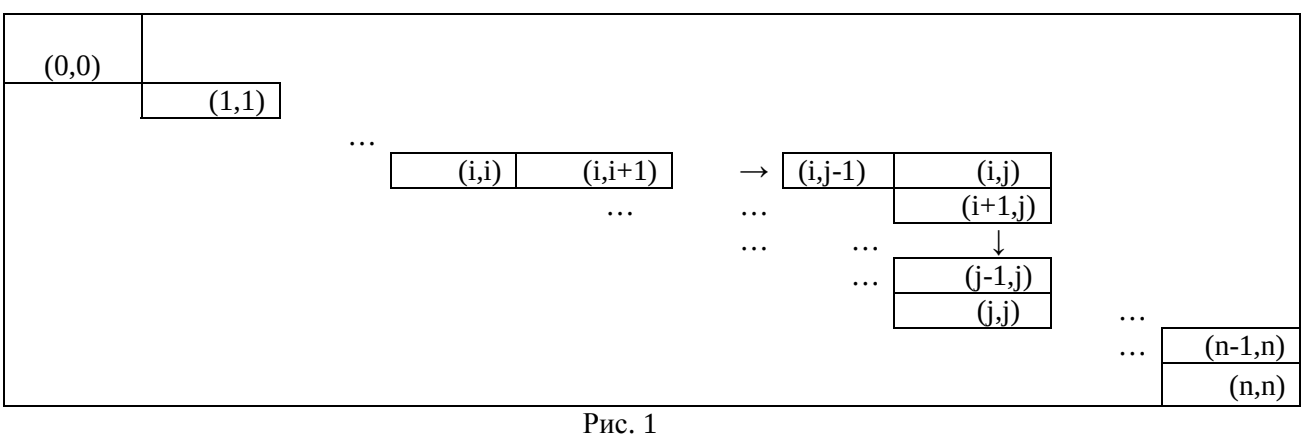

Рис. 1

При параллельном вычислении эффективно используется стратегия «двусторонней символьной мультиобработки диагоналей» [1]. На основе данной стратегии был реализован CYK-алгоритм с конвейерной организацией процесса вычисления, работа которого представлена в следующем примере.

Пример 3. Грамматика и входная цепочка используются с примера 1. Для наглядности, следует расположить диагонали треугольной матрицы построчно. На рис. 2 показан процесс вычисления после 5-го такта (полное время вычисления 8 тактов).

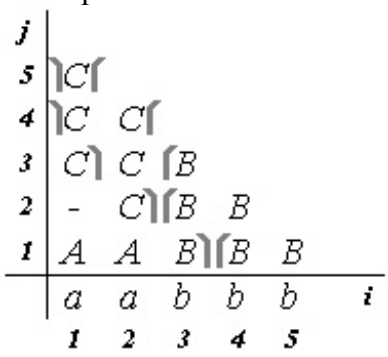

Рис. 2

Еще одним вариантом мультиобработки, параллельном вычислении матрицы, является «синхронное вычисление элементов диагоналей». Расчет элементов треугольной матрицы для фиксированной диагонали, при условии, что предыдущая диагональ уже вычислена, может происходить независимо.

На рис. 3 показан процесс вычисления диагонали, элементы которой распределяются на proc частей, согласно количеству параллельных веток.

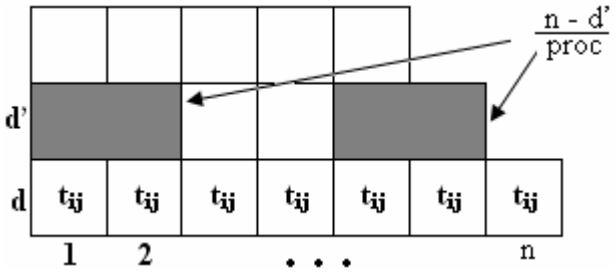

Рис. 3

Каждая часть вычисляется отдельно, независимо от другой части на этой же диагонали. После завершения вычисления всех частей, получаем полностью обработанную диагональ.

3. Реализация

Средствами <AA> спроектированы и реализованы:

- ГСП П-программа (сортировка и поиск);
- алгоритм Кока–Янгера–Касами в последовательной и параллельной формах.

В процессе реализации ГСП П-программы решены следующие задачи: реализация алгоритмов сортировки и поиска, представленных в форме ГСП П-программы при этом предусмотрена возможность обработки больших массивов с равномерным разделением на подмассивы, их локальной сортировкой и последующим слиянием. Отметим, что был реализован весьма не тривиальный адаптивный алгоритм параллельной сортировки АЛЬТ/А, частным случаем АЛЬТ/А служит известный алгоритм Дейкстры ЧЕЛНОК (прямые вставки). При этом оказалась необходимой реализация памяти, функционирующей по принципу «эластичной ленты», обобщающей известные структуры памяти как «магазин» и «очередь», а также их сочетания. По каждому из использованных алгоритмов сортировки посредством метаправила переинтерпретации получены соответствующие алгоритмы поиска.

Отметим, что частным случаем ГСП служат проекты алгоритмов представленные в САА. Поэтому реализация анализатора МАТРИЦА осуществлена по вышеизложенной схеме – последовательная модель и перенос ее на параллельную кластерную реализацию. Распараллеливание осуществлялось в процессе встречной обработке диагоналей, с исключением тупиков и согласованием обработанных ветвей.

Последовательная реализация позволила оценить свойства и перспективы для последующей разработки и интеграции в параллельные среды (кластер и GRID-технологии). Для проектирования соответствующих алгоритмов использован инструментарий, созданный сотрудниками Института программных систем НАН Украины. Для программной реализации использован язык программирования Си/Си++ и библиотеки MPI.

Заключение

В рамках современного состояния алгебры алгоритмики [14] в данной работе получены такие результаты:

- рассмотрены методы матричной символьной мультиобработки: параллельные алгоритмы сортировки, поиска и синтаксического анализа;
- спроектированы последовательные, параллельные (синхронные и асинхронные) классы алгоритмов;
- одновременно с проектированием алгоритмов осуществлена их программная реализация, с использованием существующего инструментария;
- по полученным результатам проведено статистический анализ эффективности полученных реализаций. Механизм ГСП использован в разработанной базе данных по успеваемости студентов в ВУЗе.

Тем самым реализована простейшая база данных в корой сортировка сочетается с поиском для получения как индивидуальной, так и групповой обрабатываемой информации с ее доступностью современными Веб технологиями.

Схемный подход характерный для алгебры алгоритмики в целом, и выполненного исследования открывает в качестве перспективы возможность последующих разработок:

- использовать аппарат ГСП при проектировании различных задач последовательной и параллельной обработки информации, в частности финансово-экономической, обучающей (дистанционного обучения), связанной с описанием семантики Веб и прочее;

- созданных в ИПС инструментов связано с дальнейшим развитием инструментария: реляционные базы данных (включая и корпоративные), создание нетривиальных алгоритмов символьной мультиобработки их программной и аппаратной реализации, совершенствование технологии скрин-ридеров, разработка средств взаимодействия с компьютером лиц с физическими ограничениями (включая недостатки зрения).

1. *Ноден П., Китте К.* Алгебраическая алгоритмика. – М.: Мир. – 1999. – 720 с.
2. *Чарнецкий К., Айзенекер У.* Порождающее программирование. Методы, инструменты, применение. – Изд. дом. Питер, 2005. – 736 с.
3. *Глушков В.М. , Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – Киев: Наук. думка., 1989. – 328 с.
4. *Цейтлин Г.Е.* Алгебры Глушкова и теория клонов // Кибернетика и системный анализ. – 2003. – № 4. – 48–58 с.
5. *Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., и др.* Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. – М.: Финансы и статистика, 1989. – 208 с.
6. *Цейтлин Г.Е., Суржко С.В., Ющенко К.Л., Шевченко А.И.* Алгоритмические алгебры. – Киев: 1997. – 342 с.
7. *Цейтлин Г.Е.* Конструирование алгоритмов символьной обработки // Кибернетика и системный анализ. – 1993. – № 2 – С. 17–30.
8. *Яценко О.А.* Інтегровані алгебро-алгоритмічні моделі мультиобробки // Вісн. Київськ. нац. у-ту. Серія: фіз.-мат. науки. Спец. випуск за матеріалами Міжнар. конф. “Теоретичні та прикладні аспекти побудови програмних систем” (TAAPSD’2004). – 2004.
9. *Яценко Е.А., Мохница А.С.* Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ // Проблемы программирования. Спец. выпуск по материалам 4-й Междунар. научно-практической конф. по программированию УкрПРОГ'2004. – 2004. – № 2/3. – С. 440–450.
10. *Кнут Д.* Искусство программирования для ЭВМ // Сортировка и поиск. – 1978. – Т.3. – 824 с.
11. *Цейтлин Г.Е.* Введение в алгоритмику. – Киев: Сфера, 1998. – 310 с.
12. *Янгер Д.Х.* Распознавание и анализ контекстно-свободных языков за время n3 // Проблемы математической логики. – М.: Мир, 1970. – С. 344–362.
13. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Методы символьной мультиобработки. – Киев: Наук. думка, 1980. – 252 с.
14. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., и др.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев: Академпериодика, 2007– 634 с.

Получено 14.12.2007

Об авторах:

Цейтлин Георгий Евсеевич,
доктор физико-математических наук,
профессор,

Иовчев Владимир Александрович,
аспирант, Института программных систем,

Мусихин Александр Александрович,
аспирант, Института программных систем.

Место работы авторов:

Институт программных систем
НАН Украины,
03680, Киев-187,
Проспект Академика Глушкова, 40.

e-mail: tseytlin@vikno.relc.com,
badt@meta.ua, molnija@ukr.net