УДК 681.3

СРЕДСТВА ПАРАМЕТРИЧЕСКИ УПРАВЛЯЕМОЙ ГЕНЕРАЦИИ АЛГОРИТМОВ НА ОСНОВЕ АЛГЕБРЫ ГИПЕРСХЕМ

Е.А. Яценко

Институт программных систем НАН Украины,
03187, Киев, проспект Академика Глушкова 40.
Тел.: 526 3559, e-mail: oayat@ukr.net

Предложен подход к разработке последовательных и параллельных алгоритмов на основе использования инструментария параметрически управляемой генерации схем программ. Инструментарий базируется на абстрактно-автоматной модели процесса генерации регулярных схем, ассоциированных с ней алгебрах гиперсхем и грамматиках структурного проектирования. Программные средства ориентированы на построение алгоритмов и гиперсхем в режиме поуровневого диалогового конструирования, обеспечивающем их синтаксическую правильность.

The approach to development of serial and parallel algorithms, which is based on usage of tools for parameter-driven generation of schemes, is proposed. The tools are based on the abstract-automaton model of regular schemes generation process, associated algebras of hyperschemes and grammars of structured design. Software facilities are designed for construction of algorithms and hyperschemes in the mode of dialogue constructing, providing their syntactic regularity.

Введение

Одним из важных направлений информатики является алгебраическое программирование, возникшее на стыке алгебры, логики, схематологии и ориентированное на решение проблем формализации, обоснования правильности и трансформации программ [1]. К указанному направлению относится, в частности, алгебра алгоритмики, базирующаяся на высокоуровневых спецификациях алгоритмов и программ, восходящим к системам алгоритмических алгебр В.М. Глушкова (САА). Задачи построения эффективных моделей и методов как последовательной, так и параллельной обработки занимают одно из центральных мест в алгебре алгоритмики. При этом важной остается проблема повышения адаптивности программ к конкретным условиям их использования. В частности, она может быть решена за счет использования параметрически управляемой генерации высокоуровневых спецификаций алгоритмов посредством схем более высокого уровня, называемых гиперсхемами [2].

В данной статье предложен подход к разработке последовательных и параллельных алгоритмов на основе алгебр гиперсхем и инструментальных средств генерации алгоритмов, разработанных в [3]. Упомянутые программные средства являются одним из основных компонентов созданного Интегрированного инструментария Проектирования и Синтеза программ (ИПС) [1]. Разрабатываемые в системе спецификации алгоритмов оформляются посредством структурированных (регулярных) схем программ (РС). Прототипом инструментария послужили рассмотренные в [2] алгебраический аппарат и инструментальная система структурного синтеза алгоритмов и программ по их иерархическим спецификациям (МУЛЬТИПРОЦЕССИСТ). Отличием программных средств, предлагаемых в данной работе, является использование метода диалогового конструирования синтаксически правильных программ [1].

Процессы генерации программ в работе формализуются посредством операторных регулярных представлений, – регулярных гиперсхем, которые в сочетании со средствами управления выводом используются для представления грамматик структурного проектирования (ГСП) [1] – алгебро-грамматических систем подстановок, левая и правая части которых являются термами в САА. Подход проиллюстрирован на примерах из области сортировки и линейной алгебры.

Рассматриваемый в статье алгебраический аппарат и средства его программной поддержки принадлежат к методам трансформационного синтеза. При этом среди упомянутых методов смежными к данному являются, в частности, средства переписывания термов [4, 5], смешанные вычисления [6], конкретизирующее программирование [7], гиперпрограммирование [8] и макрогенерация.

Статья состоит из четырех разделов. Раздел 1 посвящен абстрактно-автоматной модели процесса генерации программ и связанному с ней аппарату гиперсхем, ориентированному на генерацию последовательных и параллельных РС. В разделе 2 рассматриваются механизмы управления выводом в грамматиках структурного проектирования, основывающиеся на алгебре гиперсхем. В разделе 3 описаны разработанные программные средства, ориентированные на автоматизацию процессов генерации РС по гиперсхемам.

©Е.А. Яценко, 2012

1. Алгебраический аппарат генерации регулярных схем программ

Данный раздел посвящен рассмотрению абстрактно-автоматной модели процесса генерации схем программ и связанной с ней алгебры гиперсхем (АГС). Предложены примеры применения АГС для представления адаптивных алгоритмов сортировки.

В основу рассматриваемого алгебраического аппарата положена абстрактно-автоматная модель ЭВМ и САА Глушкова [1, 2]. Подобно модели ЭВМ, модель параметрически управляемого генератора схем функционирует по принципу обратной связи. В качестве управляющего автомата используется магазинный автомат Ψ, а в качестве операционного автомат Φ, снабженный лентой L. На ленте L формируется генерируемая регулярная схема. Множество M состояний автомата Φ ассоциируется с параметрами, управляющими генерацией схем. Элементы информационного множества $P = M \times L$ (где L – множество состояний ленты L) называются состояниями операционной структуры. На каждом шаге работы из операционного в управляющий автомат поступает набор значений логических условий, определенных на множестве P, в зависимости от которых, а также от содержимого магазина M, управляющий автомат инициирует выполнение некоторого оператора. Множество операторов состоит из терминальных и нетерминальных операторов. Выполнение терминального оператора состоит в изменении текущего состояния операционной структуры, что, в частности, может выражаться в записи на ленту L некоторого текста. Выполнение нетерминального оператора A при текущем состоянии $p \in P$ состоит в записи в магазин M некоторого терма $F(A, p)$ и его последующей интерпретации управляющим автоматом; более подробно назначение функции F будет рассмотрено далее. Терм $F(A, p)$ является аналогом понятий макроопределения, процедуры, подпрограммы и др. Магазин M используется при обработке вложенных и рекурсивных термов. Генерируемая схема представляет собой содержимое ленты L в заключительном состоянии операционной структуры.

Приведенной абстрактно-автоматной модели в работе [2] поставлена в соответствие алгебра гиперсхем – аппарат для формализации параметрически управляемых алгоритмов генерации операторных представлений в САА. АГС представляет собой двухосновную алгебру $\sigma = <U, V>$, где U – множество операторов, заданных на множестве P; V – множество условий четырехзначной логики, представляющих собой отображения множества P в множество $\{0, \mu, \eta, 1\}$, где 0 – “ложь”, μ – “неопределенность”, η – “не вычислено”, 1 – “истина”. Применение оператора $A \in U$ к состоянию $p \in P$, приводит к переводу операционной структуры в новое состояние $A(p) \in P$ и записи на ленту некоторого (быть может, пустого) фрагмента $F(A, p)$ текста генерируемой РС.

В сигнатуру АГС входят операции, аналогичные операциям САА: дизъюнкция ($\alpha \vee \beta$); конъюнкция ($\alpha \wedge \beta$); отрицание ($\overline{\alpha}$); композиция A * B; альтернатива ($[\alpha]$ A, B); цикл $\{[\alpha]$ $A\}$; операция выбора (переключатель): ВЫБОР($\alpha_1 \to A_1$, $\alpha_2 \to A_2$, …, $\alpha_n \to A_n$) и другие операции [2].

Для упомянутых операций в работе [2] определена функция $F(A, p)$, задающая способ генерации текста для операций АГС над аргументами, для которых $F(A, p)$ известна. Например, операция альтернативы ($[\alpha]$ A, B), где $\alpha \in V$; $A, B \in U$; порождает оператор $C = ([\alpha]$ A, $B)$ такой, что $\forall p \in P$:

$$F(C, p) = \begin{cases} F(A, p), \text{если } \alpha(p) = 1; \\ F(B, p), \text{если } \alpha(p) = 0; \\ ([\alpha[p)]\, F(A, p), F(B, A(p))), \text{если } \alpha(p) = \eta; \\ e, \text{сли } \alpha(p) = \mu, \end{cases}$$

где e — пустое слово. В соответствии с данным определением, при истинном $\alpha(p)$ порождается текст, соответствующий оператору A, при ложном – оператору B, а при не вычисленном $\alpha(p)$ генерируется сама операция альтернативы.

Определения функции F для остальных операторных конструкций приведены в [2]. Отметим, что для каждого элементарного оператора A и элементарного условия α, входящих в систему образующих множеств U и V соответственно, разработчиком алгоритма также задается F – прообраз данного оператора или условия в порождаемой РС.

Представления операторов в АГС посредством суперпозиции операций из ее сигнатуры и образующих элементов (базисных операторов и условий) называются регулярными гиперсхемами (РГС). Каждая РГС A, воздействуя на состояние $p \in P$, порождает РС F (A, p). РГС, имеющие многоуровневую структуру и допускающие рекурсию, были названы АГС-схемами [2].

Пример 1. Проиллюстрируем применение аппарата АГС на задаче ввода и сортировки некоторого потока входных числовых массивов. В приведенном далее алгоритме входные массивы, длина которых n меньше некоторого порога *MIN*, обрабатываются алгоритмом сортировки вставками (*insertionSort*), массивы длиной больше *MAX* – быстрой сортировкой (*quickSort*); массивы, попадающие в интервал [*MIN*, *MAX*], – алгоритмом сортировки слиянием (*mergeSort*) [9].

Описанному процессу соответствует РС

СОРТИРОВКА = ИНИЦ * {[КОНЕЦ_ПОТОКА] ВВОД_МАССИВА(A) *

$$* ([n < MIN]\ insertionSort(A, n),$$
$$([n > MAX]\ quickSort(A, n), mergeSort(A, n)))\}.$$

Пусть заранее известно, что описанный схемой алгоритм будет применяться в условиях, когда длина всех вводимых массивов находится в определенном диапазоне, скажем, не меньше *MIN*. Тогда приведенная РС становится избыточной. Рассматривая ее как гиперсхему, можно полагать, что на этапе генерации РС по данной схеме условие $n < MIN$ принимает значение 0, тогда как $n < MAX$ принимает значение η. Полагая, что отображение *F* является тождественным на множестве базисных операторов и условий, входящих в приведенную схему, получим сокращенную РС

$$F(\text{СОРТИРОВКА}, P_0) = \text{ИНИЦ} * \{[\text{КОНЕЦ\_ПОТОКА}]\ \text{ВВОД\_МАССИВА}(A) *$$
$$* ([n > MAX]\ quickSort(A, n), mergeSort(A, n)))\}.$$

где P_0 – начальное состояние информационного множества.

Таким образом, устанавливая различные значения параметра *n* гиперсхемы СОРТИРОВКА (в рассмотренном случае было установлено значение $n \geq MIN$), можно получить конкретизированную РС, ориентированную на специфику ее использования и оптимальную для заданных условий.

Пример 2. В работе [9] представлен более сложный вариант схемы адаптивной сортировки, построенный с использованием методов машинного обучения. Упомянутый алгоритм также может рассматриваться как гиперсхема; далее приведен его фрагмент в естественно-лингвистическом виде:

```
ГИПЕРСХЕМА adaptiveSort(A, n)
====
ЕСЛИ ‘size <= 30’ ТО
  ЕСЛИ ‘runs <= 0.7’ ТО "insertionSort(A, n)"
  ИНАЧЕ ЕСЛИ ‘runs > 0.7’ ТО
        ЕСЛИ ‘size <= 10’ ТО "insertionSort(A, n)"
        ИНАЧЕ ЕСЛИ ‘size > 10’ ТО "quickSort(A, n)"
            КОНЕЦ ЕСЛИ
        КОНЕЦ ЕСЛИ
      КОНЕЦ ЕСЛИ
  КОНЕЦ ЕСЛИ
ИНАЧЕ ЕСЛИ ‘size > 30’ ТО
      ЕСЛИ ‘size <= 40’ ТО "insertionSort(A, n)"
      ИНАЧЕ ЕСЛИ ‘size > 40’ ТО "quickSort(A, n)"
          КОНЕЦ ЕСЛИ
      КОНЕЦ ЕСЛИ
    КОНЕЦ ЕСЛИ
КОНЕЦ ЕСЛИ
```

В приведенной схеме выбор одного из алгоритмов осуществляется на основе длины *n* входного массива *A* и степени его отсортированности (*runs*). Пусть заранее известно, что описанный приведенной схемой алгоритм будет применяться в условиях, когда длина всех вводимых массивов $n < 20$, а степень отсортированности *runs* = 0.9. Тогда в результате генерации получим сокращенную САА-схему:

```
СХЕМА adaptiveSort'(A, n)
====
ЕСЛИ ‘size <= 10’ ТО "insertionSort(A, n)"
    ИНАЧЕ ЕСЛИ ‘size > 10’ ТО "quickSort(A, n)"
    КОНЕЦ ЕСЛИ
КОНЕЦ ЕСЛИ
```

По аналогии с модифицированной САА (САА-М) в [2] была предложена модифицированная АГС, ориентированная на последовательную и параллельную генерацию последовательных и параллельных РС (ПРС) в САА-М. В сигнатуру модифицированной АГС, в частности, входит операция асинхронной дизъюнкции операторов *A* // *B*, состоящая в параллельном выполнении операторов *A* и *B*. Для генерации асинхронной дизъюнкции используется операция ее последовательного порождения (//)(*A*, *B*) или асинхронного порождения (///)(*A*, *B*). Для синхронизации процессов, инициируемых асинхронными дизъюнкциями, используются контрольные точки и синхронизаторы. С каждой контрольной точкой $T(\alpha)$ связано условие α, имеющее значение “ложь” до тех пор, пока вычислительный процесс не достиг точки $T(\alpha)$, и принимающее значение “истина” с момента достижения данной точки. Синхронизатор $S(\alpha)$ осуществляет задержку вычислительного процесса при значении α “ложь” до тех пор, пока α не получит значение “истина”. Применение данных конструкций рассматривается далее в разделе 2.

2. Управление выводом в грамматиках структурного проектирования

В данном разделе рассматривается применение аппарата алгебры гиперсхем для представления алгоритмов управления выводом в ГСП; подход проиллюстрирован на примере генерации параллельного алгоритма из области линейной алгебры.

Под ГСП [1] понимается объект $G = (T, N, \alpha, P, D)$, где $T = \Sigma \cup S$ – терминальный алфавит; Σ – совокупность базисных условий, операторов, объектов, абстрактные типы данных (АТД), абстрактные типы памяти (АТП), определяющие степень конкретизации проектируемого класса программ C; S – совокупность разделителей – символов операций сигнатуры САА-М, скобок, ограничителей и др.; N – нетерминальный алфавит, к которому принадлежат метапеременные: логические, операторные, объектные, АТД, АТП, характеризующие степень абстракции в процессе проектирования программ; $\alpha \in N$ – аксиома грамматики, идентифицирующая проектируемый класс C; $P = \{u_i \to v_i | \, i = 1, 2, \ldots, k\}$ – совокупность подстановок логического, операторного, объектного типов, которые детализуют АТД и АТП и применяются при проектировании алгоритмов и программ класса C; D – выбранный механизм управления выводом в ГСП G. Отмеченный механизм является алгоритмом, в котором задан порядок применения продукций в процессе вывода (порождения) схем. Этот алгоритм может быть представлен в алгебре гиперсхем. Особенный интерес представляют матричные ГСП с последовательным, параллельным или комбинированным применением подстановок в обобщенных матричных продукциях. При этом подстановки, применяемые параллельно, записываются в столбец.

Пример 3. Проиллюстрируем процесс взаимосвязанного проектирования структур управления и данных на примере параллельного алгоритма умножения матриц. Пусть даны две прямоугольные матрицы, $A = (a_{lj})_{M \times N}$ и $B = (b_{lj})_{N \times Q}$. Элементы результирующей матрицы $C = (c_{lj})_{M \times Q} = A \times B$ определяются по формуле

$$c_{lj} = \sum_{k=0}^{N-1} a_{lk} \cdot b_{kj}, l = 0,...,M-1, j = 0,...,Q-1,$$

в которой нумерация элементов матриц, а также строк и столбцов ведётся с нуля.

В рассматриваемом далее алгоритме используется метод распараллеливания, предложенный в [10]. Элементы результирующей матрицы нумеруются по правилу

$$n_{lj} = l \cdot N + j.$$

Вычисления распределяются между K процессорами так, чтобы первый процессор вычислял первые $\frac{M \cdot Q}{K}$ элементов результирующей матрицы, второй – следующие $\frac{M \cdot Q}{K}$ и т.д. Таким образом, исходная матрица A и конечная матрица C разделяются на горизонтальные полосы (блоки), показанные на рис. 1. В i-й параллельной ветви блок A_i матрицы A умножается на матрицу B, в результате чего получаем блок C_i матрицы C. При этом нумерация ветвей также ведется с 0; i = 0, …, K–1.

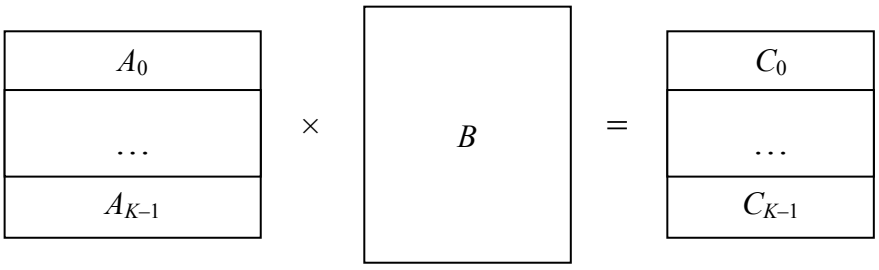

Рис. 1. Разбиение матриц на блоки при параллельном умножении матриц K ветвями

Общая регулярная схема параллельного алгоритма имеет вид:

УмножениеМатриц-П(K) =
= СТАРТ(K) ∗
∗ (Ветвь (A_0, B) // Ветвь (A_2, B) // … // Ветвь (A_{K-1}, B)) ∗
∗ S(ОБР_ЗАК) ∗ ФИН,

где СТАРТ(K) – оператор инициализации матриц и подготовки к запуску K параллельных ветвей; Ветвь (A_i, B) – ветвь, выполняющая умножение i-го блока матрицы A на матрицу B; S(ОБР_ЗАК) – оператор синхронизации, выполняющий ожидание завершения вычислений во всех ветвях; ФИН – заключительный оператор (вывод результирующей матрицы C).

САА-схема, детализирующая составной оператор Ветвь (A_i, B), имеет следующий вид:

Ветвь (A_i, B) =
  (*start* := M / K * i) *
* (*end* := M / K * (i + 1)) *
* ДЛЯ (l) от (*start*) до (*end*–1) ЦИКЛ
        ДЛЯ (j) от (0) до (Q–1) ЦИКЛ
            (*value* := A[l][0] * B[0][j]) *
            ДЛЯ (k) от (1) до (N–1) ЦИКЛ
                (*value* := *value* + A[l][k] * B[k][j])
            КОНЕЦ ЦИКЛА *
            * (C[l][j] := *value*)
        КОНЕЦ ЦИКЛА
  КОНЕЦ ЦИКЛА *
* T(ОБР_ЗАК),

где T(ОБР_ЗАК) – контрольная точка, фиксирующая момент завершения вычислений в i-й ветви.

Рассмотрим далее матричную ГСП $G_1 = (T_1, N_1, \alpha_1, P_1, D_1)$, предназначенную для проектирования класса ПРС УмножениеМатриц-П(K). ГСП порождает алгоритмы из указанного класса с различным количеством параллельных ветвей K в зависимости от количества ресурсов, находящихся в наличии. В приведенной далее схеме грамматики G_1 используется параметрическая запись продукций, обеспечивающая наращивание параллельных ветвей и связанных с ними структур данных A_i путем изменения параметра i от 0 до K–1. Система подстановок P_1 ГСП G_1 имеет следующий вид:

$$m_0: \left\| \ \alpha_1 \to \text{СТАРТ}(0) * \text{ПРС1} * S(\text{ОБР\_ЗАК}) * \ \text{ФИН} \right\|;$$

$$m_1: \left\| \ \text{ПРС1} \to \text{Ветвь}(A_0, B) \ // \ \text{ПРС1} \right\|;$$

$$m_2: \left\| \begin{array}{l} \text{Ветвь}(A_i, B) \ // \ \text{ПРС1} \to \text{Ветвь}(A_i, B) \ // \ \text{Ветвь}(A_{i+1}, B) \ // \ \text{ПРС1} \\ \text{СТАРТ}(i) \to \text{СТАРТ}(i+1) \end{array} \right\|, \text{ при } i = 0, 1, \ldots, K–3;$$

$$m_3: \left\| \begin{array}{l} \text{Ветвь}(A_i, B) \ // \ \text{ПРС1} \to \text{Ветвь}(A_i, B) \ // \ \text{Ветвь}(A_{i+1}, B) \\ \text{СТАРТ}(i) \to \text{СТАРТ}(i+2) \end{array} \right\|, \text{ при } i = K–2.$$

Здесь ПРС1, СТАРТ(i) $\in N_1$ — операторные метапеременные; A_i – объектная метапеременная.

В процессе вывода регулярной схемы по грамматике G_1 матрицей подстановок m_1 формируется ветвь с номером 0. Затем матрицей m_2 рекурсивно формируются следующие ветви ПРС (при i = 0, 1, …, K–3). Процесс завершается применением матрицы m_3 (при $i = K$–2).

Рассмотрим далее процесс построения гиперсхем, представляющих алгоритмы управления выводом в бесконтекстных ГСП, в соответствии с методом, приведенным в [2]. Пусть на множестве $P = M \times L$, где M – информационное множество, ассоциируемое с параметрами управления выводом в ГСП $G = (T, N, \alpha, P, D)$, задана АГС $\sigma = <U, V>$. Продукции p_i: $v_i \to w_i$ грамматики G с одинаковыми левыми частями записываются в виде множества P' уравнений f_i:

$$P' = \{ f_i: v_i = w_{i1} \ ; w_{i2} \ ; \ \ldots \ ; w_{it_i} | \ i = 1, 2, \ldots, n \},$$

где t_i – количество альтернативных продукций в i-м уравнении; v_i – операторные и логические метапеременные (нетерминалы); w_{ij} (j = 1, 2, …, t_i) – правые части продукций, представляющие собой операторы и условия, в которые могут входить как терминальные, так и нетерминальные символы. Упомянутым метапеременным ставятся в соответствие одноименные составные операторы и условия АГС. Базисным операторам и условиям САА, входящим в правые части продукций грамматики G, также сопоставляются одноименные базисные элементы АГС σ. Применение базисного оператора или условия АГС, принявшего значение η, к состоянию $p \in P$, приводит к записи на ленту их текста без изменений. Знаки операций синхронной и асинхронной дизъюнкции (если такие операции имеются), входящие в выражения w_{ij}, заменяются знаками операций асинхронного или последовательного порождения синхронной и соответственно асинхронной дизъюнкции. Нетерминалам v_i, которые входят в уравнения f_i, содержащие несколько альтернативных продукций, ставится в соответствие составной оператор, представляющий собой композицию некоторого оператора $O_i \in U$ и операции выбора (см. раздел 1). В рассматриваемом далее примере в качестве оператора O_i был использован оператор приращения значения параметра i на единицу, – ИНК(i). В операцию выбора в качестве аргументов входят условия α_{ij}, такие, что $p \in P$: $\alpha_{ij}(p) \neq \eta$ (j = 1, 2, …, t_i). Оператор O_i в зависимости от текущего состояния

p устанавливает значение “истина” одного и значения “ложь” остальных условий α_{ij}. При выполнении оператора O_i запись текста на ленту не производится. Операторами упомянутой конструкции выбора являются выражения, полученные из w_{ij}, входящих в уравнение f_i. Грамматике G ставится в соответствие АГС-схема, детализующая оператор v_1:

$$v_1 = O_1 * \text{ВЫБОР}([\alpha_{11}] \to w_{11}, [\alpha_{12}] \to w_{12}, \ldots, [\alpha_{1\,t_1}] \to w_{1t_1});$$

$$v_2 = O_2 * \text{ВЫБОР}([\alpha_{12}] \to w_{12}, [\alpha_{22}] \to w_{22}, \ldots, [\alpha_{2\,t_2}] \to w_{2t_2});$$

$$\ldots$$

$$v_n = O_n * \text{ВЫБОР}([\alpha_{1n}] \to w_{1n}, [\alpha_{2n}] \to w_{2n}, \ldots, [\alpha_{n\,t_n}] \to w_{nt_n}).$$

Применив АГС-схему v_1 к некоторому состоянию $p \in P$, получим на ленте ПРС $F(v_1, p)$. При этом $F(v_1, p) \in L(G)$, где $L(G)$ – язык, порождаемый грамматикой G.

АГС-схемы могут применяться для представления алгоритмов управления выводом не только в бесконтекстных грамматиках. При этом проверка правых и левых контекстных условий может осуществляться посредством операций левого и соответственно правого умножения оператора на условие. В примере, приведенном далее, такая проверка не производится, а выбор подстановки выполняется в зависимости от текущего значения параметра i.

Пример 4. Рассмотрим гиперсхему *MatrixMult*, представляющую алгоритм управления выводом в ГСП G_1 (см. пример 3):

$$MatrixMult = (i := 0) * (K := 4) * \text{СТАРТ}(K) * \text{ПРС1} * S(\text{ОБР\_ЗАК}) * \text{ФИН},$$

$$\text{ПРС1} = \text{ВЫБОР}([(i >= 0) \wedge (i < K–1)] \to$$

$$(//)(\text{Ветвь}\ (A_i, B), \text{ПРС1}),$$

$$[i = K–1] \to \text{Ветвь}\ (A_i, B)) *$$

- ИНК(i).

Для генерации операции асинхронной дизъюнкции в гиперсхеме использована операция ее последовательного порождения (см. раздел 1). Аксиоме α_1 в гиперсхеме поставлен в соответствие составной оператор *MatrixMult*, метапеременной ПРС1 – одноименный составной оператор. Полагаем, что операторы СТАРТ(K), Ветвь (A_i, B), ФИН являются тождественными на информационном множестве P, а их выполнение состоит в сохранении на выходной ленте текста одноименного оператора с подставленным текущим значением параметров i и K.

В алгоритме *MatrixMult* вначале осуществляется инициализация параметров i и K. Параметру K, соответствующему количеству параллельных ветвей, в гиперсхеме присвоено значение 4. Далее генерируется текст оператора СТАРТ(4). Затем применяется составной оператор ПРС1, в котором, с помощью изменения параметра i от 0 до K–1 и операции выбора рекурсивно формируются ветви Ветвь (A_i, B). Генерация ПРС завершается записью на ленту текста базисного оператора ФИН.

Применив гиперсхему *MatrixMult* к состоянию $p \in P$ при значении параметра $K = 4$, получим на ленте ПРС УмножениеМатриц-П(4):

$$\text{УмножениеМатриц-П}(4) = \text{СТАРТ}(4) *$$

$$* (\text{Ветвь}\ (A_0, B) // \text{Ветвь}\ (A_1, B) // \text{Ветвь}\ (A_2, B) // \text{Ветвь}\ (A_3, B)) *$$

$$* S(\text{ОБР\_ЗАК}) * \text{ФИН}.$$

Для проверки эффективности рассмотренного параллельного алгоритма умножения матриц была разработана соответствующая многопоточная программа на языке C++. Программа выполнена на многоядерной архитектуре Intel Core 2 Quad CPU, 2.51 ГГц; ОС Windows XP. Время ее выполнения показано на рис. 2. Ускорение при выполении на 2, 3 и 4 процессорах составило 2; 2.9 и 3.9 соответственно.

Отметим, что в работах [1–3] рассмотренный аппарат ГСП и гиперсхем был использован также для генерации асинхронных схем сортировки. Автоматизировать построение гиперсхем, а также выполнять генерацию алгоритмов и программ позволяет инструментарий, рассматриваемый далее в разделе 3.

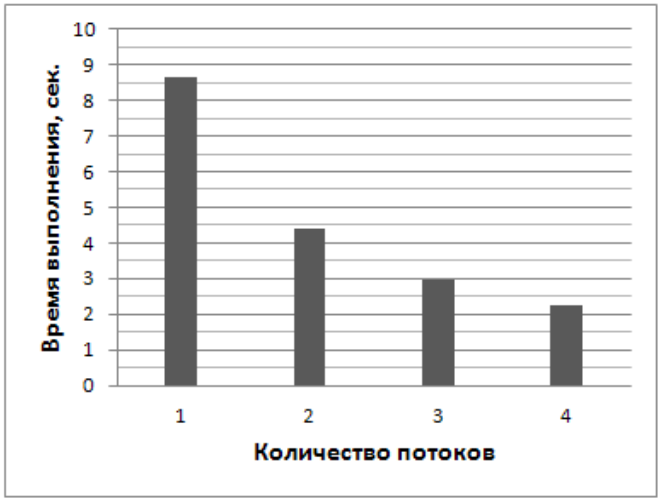

Рис. 2. Время выполнения параллельной программы умножения матриц на 4-ядерном процессоре; размер входных матриц 1000 × 1000 элементов

3. Инструментарий генерации алгоритмов по гиперсхемам

Рассмотренный подход к генерации схем программ был реализован в инструментарии, позволяющем интерпретировать базисные операторы и условия гиперсхем. Упомянутый инструментарий является одним из основных компонентов системы проектирования и синтеза программ [1, 3]. Прототипом данного инструментария послужила система МУЛЬТИПРОЦЕССИСТ, в которой выполнялся синтез схем по АГС-схемам, а также генерация произвольных текстов по Т-схемам [11]. В отличие от упомянутой системы, в разрабатываемом инструментарии осуществляется проектирование АГС и САА-схем в диалоговом режиме, который обеспечивает их синтаксическую правильность. Построение схем выполняется с использованием диалогового конструктора параллельных объектно-ориентированных программ (ДСП-конструктора) [1]. Основная идея последнего состоит в поуровневом конструировании схем программ сверху вниз путем детализации конструкций языка и показано в виде дерева (рис. 3).

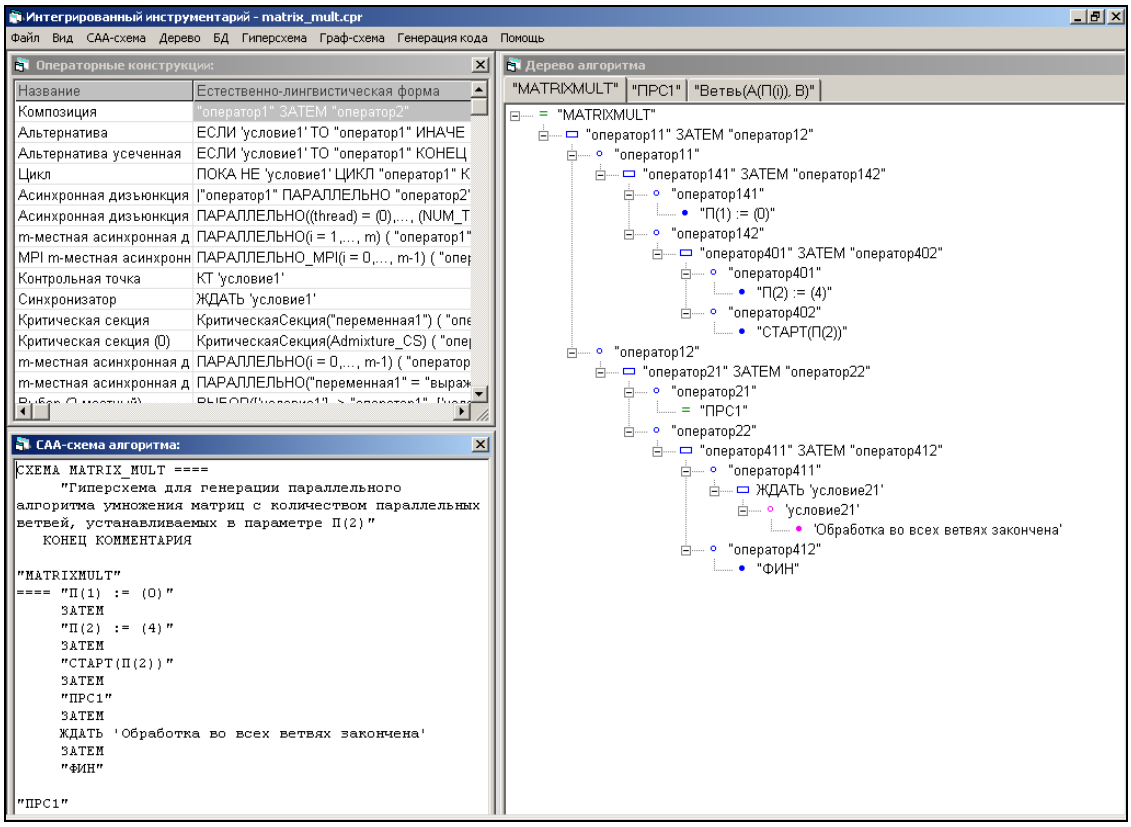

Рис. 3. ДСП-конструктор с построенной гиперсхемой умножения матриц

На каждом шаге построения схемы система предоставляет пользователю на выбор только те конструкции языка, подстановка которых в формируемый текст не нарушает синтаксическую правильность САА или АГС-схемы. По сконструированной таким образом гиперсхеме может быть выполнена генерация САА-схемы, а по САА-схеме – синтез программы на выбранном целевом языке программирования (рис. 4). САА-схемы и гиперсхемы имеют одинаковый синтаксис, что позволяет гибко использовать параметры управления генерацией схем, задаваемые на уровне базисных операторов и условий.

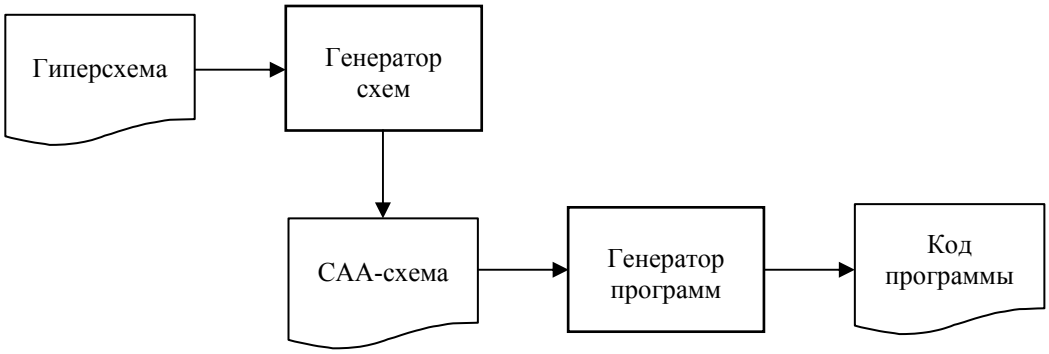

Рис. 4. Последовательность генерации алгоритмов и программ в интегрированном инструментарии

Рассмотрим процесс генерации ПРС по АГС-схемам, оформленным в языке САА/1 [1]. Перед началом генерации пользователь имеет возможность указать опции: название генерируемой схемы, комментарий к ней, имя файла схемы. АГС-схема выполняется интерпретатором во взаимодействии с синтаксическим анализатором, распознающим подлежащие выполнению конструкции. В процессе интерпретации производится обход дерева синтаксического анализа гиперсхемы, полученного на этапе ее ДСП-конструирования.

В ходе такого выполнения формируется текст порождаемой ПРС. Языковые конструкции обрабатываются рекурсивно в соответствии с определением функции F (см. раздел 1). При обработке составных операторов гиперсхемы выполняется загрузка соответствующего поддерева синтаксического анализа. Текст базисных операторов и условий АГС-схем оформляется с помощью параметризации. При этом для облегчения обработки параметры обозначаются в тексте базисных и других элементов гиперсхемы в виде П(i), где i – номер параметра. Например, базисный оператор присваивания имеет следующий вид: “П(1) := (0)”. Выполнение элементарного оператора (вычисление элементарного условия) на этапе генерации ПРС состоит в обработке его текста ПРС-генератором, который в контексте значений параметров, ассоциированных с информационным множеством M, строит текст элементарного оператора или условия порождаемой ПРС. Если при этом текст условия преобразуется в константу 0, μ или 1, считается, что данное условие принимает соответствующее значение на этапе ПРС-генерации. В противном случае считается, что условие принимает значение η, а построенный ПРС-генератором текст является текстом соответствующего условия порождаемой ПРС. Имена параметров и их значения в процессе обработки заносятся в таблицу. Параметры, входящие в различные элементы гиперсхемы (составные операторы, и др.) заменяются соответствующими значениями.

Таким образом, описанный инструментарий позволяет автоматизировать вывод ПРС по описанию информационно-алгоритмической модели этой области посредством ГСП и АГС-схем.

Заключение

Предложен подход к разработке последовательных и параллельных алгоритмов на основе использования средств параметрически управляемой генерации схем программ. Инструментальные средства базируются на абстрактно-автоматной модели процесса генерации регулярных схем, ассоциированных с ней алгебрах гиперсхем и грамматиках структурного проектирования. Инструментарий ориентирован на построение САА и АГС-схем в режиме поуровневого диалогового конструирования, обеспечивающем их синтаксическую правильность.

Отметим сходство АГС с концепцией шаблонов проектирования [12]. Гиперсхемы являются параметризованными повторно используемыми компонентами для решения определенного класса задач. Указание конкретных значений параметров в гиперсхеме и последующая интерпретация гиперсхем позволяет получить схемы алгоритмов, адаптированные к конкретным условиям использования.

1. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев: Академпериодика, 2007. – 631 с.
2. *Ющенко Е.Л., Цейтлин Г.Е., Галушка А.В.* Алгебро-грамматические спецификации и синтез структурированных схем программ // Кибернетика. – 1989. – № 6. – С. 5–16.

3. *Яценко Е.А.* Алгебры гиперсхем и интегрированный инструментарий синтеза программ в современных объектно-ориентированных средах // Кибернетика и системный анализ. – 2004. – № 1. – С. 47–52.
4. *Дорошенко А.Е., Шевченко Р.С.* Применение систем переписывания термов к анализу исходного программного кода. // Проблеми програмування. – 2008. – № 2–3. – С. 305–312.
5. *Дорошенко А.Е., Жереб К.А.* Техника и инструментарий переписывающих правил для инженерии программного обеспечения графических ускорителей // Инженерия программного обеспечения. – 2010. – № 4. – С. 35–49.
6. *Ершов А.П.* О сущности трансляции // Программирование. – 1977. – № 5. – С. 21–39.
7. *Касьянов В.Н.* Оптимизирующие преобразования программ. – М.: Наука, 1988. – 335 с.
8. *Жоголев Е.А., Соболева В.В.* Универсальная оболочка систем гиперпрограммирования // Программирование. – 1999. – № 5. – С. 62–70.
9. *Яценко Е.А.* О применении машинного обучения для проектирования адаптивных программ сортировки в алгебре алгоритмов // Проблеми програмування. – 2011. – № 2. – С. 23–33.
10. *Миронов А.А., Карпов А.Н.* Параллельные алгоритмы обработки данных. – http://www.viva64.com/ru/a/0032/#ID0EY5JO.
11. *Многоуровневое* структурное проектирование программ: Теоретические основы, инструментарий / Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К. – М.: Финансы и статистика, 1989. – 208 с.
12. *Mattson T.G., Sanders B.A., Massingill B.L.* Patterns for Parallel Programming. – Addison-Wesley Professional, 2004. – 384 p.