УДК 680.3

Д.А. Толстолужский[1], Е.Г. Толстолужская[2], Г.А. Поляков[3]

[1] *Харьковский национальный университет им. В.Н. Каразина, Харьков*
[2] *Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков*
[3] *Академия наук прикладной радиоэлектроники, Россия*

# МЕТОД СИНТЕЗА СИ-ПРОГРАММ СМЕСИ АЛГОРИТМОВ

*В статье рассматривается метод формального синтеза Си-программ смесей исходных алгоритмов и ациклической Си-программы – смеси.*

***Си-программа, граф программы, смесь алгоритмов, коэффициент загрузки оборудования***

## Введение

**Актуальность исследования.** Одним из путей повышения эффективности параллельных процессоров и многопроцессорных ЭВМ является увеличение коэффициента загрузки оборудования, определяющего степень отличия фактической производительности параллельных ЭВМ различных классов и конфигураций от ее максимально возможного значения.

Место и роль этого направления следующим образом определяются в работе [1]: «В более совершенных многопроцессорных системах можно обеспечить высокую степень загруженности с помощью «мультипрограммирования» нового типа, при котором в систему вводятся новые программы, если данная программа не может полностью загрузить все процессоры. За счет того может быть достигнута полная занятость».

Метод смеси алгоритмов ориентирован на повышение эффективности параллельных процессоров и многопроцессорных ЭВМ за счет обеспечения максимальной (100%-й) или требуемой загрузки оборудования (при использовании любой комбинации других методов параллелизма). Возможность достижения 100%-й загрузки поддерживается путем одновременной реализации подмножеств операций/функций, относящихся к различным алгоритмам смеси совместно выполняемых задач.

**Цель статьи.** Статья посвящена решению задачи формального синтеза исходных текстов Си-программ для смесей алгоритмов и задачи синтеза эквивалентных ациклических Си-программ смеси алгоритмов.

**Постановка задачи исследования.** Исходными предпосылками при разработке метода синтеза Си-программ смеси алгоритмов являются следующие положения:

- распараллеливание на уровне операций/функций является более общим и более эффективным, чем общепринятый вариант распараллеливания циклов, поскольку включает распараллеливание на уровне циклов и распараллеливание скалярных вычислений [1, 2];
- статическое распараллеливание позволяет получать более эффективные модели параллельных процессов (по сравнению с динамическим планированием) за счет «вынесения» из времени решения задач временных затрат на распараллеливание, а также за счет получения более качественных (чем при динамическом планировании) временных планов решения задач [2].

**Исходные данные:** множество $P = \{P^v\}$, $v = 1...kz$, Си-программ алгоритмов $P^v = \{P_j^v\}$, включаемых в состав формируемой смеси из kz алгоритмов (принимается, что допустимы любые типы алгоритмов: неразветвляющиеся, разветвляющиеся, циклические простые и сложные с различными типами циклов и их произвольные сочетания).

**Требуется:**

- синтезировать текст «базовой» Си-программы смеси, которая является текстовой спецификацией смеси исходных Си-программ алгоритмов;
- синтезировать текст ациклической Си-программы, логически эквивалентной базовой Си-программе смеси алгоритмов.

## Результаты исследования

Обобщенный алгоритм автоматического синтеза текста Си-программ для множества одновременно выполняемых алгоритмов (смеси алгоритмов), и синтеза соответствующих параллельных временных моделей смеси алгоритмов представлен на рис. 1.

Рассмотрим состав и назначение основных этапов обобщенного алгоритма.

**Этап 1** (символ 2, рис. 1)**.** Целью этапа является автоматическое формирование для заданного множества $P = \{P^v\}$, $v = 1...kz$, алгоритмов $P^v$ «базовой» Си-программы, являющейся текстовой спецификацией смеси исходных алгоритмов.

Решение этой задачи включает выполнение следующих действий:

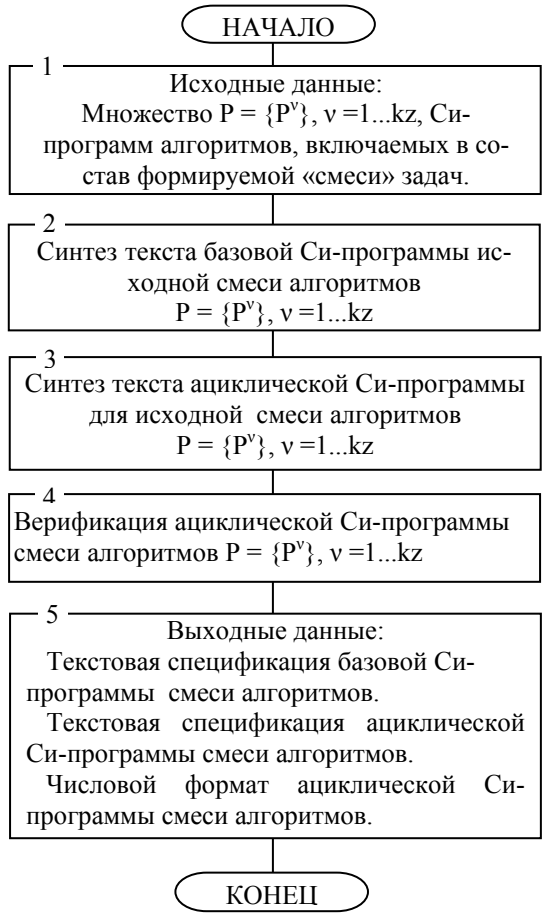


Рис. 1.Обобщенный алгоритм синтеза Си-программ смеси алгоритмов

- анализ идентификаторов переменных в исходных Си-программах и исключение «пересечения» программ по именам данных путем модификации идентификаторов;
- синтез декларативной части формируемой общей Си-программы смеси исходных алгоритмов путем объединения деклараций программ, входящих в состав смеси;
- синтез текста операторной части общей Си-программы смеси путем объединения «тел» Си-программ, входящих в состав смеси, с учетом выполненной модификации идентификаторов.

Используем конкретный пример для иллюстрации содержания основных этапов обобщенного алгоритма синтеза Си-программ смеси алгоритмов. Исходные тексты алгоритмов представляют рис. 2, 3. Результат синтеза базовой Си-программы смеси алгоритмов представлен на рис. 4.

Синтез базовой Си-программы смеси алгоритмов включает следующие шаги:

- формирование общей декларации данных смеси (отметим, что исходные программы не имеют совпадающих идентификаторов переменных и, в связи с этим, переименование данных отсутствует);
- объединенное множество операторов «scanf (…)» ввода входных данных;

```
#include <stdio.h>
void main(void)
{
int i;
int x[2],y[2],z[2];
      for(i=0;i<2;i++)
      scanf("%d %d\n",&x[i],&y[i]);
      for(i=0;i<2;i++)
      {
          z[i] = x[i] + y[i];
          printf("%4d\n",z[i] );
      }
}
```

Рис. 2. Си-программа циклического алгоритма

```
#include <stdio.h>
void main(void)
{
int a,b,c,r;
int k,l,p;
int s;
  scanf("%d %d %d\n",&a,&b,&c);
  k = a * b;
  l = b % a;
          if (k < a-c)
          {
                    r = l * 2;
                    p = k + l;
          }
          else
          {
                    p = 2 * l;
                    r = l - k;
          }
          s = p - r;
          printf("%4d\n",s);
}
```

Рис. 3. Си-программа разветвляющегося алгоритма

- циклическую операторную часть первого алгоритма и разветвляющуюся операторную часть второго алгоритма;
- объединенное множество операторов вывода результатов «printf (… )».

**Этап 2** (символ 3, рис. 1). Целью этапа является синтез «ациклической» Си-программы смеси, исходя из полученной при выполнении первого этапа смеси исходных Си-программ алгоритмов. Решение задачи обеспечивается путем «развертывания» циклической части Си-программы смеси алгоритмов.

Результат синтеза ациклической Си-программы смеси алгоритмов представлен на рис. 6. Следует отметить некоторые особенности синтезированной ациклической программы:

- исключение из декларации идентификаторов массивов и введение идентификаторов элементов массивов в виде простых переменных (x_0,y_0,…);

```
#include <stdio.h>
void main(void)
{
int  i,a,b,c,k,l,p,r, s;
int  x[2],y[2],z[2];
  scanf("%d %d %d\n",&a,&b,&c);
  for(i=0;i<2;i++)
        scanf("%d %d\n",&x[i],&y[i]);
  for(i=0;i<2;i++)
  {
        z[i] = x[i] + y[i];
        printf("%4d\n",z[i]);
  }
  k = a * b;
  l = b % a;
  if (k < a-c)
  {
        r = l * 2;
        p = k + l;
  }
  else
  {
        p = 2 * l;
        r = l - k;
  }
  s = p - r;
  printf("%4d %4d\n",s);
}
```

Рис. 4 . Базовая Си-программа смеси алгоритмов

- замена операторов ввода массивов исходных данных «for(i=0;i<2;i++) scanf("%d%d\n", &x[i],&y[i]);» логически эквивалентным множеством операторов ввода значений простых переменных «scanf("%d",&a); …, scanf("%d",&y_1);»;
- замена операторов вычисления значений элементов массива «for(i=0;i<2;i++) {z[i]=x[i]+y[i]; printf("%4d\n",z[i]);}» эквивалентным множеством операторов присваивания и вывода значений:

«z_0 = x_0 + y_0; printf(" %3d ",z_0);
z_1 = x_1 + y_1; printf(" %3d ",z_1);».

Отметим, что операторная часть ациклической программы смеси, соответствующая разветвляющемуся алгоритму «смеси», остается без изменений. Анализ синтезированного текста ациклической Си-программы смеси с помощью компилятора языка Си подтверждает ее корректность.

Графическая спецификация синтезированной ациклической Си-программы смеси в виде соответствующего Си-графа представляет рис. 6.

**Этап 3** (символ 4, рис. 1). Исходными данными для третьего этапа является текст ациклической Си-программы смеси алгоритмов, полученный на предшествующем этапе. Задачей третьего этапа является верификация ациклической Си-программы смеси алгоритмов [3].

Результаты верификации представлены на рис. 7.

```
#include <stdio.h>
void main(void)
{
int x_0,y_0,x_1,y_1,z_0,z_1,a,b,c;
int k,l,p,r,s;
  scanf("%d",&a);      scanf("%d",&b);
  scanf("%d",&c);      scanf("%d",&x_0);
  scanf("%d",&y_0);    scanf("%d",&x_1);
  scanf("%d",&y_1);
  z_0 = x_0 + y_0;     printf(" %3d ",z_0);
  z_1 = x_1 + y_1;     printf(" %3d ",z_1);
  k = a * b;
  l = b % a;
  if (k < (a - c))
  {
        r = l * 2;
        p = k + l;
  }
  else
  {
        p = 2 * l;
        r = l - k;
  }
  s = p - r;
  printf(" %3d ",s);
}
```

Рис. 5. Ациклическая Си-программа смеси алгоритмов

Они подтверждают логическую эквивалентность текстовых спецификаций исходной Си-программы смеси алгоритмов и ациклической Си-программы смеси.

Результаты сравнения ациклической и базовой Си-программ смеси (рис. 5 и рис. 4 соответственно) позволяют отметить их следующие принципиальные отличия.

**1**. Рассматриваемые Си-программы относятся к двум различным классам программ – ациклических («развернутых») и циклических («свернутых») соответственно. Особенностями структуры ациклической Си-программы смеси по сравнению с базовой Си-программой смеси являются:

- изменение описательной части путем перехода от переменных-массивов к простым переменным, представляющим соответствующие элементы массивов;
- «размножение» операторов присваивания, отражающее изменение типов данных;
- исключение из базовой Си-программы операторов типов «<», «*upl*», «++», «*l.o*», «*a.o*», «*&*», «*», «*bp*» «*bpv*», связанных с организацией циклов;
- наличие в ациклической Си-программе только операторов (типа «+», «-», «*», «%») вычисления значений простых переменных, операторов присваивания (типа «=») значений переменным и операторов «<», «*upl*» управления разветвлением параллельного процесса.

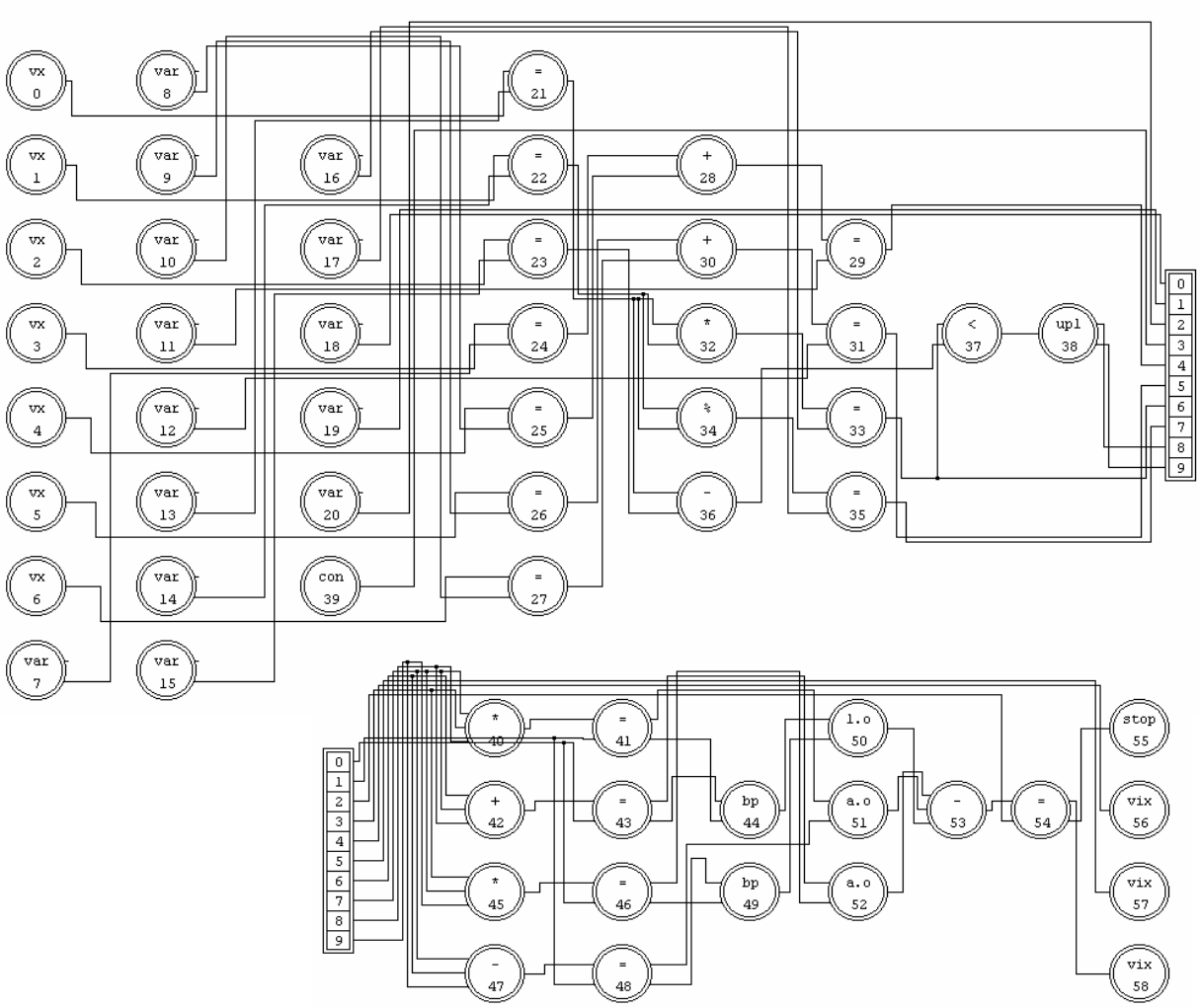


Рис. 6. Си-граф ациклической Си-программы смеси алгоритмов

```
ТЕСТ КОРРЕКТНОСТИ ФАЙЛОВ:
максимальное количество элементов :  0 - 58
максимальное количество связей :     0 - 73
ТЕСТ СООТВЕТСТВИЯ ЧИСЛА СОПРЯЖЕННЫХ И ВНЕШНИХ СВЯЗЕЙ: OK

ТЕСТ ЧИСЛА СВЯЗЕЙ ПО СОПРЯЖЕННЫМ ЭЛЕМЕНТАМ:  OK

ТЕСТ ЧИСЛА СВЯЗЕЙ ПО ВНЕШНИМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ВЫВОДОВ ПО СОПРЯЖЕННЫМ ЭЛЕМЕНТАМ: OK

ТЕСТ СООТВЕТСТВИЯ ВЫВОДОВ ПО ВНЕШНИМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ЧИСЛА ВХОДОВ ЭЛЕМЕНТА И КОЛИЧЕСТВА ЕГО СОПРЯЖЕННЫХ: OK
```

Рис. 7. Результаты верификации
ациклической Си-программы смеси алгоритмов

**2.** Ациклическая Си-программа смеси алгоритмов имеет меньшую (по сравнению с базовой Си-программой) общую сложность (суммарное количество операторов различного назначения) и меньшую структурную сложность (определяемую количеством операторов условной и безусловной передач управления): общее количество «операторов/операций» в базовой Си-программе и ее временной модели $N_Б = 74$, в ациклической $N_{AC} = 59$, сокращение общего числа операторов $\Delta(N) = N_Б/N_{AC} = 1{,}25$ раз; количество операторов управления – соответственно $C_Б = 22$ и $C_{AC} = 6$, сокращение числа операторов управления $\Delta(C) = C_Б / C_{AC} = 3{,}66$ раза.

Причиной такого снижения является исключение из процесса решения задачи операторов, связанных с организацией циклов в исходной Си-программе, реализуемых при каждом выполнении тела цикла (к ним относятся операторы типов «<», «*upl*», «++», «*l.o*», «*a.o*», «&», «*», «*bp*», «*bpv*»).

**3.** Исключение из процесса решения задачи операторов, связанных с организацией циклов в базовой Си-программе смеси, обусловливает различие в структурной сложности (на уровне естественных частей) базовой и ациклической Си-программ смеси (рис. 8, 9). Как видно, базовая Си-программа смеси включает десять естественных частей, ациклическая – только четыре. Значительное снижение структурной сложности при переходе к ациклической программе смеси создает предпосылки к существенному облегчению процесса отладки ациклической Си – программы смеси алгоритмов и увеличению вероятности правильного выполнения алгоритмов.

**4.** Уменьшение количества выполняемых операторов, обусловленное переходом к ациклической Си-программе смеси алгоритмов, обеспечивает уменьшение емкости программной памяти и создает потенциальную возможность уменьшения времени решения смеси задач.

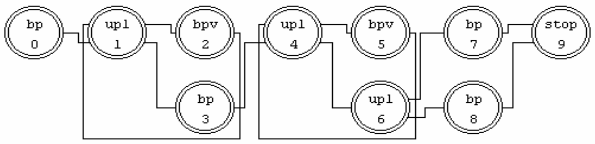


Рис. 8. Си-граф естественных частей ациклической Си-программы смеси

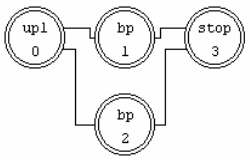


Рис. 9 .Си-граф естественных частей базовой Си-программы смеси

**5**. Параллельная реализация смеси обеспечивает увеличение коэффициента загрузки процессоров: для NM = 1…5 $S_{AC}$= f3(NM) принимает значения (в %): 100, 99,56, 99,56, 87,31, 77,61 (рис. 10) по сравнению со значениями этого коэффициента S(NM) = 100, 98,7, 77, 95, 62,30, 49,84 (рис. 11) при решении системы уравнений с применением только метода совмещения независимых операций.

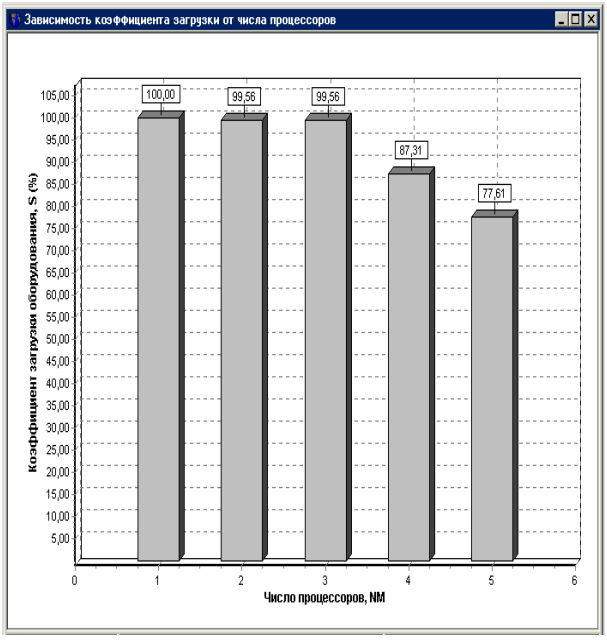


Рис. 10. Коэффициент $S_{AC}$ = f3(NM) загрузки процессоров при выполнении модели смеси (NM =1…5)

## Выводы

1. Переход к одновременному выполнению смеси алгоритмов – один из путей повышения загрузки и, как следствие, повышения эффективности параллельных процессоров и ЭВМ.

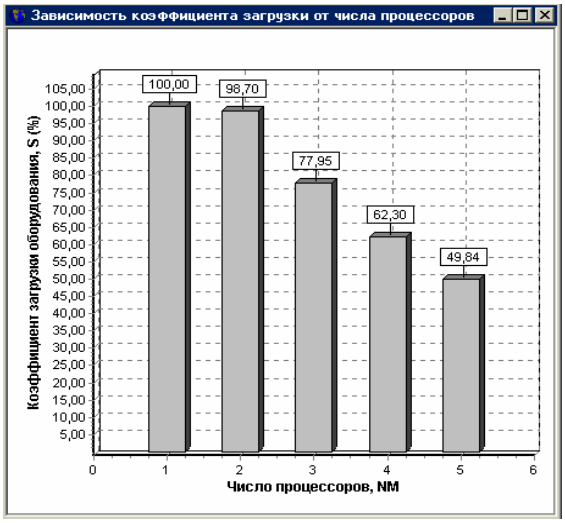


Рис. 11. Коэффициент S = f3(NM) загрузки процессоров

2. Разработанный метод обеспечивает возможность формального синтеза Си-программ смесей исходных алгоритмов и ациклической Си-программы – смеси.

3. Проведенный анализ подтверждает возможность существенного увеличения загрузки и сокращения количества выполняемых операторов программы ациклической смеси по сравнению с исходной смесью Си-программ за счет исключения операторов управления циклами при определенном увеличении емкости памяти программ.

4. Результаты метода обеспечивают поддержку решения ряда практически важных задач (например, автоматический синтез параллельных временных моделей алгоритмов, решение задачи автоматизации синтеза параллельных программ для различных классов существующих и перспективных параллельных процессоров и параллельных ЭВМ).

## Список литературы

*1. Степанов А.Н. Архитектура вычислительных систем и компьютерных сетей. – СПб.: Питер, 2007. – 509 с.*

*2. Поляков Г.А. Глобально-параллельные время – параметризованные программы – новый подход к синтезу структур и выполнению параллельных программ в АСУ реального времени // Тезисы докладов Всесоюзной конференции «Программное обеспечение вычислительных сетей и систем реального времени». – К., 1981. – С. 130-132.*

*3. Поляков Г.А., Толстолужский Д.А. Компиляционная методика верификации статических и динамических объектов автоматического проектирования мультипараллельных цифровых устройств // Прикладная радиоэлектроника. – 2005. – Т. 4, № 1. – С. 161-167.*

*Поступила в редколлегию 11.01.2008*

**Рецензент:** д-р техн. наук, ст. научн. сотр. В.В. Баранник, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.