Модел. и анализ информ. систем. Т. **18**, № **4** (2011) 80–93

УДК 519.6+681.3

Ингибиторная сеть Петри, выполняющая произвольный заданный нормальный алгорифм[1] Маркова

Зайцев Д.А.

Международный гуманитарный университет

e-mail: zsoftua@yahoo.com

получена 22 ноября 2010 года

Ключевые слова: нормальный алгорифм Маркова, ингибиторная сеть Петри, кодирование, шифр

Построена ингибиторная сеть Петри с фиксированной структурой, которая выполняет произвольный заданный нормальный алгорифм Маркова. Алгорифм и его входная цепочка шифруются целыми неотрицательными числами и помещаются в выделенные позиции сети Петри, реализующей применение подстановок алгорифма к цепочке символов. Использованы правила кодирования последовательных, ветвящихся и циклических процессов сетями Петри. По завершении работы сети выходная цепочка восстанавливается (дешифруется) из целочисленной формы представления. В парадигме вычислений на сетях Петри построенная сеть обеспечивает совместимость систем.

1. Введение

Универсальная ингибиторная сеть Петри [1], выполняющая произвольную заданную ингибиторную сеть Петри, является теоретической основой для новой парадигмы вычислений, организованных на компьютерах с процессорами сетей Петри и программами в форме сетей Петри. Традиционная программа – это последовательность команд, и в процессе программирования изначально независимые события упорядочивают искусственно в некоторую последовательность. В дальнейшем возникает обратная задача – распараллеливание, при выполнении программ на многопроцессорных (многоядерных) компьютерах. Описание процессов сетью Петри позволяет сохранить изначальную независимость (асинхронность) событий и осуществлять их синхронизацию лишь в полном соответствии с семантикой предметной области. Таким образом, программа в форме сети Петри сохраняет естественный параллелизм предметной области, что позволяет существенно ускорить вычисления.

[1] Традиционно сохранен оригинальный термин А.А. Маркова [3].

Однако в настоящее время имеются значительные объемы алгоритмов, представленных в классических парадигмах на основе машины Тьюринга [2] и нормальных алгорифмов Маркова [3]. В этой связи целесообразно построение специальных сетей Петри, эмулирующих указанные алгоритмические системы. Технически они могут быть реализованы как соответствующие сопроцессоры к основному процессору сетей Петри, что позволит обеспечить преемственность программ. Продукционная форма записи, принятая в нормальных алгорифмах Маркова, широко применяется в обработке текстовой информации, символьных преобразованиях формул, системах искусственного интеллекта.

В [4] построена сеть Петри, которая выполняет машину Тьюринга. Целью настоящей работы является построение ингибиторной сети Петри, выполняющей произвольный заданный нормальный алгорифм Маркова; использовано шифрование векторов и цепочек символов, а также кодирование операторов и программ [1, 4].

Преимуществом шифрования натуральными числами [1, 4] перед классической техникой доказательств [5] является получение результирующей сети с фиксированной структурой, в которую произвольная заданная сеть Петри (машина Тьюринга, нормальный алгорифм Маркова) вводится в виде маркировки выделенных позиций.

2. Нормальный алгорифм Маркова и сеть Петри

Нормальный алгорифм Маркова (НАМ) [3] представляет собой последовательность подстановок слов. Пусть A – некоторый алфавит; A^* будем обозначать множество всех слов в алфавите A (включая пустое слово). Подстановкой будем называть упорядоченную пару $p = (\alpha, \beta)$, где $\alpha, \beta \in A^*$. Различают обычные и заключительные подстановки; для записи обычной и заключительной подстановок в виде строки используют специальные символы $\rightarrow$, $\mapsto$ соответственно, не входящие в алфавит A: $\alpha \rightarrow \beta$, $\alpha \mapsto \beta$.

НАМ – это пара $M = (A, P)$, где A – конечный алфавит символов и $P = (p_1, p_2, ..., p_N)$ – конечная последовательность подстановок, каждая из которых имеет одну из двух допустимых форм: $\alpha \rightarrow \beta$ либо $\alpha \mapsto \beta$, причем $\alpha, \beta \in A^*$. Подстановка p вида $\alpha \rightarrow \beta$ применима к слову вида $X = \sigma\alpha\omega$, где $\sigma, \omega \in A^*$; в результате применения подстановки получаем слово $X' = \sigma\beta\omega$; процесс обозначим как $X \stackrel{p}{\Rightarrow} X'$.

НАМ преобразует заданное входное слово $X \in A^*$ в выходное слово $Y \in A^*$, $Y = M(X)$ по следующим правилам:

- (a) подстановки выбираются последовательно;
- (b) слово просматривается слева направо;
- (c) если текущая подстановка применима, то выполняется замена самого левого ее вхождения;
- (d) если отсутствуют применимые подстановки либо применена заключительная подстановка, то останов, иначе перейти к (a).

Таким образом, выполнение НАМ представляет собой некоторую последовательность подстановок, применяемых к входному слову и преобразующих его в выходное слово:

$$X \overset{p_{i_1}}{\Rightarrow} X^1 \overset{p_{i_2}}{\Rightarrow} X^2 \overset{p_{i_3}}{\Rightarrow} ... \overset{p_{i_k}}{\Rightarrow} X^k = Y.$$

Использованы определения ингибиторной сети Петри (ИСП), представление ее динамики в виде уравнения состояния, а также обозначения, описанные в [1, 4]. Напомним, что сеть Петри [5, 6] представляет собой двудольный ориентированный граф; одну долю вершин составляют позиции, изображенные в виде окружностей, вторую – переходы, изображенные в виде прямоугольников. Динамические элементы – фишки находятся внутри позиций и перемещаются в сети в результате срабатывания переходов; фишки изображают точками либо числом, равным количеству фишек. Условием возбуждения перехода для обычной дуги служит наличие фишек во входной позиции, ингибиторной дуги – отсутствие фишек. На конце ингибиторной дуги изображают небольшую окружность; заполненный круг служит для обозначения проверочной дуги, являющейся представлением пары встречных дуг.

3. Шифрование НАМ в целых неотрицательных числах

Для выполнения на ИСП, заданный НАМ следует зашифровать (рис. 1) в виде определенного набора скалярных целых неотрицательных величин, которые затем могут быть размещены в позициях сети Петри (рис. 2). Для шифрования слов символы алфавита A занумеруем: $A = \{a_1, ..., a_{|A|}\}$; слово $X = a_{i_1} a_{i_2} ... a_{i_n}$ в алфавите A представим как вектор $\bar{x} = (i_1, i_2, ..., i_n)$ и применим для его шифрования и последующего дешифрования правила и рекурсивные процедуры MulAdd и ModDiv, описанные в [1]; шифр обозначим $S(X)$. Заметим, что для последовательного доступа к элементам с начала вектора шифрование следует выполнить с конца.

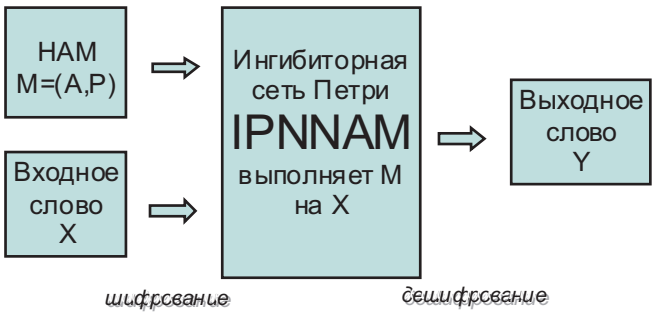

Рис. 1. Концепция выполнения НАМ на ИСП

Обработка входной строки X при выполнении НАМ требует передвижения вдоль строки как вправо, так и влево; для реализации правил НАМ требуется лишь доступ к одному текущему символу (его коду) и обеспечение возможности последующего

перемещения на один символ (вправо либо влево). Правила доступа похожи на порядок работы с лентой машины Тьюринга, поэтому для представления строки будем использовать шифрование, предложенное для выполнения на сетях Петри машин Тьюринга [2]. Однако для простого распознавания концов строки шифрование символов алфавита A начнем с единицы (с основанием $b = |A| + 1$); тогда нулевой код будет соответствовать пустой строке.

Произвольную просматриваемую строку X будем представлять в форме $X = a_l^L a_{l-1}^L ... a_2^L a_1^L a a_1^R a_2^R ... a_{r-1}^R a_r^R$, где a – текущий просматриваемый символ. Обозначим $X^L = a_1^L a_2^L ... a_{l-1}^L a_l^L$, $X^R = a_1^R a_2^R ... a_{r-1}^R a_r^R$; отметим, что строка X^L составлена в обратном порядке для создания двух стеков по обе стороны от текущего символа. Шифр строки X будем представлять тройкой целых неотрицательных чисел (lX, x, rX), где $lX = S(X^L)$, $x = S(a)$, $rX = S(X^R)$.

Утверждение 1. *Алгоритмы RightShift (LeftShift), представленные в листинге 1, реализуют перемещение вдоль шифра строки на один символ вправо (влево).*

```
void RightShift(luint *l, uint *c, luint *r, uint b)
{
  MulAdd(l,b,(*c));  /* l=l*b+c */
  ModDiv(c,r,b);     /* c=r mod b, r=r div b */
}

void LeftShift(luint *l, uint *c, luint *r, uint b)
{
  MulAdd(r,b,(*c)); /* r=r*b+c */
  ModDiv(c,l,b);    /* c=l mod b, l=l div b */
}
```

Листинг 1. Алгоритмы работы с шифрами строк

Тип uint обозначает целое без знака (с диапазоном, достаточным для хранения шифра одного символа); тип luint представляет длинное целое без знака (с диапазоном, достаточным для хранения шифра строки).

Поскольку правила работы НАМ предполагают последовательный порядок просмотра подстановок (с возвратом к началу), целесообразно представить последовательность подстановок P в виде одной строки; для разделения подстановок используем дополнительный символ, который обозначим как ",". Тогда строка подстановок имеет вид $P =$"$p_1, p_2, ..., p_{n-1}, p_n$" в расширенном алфавите $A \cup \{, , \to, \mapsto\}$. При нумерации будем размещать дополнительные символы в конце алфавита таким образом, что $S(,) = b - 3$, $S(\to) = b - 2$, $S(\mapsto) = b - 1$. Для единообразия основание расширенного алфавита будет использовано в дальнейшем также для шифрования входной (рабочей) строки.

Таким образом, шифр НАМ представлен семью позициями lP, p, rP, lX, x, rX, b ингибиторной сети Петри, которая выполняет нормальный алгорифм Маркова (ИСПНАМ), изображенными на рис. 2. Ввиду частого использования будем предполагать заготовленными заранее шифры вспомогательных символов $b1 = b - 1$, $b2 = b - 2$, $b3 = b - 3$, размещенные в одноименных позициях.

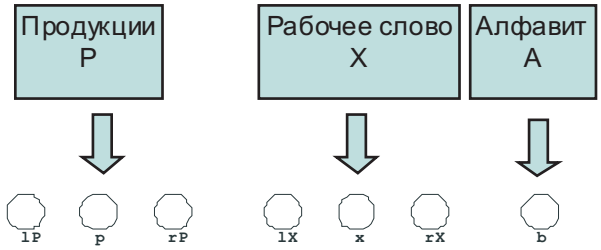

Рис. 2. Размещение шифра НАМ в позициях ИСП

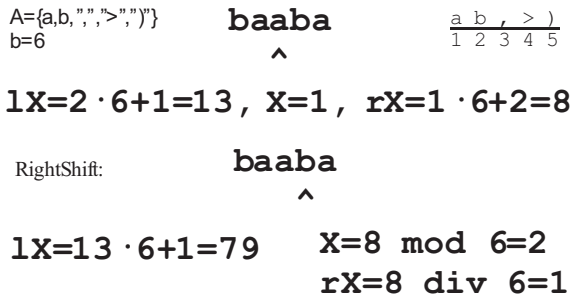

Рис. 3. Пример работы с шифром

Примеры НАМ и их шифров приведены в приложении; ручное выполнение алгорифмов и вычисление шифров позволит изучить особенности шифрования; на рис. 3 пояснен алгоритм преобразования шифра строки при выполнении перемещения вправо (RightShift). Для каждого алгорифма указан алфавит в последовательности шифрования символов; для обозначения символа "→" использован знак ">", а для обозначения символа "↦" – знак ") ". Отметим, что символы {x,y,*} являются вспомогательными и используются только в подстановках.

4. Уточнение правил работы НАМ с учетом шифрования

Для построения ИСПНАМ правила работы НАМ следует уточнить с учетом описанного шифрования. В качестве элементарных операций будем использовать сравнение шифров, присваивание шифра текущего символа и процедуры перемещения вдоль строк RightShift и LeftShift, которые в соответствии с [1, 4], могут быть достаточно просто реализованы соответствующими ИСП.

Правила работы НАМ в посимвольной форме:

1. если текущий символ строки подстановок пуст, завершить;

2. иначе если текущий символ строки подстановок равен b1 или b2, то выполнить подстановку, перейти в начало рабочей строки и строки подстановок, в случае b2 завершить;

3. иначе если текущий символ рабочей строки пуст, перейти в начало рабочей строки, перейти к следующей подстановке, если текущий символ строки подстановок пуст, завершить;

4. иначе если текущий символ строки подстановок не равен текущему символу рабочей строки, вернуться в начало текущей подстановки и в соответствующую позицию рабочей строки, перейти к следующему символу рабочей строки;

5. иначе (если текущий символ строки подстановок равен текущему символу рабочей строки) перейти к следующему символу рабочей строки и следующему символу строки подстановок;

6. повторить с (1).

Правила используют следующие вспомогательные действия (процедуры), которые могут быть описаны с использованием выбранных элементарных операций:

- выполнить подстановку (Substr);
- перейти в начало строки (Rewind);
- перейти к следующей подстановке (NextP);
- вернуться в начало текущей подстановки и в соответствующую позицию рабочей строки (Rewind1).

Лемма 1. *Алгоритм Rewind (листинг 2) реализует перемещение в начало строки.*

Доказательство. Если текущий символ *c позиционирован на крайнем левом символе строки, то (*l)==0 и цикл не выполняется, иначе в цикле выполняется перемещение влево на один символ (LeftShift) до достижения указанного условия. □

Лемма 2. *Алгоритм Rewind1 (листинг 2) реализует перемещение в начало текущей подстановки и в соответствующую позицию рабочей строки.*

Доказательство. В цикле выполняется перемещение влево вдоль рабочей строки и строки подстановок (LeftShift). Если текущая подстановка является первой в строке подстановок, то выход из цикла по условию (*p)==0, иначе выход из цикла по разделителю (*p)==(b-3). Затем вернуться к первому символу после разделителя (RightShift) в строке подстановок и соответственно в рабочей строке. Заметим, что перемещения (LeftShift, RightShift) обеспечивают правильную работу в случае возврата к крайнему левому символу текущей строки. □

Лемма 3. *Алгоритм NextP (листинг 2) реализует перемещение к следующей подстановке.*

Доказательство. В цикле выполняется перемещение вправо вдоль строки подстановок (RightShift). Если текущая подстановка является последней в строке подстановок, то выход из цикла по условию (*p)==0, иначе выход из цикла по разделителю (*p)==(b-3). В случае не последней подстановки (*p)>0 выполняется пропуск разделителя (RightShift). Таким образом реализовано позиционирование на первый символ следующей подстановки (либо на конец строки подстановок). □

Лемма 4. *Алгоритм Substr (листинг 2) реализует выполнение текущей подстановки.*

Доказательство. Вначале выполняется позиционирование на последний символ текущей продукции (в случае пустой правой части текущий символ равен b1 или b2). В цикле выполняется перемещение вправо вдоль строки подстановок (RightShift). Если текущая подстановка является последней в строке подстановок, то выход из цикла по условию (*p)==0, иначе выход из цикла по разделителю (*p)==(b-3). Возврат на последний символ обеспечивает перемещение влево (LeftShift).

Затем выполняется копирование правой части подстановки в текущую строку. В цикле выполняется перемещение влево вдоль строки подстановок (LeftShift) до разделителя b1 или b2. Перед каждым перемещением вдоль строки продукций текущий символ рабочей строки добавляется к шифру правой части рабочей строки (MulAdd). После этого в качестве текущего символа рабочей строки присваивается текущий символ строки продукций (*x)=(*p); указанный порядок гарантирует неиспользование разделителя частей подстановки при выходе из цикла.

Для завершения подстановки необходимо удалить ее левую часть из шифра рабочей строки. Поскольку индивидуальное использование MulAdd сохранило неизменным шифр левой части рабочей строки lX, его содержимое соответствует позиционированию на последнем символе вхождения левой части продукции. В цикле выполняется перемещение вправо вдоль строки подстановок (LeftShift). Если текущая подстановка является первой в строке подстановок, то выход из цикла по условию (*p)==0, иначе выход из цикла по разделителю (*p)==(b-3). Перед выполнением перемещения из шифра левой части рабочей строки извлекается символ (ModDiv); указанный порядок гарантирует неиспользование разделителя подстановок при выходе из цикла. □

```
void Rewind (luint *l, uint *c, luint *r, uint b) {
   while((*l)>0)
     LeftShift(l,c,r,b);
}

void Rewind1(luint *lP,uint *p,luint *rP, luint *lX,uint *x,luint *rX, uint b) {
  while((*p)>0 && (*p)!= (b-3)) /* (b-3) - code of "," */
  {
    LeftShift(lP,p,rP,b);
    LeftShift(lX,x,rX,b);
  }
  RightShift(lP,p,rP,b);
  RightShift(lX,x,rX,b);
}

void NextP(luint *lP, uint *p, luint *rP, uint b) {
  while((*p)>0 && (*p)!=(b-3)) /* (b-3) - code of "," */
    RightShift(lP,p,rP,b);
  if((*p)>0) RightShift(lP,p,rP,b);
}

void Substr(luint *lP, uint *p, luint *rP, luint *lX, uint *x, luint *rX, uint b) {
  uint x1;
```

```
  /* find the end of the current production */
  while((*p)>0 && (*p)!=(b-3)) /* (b-3) - code of "," */
    RightShift(lP,p,rP,b);
  LeftShift(lP,p,rP,b);

  /* insert the right part */
  while((*p)<(b-2)) /* (b-2) - code of "->"; (b-1) - code of "|->" */
  {
    mul_add(rX,b,(*x));
    (*x)=(*p);
    LeftShift(lP,p,rP,b);
  }
  LeftShift(lP,p,rP,b);

  /* erase the left part */
  while((*p)>0 && (*p)!=(b-3)) /* (b-3) - code of "," */
  {listing
    mod_div(&x1,lX,b);
    LeftShift(lP,p,rP,b);
  }
  RightShift(lP,p,rP,b);
} /* substr */
```

Листинг 2. Вспомогательные алгоритмы для выполнения НАМ

Теорема 1. *Алгоритм AIPNNAM реализует правила работы НАМ с учетом шифрования.*

Доказательство. После ввода шифра НАМ в правые части осуществляется перемещение влево (RightShift) для позиционирования текущего символа строки продукций и рабочей строки. Повторение основного цикла контролируется логической переменной goon. Последовательности условий и действий, указанные в условных операторах, полностью соответствуют правилам (1) – (6) работы НАМ в посимвольной форме. Корректность использованных вспомогательных процедур Rewind, Rewind1, NextP, Substr доказана в леммах 1 – 4. □

```
void AIPNNAM() {
  int goon=1;

  input_NAM(rP,rX,&b);
  p=x=lP=lX=0;

  RightShift(lP,&p,rP,b);
  RightShift(lX,&x,rX,b);

  while(goon)
  {
    if(p==0) goon=0;
    else
      if(p>=(b-2))
      {
```

```
        if(p==(b-1)) goon=0;
        Substr(lP,&p,rP,lX,&x,rX,b);
        Rewind(lP,&p,rP,b);
        Rewind(lX,&x,rX,b);
      }
      else
      {
        if(x==0)
        {
        Rewind(lX,&x,rX,b);
        NextP(lP,&p,rP,b);
        if(p==0) goon=0;
        }
        else
          if(p!=x)
        {
          Rewind1(lP,&p,rP,lX,&x,rX,b);
          RightShift(lX,&x,rX,b);
        }
        else
        {
          RightShift(lP,&p,rP,b);
          RightShift(lX,&x,rX,b);
        }
      }
  }
  Rewind(lX,&x,rX,b);
  Rewind(lP,&p,rP,b);
  if(x>0) LeftShift(lX,&x,rX,b);
  if(p>0) LeftShift(lP,&p,rP,b);
  output_NAM(rP,rX,b);
} /* AIPNNAM */
```

Листинг 3. Алгоритм выполнения НАМ (AIPNNAM)

Алгоритм AIPNNAM закодирован на языке Си с использованием библиотеки MPI для работы со сверхдлинными целыми числами и протестирован на ряде НАМ.

5. Композиция ИСПНАМ

Композиция ИСПНАМ выполнена по алгоритму AIPNNAM в соответствии с методикой представления алгоритмов ингибиторными сетями Петри, порядком работы с переменными и вспомогательными сетями [1, 4]. Напомним, что предусмотрены шаблоны ИСП, реализующие операторы языка программирования, а также специальные пунктирные дуги, реализующие копирование значений переменных во внутренние контактные позиции операций (процедур). Для удобства представления ИСПНАМ введем дополнительные обозначения вспомогательных подсетей (процедур) для строк, указанных с помощью суффикса (P либо X соответственно):

- RightShiftP:: RightShift(lP, p, rP, b);

- RightShiftX:: RightShift(lX, x, rX, b);
- LeftShiftP:: LeftShift(lP, p, rP, b);
- LeftShiftX:: LeftShift(lX, x, rX, b);
- RewindP:: Rewind(lP, p, rP, b);
- RewindX:: Rewind(lX, x, rX, b).

Позиции, соответствующие параметрам указанных процедур, будем опускать на графическом представлении сети.

Лемма 5. *Сети, представленные на рис. 4, реализуют процедуры LeftShift, RightShift, Rewind, Rewind1, NextP, Substr.*

Доказательство. Следствие применения методики кодирования алгоритмов ингибиторной сетью Петри [1] к процедурам LeftShift, RightShift, Rewind, Rewind1, NextP, Substr. □

Теорема 2. *Сеть IPNNAM, представленная на рис. 5, выполняет произвольный заданный алгорифм Маркова.*

Доказательство. Следует доказать, что сеть IPNNAM реализует алгоритм AIPNNAM, что является следствием применения методики кодирования алгоритмов ингибиторной сетью Петри [1] к алгоритму AIPNNAM с учетом вспомогательных процедур, перечисленных в лемме 5. □

Предполагаем, что перед началом работы сети IPNNAM в позиции rP, rX, b введены значения шифров строки подстановок и входной строки (начальной рабочей строки), а также основание для шифрования соответственно. Для запуска сети фишку помещают в позицию s; завершение работы сети индицирует попадание фишки в позицию f (отсутствуют возбужденные переходы). Шифр результата представлен содержимым позиции rX.

Кроме того, предполагаем единичную начальную разметку позиции goon, что также использовано для быстрого обнуления goon с помощью выполнения декремента одним переходом. Фрагменты для вычисления вспомогательных значений b1, b2, b3, а также перемещения вправо RightShift (аналогичного перемещению влево) не приведены. Напомним, что использованы сети ModDiv, MulAdd, NE, GE, Assign, описанные в [1, 4].

Для удобства графического представления использовано наложение позиций: все позиции с одинаковым именем логически являются одной и той же позицией. Позиция с именем cond использована для временного хранения вычисленного условия ветвления.

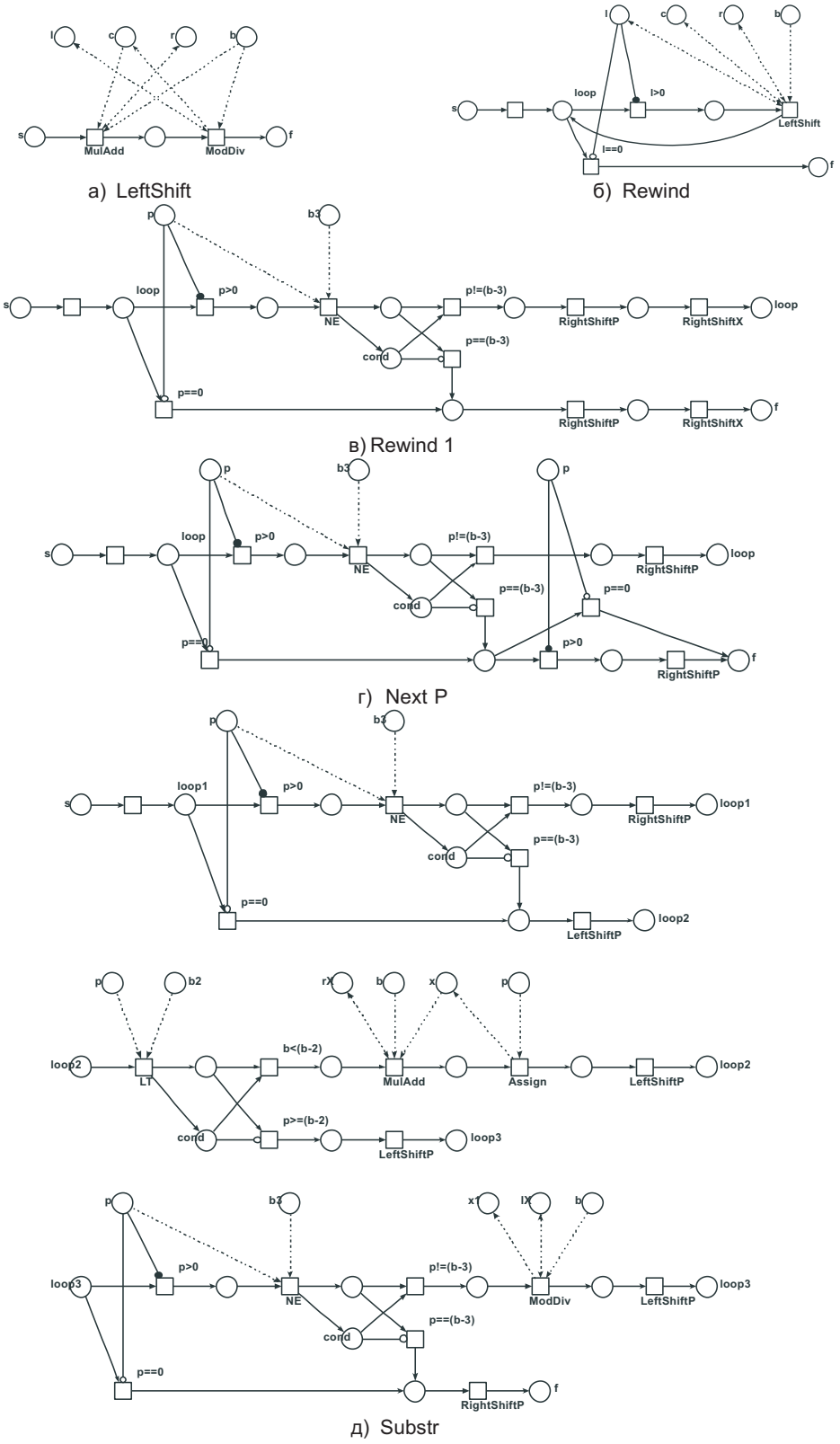

Рис. 4. Компоненты ИСПНАМ (IPNNAM)

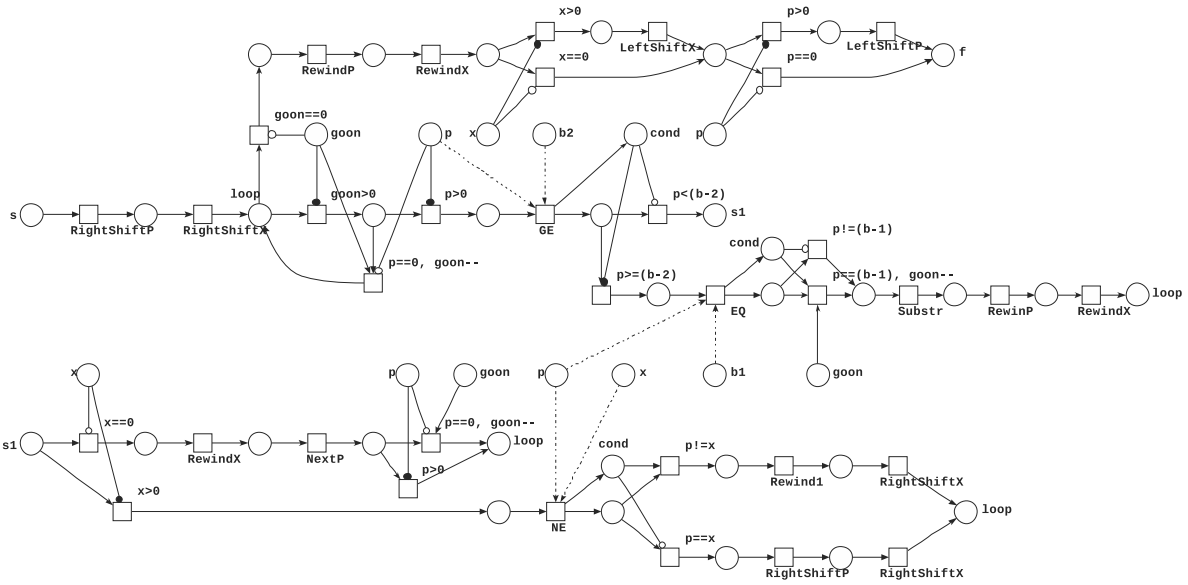

Рис. 5. Ингибиторная сеть Петри, выполняющая нормальный алгорифм Маркова ИСПНАМ (IPNNAM)

6. Выводы

Впервые построена ингибиторная сеть Петри с фиксированной структурой, выполняющая произвольный заданный алгорифм Маркова. Шифр алгорифма Маркова представлен маркировкой семи выделенных позиций; шифр рабочей строки представлен шифрами текущего символа и двух стеков по обе стороны от него, что позволяет организовать работу с использованием незначительного числа вспомогательных позиций. Использована методика кодирования программ ингибиторными сетями Петри и вспомогательные сети, реализующие логические, арифметические операции и копирование маркировок.

В новой парадигме вычислений, организованных на сетях Петри, реализация традиционных алгоритмических систем обеспечивает совместимость концепций. Отметим, что продукционные формы нормальных алгорифмов Маркова широко применяют для спецификации интеллектуальных систем.

Список литературы

1. Zaitsev D.A. Universal Inhibitor Petri Net // Proceedings of the 17th German Workshop on Algorithms and Tools for Petri Nets. Cottbus, Germany, October 07-08. 2010. P. 1–15. (http://CEUR-WS.org/Vol-643/).

2. Turing A.M. On Computable Numbers with an Application to the Entscheidungsproblem // Proceedings of the London Mathematical Society. 1936. 42. P. 230—265.

3. Марков А.А. Теория алгорифмов // Тр. МИАН. 1954. 42. 375 с.

4. Зайцев Д.А. Построение сети Петри исполняющей машину Тьюринга // Материалы IV международной научно-технической конференции «Компьютерная математика в науке, инженерии и образовании» (CMSEE-2010), г. Полтава, 1–31 октября 2010 г. Киев: Изд-во НАН Украины, 2010. С. 12–14.

5. Котов В.Е. Сети Петри. М.: Наука, 1984. 160 с.

6. Слепцов А.И., Юрасов А.А. Автоматизация проектирования управляющих систем гибких автоматизированных производств / Под ред. Б.Н. Малиновского. Киев: Техніка, 1986. 160 с.

Приложение. Примеры шифрования НАМ

№	Назначение	Спецификация		Шифр
1	Стирание символов a, b	A	abc,>)	7
		P	a>,b>	12923
		X	acaacbaacba	381459289
		Y	ccc	171
2	Дописывание цепочки aabbcc в конец строки	A	abcx,>)	8
		P	xa>ax,xb>bx,xc>cx,x)aabbcc,>x	93518363095517155420193676
		X	ccbcb	9883
		Y	ccbcbaabbcc	3661932187
3	Обратный порядок символов	A	abcxy,>)	9
		P	xx>y,yx>y,ya>ay,yb>by,yc>cy,y),xaa>axa,xab>bxa,xac>cxa,xba>axb,xbb>bxb,xbc>cxb,xca>axc,xcb>bxc,xcc>cxc,>x	8463636869530269624567207595434984259917833671785036771359568709928518476877331159825245871193090361
		X	baca	983
		Y	acab	1567
4	Остаток от деления в унарной системе счисления	A	|,>)	5
		P	|||||> (делитель 5)	10156
		X	|||||||||||||||||||| (делимое 23)	2980232238769531
		Y	||| (остаток 3)	31
5	Деление в унарной системе счисления	A	|*,>)	6
		P	*|||||>|*,*|>*,*),>* (делитель 5)	1691055643342580
		X	|||||||||||||||||||| (делимое 23)	157946044610720563
		Y	|||| (частное 4)	259

Inhibitor Petri Net that Executes an Arbitrary Given Markov Normal Algorithm

Zaitsev D.A.

Keywords: normal algorithm of Markov, inhibitor Petri net, encoding, cipher

The inhibitor Petri net with a fixed structure that executes an arbitrary given Markov normal algorithm was constructed. The algorithm and its input string are encoded by nonnegative integer numbers and put into dedicated places of the Petri net which implements the application of algorithm productions over the string of symbols. The rules of the sequential, branching and cyclic processes encoding by Petri nets were used. At the completion of the net work, the output string is restored (decoded) from the integer form of representation. Within the paradigm of computations on Petri nets the net built provides the compatibility of systems.

Сведения об авторе:
Зайцев Дмитрий Анатольевич,
Международный гуманитарный университет, г. Одесса, Украина,
доктор технических наук, профессор.
Сайт: *http* : *//member.acm.org/∼daze*, *http* : *//daze.ho.ua* .
Основные научные результаты:
анализ бесконечных сетей Петри с регулярной структурой; кланы систем линейных алгебраических уравнений (функциональные подсети), оптимальный коллапс взвешенного графа; композиционный анализ сетей Петри; функциональная эквивалентность, передаточная функция и эквивалентные преобразования временных сетей Петри; временные сети Петри с многоканальными переходами, уравнение состояний, частичные инварианты; синтез функций непрерывной (нечеткой) логики, заданных таблично.