УДК 519.6, 510.51

Д.А. ЗАЙЦЕВ

УНИВЕРСАЛЬНАЯ СЕТЬ ПЕТРИ

Ключевые слова: *универсальная ингибиторная сеть Петри, универсальная машина Тьюринга, алгоритм, шифрование, поток управления.*

ВВЕДЕНИЕ

Известно, что ингибиторные, синхронные, приоритетные и другие расширенные классы сетей Петри являются универсальной алгоритмической системой [1, 2]. Для таких алгоритмических систем, как машины Тьюринга известны примеры построения универсальной машины [3]. В этой связи определенный интерес представляет построение универсальной сети Петри, которая может выполнять произвольную заданную сеть Петри. Это составляет цель настоящей статьи. Предварительный вариант работы был представлен на 17-м Немецком семинаре по алгоритмам и программам для сетей Петри [4].

Помимо побочных результатов по кодированию сетей Петри целыми числами, аналогичных отмеченным в [9] для универсальных машин Тьюринга, основной мотивацией построения универсальной сети Петри является тенденция применения сетей Петри не только для моделирования, но и как графической основы параллельных асинхронных языков программирования. В этом случае универсальная сеть Петри представляет прототип процессора, предназначенного для выполнения программ, написанных на языке сетей Петри.

Возможно также применение модификаций универсальной сети для анализа свойств сетей Петри (таких как ограниченность, консервативность, живость) на основе сохранения в зашифрованном виде не только текущей трассы запуска переходов, а всего графа достижимых (покрывающих) маркировок.

1. КОНЦЕПЦИЯ УНИВЕРСАЛЬНОЙ СЕТИ ПЕТРИ

Построение универсальной сети выполним в классе ингибиторных сетей Петри [1, 2]. Соответствующую универсальную ингибиторную сеть Петри обозначим UIPN. Ввиду недетерминированного характера динамики сети Петри наиболее близким аналогом являются недетерминированные машины Тюринга [3].

Поскольку интерес представляет построение универсальной сети UIPN с фиксированной структурой, единственным средством представления входной и выходной информации является маркировка фиксированного числа определенных позиций сети UIPN. Поэтому необходимо задать некоторые правила взаимно однозначного шифрования графа сети Петри и ее маркировки фиксированным числом целых неотрицательных чисел. Пусть заданы соответствующие правила шифрования, а также sXIPN — шифр графа сети Петри XIPN и sQXIPN — шифр маркировки QXIPN.

Концепция достижимости маркировки в сети Петри подразумевает наличие соответствующей допустимой последовательности срабатываний переходов [1, 2]. Однако использование лишь маркировок QXIPN в определении UIPN не гарантирует получения всех допустимых последовательностей срабатывания переходов сети XIPN. Пусть заданы также правила шифрования последовательностей срабатывания переходов и sZQXIPN — шифр допустимой последовательности

срабатывания переходов ZQXIPN, переводящей сеть Петри XIPN из маркировки Q0XIPN в маркировку QXIPN.

Определение 1. Сеть Петри UIPN является универсальной ингибиторной сетью Петри тогда и только тогда, когда для произвольной заданной ингибиторной сети XIPN и ее начальной маркировки Q0XIPN сеть UIPN останавливается в маркировке (sQXIPN, sZQXIPN) такой, что маркировка QXIPN достижима в XIPN с помощью последовательности срабатываний переходов ZQXIPN и любая маркировка (sQXIPN, sZQXIPN), в которой останавливается UIPN, является шифром маркировки QXIPN, достижимой в XIPN из начальной маркировки Q0XIPN с помощью последовательности ZQXIPN.

Требование возможности останова UIPN даже в случае не тупиковой маркировки QXIPN сети XIPN связано с обеспечением проверки (наблюдения) любой достижимой маркировки (и последовательности срабатывания переходов) и абстрагированием от реализации UIPN, в противном случае необходимо добавить ограничения для исключения из рассмотрения промежуточных маркировок сети UIPN.

2. ФОРМАЛЬНОЕ ПРЕДСТАВЛЕНИЕ ИНГИБИТОРНОЙ СЕТИ ПЕТРИ

Граф ингибиторной сети Петри [1, 2] — это четверка $G = (P, T, B, D)$, где $P = \{p_1, \dots, p_m\}$ — конечное множество вершин, называемых позициями; $T = \{t_1, \dots, t_n\}$ — конечное множество вершин, называемых переходами; отображения $B\colon P \times T \to \mathbb{N} \cup \{-1\}$ и $D\colon T \times P \to \mathbb{N}$ задают входящие и исходящие дуги переходов соответственно и их кратность, $\mathbb{N}$ — множество целых неотрицательных чисел; нулевое значение отображений B, D означает отсутствие дуги, ненулевое — кратность дуги, специальное значение –1 означает ингибиторную дугу. Отображения могут быть представлены соответствующими матрицами: $B = ||b_{i,j}||$, $b_{i,j} = B(p_j, t_i)$ и $D = ||d_{i,j}||$, $d_{i,j} = D(t_i, p_j)$.

Состояние сети называют маркировкой и представляют отображением $Q\colon P \to \mathbb{N}$, задающим количество динамических элементов — фишек в позициях сети. Ингибиторная сеть Петри [1, 2] — это пара $N = (G, Q_0)$, где G — граф сети, а Q_0 — ее начальная маркировка. Маркировка может быть представлена соответствующим вектором: $Q = ||q_j||$, $q_j = Q(p_j)$. Таким образом, ингибиторная сеть Петри задана парой чисел, парой матриц и вектором: $N = (m, n, B, D, Q_0)$.

Графически позиции представляют окружностями, переходы — прямоугольниками, фишки изображают точками, находящимися внутри позиций. Если значение маркировки позиции велико, то внутри позиции записывают число, равное количеству фишек. Для графического представления ингибиторной дуги используют полый круг на ее конце. Ингибиторные дуги проверяют маркировку позиции на равенство нулю. Дуги с закрашенным кругом на конце являются вспомогательным обозначением для пары дуг противоположного направления с одинаковой кратностью. Они используются для проверки ненулевой маркировки позиции. Примеры ингибиторных сетей даны в приложении А.

Динамика ингибиторной сети представляет собой пошаговый процесс изменения ее маркировки в результате срабатывания переходов [1, 2] и может быть формально описана следующей системой:

$$\begin{cases} q_j^k = q_j^{k-1} - x(b_{l,j}) + d_{l,j},\ j = \overline{1, m}, \\ u(t_i) = \bigwedge\limits_{j=1,m} (y(b_{i,j}) \wedge (q_j^{k-1} = 0)) \vee (\bar{y}(b_{i,j}) \wedge (q_j^{k-1} \geq b_{i,j})),\ i = \overline{1, n}, \\ u(t_l) = 1,\ \ l \in \overline{1, n}, \\ k = 1, 2, \dots, \\ x(b) = \begin{cases} b, b \geq 0, \\ 0, b = -1, \end{cases} \quad y(b) = \begin{cases} 0, b \geq 0, \\ 1, b = -1. \end{cases} \end{cases} \tag{1}$$

Первая строка системы (1) описывает изменение маркировки при срабатывании перехода t_l; функция $u(t_i)$ во второй строке представляет условие возбуждения перехода t_i на текущем шаге k; третья строка задает недетерминированный выбор срабатывающего перехода t_l из множества возбужденных; четвертая строка задает порядок следования шагов; вспомогательные отображения x и y служат для задания декремента маркировки и распознавания ингибиторной дуги соответственно.

3. ШИФРОВАНИЕ ИНГИБИТОРНОЙ СЕТИ ПЕТРИ

Представим шифр графа ингибиторной сети Петри, ее текущую маркировку и соответствующую последовательность срабатывания переходов в виде маркировки десяти выделенных позиций универсальной сети UIPN (рис. 1). Примеры шифрования сетей приведены в приложении Б.

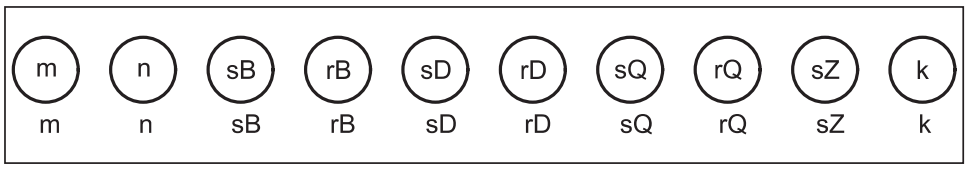


Рис. 1. Представление шифров сети Петри и последовательности срабатывания переходов

3.1. Шифрование вектора. Пусть Q — некоторый вектор (строка), содержащий m целых неотрицательных элементов. Предполагаем, что индексация элементов начинается с нуля. Пусть также найдено значение $r = \max\limits_j q_j + 1$.

Представим функцию шифрования вектора:

$$s = \varphi(Q) = \sum_{j=0}^{m-1} r^j \cdot q_j .$$

Утверждение 1. Функция шифрования вектора является инъективной.

Соответствующая функция дешифрования может быть представлена как

$$q_i = \psi(s, j) = (s \operatorname{div} r^j) \bmod r.$$

Указанное шифрование является формой представления чисел в позиционной системе счисления с основанием r.

Шифрование может быть выполнено рекуррентно:

$$s_j = s_{j-1} \cdot r + q_{m-1-j}, \;\; s_0 = q_{m-1}, \;\; j = \overline{1, m-1}, \tag{2}$$

где шифр вектора Q равен $s = s_{m-1}$.

Дешифрование может быть также выполнено рекуррентно:

$$q_j = s_{m-1-j} \bmod r, \;\; s_{m-1-(j+1)} = s_{m-1-j} \operatorname{div} r, s_{m-1} = s, \; j = \overline{0, m-1}. \tag{3}$$

3.2. Шифрование матрицы. Пусть A — некоторая $n \times m$-матрица с целыми неотрицательными значениями элементов. Предполагаем, что индексация элементов начинается с нуля. Пусть также найдено значение

$$r = \max_{i,j} a_{i,j} + 1 .$$

При шифровании будем представлять матрицу как вектор с разложением по строкам. Тогда матрица A может быть зашифрована как

$$s = \varphi(A) = \sum_{i=0}^{n} \sum_{j=0}^{m} r^{(m \cdot i + j)} \cdot a_{i,j} .$$

Утверждение 2. Функция шифрования матрицы является инъективной. Соответствующая функция дешифрования может быть представлена как

$$a_{i,j} = \psi(A, i, j) = (s \operatorname{div} r^{(m \cdot i + j)}) \bmod r.$$

Шифрование может быть выполнено рекуррентно:

$$s_\upsilon = s_{\upsilon-1} \cdot r + a_{i,j},\ s_0 = a_{n-1,m-1},$$
$$\upsilon = \overline{1, n \cdot m - 1},\ i = \upsilon \operatorname{div} n,\ j = \upsilon \bmod n, \tag{4}$$

где шифр матрицы A равен $s = s_{n \cdot m-1}$.

Дешифрование может быть также выполнено рекуррентно:

$$a_{i,j} = s_{n \cdot m-1-\upsilon} \bmod r,\ s_{n \cdot m-1-(\upsilon+1)} = s_{n \cdot m-1-\upsilon} \operatorname{div} r,\ s_{n \cdot m-1} = s,$$
$$\upsilon = \overline{0, n \cdot m - 1},\quad i = \upsilon \operatorname{div} n,\quad j = \upsilon \bmod n. \tag{5}$$

3.3. Шифрование графа ингибиторной сети Петри. Граф представлен парой матриц B и D, являющихся соответственно матрицами входящих и исходящих дуг переходов. Обычно нулевое значение элемента матрицы указывает на отсутствие соответствующей дуги, ненулевое — на ее кратность. Представление ингибиторных дуг матрицы B требует специальных соглашений, чтобы избежать использования отрицательных значений. Пусть k — кратность дуги. Тогда для ее представления будем использовать значение $k+1$. Значение, равное единице, зарезервируем для представления ингибиторной дуги.

Целесообразно раздельное шифрование в соответствии с (4) и хранение в отдельных позициях шифров матриц B и D, а также соответствующих значений r. Для хранения зашифрованного графа сети Петри будем использовать шесть позиций с именами m, n, sB, rB, sD, rD, изображенных на рис. 1, маркировка которых содержит значения m, n, sB, rB, sD, rD соответственно.

3.4. Шифрование маркировки. Маркировка сети Петри, содержащей m позиций, задана вектором Q размера m с целыми неотрицательными компонентами $q_j = Q(p_j)$. Для хранения зашифрованной в соответствии с (2) маркировки будем использовать три позиции с именами m, sQ, rQ, изображенные на рис. 1, маркировка которых содержит значения m, sQ, rQ соответственно.

3.5. Шифрование последовательности срабатывания переходов. Последовательность срабатывания переходов Z длины k может быть представлена вектором $\overline{Z}$ размера k с целыми неотрицательными компонентами $z_j = l$, где l — номер перехода t_l, срабатывающего на шаге j. Для хранения зашифрованной в соответствии с (2) последовательности будем использовать три позиции с именами k, sZ, n, изображенные на рис. 1, маркировка которых содержит значения k, sZ, n соответственно.

Отметим, что позиции m, n используются в качестве параметров для шифрования (дешифрования) графа сети Петри, маркировки и последовательности срабатывания переходов.

3.6. Шифрование множества возбужденных переходов. Множество возбужденных переходов сети Петри является вспомогательной информацией для недетерминированного выбора срабатывающего перехода t_l $(u_l = 1)$ на текущем шаге. Для представления множества возбужденных переходов используем вектор $\overline{u}$ длины n, компонентами которого являются индикаторы возбуждения u_i соответствующих переходов t_i, $i = \overline{0, n-1}$, вычисленные в соответствии с (1). Тогда для шифрования $\overline{u}$ применим правила шифрования вектора (2) при $r = 2$.

4. АЛГОРИТМ ВЫПОЛНЕНИЯ ИНГИБИТОРНОЙ СЕТИ ПЕТРИ

По системе (1) в соответствии с выбранным способом шифрования графа, маркировки и последовательности срабатывания переходов сети Петри построим алгоритм AUIPN выполнения ингибиторной сети Петри на Си-подобном псевдоязыке:

```
void  AUIPN()
{
    uint  u,  l;

    inputXIPN();
    k=1;  sZ=0;
    while(NonDeterministic())
    {
         CheckFire(&u);
         if(u==0)  goto  out;
         PickFire(u,  &l);
         Fire(l);
         MUL_ADD(&sZ,n,l-1);
         k++;
    }
    out:outputXIPN();
}
```

В алгоритме использованы следующие переменные и процедуры: `u` — шифр индикатора возбужденных переходов, `l` — номер срабатывающего перехода, `k` — номер шага; `CheckFire` — проверка условий возбуждения переходов, `PickFire` — выбор срабатывающего перехода, `Fire` — срабатывание перехода; `NonDeterministic` — недетерминированный выбор числа из множества $\{0, 1\}$. Представим алгоритмы вспомогательных процедур `MUL_ADD`, `MOD_DIV`, реализующих рекуррентное шифрование/дешифрование в соответствии с (2), (3):

```
void  MUL_ADD(&x,y,z)
{
    (*x) = (*x) * y + z;
}

void  MOD_DIV(&m,&x,y)
{
    (*m) = (*x) mod y;
    (*x) = (*x) div y;
}
```

Алгоритм процедуры `CheckFire`:

```
void  CheckFire(uint  *u)
{
    uint  i,  j,  qj,  bij,  ui,  uij;
    uint  sB1,  sQ1;

    sB1=sB;  &u=0;
    for(i=n;  i>0;  i--)
    {
       sQ1=sQ;
       ui=1;
       for(j=m;  j>0;  j--)
```

```
        {
              MOD_DIV(&qj,&sQ1,rQ);
              MOD_DIV(&bij,&sB1,rB);
              uij=1;
              if(bij==0)  continue;
              bij--;
              if(bij==0)  uij=(qj==0);
              else  uij=(qj>=bij);
              ui=ui  &&  uij;
        }
        MUL_ADD(&u,2,ui);
    }
}
```

Лемма 1. Алгоритм `CheckFire` формирует множество переходов, возбужденных в текущей маркировке.

Доказательство. Алгоритм представляет последовательное вычисление компонентов вектора $\bar{u}$ в соответствии со второй строкой системы (1) и его одновременное шифрование (2) в переменной `u` после вычисления текущей компоненты в переменной `ui`. Цикл по переменной `i` задает перебор всех переходов, вложенный цикл по переменной `j` задает перебор всех позиций для текущего перехода. Порядок последовательного дешифрования элементов вектора Q и матрицы B соответствует порядку изменения индексов циклов в соответствии с (3) и (5). □

Алгоритм процедуры `PickFire`:

```
void  PickFire(uint u,  uint *l)
{
    uint  ui, i;

    i=0;
    while(u>0)
    {
       MOD_DIV(&ui,&u,2);
       i++;
       if(ui==0)  continue;
       if(NonDeterministic())  goto  out;
    }
out:*l=i;
}
```

Лемма 2. Алгоритм `PickFire` выполняет выбор произвольного срабатывающего перехода из множества возбужденных.

Доказательство. Условие выбора срабатывающего перехода соответствует третьей строке системы (1) и порядку дешифрования вектора $\bar{u}$ в соответствии с (3). Для недетерминированного выбора возбужденного перехода использована функция `NonDeterministic` для выхода из цикла. Условие $u > 0$ обеспечивает завершение цикла после обработки последнего возбужденного перехода, который будет выбран для срабатывания. □

Алгоритм процедуры `Fire`:

```
void  Fire(uint  l)
{
    uint  rQ1,  maxQ1,  shift,  qj,  bij,  dij,  j;
    uint  sB1,  sD1,  sQ1;

    sB1=sB;  sD1=sD;  sQ1=0;  rQ1=rQ+rD-1;  maxQ1=0;
```

```
        shift=(n-l)*m;
        while(shift--)
        {
           MOD_DIV(&b,&sB1,rB);
           MOD_DIV(&d,&sD1,rD);
        }

        for(j=m; j>0; j--)
        {
           MOD_DIV(&qj,&sQ,rQ);
           MOD_DIV(&bij,&sB1,rB);
           if(bij>0)  bij--;
           MOD_DIV(&dij,&sD1,rD);
           qj=qj-bij+dij;
           maxQ1=MAX(qj,maxQ1);
           MUL_ADD(&sQ1,rQ1,qj);
        }
        sQ=0;  rQ=maxQ1+1;

        for(j=m;  j>0;  j--)
        {
           MOD_DIV(&qj,&sQ1,rQ1);
           MUL_ADD(&sQ,rQ,qj);
        }
    }
```

Лемма 3. Алгоритм Fire реализует изменение маркировки в результате срабатывания указанного перехода.

Доказательство. Алгоритм реализует перевычисление маркировки в соответствии с первой строкой системы (1) и выбранным порядком дешифрования матриц B, D и вектора Q в соответствии с (5) и (3). Значение переменной `shift` соответствует количеству пропускаемых элементов для позиционирования на начало строки срабатывающего перехода с номером l. Затем в первом цикле по переменной `j` выполняется предварительное перевычисление шифра (2) маркировки в переменной `sQ1`. При этом использовано значение `rQ1`, обеспечивающее хранение наибольшего допустимого значения элемента новой маркировки `rQ+rD-2`. Для предотвращения нарастания `rQ` во втором цикле по переменной `j` выполняется окончательное перевычисление шифра (2) маркировки в переменной `sQ` с учетом фактического значения максимального элемента `maxQ1`. □

Теорема 1. Алгоритм `AUIPN` реализует динамику произвольной заданной ингибиторной сети Петри.

Доказательство. Покажем, что алгоритм `AUIPN` перевычисляет маркировку ингибиторной сети Петри в соответствии с системой (1) и сохраняет использованную последовательность срабатывания переходов. Алгоритм выполнения шага представлен циклом `while` алгоритма `AUIPN` и полностью соответствует системе (1). Вначале процедура `CheckFire` определяет множество возбужденных переходов и формирует шифр (2) соответствующего индикатора возбужденных переходов `u` (лемма 1). При отсутствии возбужденных переходов $u == 0$ выполнение алгоритма прекращается, что соответствует тупиковой маркировке. Процедура `PickFire` выполняет недетерминированный выбор срабатывающего перехода из множества возбужденных. Переменная `l` возвращает номер срабатывающего перехода (лемма 2). Процедура `Fire` реализует изменение текущей маркировки в результате срабатывания перехода с номером `l` с ее одновремен-

ным шифрованием (2) (лемма 3). Затем в шифр (2) последовательности срабатывания переходов sZ добавляется номер l и значение текущего шага k увеличивается на единицу. Недетерминированный выход из цикла соответствует определению 1. □

Алгоритм AUIPN был также закодирован на языке Си с использованием библиотеки MPI для представления сверхдлинных целых и протестирован на ряде сетей Петри.

Теорема 2. Алгоритм AUIPN может быть представлен ингибиторной сетью Петри.

Доказательство теоремы 2 непосредственно вытекает из того факта, что ингибиторная сеть Петри является универсальной алгоритмической системой [1] и алгоритм AUIPN использует только целые неотрицательные скалярные переменные, значения которых могут быть представлены маркировками соответствующих позиций сети Петри.

Для конструктивного доказательства теоремы 2 соответствующая сеть будет построена по алгоритму AUIPN в последующих разделах настоящей статьи.

5. ПРИНЦИПЫ КОДИРОВАНИЯ АЛГОРИТМА ИНГИБИТОРНОЙ СЕТЬЮ ПЕТРИ

Известны различные подходы к кодированию алгоритма сетью Петри, основанные на принципах комбинирования потоков данных и потоков управления [2, 5, 6]. Используем непосредственное кодирование основных операторов языка программирования Си для представления одного потока управления. Каждая переменная представлена соответствующей позицией сети Петри. Все переменные являются глобальными. Поток управления моделируется трассой прохождения одной фишки из начальной позиции start в конечную позицию finish.

Для унифицированной организации работы с переменными будем представлять операторы языка программирования в виде схемы, изображенной на рис. 2.

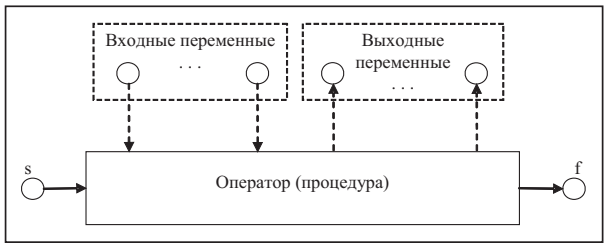


Рис. 2. Схема представления оператора языка программирования

Для обеспечения повторного прохождения потока управления через операторы (процедуры) примем следующие соглашения: все внутренние позиции имеют нулевую разметку; перед началом работы входные переменные копируются во входные позиции оператора; работа оператора запускается фишкой в позиции start (s); оператор завершает свою работу при попадании фишки в позицию finsh (f); при завершении работы все позиции оператора становятся пустыми, за исключением выходных позиций, в которых находится результат. Штриховые дуги обозначают следующие дополнительные правила формирования значений входных и выходных переменных оператора: при запуске содержимое переменной копируется в локальную входную позицию оператора; после завершения переменная очищается и в нее перемещается значение из локальной выходной позиции оператора (рис. 3). В случае нескольких переменных создаются цепочки COPY для последовательного копирования входных переменных и цепочки CLEAN, MOVE для перемещения выходных переменных. Последовательность операций CLEAN, MOVE далее обозначена ASSIGN. Представленная схема обеспечивает корректную работу с переменными в общем случае. Работа с переменными может быть оптимизирована, когда они являются временными либо входными и выходными одновременно.

Для вычисления выражений может быть использован подход потоков данных: выполнение операций упорядочивается в соответствии с их приоритетами. Выходные позиции операции совмещаются с входными позициями следующей операции.

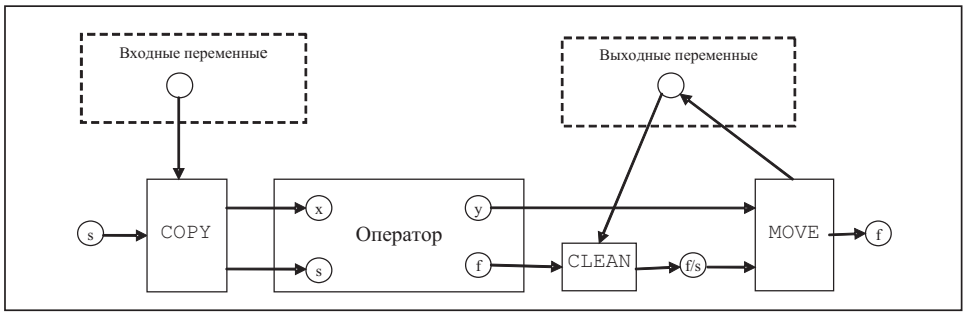


Рис. 3. Схема формирования входных и выходных переменных

Далее рассмотрим представление основных управляющих конструкций языка программирования: последовательности (`operator1; operator2`) ветвления (`if(condition()) operator1; else operator2`), цикл `while` (`while(condition()) operator`) и цикл `for` (`for(i=n;i>0;i--) operator`). Абстрагируемся от используемых переменных.

Лемма 4. Алгоритмические управляющие конструкции языка программирования могут быть закодированы ингибиторной сетью Петри в форме, изображенной на рис. 4.

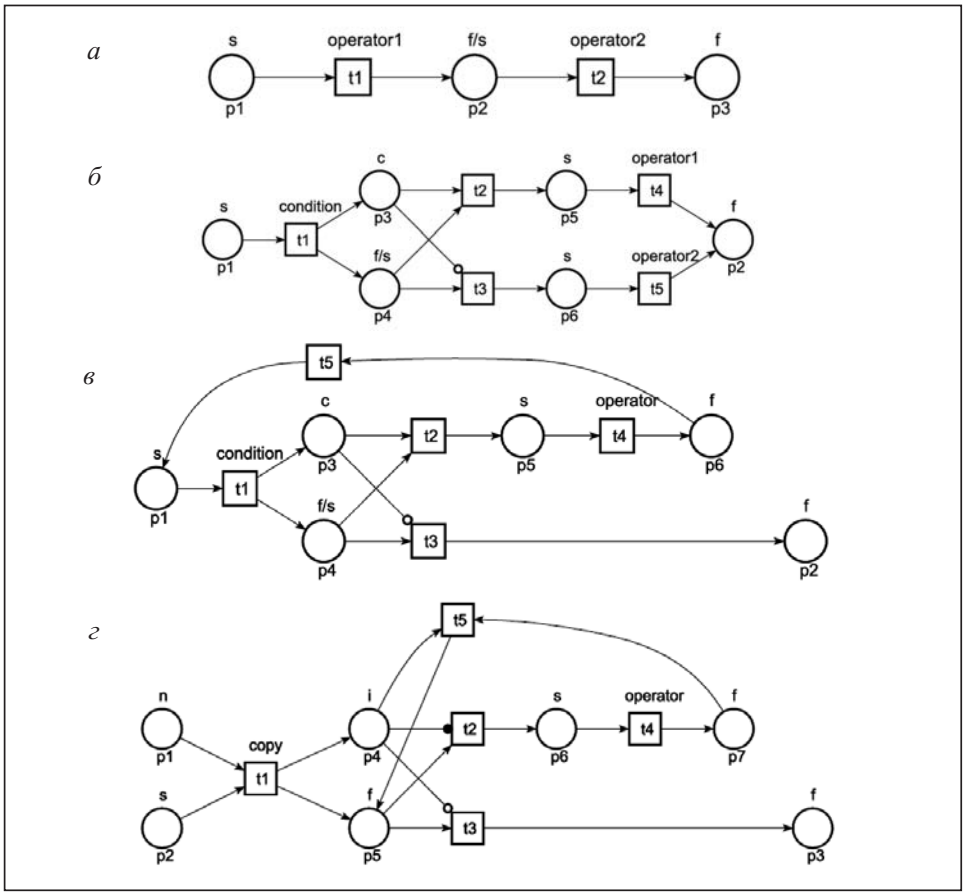


Рис. 4. Схема кодирования управляющих конструкций языка программирования: последовательность (*а*), ветвление (*б*), цикл `while` (*в*), цикл `for` (*г*)

Для каждой управляющей конструкции корректность представления может быть доказана путем классификации всех допустимых последовательностей срабатывания переходов и их сравнения с порядком выполнения операторов в конструкциях языка программирования [2]. Отметим, что в соответствии с рис. 4, *а* суперпозиция операторов при кодировании программы выполняется путем наложения (совмещения) выходной позиции `f` первого оператора с входной позицией s второго оператора. Совмещение позиций глобальных переменных с контактными позициями операторов использовано также при присоединении вспомогательных сетей копирования значений переменных (`CLEAN`, `COPY`, `MOVE`, `ASSIGN`), представленных штриховыми дугами на рис. 2.

Реализация основных алгебраических и логических операций сетями Петри рассмотрена в [2, 6]. В некоторых случаях целесообразно представить непосредственно наиболее часто встречающиеся операции (как, например, `MUL_ADD` и `MOD_DIV` для шифрования и дешифрования сети Петри). В приложении А приведены сети, выполняющие операции, которые использованы в алгоритме `AUIPN`.

Лемма 5. Сети, представленные в приложении А, реализуют указанные операции.

Для каждой представленной сети доказательство корректной реализации требуемой операции можно построить на основе классификации всех допустимых последовательностей срабатывания переходов [2, 6].

6. КОМПОЗИЦИЯ УНИВЕРСАЛЬНОЙ ИНГИБИТОРНОЙ СЕТИ ПЕТРИ UIPN

Закодируем алгоритм работы универсальной сети Петри `AUIPN` ингибиторной сетью Петри в соответствии с правилами, описанными в разд. 5. Отметим, что лемма 4 и лемма 5 (Приложение А) перечисляют все управляющие конструкции и все операции, используемые в алгоритме `AUIPN`. Получим сеть `UIPN`, изображенную на рис. 5 и 6. Для представления переменных алгоритма используются совмещенные позиции: все позиции с одинаковым именем логически являются одной и той же позицией. Совмещенные позиции упрощают графическое представление сети. Предполагаем, что перед запуском сети `UIPN` шифр исполняемой сети `XIPN` загружен в позиции, изображенные на рис. 1, а после останова сети `UIPN` шифр маркировки и последовательности срабатывания переходов сети `XIPN` считан из соответствующих позиций. Штриховые дуги обозначают рассмотренные в разд. 5 соглашения по копированию входных и выходных переменных. Двухнаправленные дуги использованы для работы с переменными, которые одновременно являются входными и выходными. В этом случае копирование можно оптимизировать, дважды выполняя операции `MOVE` без очистки. В некоторых случаях для копирования входной переменной вместе с ее одновременной очисткой целесообразно использовать `MOVE` вместо `COPY`. В качестве соответствующего обозначения используется пунктир. Подстановка перехода подразумевает копирование соответствующей подсети с совмещением контактных позиций [7]. В общем случае подстановка перехода требует указания отображения входных и выходных позиций. В представленных сетях отображение позиций определено контекстом использования операции и не указано на рисунках.

Теорема 3. Сеть UIPN является универсальной ингибиторной сетью Петри.

Доказательство теоремы 3 непосредственно следует из теоремы 1 и корректности использованных правил кодирования последовательного алгоритма ингибиторной сетью Петри (лемма 4), а также корректности сетей, реализующих использованные операции (лемма 5).

Отметим, что сеть UIPN изображена на рис. 5, 6 покомпонентно в соответствии с использованными процедурами, операциями и правилами работы с переменными. Определенный интерес представляют компоновка UIPN в форме единой ингибиторной сети Петри и ее исполнение в среде некоторой моделирующей системы, имитирующей процесс срабатывания переходов.

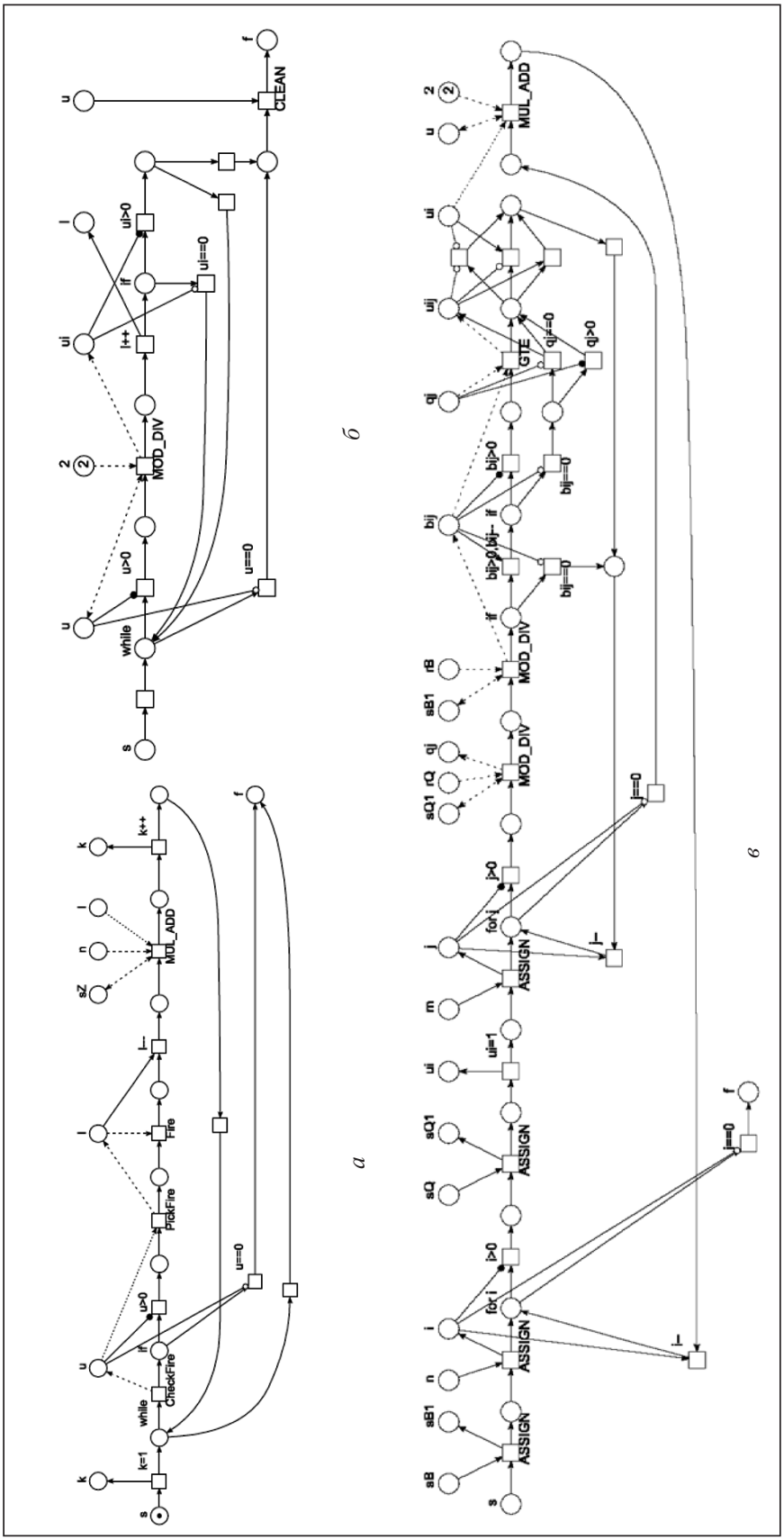


Рис. 5. Универсальная ингибиторная сеть Петри UIPN (*а*); подсеть PickFire (*б*); алгоритм процедуры CheckFire (*в*)

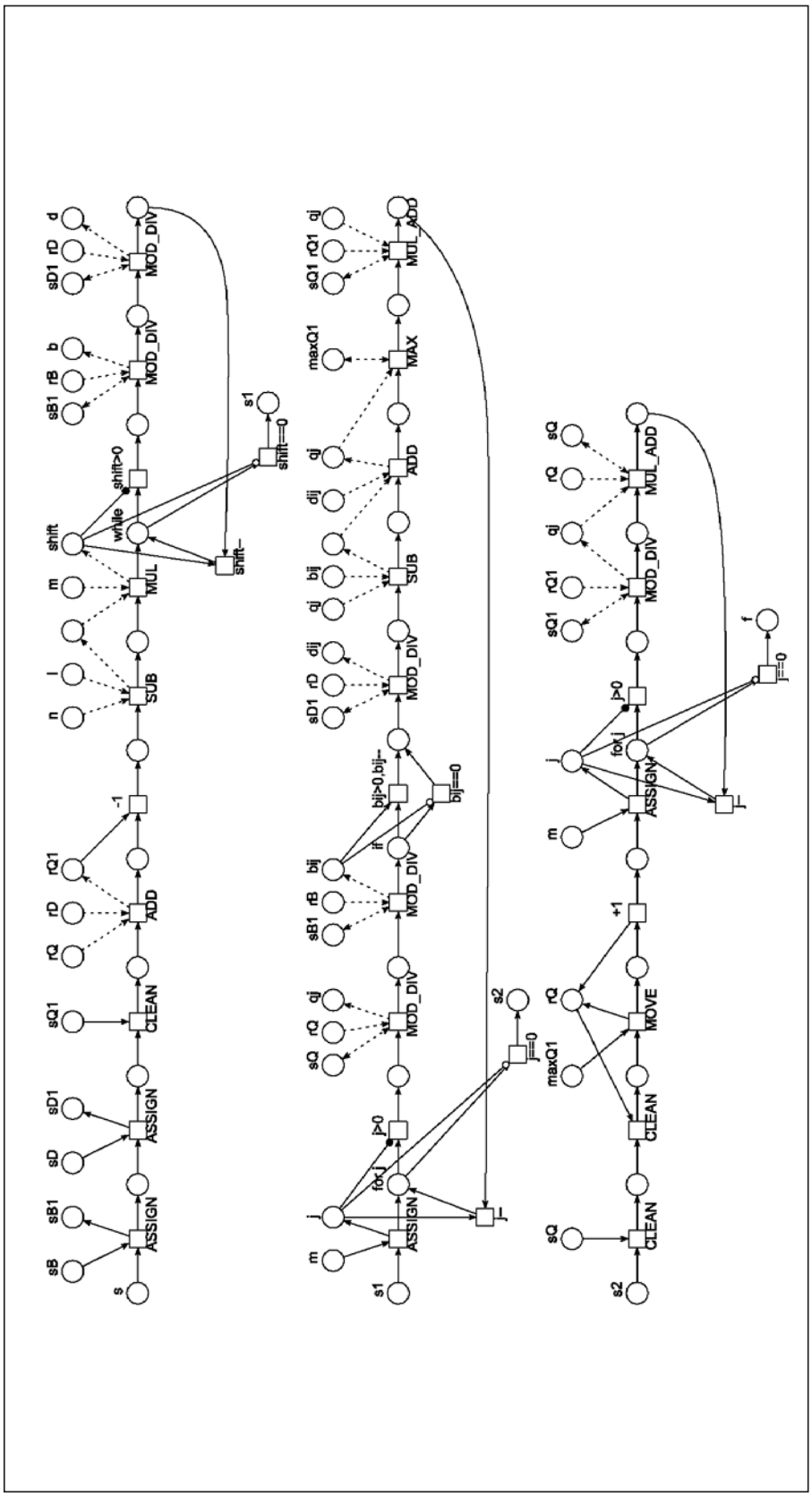


Рис. 6. Подсеть `Fire`

ЗАКЛЮЧЕНИЕ

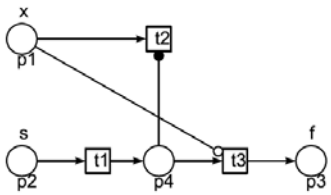
В настоящей статье выполнено построение универсальной ингибиторной сети Петри, исполняющей произвольную заданную ингибиторную сеть Петри. В соответствии с оценками, полученными в [8], сеть содержит около 500 позиций, 500 переходов, 1500 дуг и моделирует динамику заданной ингибиторной сети за время $o(k^2 / \log k)$ с размером памяти $o(k)$, где k — количество запущенных переходов.

Возможно построение универсальных сетей и в других классах сетей Петри (приоритетных, синхронных, временных), являющихся универсальной алгоритмической системой. Кроме того, возможны комбинированные построения, например ингибиторной сети, выполняющей произвольную синхронную сеть.

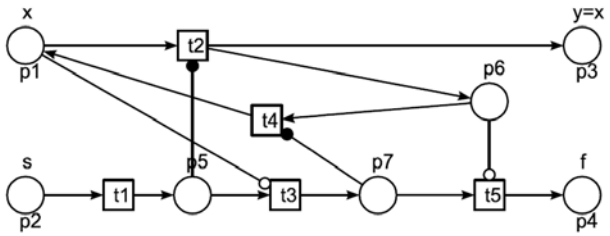
Примеры построения универсальных машин Тьюринга с минимальным количеством использованных символов/состояний даны в работах [9, 10]. В этой связи определенный интерес представляет построение универсальной сети Петри с минимальным количеством позиций (переходов), минимальным значением маркировки.

ПРИЛОЖЕНИЕ А. РЕАЛИЗАЦИЯ ИСПОЛЬЗОВАННЫХ ОПЕРАЦИЙ

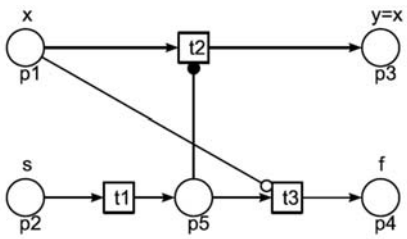
1. Очистить переменную $(x = 0)$
CLEAN::



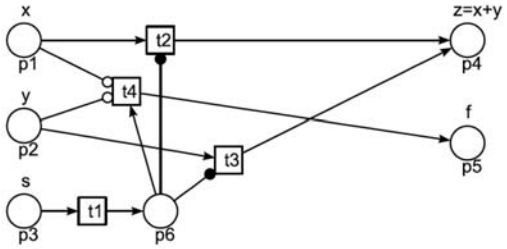
2. Копировать значение переменной $(y = x)$
COPY::



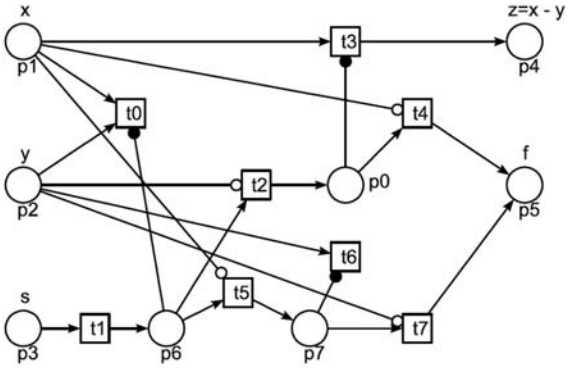
3. Переместить значение переменной $(y = x,\ x = 0)$
MOVE::



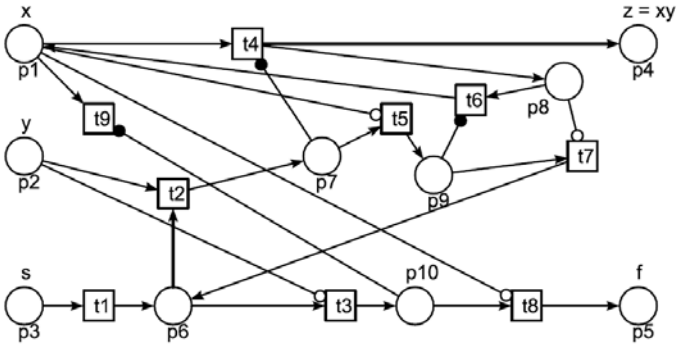
4. Сложение $(z = x + y)$
ADD::



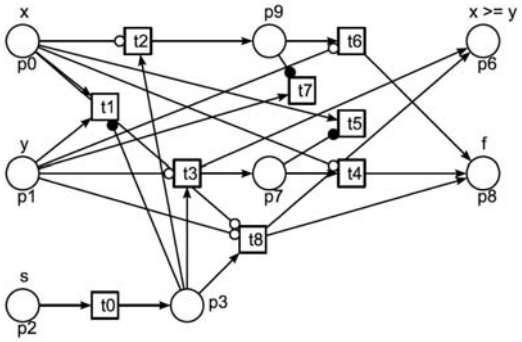
5. Вычитание $(z = x \dot{-} y)$: условное вычитание $z = \begin{cases} x - y, x \geq y \\ 0, x < y \end{cases}$
SUB::



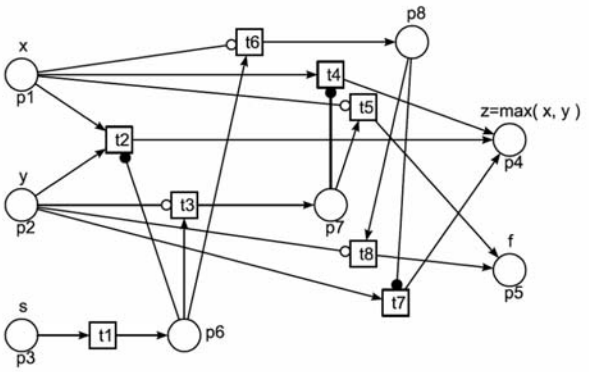
6. Умножение $(z = x \cdot y)$
MUL::



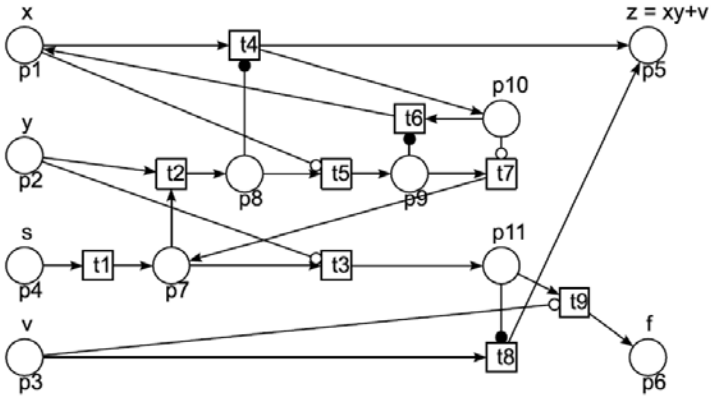
7. Сравнение $(x \geq y)$
GTE::



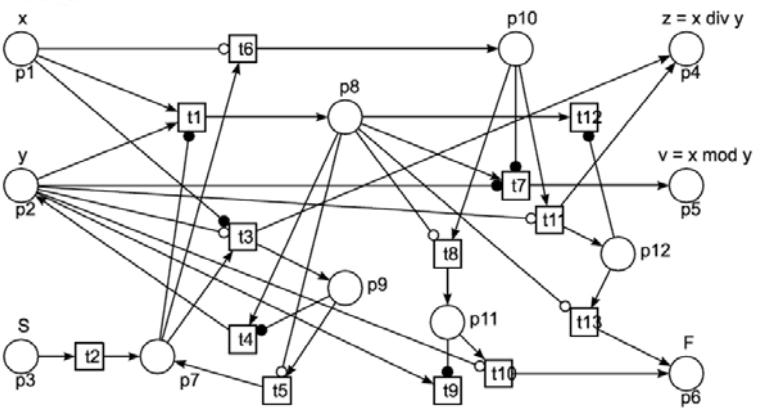
8. Максимум $(z = \max(x, y))$
MAX::



9. Добавить к шифру
MUL_ADD::



10. Извлечь из шифра $(z = x \operatorname{div} y,\ v = x \bmod y)$
MOD_DIV::



ПРИЛОЖЕНИЕ Б. ПРИМЕРЫ ШИФРОВАНИЯ СЕТЕЙ

Таблица 1. Шифрование графа сети

Сеть	*m*	*n*	*sB*	*rB*	*sD*	*rD*
add	6	4	21180169496	3	282946	2
max	8	8	254813592433189871074065241412	3	293862152152879368	2
mul	10	9	64654907206110145566888903466348174395265 4	3	19352259085292454555975681	2

Таблица 2. Шифрование маркировки

Сеть	**Маркировка**	*Q*	*sQ*	*rQ*
add	addQ0	(2,3,1,0,0,0)	2880	4
add	addQ	(0,0,0,5,1,0)	186	6
max	maxQ0	(2,3,1,0,0,0,0,0)	46080	4
max	maxQ	(0,0,0,3,1,0,0,0)	832	4
mul	mulQ0	(2,3,1,0,0,0,0,0,0,0)	737280	4
mul	mulQ	(0,0,0,6,1,0,0,0,0,0)	722701	7

Таблица 3. Шифрование последовательности срабатывания переходов

Сеть	*Q*0	*Q*	*Z*	*sZ*	*k*
add	addQ0	addQ	t1,t3,t2,t2,t3,t3,t4	2411	7
max	maxQ0	maxQ	t1,t2,t2,t6,t7,t8	4983	6
mul	mulQ0	mulQ	t1,t2,t4,t4,t5,t6,t6,t7,t2,t4,t4,t5,t6,t6, t7,t2,t4,t4,t5,t6,t6,t7,t3,t9,t9,t8	109815712212339723705298	26

СПИСОК ЛИТЕРАТУРЫ

1. Agerwala T. A complete model for representing the coordination of asynchronous processes. — Baltimore: John Hopkins University, Hopkins Computer Science Program, Res. Rep., No. 32, July 1974. — 58 p.
2. Котов В.Е. Сети Петри. — М.: Наука, 1984. — 160 с.
3. The universal Turing machine. A half-century survey / Rolf Herken (ed.), Wien; New York: Springer-Verlag, 1994. — 609 p.
4. 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.
5. Слепцов А.И., Юрасов А.А. Автоматизация проектирования управляющих систем гибких автоматизированных производств / Под ред. Б.Н. Малиновского. — К.: Техніка, 1986. — 160 с.
6. Слепцов А.И. Уравнения состояний и эквивалентные преобразования нагруженных сетей Петри (алгебраический подход) // Формальные модели параллельных вычислений: Докл. и сообщ. Всесоюз. конф. — Новосибирск, 1988. — С. 151–158.
7. Зайцев Д.А. Композиционный анализ сетей Петри // Кибернетика и системный анализ. — 2006. — № 1. — С. 143–154.
8. Zaitsev D.A. Complexity of universal inhibitor Petri net // Proc. of the 18th German Workshop on Algorithms and Tools for Petri Nets, Hagen, Germany, September 29–30, 2011. — P. 62–71.
9. Minsky M. Size and structure of universal Turing machines using tag systems / Recursive Function Theory, Symposium in Pure Mathematics, **5**, AMS, Provelence, 1962. — P. 229–238.
10. Rogozhin Y. Small universal Turing machines // TCS. — 1996. — **168**(2). — P. 215–240.

Поступила 18.03.2010