УДК 519.16

# Алгоритм структурирования сетей потоков работ на основе минимизации числа ИЛИ-слияний

Е. Е. Сурикова
Ярославский государственный университет им. П. Г. Демидова
E-mail: helen.410@mail.ru

**Аннотация**

Исследована проблема повышения обратимости сети потока работ (workflow). Разработан алгоритм структурирования сети на основе минимизации числа ИЛИ-слияний на промежуточных вершинах. Проведён анализ полученного алгоритма и возможностей его применения.

*Ключевые слова:* сети Петри, потоки работ, ИЛИ-слияние, обратимость, структурирование.

## Введение

Поток работ (англ. workflow) [1; 2] — графическое представление потока задач в процессе и связанных с ним подпроцессов, включая последовательность решений и работ. Потоки работ используются для формализации управления всевозможными технологическими процессами, бизнес-процессами и т. д.

Для изображения потока работ используют блок-схему или граф, который состоит из позиций, переходов, стрелок. Разветвления блок-схемы имеют логические символы «и», «или». Стрелки используют для отображения последовательности выполнения операций или потока объектов (документы, ресурсы). В потоках работ возможны циклы. Однако есть и ограничения, в частности, существуют одно начальное и одно конечное состояния.

Для формального описания потока работ используют специальный подкласс сетей Петри [3; 4] — так называемые сети потоков работ (WF-сети).

В данной работе исследуется проблема автоматической оптимизации потоков работ для повышения уровня обратимости сети, в частности, автоматическое структурирование на основе минимизации числа промежуточных ИЛИ-слияний, то есть уменьшения количества промежуточных (не финальных) позиций, имеющих более

одной входной дуги. Показано, что не от всех ИЛИ-слияний можно избавиться без повреждения языка сети. Представлен эффективный алгоритм, позволяющий выполнить максимально возможное структурирование.

Практическая значимость алгоритма состоит в следующем:

- В модифицированной сети повышается её обратимость: больше возможностей выполнить «откат» в случае сбоя на поздних этапах выполнения процесса.
- Появляется возможность использования принципа разделения ответственности между исполнителями, поскольку становится проще отслеживать историю.
- Появляется возможность эффективного предварительного распределения альтернативных ветвей (подпроцессов) между различными исполнителями.

**Основные определения**

*Сетью Петри* называется набор $N = (P, T, F)$, где $P$ — конечное множество позиций; $T$ — конечное множество переходов, $P \cap T = \emptyset$; $F : (P \times T) \cup (T \times P) \to Nat$ — функция инцидентности.

Графически сеть Петри изображается как двудольный ориентированный граф. Вершины-позиции изображаются кружками и характеризуют локальные состояния сети, вершины-переходы изображаются прямоугольниками и соответствуют действиям моделируемой системы. Дуги в графе соответствуют элементам $F$.

*Разметкой* (состоянием) сети $N$ называется функция вида $M : P \to Nat$, сопоставляющая каждой позиции сети некоторое натуральное число или ноль.

Графически разметка изображается при помощи маркеров (называемых «фишками») — черных точек внутри позиций. При разметке $M$ в каждую позицию $p$ помещается ровно $M(p)$ фишек. Если не хватает места на рисунке, то вместо точек в позиции рисуется число $M(p)$.

Для перехода $t \in T$ через ${}^\bullet t$ и $t^\bullet$ обозначим мультимножества его входных и выходных позиций, такие, что $\forall p \in P \quad {}^\bullet t(p) =_{def} F(p,t),\ t^\bullet(p) =_{def} F(t,p)$.

Переход $t \in T$ *готов к срабатыванию* при разметке $M$, если ${}^\bullet t \subseteq M$ (входные позиции содержат нужное число фишек).

Готовый к срабатыванию переход $t$ может *сработать*, порождая новую разметку $M' =_{def} M - {}^\bullet t + t^\bullet$ (обозн. $M \xrightarrow{t} M'$).

Фишки, находящиеся в той или иной позиции, моделируют наличие в системе того или иного ресурса, используемого или порождаемого при срабатывании переходов.

Важная область применения сетей Петри — моделирование потоков работ. Поток работ (англ. workflow) — графическое представление потока задач в процессе и связанных с ним подпроцессов, включая специфические работы, информационные зависимости и последовательность решений и работ. Потоки работ используются для формализации управления всевозможными технологическими процессами, бизнес-процессами, web-сервисами, распределенными вычислениями и т. д. В потоках работ возможны циклы, также возможно распараллеливание и синхронизация. Однако есть и ограничения, в частности, существуют одно начальное и одно конечное состояния, запрещены тупики.

Сеть Петри $N = (P, T, F)$ называется *WF-сетью* (сетью потока работ), если

1) в множестве $P$ имеются две специальные позиции $i$ и $o$, такие, что ${}^\bullet i = o^\bullet = \emptyset$;
2) любой элемент множества $P \cup T$ лежит на пути из $i$ в $o$,

Позиция $i$ называется *начальной*, а позиция $o$ — *финальной* позицией сети $N$. Начальная разметка сети потока работ состоит из одной фишки в позиции $i$ (и обозначается как $i$).

Важнейшим понятием в теории параллелизма является эквивалентность поведения двух систем. Существуют различные виды эквивалентности.

Мы рассматриваем формальные языки, распознаваемые сетями потоков работ (терминальные языки сетей Петри). При этом распознаваемыми символами являются буквы, приписанные переходам (имена действий), а начальной и терминальной разметками — разметки $i$ и $o$ соответственно.

Известно, что если две сети потоков работ равны (изоморфны их графы), то их терминальные языки тоже равны. Обратно же неверно. В дальнейшем под эквивалентностью сетей потоков работ мы понимаем равенство их языков.

### Проблема расщепления (структурирования) сети

Workflow-процессы необходимо эффективно выполнять в любых условиях, в том числе и достаточно нестабильных. Поэтому очень

желательной является возможность вернуться назад на несколько шагов (сделать откат) в случае возникновения каких-либо сбоев.

Для реализации такой возможности необходимо сделать WF-сеть обратимой, то есть сохраняющей историю процесса. При этом полученная сеть должна быть эквивалентна исходной (с точки зрения распознаваемого языка).

Очевидно, что не все сети можно сделать полностью обратимыми. Если в сети присутствует цикл, то соответствующее ИЛИ-слияние всегда приводит к сомнениям при движении по схеме снизу-вверх. В то же время мы можем попытаться «расщепить» обычное (не принадлежащее циклу) ИЛИ-слияние на несколько веток, что при движении снизу-вверх приведет к отсутствию спорных моментов. Такая оптимизация схемы значительно облегчит сопровождение процесса, повысит уровень обратимости сети и частично решит проблему «отката» при возникновении сбоя.

Несмотря на то, что задача имеет достаточно естественную интерпретацию, нам не удалось найти в литературе точного описания алгоритма её решения. Вероятно, это можно объяснить тем, что сети потоков работ представляют собой достаточно специфический подвид двудольных ориентированных графов.

**Алгоритм структурирования**

Входные данные: workflow–сеть, заданная как:

- массив дуг $D = [[1, -1], [1, -2], [-2, 1], [2, -3], [-2, 3] \dots]$, состоящий из пар (позиция, переход), отрицательное значение — входит, положительное — выходит;
- $P$ — массив позиций;
- $T$ — массив переходов.

Важные пояснения:

- все вершины пронумерованы от 0 до $n$, где 0 — это вершина $i$, а $n$ — вершина $o$;
- вершины нумеруются слоями по уровням (сначала первый уровень, потом второй и т. д.);
- данный алгоритм проходит только по позициям сети.

Также необходимо учесть, что не все сети можно сделать полностью обратимыми. Если в схеме есть цикл, то это ИЛИ-слияние невозможно удалить

Дополнительные функции:

- **copy** (вершина, с которой начинается дублирование; вершина, которой заканчивается дублирование; вершина для которой происходит дублирование) — дублирует путь;
- **delete** (вершина, с которой начинается удаление; вершина, которой заканчивается удаление) — удаляет путь из WF-сети;
- **reachability** (D[j]) — возвращает true, если входящая дуга $D[j]$ приводит к циклу через непомеченную вершину (в МассивПосещений) или без вершины (цикл, состоящий только из перехода), иначе возвращает false.

Пояснения к массивам и переменным:

- МассивПосещений — содержит 1, если посетили вершину и 0 в противном случае. Изначально состоит только из нулей. Размерности $n$.
- МассивНаличияИЛИСлияний — содержит 1, если в вершине содержится ИЛИ-слияние, 0 в противном случае. Изначально состоит только из нулей. Размерности $n$.
- МассивПНОЦ — содержит индексы переходов, не образующих циклы.
- МПП — МеткаПовторногоПрохождения (0 или 1).

Ниже приведён сам алгоритм, записанный на псевдокоде.

```
МПП ← 1
while МПП = 1 do
  МПП ← 0
  МассивПосещений ← []
  МассивНаличияИЛИСлияний ← []
  {Формируем МассивНаличияИЛИСлияний}
  i ← 0
  while i < n − 1 do
    j ← 0
    sum ← 0
    while j < length(D) do
      if D[j][0] = (−1) ∗ i then
        sum ← sum + 1
      end if
      j ← j + 1
    end while
    if sum > 1 then
      МассивНаличияИЛИСлияний[i] ← 1
```

```
    end if
    i ← i + 1
  end while
  {Проверяем наличие или-слияний в WF-сети}
  i ← 0
  sum ← 0
  while i < n − 1 do
    sum ← sum + МассивНаличияИЛИСлияний[i]
    i ← i + 1
  end while
  if sum = 0 then
    КОНЕЦ алгоритма
  end if
  {Проходим по вершинам, выясняя необходимость изменений}
  i ← 0
  while i < n − 1 do
    if МассивНаличияИЛИСлияний[i] = 1 then
      {Проверяем или-слияния на возможность удаления}
      j ← 0
      МассивПНОЦ ← []
      while j < length(D) do
        if D[j][0] = (−1) ∗ i and reachability(D[j]) = false then
          МассивПНОЦ[j] ← D[j][1]
        end if
        j ← j + 1
      end while
      j ← 0
      sum ← 0
      while j < length(МассивПНОЦ) do
        j ← j + 1
        sum ← sum + 1
      end while
      if sum > 1 then
        МПП ← МПП + 1
        {Создаем новую вершину l}
        {Дублируем все пути}
        copy(i, o, l)
        delete(i − 1, i)
        МассивПосещений[i] ← 1
```

```
            МассивПосещений[l] ← 1
            МассивНаличияИЛИСлияний[i] ← 0
            МассивНаличияИЛИСлияний[l] ← 0
          else
            МассивНаличияИЛИСлияний[i] ← 0
          end if
        else
          МассивПосещений[i] ← 1
          i ← i + 1
        end if
      end while
    end while
```

В полученном алгоритме получилось 7 циклов. Число элементов в каждом цикле не превосходит число элементов сети (позиций, переходов, дуг). Поэтому алгоритма полиномиален. Точнее, трудоемкость алгоритма — $O(n^3m)$, где $n$ — число позиций, $m$ — число переходов.

**Заключение**

Представлен алгоритм, который расщепляет максимально возможное число ИЛИ-слияний в произвольной сети потоков работ, сохраняя при этом её наблюдаемое поведение (язык). Применение данного алгоритма поможет решить множество спорных ситуаций, связанных со сбоями в сети, и даст возможность более широко использовать принцип разделения ответственности при управлении процессами.

**Список литературы**

1. *Аалст ван дер В.* Управление потоками работ: модели, методы и системы. М. : Научный мир, 2007. 320 с.
2. *Башкин В. А.* Модели потоков работ. Ярославль : ЯрГУ, 2009. 43 с.
3. *Башкин В. А.*, *Ломазова И. А.* Эквивалентность ресурсов в сетях Петри. М. : Научный мир, 2007. 208 с.
4. *Котов В. Е.* Сети Петри. М. : Наука, 1984. 160 с.