УДК 004-027.21

ПРИМЕНЕНИЕ СЕМАНТИЧЕСКИХ СЕТЕЙ ПЕТРИ-МАРКОВА ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАСПАРАЛЛЕЛИВАНИЯ АЛГОРИТМОВ

А. Н. Ивутин, заведующий кафедрой ВТ ТулГУ, к.т.н., доцент; alexey.ivutin@gmail.com
А. Г. Трошина, доцент кафедры ВТ ТулГУ, к.т.н.; atroshina@mail.ru
Д. О. Есиков, аспирант ТулГУ; mcgeen4@gmail.com

Целью работы *является создание единого математического аппарата и методов для решения задачи оптимального распараллеливания алгоритмов в тех практических ситуациях, когда параллельные процессы реализуются в вычислительной системе с известным быстродействием, объемом памяти, каналами связи и количеством процессоров. Рассматривается задача разработки и применения математического аппарата семантических сетей Петри-Маркова для оптимизации временной вычислительной сложности алгоритмов за счет их распараллеливания с учетом контекстной зависимости операторов. Впервые предложено сформировать понятие семантической сети Петри-Маркова (ССПМ) для моделирования контекстно-зависимых связей в алгоритмах, что позволит проводить структурное распараллеливание вычислительного процесса с сохранением логики взаимодействия отдельных операторов.*

Ключевые слова: *параллельное программирование, сети Петри-Маркова, семантические связи, полумарковский процесс, моделирование, функция перехода, технологии распараллеливания, семантическая сеть.*

DOI: 10.21667/1995-4565-2016-58-4-49-56

Введение

Развитие вычислительной техники как в области аппаратного, так и программного обеспечения связано с постоянным повышением быстродействия работы на ЭВМ. На начальных этапах повышение быстродействия и производительности достигалось за счет увеличения тактовой частоты. Сейчас основным направлением повышения быстродействия является использование параллельных архитектур ЭВМ и эффективных методов распараллеливания исполняемых программ.

Проблема эффективности распараллеливания алгоритмов заключается в привязке параллельного кода к определенному типу архитектуры ЭВМ. Так, например, использование библиотек MPI [1-3] более эффективно для систем с распределенной памятью, а OpenMP [4] для систем с общей памятью. Существенным недостатком MPI как систем, основанных на модели передачи сообщений, является недостаточная масштабируемость параллельных программ, а также сравнительно высокие расходы на передачу сообщений. Сложностью реализации OpenMP систем является необходимость анализа и разработки эффективного параллельного кода самим программистом. Использование системы PVM [5] позволяет пользователям использовать существующие аппаратные средства для решения достаточно сложных задач. Однако эффективное программирование для PVM требует значительной адаптации алгоритма к составу PVM и к ее характеристикам, а также решения задачи управления вычислительным процессом.

Кроме перечисленных, широкое распространение получили системы, разработанные как расширения существующих языков программирования высокого уровня (в основном С и Фортран): DVM [6-8], mpC [9], HPF [7] и пр., специализированные языки параллельного программирования: НОРМА [10, 11], ABCL [12], Linda [13], Parallaxis [14, 15], Sisal [16]. В последнее время появились системы, основанные на парадигме функционального программирования (Т-система [17]). Многие функциональные языки (Erlang [18], Haskell [19]) в той или иной мере поддерживают параллельные вычисления. Однако в этих системах следует тщательно продумывать элементы параллелизма.

Для повышения эффективности синтеза параллельных алгоритмов важным этапом является построение модели параллельного алгоритма, которая позволяет проверить принятые решения, оценить основные характеристики параллельного алгоритма, а также выбрать оптимальный вариант параллельного кода по критерию вычислительной сложности и минимизации времени исполнения.

Протекание любого вычислительного процесса подчиняется определенным правилам и условиям, направленным на достижение конкретной, заданной алгоритмом, цели. Эта цель достигается реализацией некоторой, заданной правилами, последовательности типовых операций. Выполнение типовой операции требует некоторого ресурса на входе, после обработки которого на выходе появляется другой ресурс. При этом зачастую одного и того же результата можно добиться с помощью разного набора некоторых типовых операций при выполнении условия эквивалентности входных ресурсов и эквивалентности выходных ресурсов эталонного (заданного) и модифицированного (альтернативного) вариантов обработки. Этот факт позволяет говорить о семантике операции и семантической эквивалентности операторов (группы операторов).

При разработке параллельного кода на основании последовательной программы, необходимо рассмотреть и проанализировать информационные взаимосвязи процесса вычислений и на основании таких взаимосвязей сгенерировать новый код.

Наибольшей наглядностью обладают методы графического моделирования параллельного алгоритма. Использование графов для представления параллельных процессов позволяет наглядно показать участки, которые выполняются параллельно. Распространено несколько способов визуализации параллелизма – это диаграммы потоков управления (ДПУ), диаграммы потоков данных (ДПД), системы перерисовки графов, методы описания взаимодействия объектов, а также классические DFD-диаграммы, CASE-технологий, UML, сети Петри.

Преимуществом использования сетей Петри-Маркова является то, что они не только описывают поведение системы на событийном уровне, но и предоставляют математический аппарат для оценки временных характеристик процесса. Существенным недостатком таких сетей является отсутствие учета семантических связей между позициями, что не позволяет строить эффективные распределенные схемы реализации процессов. Ликвидировать этот недостаток призваны семантические сети Петри-Маркова [20-22].

Теоретические исследования

Семантической сетью Петри-Маркова (ССПМ) называется структурно-параметрическая модель, заданная множеством:

$$\Psi = \{\Pi, M\}. \qquad (1)$$

где $\Pi = \{A, \{Z^C, \tilde{R}^C, \hat{R}^C\}, \{Z^S, \tilde{R}^S, \hat{R}^S\}$ – множество, описывающее структуру трехдольного ориентированного графа, представляющего собой сеть Петри, $A = \{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)}\}$ – конечное множество позиций; $Z^C = \{z^C_{1(Z^C)}, ..., z^C_{j(Z^C)}, ..., z^C_{J(Z^C)}\}$ – конечное множество переходов по управлению; $Z^S = \{z^S_{1(Z^S)}, ..., z^S_{j(Z^S)}, ..., z^S_{J(Z^S)}\}$ – конечное множество переходов по семантическим связям; $\tilde{R}^C = (\tilde{r}^C_{j(a)j(z^C)})$ – матрица смежности размером $J(a) \times J(z^C)$ отображающая множество позиций в множество переходов по управлению; $\hat{R}^C = (\hat{r}_{j(z^C)j(a)})$ – матрица смежности размером $J(z^C) \times J(a)$, отображающая множество переходов по управлению в множество позиций; $\tilde{R}^S = (\tilde{r}^S_{j(a)j(z^S)})$ – матрица смежности размером $J(a) \times J(z^S)$, отображающая множество позиций в множество переходов по семантическим связям; $\hat{R}^S = (\hat{r}_{j(z^S)j(a)})$ – матрица смежности размером $J(z^S) \times J(a)$, отображающая множество переходов по семантическим связям в множество позиций; $M = \{q, h(t), \Lambda\}$ – параметры, накладываемые на структуру Π и определяющие временные, вероятностные и логические характеристики ССПМ по управлению; $q = \{q_{1(z)}, ..., q_{j(z)}, ..., q_{J(z)}\}$ – вектор, определяющий вероятность начала процесса в одном из переходов множества Z; $h(t) = (h_{j(a)j(z^C)}(t))$ – полумарковская матрица размером $J(a) \times J(z^C)$; t – время; $\Lambda = (\lambda_{i(z)i(a)})$ – матрица логических условий размером $J(a) \times J(z^C)$; $I_A(z^C) = \left\{I_A(z^C_{1(Z^C)}), ..., I_A(z^C_{j(Z^C)}), ..., I_A(z^C_{J(Z^C)})\right\}$ и $O_A(z^C) = \left\{I_A(z^C_{1(Z^C)}), ..., I_A(z^C_{j(Z^C)}), ..., I_A(z^C_{J(Z^C)})\right\}$ – соответственно входная и выходная функции переходов по управлению;

$I_A(z^S) = \left\{I_A(z^S_{1(z^S)}), ..., I_A(z^S_{j(Z^S)}), ..., I_A(z^S_{J(Z^S)})\right\}$ и $O_A(z^S) = \left\{O_A(z^S_{1(z^S)}), ..., O_A(z^S_{j(Z^S)}), ..., O_A(z^S_{J(Z^S)})\right\}$ – соответственно входная и выходная функции переходов по семантическим связям.

$$\tilde{r}^S_{j(a)j(z^S)} = \begin{cases} 1, если & a_{j(a)} \in I_A(z_{j(z^C)}); \\ 0, если & a_{j(a)} \notin I_A(z_{j(z^C)}); \end{cases}$$

$$\tilde{r}^S_{j(a)j(z^S)} = \begin{cases} 1, если & a_{j(a)} \in I_A(z_{j(z^S)}); \\ 0, если & a_{j(a)} \notin I_A(z_{j(z^S)}); \end{cases}$$

$$\hat{r}^{C}_{j(z^{C})j(a)} = \begin{cases} 1, если \quad a_{j(a)} \in O_A(z_{j(z^{C})}); \\ 0, если \quad a_{j(a)} \notin O_A(z_{j(z^{C})}); \end{cases}$$

$$\hat{r}^{S}_{j(z^{S})j(a)} = \begin{cases} 1, если \quad a_{j(a)} \in O_A(z_{j(z^{S})}); \\ 0, если \quad a_{j(a)} \notin O_A(z_{j(z^{S})}); \end{cases}$$

$$h(t) = p \otimes f(t) = (p_{j(a)j(z^{C})}, f_{j(a)j(z^{C})}(t) = (h_{j(a)j(z^{C})}(t)), \quad (2)$$

$$\lambda_{j(z^{C})j(a)} = \begin{cases} \lambda\left[I_A(z_{j(z^{C})}) \right], если \quad a_{j(a)} \in O_A(z_{j(z^{C})}); \\ 0, если \quad a_{j(a)} \notin O_A(z_{j(z^{C})}); \end{cases} \quad (3)$$

$p = (p_{j(a)j(z^{C})})$ – матрица вероятностей;

$f(t) = (f_{j(a)j(z^{C})}(t))$ – матрица плотностей распределения.

Позиции ССПМ являются математическими объектами, моделирующими состояния элементов алгоритма (операторов).

Переход по управлению в ССПМ является математическим объектом, моделирующим процесс смены состояния одного или нескольких операторов. Переход по семантическим связям в ССПМ является математическим объектом, устанавливающим семантические ограничения на последовательность смены состояний одного или нескольких операторов.

Входная функция перехода по управлению $I_A(z^{C}_{j(z^{C})})$ описывает множество позиций, отображаемых в данный переход. Входная функция $I_A(z^{C}_{j(z^{C})})$ есть подмножество вершин, смежных с данным переходом, для которых $\tilde{r}^{C}_{j(a)j(z^{C})} = 1$. Входная функция перехода по семантическим связям $I_A(z^{S}_{j(z^{S})})$ описывает множество позиций, отображаемых в данный переход. Входная функция $I_A(z^{S}_{j(z^{S})})$ есть подмножество вершин, смежных с данным переходом, для которых $\tilde{r}^{S}_{j(a)j(z^{S})} = 1$.

Выходная функция перехода по управлению $O_A(z^{C}_{j(z^{C})})$ описывает множество позиций, отображаемых из данного перехода. Выходная функция $O_A(z^{C}_{j(z^{C})})$ есть подмножество вершин, смежных с данным переходом, для которых $\hat{r}^{C}_{j(z^{C})j(a)} = 1$. Выходная функция перехода по семантическим связям $O_A(z^{S}_{j(z^{S})}) = \{a_{j(a)}\}$ описывает единственную позицию $a_{j(a)}$, отображаемую из данного перехода. Выходная функция $O_A(z^{S}_{j(z^{S})})$ есть вершина, смежная с данным переходом, для которой $\hat{r}^{S}_{j(z^{S})j(a)} = 1$.

Физическим смыслом семантической связи является тот факт, что потребляемый позицией $a_{j(a)}$ ресурс вырабатывается в позициях, составляющий входную функцию перехода по семантическим связям, ведущего $a_{j(a)}$. Более того, переход по семантическим связям всегда ведет только в одну позицию, т.е.

$$\forall z^{S}_{j(z^{S})} \,|\, (z^{S}_{j(z^{S})} \in Z^{S}) \,|\, I_A(z^{S}_{j(z^{S})})| = n,$$

$$|\, O_A(z^{S}_{j(z^{S})})| = 1, \; n \geq 1;$$

$$\forall a_{j(a)} \,|\, (O_A(z^{S}_{j(z^{S})}) = a_{j(a)}, a_{j(a)} \in A, z^{S}_{j(z^{S})} \in Z^{S}),$$

$$a_{j(a)} \cap O_A(Z^{S}) = a_{j(a)}.$$

В качестве визуального отображения ССПМ можно использовать способ представления в виде ориентированных взвешенных трехдольных графов, как показано на рисунке 1, где переходы по управлению показаны чертой, а переходы по семантическим связям – треугольником.

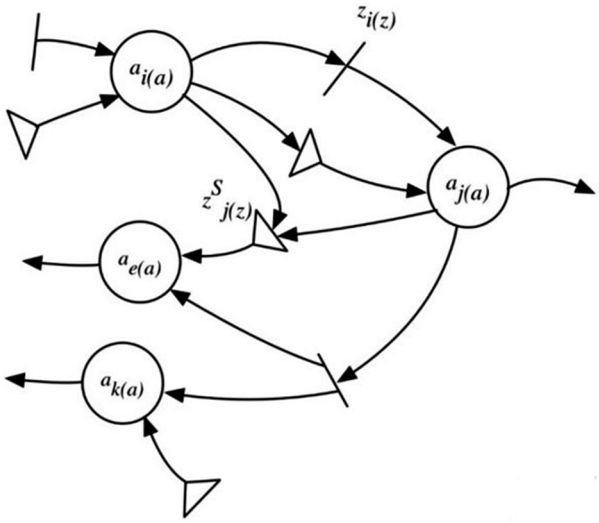

Рисунок 1 – Фрагмент семантической сети Петри-Маркова

Под примитивным переходом по управлению ССПМ понимают элемент структуры, когда мощности входной и выходной функций переходов равны 1, т.е.

$$\left| I_A(z^{C}_{j(z^{C})}) \right| = \left| O_A(z^{C}_{j(z^{C})}) \right| = 1. \quad (4)$$

Под примитивным переходом по семантическим связям ССПМ понимают элемент структуры, когда мощности входной и выходной функций переходов равны 1, т.е.

$$\left| I_A(z^{S}_{j(z^{S})}) \right| = \left| O_A(z^{S}_{j(z^{S})}) \right| = 1. \quad (5)$$

Все остальные переходы являются непримитивными. Непримитивные переходы (НП) образуют подмножества $z^C_{j(z^C)} \subset Z$ и $z^S_{j(z^S)} \subset Z$. Для них вводится специальная индексация $i(z^C n)$ и $i(z^S n)$. Среди множества непримитивных переходов могут быть выделены группа начальных, или стартовых переходов (start, $\left|I_A(z^X_{j(z^X)})\right| = 0, \left|O_A(z^X_{j(z)})\right| > 1$) и группа конечных, или поглощающих переходов (finish, $\left|I_A(z^X_{j(z^X)})\right| = 1, \left|O_A(z^X_{j(z)})\right| = 0$).

Для семантических связей не существует понятия стартового и поглощающего переходов. Физическим смыслом стартового перехода по семантическим связям является зависимость некоторого состояния $I_A(z^S_{j(z^S)}) = a_{i(a)}$ от исходных данных. Поскольку эти данные на момент начала процесса имеются в полном объеме, введение стартовых переходов по семантическим связям становится избыточным. Понятие поглощающего перехода по семантическим связям также является избыточным. На практике оно означает, что процесс переработки окончен и получен результат, который может быть использован для дальнейшей переработки. Однако для состояний возможен случай, когда оно не входит ни в одну входную функцию перехода по семантическим связям $a_{i(a)} \cup \{I_A(z^S_{j(z^S)})\} = 0$. Это означает, что полученный ресурс больше нигде не используется. Данная ситуация характерна для завершающих стадий обработки ресурса, т.е. для корректно спроектированных алгоритмов имеет место следующее утверждение:

Утверждение 1.

$$\forall a_{j(a)} \left| a_{j(a)} \cup \{I_A(z^S_{j(z^S)})\} = 0) \exists z^C_{j(E)} \in Z^C_E \right|$$
$$(a_{j(a)} \cup \{I_A(z^C_{j(E)})\} a_{j(a)}).$$

Для переходов по управлению к непримитивным также относятся:

fork, $\left|I_A(z^C_{j(z^C)})\right| = 1, \left|O_A(z^C_{j(z^C)})\right| > 1$, т.е. порождающая операция, которая описывает действие, в результате которого запускается один или несколько дополнительных процессов, выполняющихся параллельно.

join, $\left|I_A(z^C_{j(z^C)})\right| > 1, \left|O_A(z^C_{j(z^C)})\right| = 1$, т.е. поглощающая операция, которая описывает действие, в результате которого один или несколько параллельных процессов объединяются в один.

synchro, $\left|I_A(z^C_{j(z^C)})\right| = n, \left|O_A(z^C_{j(z^C)})\right| = m, n > m$,

т.е. синхронизирующая операция, которая характеризует действие, в результате которого два и более параллельных процессов ожидают завершения друг друга, после чего их независимое выполнение продолжается.

Для переходов по семантическим связям к непримитивным также относятся:

s-join, $\left|I_A(z^C_{j(z^C)})\right| > 1, \left|O_A(z^C_{j(z^C)})\right| = 1$.

Наличие s-join-перехода означает семантическую связь позиции с несколькими другими позициями, причем количество таких зависимостей определяется мощностью множества $\left|I_A(z^S_{j(z^S)})\right|$.

Важным понятием аппарата ССПМ при моделировании является понятие фишки. Под фишкой понимается некоторый указатель того, что в текущий момент времени *t* некоторая позиция $a_{j(a)}$ находится в активном состоянии.

Активное состояние позиции моделирует состояние системы при интерпретации заданного алгоритма. Состоянием называется то физическое действие, которое система выполняет в данный момент, что связано с пребыванием фишки в соответствующих позициях. Сменой физического состояния является переход по управлению от одного физического состояния к другому и моделируется изъятием фишки из одной позиции $I_A(z^C_{j(z^C)})$ и помещением ее в позиции $O_A(z^C_{j(z^C)})$, причем переход разрешен, если выражение (3) не равно 0. Данное условие является еще одним расширением классического аппарата сетей Петри, где переход разрешен, если каждая из его входных позиций имеет число фишек большее или равное числу входов в переход из рассматриваемой позиции.

Функциональным подобием процесса решения задачи в вычислительной системе является последовательность перемещений, реализуемая в виде полушагов по управлению семантической сети Петри-Маркова. Полушагом $\sigma_{i(a),i(z^C)}$ или $\sigma_{i(z^C),i(a)}$ называется перемещение по ССПМ, при котором из позиции $a_{i(a)}$ попадают в сопряженный с ней переход $z^C_{i(z^C)}$ или из перехода $z^C_{i(z^C)}$ попадают в сопряженную с ним позицию $a_{i(a)}$ для случая переходов по управлению. Каждый полушаг может быть описан множеством мощностью два:

$$\sigma_{i(a),i(z^C)} = (a_{i(a)}, z^C_{i(z^C)});$$
$$\sigma_{j(z^C),i(a)} = (z^C_{j(z^C)}, a_{j(a)}).$$

Два последовательных полушага образуют шаг. Результат выполнения шага определяется перемножением матриц смежности. Позиция

отображается в позицию, если $R_P^C = \tilde{R}^C \times \hat{R}^C$. Переход отображается в переход, если $R_T^C = \hat{R}^C \times \tilde{R}^C$.

Следует отметить, что введение понятия полушага по семантическим связям не имеет физического смысла, поскольку семантические связи описывают зависимость состояния от ресурса, вырабатываемого другими позициями вне зависимости от того в активном или не активном состоянии находится в текущий момент та или иная позиция.

Утверждение 2. Время исполнения примитивного перехода по управлению описывается δ-функцией.

Важным свойством сетей Петри для моделирования реальных систем, не рассмотренное для ССПМ, является свойство ограниченности (и, соответственно, частный случай этого свойства – безопасность). Это структурный уровень описания, указывающий на ограничения аппаратного обеспечения. Ввести понятие подсети и рассматривать ограниченность и безопасность в контексте безопасности подсети.

ССПМ, описывающая типовой процесс переработки ресурса некоторой системой, является k-ограниченной. Причем, в случае описания алгоритма (в том числе параллельного) такая сеть является 1-ограниченной. Это означает в том числе, что каждой операции fork в ССПМ обязательно должна соответствовать парная операция joint, а также выполняться следующее свойство:

$$\forall a_j(a) \in O_A(z^C_{j(z^C)}), \left| O_A(z^C_{j(z^C)}) = \{a_{j(a)}\} \right| = 1 .$$

Процесс функционирования вычислительной системы может быть представлен в виде последовательности смен состояний при реализации переходов по управлению. Каждая смена состояний характеризуется временем выполнения, вероятностью выполнения, исходным перерабатываемым ресурсом, результатом переработки ресурса. В качестве ресурса может выступать информация, физический объект. Последовательности состояний могут быть выстроены в одну из следующих структур: параллельную, линейную или с ветвлениями. Каждый компонент вычислительной системы может быть разбит на такие элементы, последовательность состояний которых включает только примитивные переходы, а также линейные участки и участки с ветвлением.

Определение 1. Линейная структура по управлению характеризует последовательное выполнение некоторого действия сразу после окончания выполнения предыдущего, т.е. последовательную смену состояний и определяется следующими зависимостями:

$$\begin{gathered} a_{i(a)} \in I_A(z^C_{j(z^C)}), a_{j(a)} \in O_A(z^C_{j(z^C)}); \\ \left| I_A(z^C_{j(z^C)}) \right| = \left| O_A(z^C_{j(z^C)}) \right| = 1; \\ a_{i(a)} \cup \{I_A(z^C_{j(z^C)})\} = 1, a_{j(a)} \cup \{O_A(z^C_{j(z^C)})\} = 1. \end{gathered} \quad (6)$$

Пример линейного участка приведен на рисунке 2.

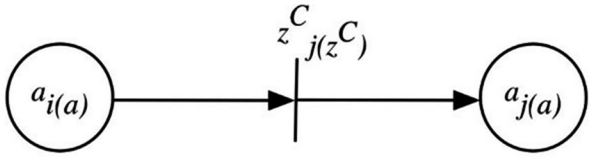

Рисунок 2 – Линейная структура по управлению

Структура с ветвлением по управлению описывает выполнение некоторого действия по условию, причем количество вариантов действий может варьироваться от 2 до n.

Среди вариантов структур с ветвлением по управлению следует выделить branch-позиции (рисунок 3), когда позиция входит во входные функции двух и более переходов по управлению, т.е. $a_{i(a)} \in I_A(z^C_{i(z^C)})$ и $a_{i(a)} \in I_A(z^C_{i(z^C)})$ при $z^C_{i(z^C)} \neq z^C_{j(z^C)}$.

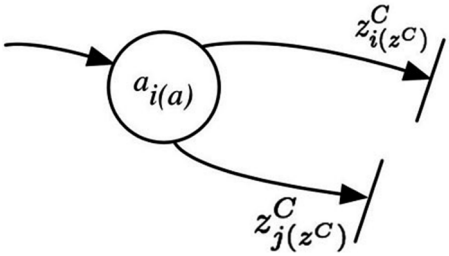

Рисунок 3 – Фрагмент семантической сети Петри-Маркова с branch-позицией

Branch-позиции описывают классические схемы альтернативного развития алгоритма управления, внося дополнительные сложности в процедуры анализа протекания информационных процессов. Следует отметить, что зачастую в branch-позициях переходам по управлению будут соответствовать и переходы по семантическим связям (однако это условие не является необходимым).

Определение 2. Семантическая подсеть Петри-Маркова со структурой:

$$\Pi_{i(n)} = \{A_{i(n)}, \{Z^C_{i(n)}, \tilde{R}^C_{i(n)}, \hat{R}^C_{i(n)}\}, \{Z^S_{i(n)}, \tilde{R}^S_{i(n)}, \hat{R}^S_{i(n)}\}\}$$

называется связной подсетью, если для двух произвольных вершин подсети из множества $A_{j(n)} \cup Z^C_{j(n)}$ существует хотя бы один путь.

Определение 3. Связная подсеть со структурой $\Pi_{i(n)} = \{A_{i(n)}, \{Z^C_{i(n)}, \tilde{R}^C_{i(n)}, \hat{R}^C_{i(n)}\}, \{Z^S_{i(n)}, \tilde{R}^S_{i(n)}, \hat{R}^S_{i(n)}\}\}$, у

которой множество переходов $Z^C_{j(n)}$ разбивается на переходы $Z^C_{j(n),B}$, для которых:

$$I_A\left(z^C_{j[z^C,j(n),B]}\right)\cap A_{j(n)}=\varnothing;$$

$$\mu\left[I_A\left(z^C_{j[z^C,j(n),B]}\right)\cap A_{j(n)}\right]=1.$$

и переходы $Z^C_{j(n),E}$, для которых:

$$O_A(z^C_{j[z^C,j(n),B]})\cap A_{j(n)}=\varnothing;$$

$$\mu=[I_A(z^C_{j[z^C,j(n),B]})z^C_{j[z^C,j(n),B]}]=1.$$

переходы, для которых выполняются условия:

$$\mu[O_A(z^C_{j[z^C,j(n)]})\cap A_{j(n)}]=1;$$

$$\mu[I_A(z^C_{j[z^C,j(n)]})\cap A_{j(n)}]=1,$$

а множество переходов $Z^S_{j(n)}$ разбивается на переходы $Z^S_{j(n),B}$, для которых:

$$I_A(z^S_{j[z^S,j(n),B]})\cap A_{j(n)}=\varnothing;$$

$$\mu[I_A(z^S_{j[z^S,j(n),B]})\cap A_{j(n)}]=1;$$

$$I_A(z^S_{j[z^S,j(n),B]})\in\{I_A(z^C_{j[z^C,j(n),B]})\},$$

переходы $Z^S_{j(n)}$, для которых:

$$I_A(z^S_{j[z^S,j(n)]})\in A_{j(n)};$$

$$O_A(z^S_{j[z^S,j(n)]})\cap A_{j(n)}=\varnothing\,||\,O_A(z^S_{j[z^S,j(n)]})\in A_{j(n)},$$

называется элементарной семантической подсетью Петри-Маркова (ЭСППМ).

Следствие из определения 3. В ЭСППМ для семантических связей ограничение на примитивность переходов не накладывается.

Важной особенностью ЭСППМ является то, что благодаря свойству достижимости, они могут применяться для представления любых алгоритмов, при условии, что все его операторы существенны, т.е. достижимы из подмножества начальных операторов, а также из них достижимо подмножество конечных операторов.

С функциональной точки зрения подобием ЭСППМ будет являться процесс интерпретации алгоритма в некотором элементе без взаимодействия с другими. Тогда переходы $z^C_{j[z^C,j(n),B]}\in Z^C_{j(n),B}$ и $z^S_{j[z^S,j(n),B]}\in Z^S_{j(n),B}$ будут стартовыми переходами ЭСППМ $\Pi_{j(n)}$, моделирующими начало процесса в элементе, причем ограничение по входным семантическим переходам вполне естественно и имеет реальный физический смысл – никакая обработка ресурса не может быть начата до тех пор, пока нет всей необходимой входной информации. Переходы $z^C_{j[z^C,j(n),E]}\in Z^C_{j(n),E}$ будут конечными переходами ЭСППМ $\Pi_{j(n)}$, моделирующими завершение обработки в соответствии с заданным алгоритмом, а остальные переходы (по управлению) являются примитивными.

Из определения 3 следует, что непосредственно в ЭСППМ непримитивные переходы по управлению типа join, fork и synchro отсутствуют. Наличие только примитивных переходов свидетельствует о том, что структурами подобного рода можно представить только последовательные процессы обработки ресурса, происходящие в некотором функциональном элементе. Сказанное, однако, не означает, что непосредственно в ЭСППМ не существует возможностей по распараллеливанию путем некоторого преобразования структур. Такое преобразование, очевидно, приведет к тому, что преобразуемая подсеть перестанет существовать в качестве элементарной подсети и породит некоторое количество новых элементарных подсетей.

Общее число одновременно реализуемых ЭСППМ в данном элементе $J(n)$ будет равно числу исполнительных элементов в нем, а непосредственно ССПМ может быть представлена объединением элементарных подсетей:

$$\Pi=\bigcup_{j(n)=1}^{J(n)}\Pi_{j(n)}\,. \qquad (7)$$

Определение 4. Любая сколь угодно сложная система, описанная ССПМ, может без потери семантических связей быть преобразована таким образом, что будет представлять собой комбинацию ЭСППМ в соответствии с выражением (7).

Пример представления системы комбинацией ЭСППМ приведен на рисунке 4.

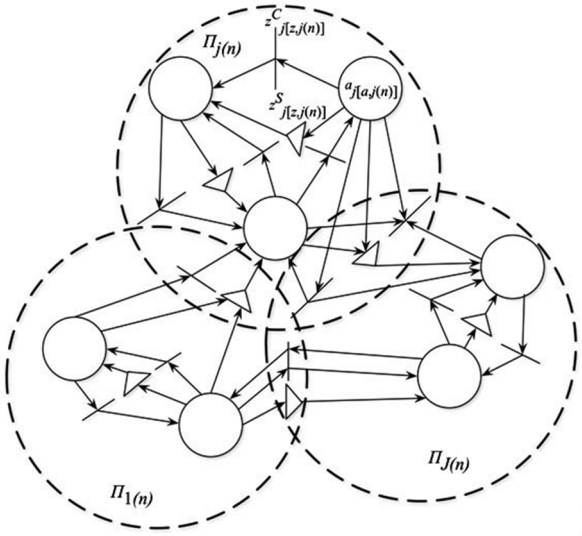

Рисунок 4 – ССПМ в виде комбинации ЭСППМ

Исполнение некоторого процесса характеризуется семантикой процесса, определяющей очередность (порядок) выполнения операций. В за-

висимости от определяемого семантикой контекста исполнения могут меняться непосредственные характеристики процесса, а именно, время выполнения отдельных операций, зависящее от трудозатрат на реализацию заданного действия, и вероятность исполнения операций, зависящая от функции распределения исходных данных для обработки.

Заключение

Новизна предлагаемого математического аппарата связана с использованием в классической теории сетей Петри марковских процессов, позволяющих решить проблему оценки временных характеристик процесса, и семантических связей, позволяющих гибко перестраивать процесс без изменения заложенной в алгоритме логики. Последнее особенно важно для случаев, когда сеть Петри строится в автоматическом режиме, например, при прямом преобразовании алгоритма или его мнемонического представления, а также в оптимизационных задачах, когда имеются ограничения по количеству исполнительных устройств, их характеристикам и другим возможностям.

Исследование выполнено при финансовой поддержке РФФИ и Правительства Тульской области в рамках научного проекта 16-41-710160 р_а.

Библиографический список

1. **Антонов А. С.** Технология программирования MPI: Учебное пособие. М.: Из-во МГУ, 2004. 71 с.

2. **Гришагин В. А., Свистунов А. Н.** Параллельное программирование на основе MPI. Н. Новгород: Изд-во Нижегор. гос. ун-та, 2005. 93 с.

3. **Корнеев В. Д.** Параллельное программирование в MPI. Новосибирск: Изд-во Сиб. отд. Рос. АН, 2000. 304 с.

4. OpenMP Application Programming Interface Version 4.5 November 2015. [Электронный ресурс]. URL: www.openmp.org/mp-documents/openmp-4.5.pdf (дата обращения: 01.11.2016).

5. PVM Parallel Virtual Machine. [Электронный ресурс]. URL: http://www.csm.ornl.gov/pvm/ (дата обращения: 01.11.2016)

6. **Коновалов Н. А. и др.** C-DVM – язык разработки мобильных параллельных программ. // Программирование. 1999. № 1. С. 54-65.

7. **Коновалов Н. А., Крюков В. А.** DVM-подход к разработке параллельных программ для вычислительных кластеров и сетей М.: ИПМ им. МВ Келдыша РАН, 2002. 42 с.

8. **Коновалов Н. А. и др.** Fortran DVM-язык разработки мобильных параллельных программ // Программирование. 1995. №. 1. С. 49-54.

9. The mpC Parallel Programming Environment: сайт. [Электронный ресурс]. URL: http://panda.ispras.ru/~mpc/ (дата обращения: 01.11.2016)

10. **Андрианов А. Н., Ефимкин К. Н., Задыхайло И. Б.** Язык Норма. Препринт ИПМ им. М. В.Келдыша АН СССР. № 165. 1985.

11. Система НОРМА: сайт. [Электронный ресурс]. URL: http://keldysh.ru/pages/norma (дата обращения: 01.11.2016).

12. The ABCL family of languages. [Электронный ресурс]. URL: http://web.yl.is.s.u-tokyo.ac.jp/pl/ abcl.html#overview (дата обращения: 02.11.2016).

13. Linda Introduction: сайт. [Электронный ресурс]. URL: http://web.archive.org/web/20080827 180428/ http://phi.sinica.edu.tw/instruct/workshop/ html/linda/ linda.html (дата обращения: 01.11.2016)

14. **Bräunl T.** Parallaxis-III: A language for structured data-parallel programming //Proceedings of the IEEE First International Conference on Algorithms and Architectures for Parallel Processing. pp. 43-52.

15.Parallaxis-III — A Structured Data-Parallel Programming Language. [Электронный ресурс]. URL: http://robotics.ee.uwa.edu.au/parallaxis/ (дата обращения: 01.11.2016)

16. Sisal Parallel Programming: сайт. [Электронный ресурс]. URL: https://sourceforge.net/ projects/sisal/

17. **Абрамов С. М., Кузнецов А. А., Роганов В. А.** Кроссплатформенная версия Т-системы с открытой архитектурой //Вычислительные методы и программирование. 2007. Т. 8. №. 1. С. 175-180.

18. Язык Erlang и программирование для мультиядерных процессоров. [Электронный ресурс]. URL: http://itc.ua/articles/yazyk_erlang_i_programmirovanie_dlya_multiyader nyh _processorov_26721/ (дата обращения: 01.11.2016).

19. **Казаков Ф. А., Легалов А. И.** Параллельное программирование в языках Haskell и Пифагор //Проблемы информатизации региона. ПИР-2001: Сб. науч. тр./ИПЦ КГТУ. Красноярск, 2002. С. 48-55.

20. **Ivutin A. N., Larkin E. V., Lutskov Y. I., Novikov A. S.** Simulation of concurrent process with Petri-Markov nets. // Life Sci J 2014; 11(11):506-511 (ISSN:1097-8135). http://www.lifesciencesite.com/lsj/life1111/086_25899life111114_506_511.pdf

21. **Ивутин А. Н., Дараган Е. И.** Теория сетей Петри и ее расширения // Известия ТулГУ. Серия: Технические науки. Вып. 10. Тула: Изд. ТулГУ, 2012. С. 211-220.

22. **Ивутин А. Н.** Оптимизация параллельных программ с использованием математического аппарата сетей Петри-Маркова // Известия ТулГУ. Серия: Технические науки. Вып. 12. Ч.2 Тула: Изд. ТулГУ, 2012. С. 249-255.

UDC 004-027.21

PARALLELIZATION OF ALGORITHMS WITH USE THE SEMANTIC PETRI-MARKOV NETS

A. N. Ivutin, PhD (technical sciences), associated professor, Head of the Department of Computer Technology, TSU, Tula; alexey.ivutin@gmail.com
A. G. Troshina, PhD (technical sciences), assistant professor of the Department of Computer Technology, TSU, Tula; atroshina@mail.ru
D. O. Yesikov, post-graduate student, TSU, Tula; mcgeen4@gmail.com

The aim of this work is *to create a unified mathematical apparatus and methods for solving the problems of optimal parallelization algorithms in practical situations where parallel processes are implemented in a computer system with known processing speed, memory volume, communication channels and the number of processors. The problem of the development and application of mathematical apparatus of se-mantic Petri-Markov nets to optimize the time computational complexity of algorithms by their parallelization taking into account contextual dependence of the operators is considered. For the first time we offer the notion of semantic Petri-Markov net (SPMN) for simulation of context-sensitive relations in the algorithms that will allow to make structural parallelization of computational process, preserving the logic of interaction of individual operators.*

Key words: *concurrent programming, Petri-Markov nets, semantic relations, semi-Markov process, simulation, transition function, paralleling technology, semantic net.*

DOI*:* 10.21667/1995-4565-2016-58-4-49-56

References

1. **Antonov A. S.** Tehnologija programmirovanija MPI: Uchebnoe posobie. M.: Iz-vo MGU, 2004, 71 p.
2. **Grishagin V. A., Svistunov A. N.** Parallel'noe programmirovanie na osnove MPI. N. Novgorod: Izd-vo Nizhegor. gos. un-ta, 2005, 93 p. (in Russian)
3. **Korneev V. D.** Parallel'noe programmirovanie v MPI. Novosibirsk: Izd-vo Sib. otd. Ros. AN, 2000, 304 p. (in Russian).
4. OpenMP Application Programming Interface Version 4.5 November 2015. [Electronic resource]. URL: www.openmp.org/mp-documents/openmp-4.5.pdf (date of access: 01.11.2016)
5. PVM Parallel Virtual Machine. [Electronic resource]. URL: http://www.csm.ornl.gov/pvm/(date of access: 01.11.2016)
6. **Konovalov N. A**. i dr. C-DVM-jazyk razrabotki mobil'nyh parallel'nyh programm. Programmirovanie, № 1, 1999, pp. 54-65. (in Russian)
7. **Konovalov N. A., Krjukov V. A.** DVM-podhod k razrabotke parallel'nyh programm dlja vychisli-tel'nyh klasterov i setej M.: IPM im. MV Keldy-sha RAN. 2002, 42 p. (in Russian)
8. **Konovalov N. A.** i dr. Fortran DVM-jazyk razrabotki mobil'nyh parallel'nyh program // Programmirovanie. 1995. №. 1. pp. 49-54. (in Russian)
9. The mpC Parallel Programming Environment: web-site. [Electronic resource]. URL: http://panda. ispras.ru/~mpc/ (date of access:01.11.2016
10. **Andrianov A. N., Efimkin K. N., Zadyhajlo I. B.** Jazyk Norma. Preprint IPM im. M.V.Keldysha AN SSSR. № 165. 1985. (in Russian)
11. Sistema NORMA: web-site. [Electronic resource]. URL: http://keldysh.ru/pages/norma (date of access: 01.11.2016) (in Russian).
12. Linda Introduction: web-site. [Electronic resource]. URL: http://web.archive.org/web/20080827180428/http://phi.sinica.edu.tw/instruct/workshop/html/linda/linda.html (date of access: 01.11.2016)
13. The ABCL family of languages. [Electronic resource]. URL: http://web.yl.is.s.u-tokyo.ac.jp/pl/abcl.html #overview (date of access: 02.11.2016)
14. **Bräunl T.** Parallaxis-III: A language for structured data-parallel programming. Proceedings of the IEEE First International Conference on Algorithms and Architectures for Parallel Processing. pp. 43-52.
15. Parallaxis-III - A Structured Data-Parallel Programming Language. [Electronic resource]. URL: http://robotics. ee.uwa.edu.au/parallaxis/ (date of access: 01.11.2016)
16. Sisal Parallel Programming: web-site. [Electronic resource]. URL: https://sourceforge.net/ projects/sisal/
17. **Abramov S. M., Kuznecov A. A., Roganov V. A.** Krossplatformennaja versija T-sistemy s otkrytoj arhitekturoj. Vychislitel'nye metody i program-mirovanie. 2007. Vol. 8. №. 1. pp. 175-180. (in Russian)
18. Jazyk Erlang i programmirovanie dlja mul'-tijadernyh processorov. [Electronic resource]. URL: http://itc.ua/articles/yazyk_erlang_i_programmirovanie_dlya_multiyader nyh _processorov_26721 / (data obrashhenija: 01.11.2016) (in Russian).
19. **Kazakov F. A., Legalov A. I.** Parallel'noe programmirovanie v jazykah Haskell i Pifagor // Problemy informatizacii regiona. PIR-2001: Sb. nauch. tr./IPC KGTU. Krasnojarsk, 2002. pp. 48-55. (in Russian)
20. **Ivutin A. N., Larkin E. V., Lutskov Y. I., Novikov A. S.** Simulation of concurrent process with Petri-Markov nets. Life Sci J 2014; 11(11):506-511 (ISSN:1097-8135). http://www.lifesciencesite.com/lsj/life1111/086_ 25899life111114_506_511.pdf
21. **Ivutin A. N., Daragan E. I.** Teorija setej Petri i ee rasshirenija. Izvestija TulGU. Serija: Tehnicheskie nauki. Vyp. 10. Tula: Izd. TulGU, 2012, pp. 211-220. (in Russian).
22. **Ivutin A. N.** Optimizacija parallel'nyh programm s ispol'zovaniem matematicheskogo apparata setej Petri-Markova. Izvestija TulGU. Serija: Tehni-cheskie nauki. Vyp. 12. Ch.2, Tula: Izd. TulGU, 2012. pp. 249-255. (in Russian).