УДК 681.3

ОБНАРУЖЕНИЕ ТУПИКОВ В ПАРАЛЛЕЛЬНЫХ ПРОГРАММАХ КАК РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ ДИОФАНТОВЫХ УРАВНЕНИЙ

О.Ф. БАБАХЬЯН
(Ростовский научно-исследовательский институт радиосвязи)

Рассмотрен метод обнаружения дедлоков в распределенных системах на этапе проектирования системы. Система представляется в виде модели посредством формальной спецификации с помощью аппарата сетей Петри.
Ключевые слова: *тестирование, параллельные программы, сети Петри, дедлоки.*

Введение. Параллельные программы с недетерминированным поведением, в которых отдельные компоненты функционируют параллельно и взаимодействуют асинхронно, получают все большее распространение. Тестирование параллельных приложений является непростой задачей, поскольку из-за недетерминированности ошибки распараллеливания сложно выявить. Скрупулезный анализ и хорошо спланированное тестирование позволяют исключить значительную долю ошибок, тем не менее, в достаточно сложных системах ошибки могут быть выявлены уже в процессе эксплуатации. Стоимость исправления таких ошибок может быть весьма существенной. Моделирование проектируемой системы сетью Петри на этапе ее создания благодаря развитым инструментам позволяет разработчикам эффективно обнаруживать и исправлять ошибки на ранних стадиях.

Сети Петри – одно из современных и наиболее эффективных средств графического и математического моделирования систем самых различных классов [1]. Это мощный инструмент для описания систем, использующих параллелизм, синхронизацию и разделяемые ресурсы.

Цель настоящей работы – рассмотрение метода обнаружения дедлоков в распределенных системах. Используя описания сетями Петри на примере вычислительной системы (ВС), показать, что система характеризуется отсутствием или наличием тупиков, воспользовавшись методом построения усеченного множества решений или Truncated Set of Solutions (TSS) [3] для нахождения *T*-инварианта.

Построение формальной модели вычислительной системы. ВС представлена в виде ординарной сети Петри с одноцветными фишками [1, 2]. Пусть ВС состоит из двух вычислительных устройств (ВУ), одно из которых главное (ВУГ), второе – подчиненное (ВУП). Рабочий цикл ВУ складывается из трех этапов: начало работы *(BEGIN),* прием или посылка сообщений *(INIT),* окончание работы *(END).* На этапе *BEGIN* ВУГ посылает устройству ВУП сигнал о начале работы и переходит в состояние ожидания ответа. ВУП, получив сигнал о начале работы, переходит в активное состояние и выдает подтверждающее сообщение о готовности. ВУГ, получив подтверждающий сигнал, также переходит в активное состояние. На этом этап *BEGIN* заканчивается. Находясь в состоянии *INIT* , ВУГ может либо передавать задания для выполнения устройству ВУП, либо перейти в состояние *END*. В свою очередь ВУП может также передавать устройству ВУГ задания для обработки, но самостоятельно в состояние *END* перейти не может. Если ВУГ (ВУП) передало задание устройству ВУП (ВУГ) для обработки, то оно переходит в состояние ожидания. Только после того, как будет получен подтверждающий сигнал, ВУГ (ВУП) может выполнить действия по инициализации новых заданий. Инициатива по переходу в состояние *END* может исходить только от ВУГ. При этом ВУГ посылает устройству ВУП сигнал о завершении работы и переходит в неактивное состояние.

Построим основанную на понятиях сетей Петри модель, которая описывает функционирование рассмотренной выше ВС. Подобные модели в дальнейшем будем называть СП-моделями (рис.1).

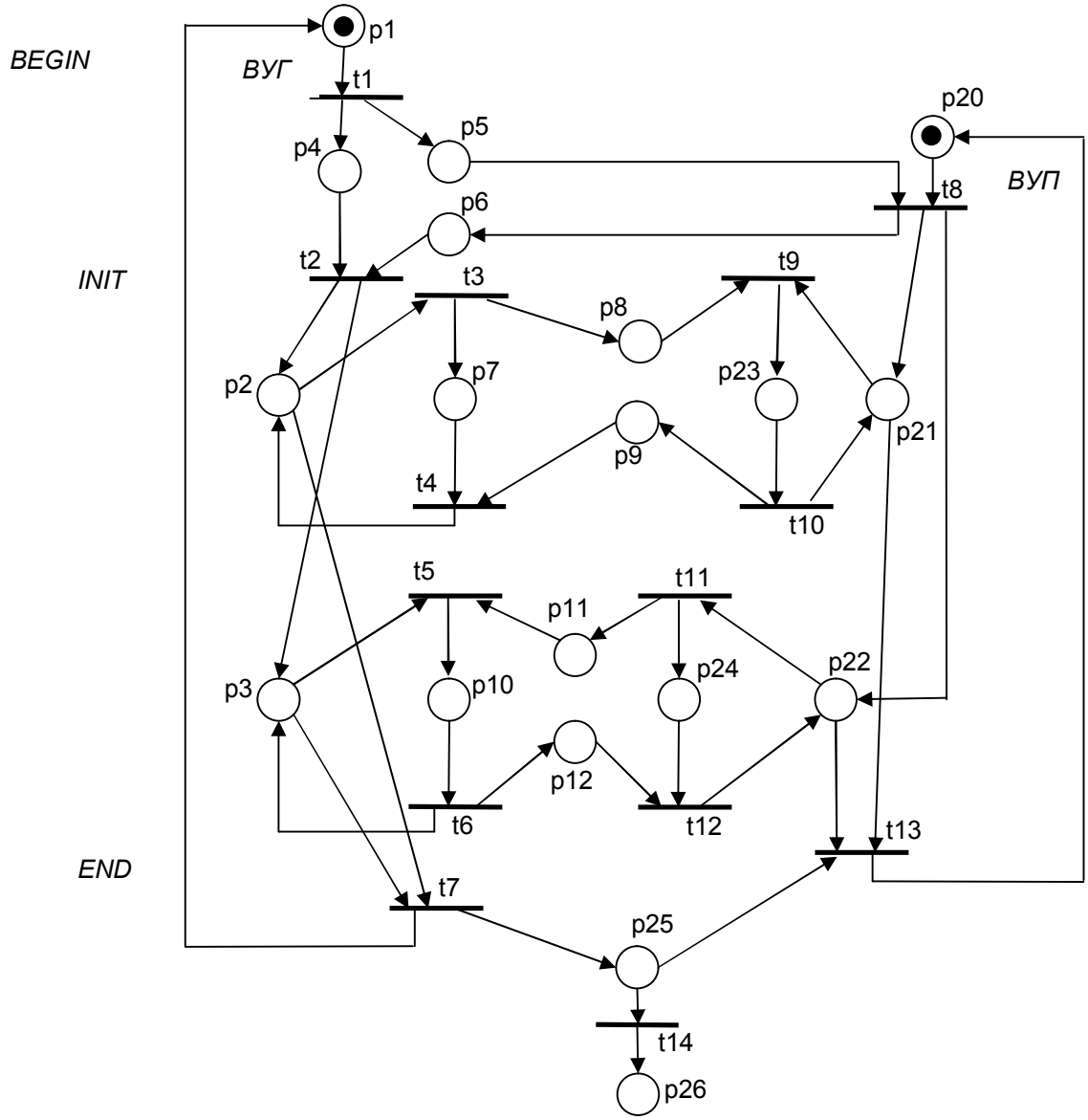

Рис. 1. СП-модели вычислительной системы

Начальное состояние полученной модели описывается разметкой:

$$\mu_0 = (\mathrm{p}_1, \mathrm{p}_4, \mathrm{p}_5, \mathrm{p}_6, \mathrm{p}_2, \mathrm{p}_7, \mathrm{p}_8, \mathrm{p}_9, \mathrm{p}_3, \mathrm{p}_{10}, \mathrm{p}_{20}, \mathrm{p}_{21}, \mathrm{p}_{23}, \mathrm{p}_{22}, \mathrm{p}_{11}, \mathrm{p}_{12}, \mathrm{p}_{24}, \mathrm{p}_{25}, \mathrm{p}_{26}) =$$
$$= (1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0).$$

Алгоритм построения TSS для системы линейных однородных диофантовых уравнений (СЛОДУ) и его реализация подробно описаны в работе [3], поэтому приведем лишь необходимые факты и определения.

Пусть дана СЛОДУ *S* вида:

$$S = \begin{cases} L_1(x) = a_{11}x_1 + \cdots + a_{1q}x_q = 0 \\ L_2(x) = a_{21}x_1 + \cdots + a_{2q}x_q = 0 \\ \cdots \\ L_p(x) = a_{p1}x_1 + \cdots + a_{pq}x_q = 0 \end{cases}. \quad (1)$$

Рассмотрим множество векторов канонического базиса $M_0^{'} = \{e_1, \ldots, e_q\}$ и первое уравнение $L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1q}x_q = 0$ системы S вида (1). С помощью функции $L_1(x)$ разо-

бьем элементы множества $M_0^{'}$ на три группы: $M_1^0 = \{e|L_1(e) = 0\}$, $M_1^+ = \{e|L_1(e) > 0\}$ и $M_1^- = \{e|L_1(e) < 0\}$. Ясно, что если одно из множеств $M_1^0 \cup M_1^+$ или $M_1^0 \cup M_1^-$ пусто, то уравнение $L_1(x) = 0$ не имеет нетривиальных решений в множестве натуральных чисел. Если хотя бы два из множеств M_1^0, M_1^+, M_1^- непусты, тогда множество решений уравнения $L_1(x) = 0$ можно определить по следующей формуле:

$$M_1 = M^0 \cup \{y_{ij}|y_{ij} = -L_1(e_i) \cdot e_j + L_1(e_j) \cdot e_i, e_j \epsilon M^+, e_i \epsilon M^-\}. \quad (2)$$

Полученное множество M_1 с помощью функции $L_2(x)$ тоже разбирается на три группы: $M^0 = \{e|L_2(e) = 0\}$, $M^+ = \{e|L_2(e) > 0\}$ и $M^- = \{e|L_2(e) < 0\}$. Далее повторяются все действия, аналогичные предыдущему уравнению. Алгоритм заканчивает работу, когда все уравнения данной СЛОДУ будут обработаны и элементы множества M_j составляют TSS этой СЛОДУ.

Построим *T*-инвариант для доказательства живости сети. *T*-инвариант соответствует последовательности срабатывания переходов, переводящей сеть из разметки M в ту же самую разметку M. Если *T*-инвариант содержит все переходы сети, то она жива.

Для нахождения *T*-инварианта сети построим целочисленную *n×m* матрицу инцидентности сети Петри A (табл. 1), где n и m — мощности множеств P и T соответственно. В матрице A элемент $A_{ij} = 1$, если при выполнении перехода j фишка добавляется в позицию i, $A_{ij} = -1$, если при выполнении перехода j фишка убирается из позиции i, $A_{ij} = 0$, если при выполнении перехода j число фишек в позиции i не изменяется.

Таблица 1

Построение матрицы инцидентности сети Петри (см. рис.1)

Состояние сети Петри	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}
p_1	–1	0	0	0	0	0	1	0	0	0	0	0	0	0
p_2	0	1	–1	1	0	0	–1	0	0	0	0	0	0	0
p_3	0	1	0	0	–1	1	–1	0	0	0	0	0	0	0
p_4	1	–1	0	0	0	0	0	0	0	0	0	0	0	0
p_5	1	0	0	0	0	0	0	–1	0	0	0	0	0	0
p_6	0	–1	0	0	0	0	0	1	0	0	0	0	0	0
p_7	0	0	1	–1	0	0	0	0	0	0	0	0	0	0
p_8	0	0	1	0	0	0	0	0	–1	0	0	0	0	0
p_9	0	0	0	–1	0	0	0	0	0	1	0	0	0	0
p_{10}	0	0	0	0	1	–1	0	0	0	0	0	0	0	0
p_{11}	0	0	0	0	–1	0	0	0	0	0	1	0	0	0
p_{12}	0	0	0	0	0	1	0	0	0	0	0	–1	0	0
p_{20}	0	0	0	0	0	0	0	–1	0	0	0	0	1	0
p_{21}	0	0	0	0	0	0	0	1	–1	1	0	0	–1	0
p_{22}	0	0	0	0	0	0	0	1	0	0	–1	1	–1	0
p_{23}	0	0	0	0	0	0	0	0	1	–1	0	0	0	0
p_{24}	0	0	0	0	0	0	0	0	0	0	1	–1	0	0
p_{25}	0	0	0	0	0	0	1	0	0	0	0	0	–1	–1
p_{26}	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Проанализируем показанную на рис.1 сеть Петри, описывающую модель функционирования вычислительной системы. Для этого воспользуемся уравнением состояний:

$$Ax = 0, \quad (3)$$

где A – целочисленная $n \times m$ матрица инцидентности сети Петри; n и m – мощности множеств P и T соответственно; Ax – m-мерный вектор Париха.

С помощью уравнения состояния (3) определим *T*-инвариант сети Петри. Размерность матрицы инцидентности A в уравнении (3) для данной сети Петри составляет 19×14 (19 уравнений, 14 неизвестных), а вектора Париха x – 14.

В соответствии с *TSS*-алгоритмом система диофантовых уравнений (1) имеет четыре решения:

$$x_1 = \{0,0,0,0,1,1,0,0,0,0,1,1,0,0\}^T,$$
$$x_2 = \{1,1,0,0,0,0,1,1,0,0,0,0,1,0\}^T,$$
$$x_3 = \{1,1,2,2,0,0,1,1,2,2,0,0,1,0\}^T,$$
$$x_4 = \{1,1,1,1,1,1,1,1,1,1,1,1,1,0\}^T.$$

На основе решений этой системы можно построить *T*-инварианты (табл.2).

Таблица 2

T-инварианты сети Петри (см. рис.1)

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}
0	0	0	0	1	1	0	0	0	0	1	1	0	0
1	1	0	0	0	0	1	1	0	0	0	0	1	0
1	1	2	2	0	0	1	1	2	2	0	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	0

В табл.2 столбец t_{14} содержит все нули, но поскольку он отвечает за завершение работы, то это не приводит к неверной работе сети. Таким образом, его можно не учитывать при анализе. Это значит, все переходы в сети Петри являются живыми при данной начальной разметке, т.е. каждый переход в сети срабатывает хотя бы один раз. Следствием чего является отсутствие тупиков, когда в системе не может возникнуть ситуация взаимной блокировки вычислений или доступа к данным (общим или распределенным). Действительно, тупик возникает в сети Петри, если нельзя запустить один или несколько переходов. Так как все переходы живые, то сеть Петри обладает требуемым свойством: отсутствием тупиков.

Заключение. Приведен пример построения имитационной модели функционирования вычислительной системы с помощью аппарата сети Петри. Анализ предлагаемой модели выполняется с использованием матрицы инцидентности и TSS-алгоритма решения систем линейных диофантовых уравнений над множеством натуральных чисел. Построенная модель обладает необходимым свойством — является живой. Таким образом, показано, что предложенный метод можно применять на этапах построения систем, использующих параллелизм, синхронизацию и разделяемые ресурсы для анализа такого свойства системы, как наличие или отсутствие дедлоков.

Библиографический список

1. Питерсон Дж. Теория сетей Петри и моделирование систем : пер. с англ. / Дж. Питерсон. – М.: Мир, 1984. – 264 с.

2. Котов В.Е. Сети Петри / В. Е. Котов. – М.: Наука, 1984. – 160 с.

3. Krivoi S. A criteria of Compatibility Systems of Linear Diophantine Constraints / S. Krivoi // Lecture Notes in Comp. Science. – 2002. – №2328. – P.264-271.

Материал поступил в редакцию 18.05.2011.

References

1. Piterson Dzh. Teoriya setej Petri i modelirovanie sistem : per. s angl. / Dzh. Piterson. – M.: Mir, 1984. – 264 s. – In Russian.
2. Kotov V.E. Seti Petri / V. E. Kotov. – M.: Nauka, 1984. – 160 s. – In Russian.
3. Krivoi S. A criteria of Compatibility Systems of Linear Diophantine Constraints / S. Krivoi // Lecture Notes in Comp. Science. – 2002. – #2328. – P.264-271.

DETECTION OF DEADLOCKS IN PARALLEL PROGRAMS AS SOLUTION OF LINEAR DIOPHANTINE EQUATIONS

O.F. BABAKHYAN
(Rostov Scientific research Institute for Radiocommunication)

The method of detecting deadlocks in the distributed systems at the design stage of the system is considered. The system is presented in the form of a model through the formal specification by means of Petri nets.
***Keywords**: testing, parallel programs, Petri nets, deadlocks.*