СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2015. – № 2(80).** – 80–86

УДК 62-50:519.216

СИМВОЛЬНЫЕ МЕТКИ В СЕТЯХ ПЕТРИ ПРИ АНАЛИЗЕ ПРОГРАММ*

А.А. ВОЕВОДА[1], Д.О. РОМАННИКОВ[2]

[1] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, доктор технических наук, профессор кафедры автоматики. E-mail: ucit@ucit.ru*
[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, кандидат технических наук, доцент кафедры автоматики. E-mail: rom2006@gmail.ru*

В данной статье предлагается идея нового вида сетей Петри. Выполнен обзор различных подходов к представлению программ, где акцент сделан на представление программы в виде модели, основанной на сети Петри и последующей ее интерпретации. Все известные авторам модели программ, основанные на представлении с использованием сетей Петри, предполагают, что их моделирование выполняется на одном, заранее заданном наборе значений переменных. Таким образом, для проверки на всем множестве значений переменных необходимо выполнить перебор всех возможных значений, что приводит к гигантскому росту требуемых вычислений и фактической невозможности осуществить такую проверку. В работе предлагается использовать модифицированный вид сетей Петри, где вместо обычных значений в метках сети должны быть использованы метки, содержащие символы. Причем эти символы должны однозначно соответствовать переменным из оригинальной программы. Безусловно, такой подход усложняет анализ интерпретируемой модели, но также он *делает возможным* анализ модели программы в сетях Петри на всем спектре значений переменных. В работе приводится процесс построения такой сети Петри на примере простой программы, в которой содержатся условные операторы и операторы присваивания для небольшого количества переменных, но предлагаемые принципы построения модифицированной сети Петри могут быть применены и для больших и сложных примеров. Работа заканчивается выводами о возможностях и перспективах предлагаемой идеи, а также вариантах дальнейших исследований, к которым относятся разработка формальных алгоритмов построения сети и ее анализа.

Ключевые слова: программное обеспечение, тестирование, входные интервалы, формальная верификация, динамическая верификация, верификация, проверка моделей, модели программного обеспечения, графы, тотальная корректность программ

DOI: 10.17212/2307-6879-2015-2-80-86

* Статья получена 12 февраля 2015 г.

ВВЕДЕНИЕ

Научное и практическое сообщества за все время предложили множество способов представления программ. В основном это различные модификации графа управления (Control Flow Graph) [1–7], представление в виде сетей Петри [8–13] или представление в виде тотального графа переходов (Total Transition Graph) [14–15]. В данной работе представлена идея подхода к представлению программы в виде сети Петри, метки которой содержат символы вместо обычного представления со значениями.

ПРЕДСТАВЛЕНИЕ ПРОГРАММЫ

В работах [8–10] программа представлена в виде сети Петри, где операции выполняются при переходах между состояниями, состояния содержат «слепок» программы в определенный момент времени исполнения, а метки в сети являются аналогом управления в реальной программе. Все вышеприведенные представления в сетях Петри основывались на том, что метки содержат численную информацию о переменных, что приводит к невозможности проверки даже достаточно простой программы на всем спектре значений используемых переменных.

Если в представлениях [8–13] заменить метки с численным значениями на метки, в которых содержатся символы, то станет теоретически возможно проверять весь спектр возможных значений используемых переменных при той же асимптотической сложности. Рассмотрим пример программы, содержащей несколько условных операторов, на рис. 1.

```
a,b,c;
if (a > b) {
  c = 10;
} else {
  c = 12;
}
b = 2*c;
if (b == 3*c) {
  c = 18;
} else {
  b = 22;
}
```

Рис. 1. Пример программы с условными переходами

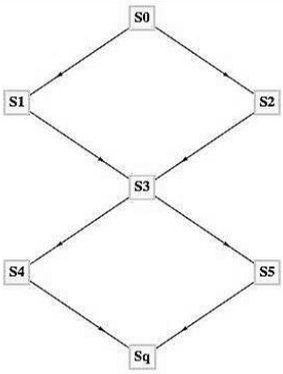

Рис. 2. Граф переходов для программы из рис. 1

На рис. 2 представлен граф программы из рис. 1. Граф по своей структуре полностью аналогичен сети Петри для той же программы. Начальная метка с типом (INT, INT, INT) содержалась бы в узле S0, переходы S0 → S1, S0 → S2, S3 → S4, S3 → S5 содержали бы защитные условия соответствующих условных операторов. Однако, как было сказано ранее, возможности по анализу данной сети крайне ограничены. Если модифицировать метки вышеприведенной сети Петри на метки, в которых будут содержаться символы вместо значений (a, b, c), то станет возможным выполнение анализа с учетом *всех возможных значений переменных типа данных*.

ВЫВОДЫ

Представленная идея использования «символьных» меток (меток, в которых содержатся символы вместо соответствующих им численных значений в одном из возможных вариантов исполнения программы) в сетях Петри для расширения возможностей анализа: появляется возможность выполнять проверки не только на конкретных значениях, но и на всем диапазоне возможных значений используемого типа данных. Данное достоинство позволяет находить большее количество ошибок, и при этом структура сети Петри остается неизменной.

Дальнейшим направлением исследования является разработка формальных алгоритмов преобразования программы в модифицированную сеть Петри и разработка алгоритмов анализа полученной модели.

СПИСОК ЛИТЕРАТУРЫ

1. *Bush W.R., Pincus J.D., Sielaff D.J.* A static analyzer for finding dynamic programming errors // Software: Practice and Experience. – 2000. – Vol. 30, iss. 7. – P. 775–802. – doi: 10.1002/(SICI)1097-024X(200006)30:7<775::AID-SPE309>3.0.CO;2-H.

2. *Falk H., Marwedel P.* Source code optimization techniques for data flow dominated embedded software. – New York: Springer Science; Business Media, 2004. – 226 p. – doi: 10.1007/978-1-4020-2829-8.

3. Coherent clusters in source code / S. Islam, J. Krinke, D. Binkley, M. Harman // The Journal of Systems and Software. – 2014. – Vol. 88. – P. 1–24. – doi:10.1016/j.jss.2013.07.040.

4. *Horwitz S., Reps T., Binkley D.* Interprocedural slicing using dependence graphs // ACM Transactions on Programming Languages and Systems (TOPLAS).– 1990. – Vol. 12, iss. 1. – P. 26–60. – doi: 10.1145/77606.77608.

5. *Ferrante J., Ottenstein K.J., Warren J.D.* The program dependence graph and its use in optimization // ACM Transactions on Programming Languages and Systems (TOPLAS). – 1987. – Vol. 9, iss. 3. – P. 319–349. – doi: 10.1145/24039.24041.

6. *Scholz B., Blieberger J., Fahringer T.* Symbolic pointer analysis for detecting memory leaks // Proceedings of ACM SIGPLAN Workshop on "Partial Evaluation and Semantics-Based Program Manipulation" (PEPM'00), Boston, Massachusetts, USA, January 22–23, 2000. – New York: ACM Press, 2000. – P. 104–113. – doi: 10.1145/328690.328704.

7. Fahringer T., Scholz B. Advanced symbolic analysis for compilers. – Berlin; Heidelberg: Springer-Verlag, 2003. – 136 p. – (Lecture Notes in Computer Science; vol. 2628). – doi: 10.1007/3-540-36614-8.

8. *Романников Д.О.* Разработка программного обеспечения с применением UML диаграмм и сетей Петри для систем управления локальным оборудованием: дис. … канд. техн. наук: 05.13.11 / Новосибирский государственный технический университет. – Новосибирск, 2012. – 195 с.

9. *Коротиков С.В.* Применение сетей Петри в разработке программного обеспечения центров дистанционного контроля и управления: дис. … канд. техн. наук: 05.13.11 / Новосибирский государственный технический университет. – Новосибирск, 2007. – 216 с.

10. *Воевода А.А., Романников Д.О., Зимаев И.В.* Применение UML диаграмм и сетей Петри при разработке встраиваемого программного обеспечения // Научный вестник НГТУ. – 2009. – № 4 (37). – С. 169–174.

11. *Коротиков С.В., Воевода А.А.* Применение сетей Петри в разработке программного обеспечения центров дистанционного управления и контроля // Научный вестник НГТУ. – 2007. – № 4 (29). – С. 15–32.

12. *Воевода А.А., Марков А.В.* Методика автоматизированного проектирования программного обеспечения функционирования сложных систем на основе совместного использования UML диаграмм и сетей Петри // Современные технологии. Системный анализ. Моделирование. – 2014. – № 2 (42). – С. 110–115.

13. *Воевода А.А., Марков А.В., Романников Д.О.* Разработка программного обеспечения: проектирование с использованием UML диаграмм и сетей Петри на примере АСУ ТП водонапорной станции // Труды СПИИРАН. – 2014. – Вып. 3 (34). – С. 218–231.

14. *Кларк Э., Грамберг О., Пелед Д.* Верификация моделей программ. Model Checking: пер. с англ. – М.: МЦНМО, 2002. – 416 с.

15. *Карпов Ю.Г.* Model checking. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010. – 560 с.

Воевода Александр Александрович, доктор технических наук, профессор кафедры автоматики Новосибирского государственного технического университета. Основное направление научных исследований – управление многоканальными объектами. Имеет более 200 публикаций. E-mail: ucit@ucit.ru

Романников Дмитрий Олегович, кандидат технических наук, доцент кафедры автоматики Новосибирского государственного технического университета. Основные направления научных исследований: формальная верификация, проверка моделей. Имеет 34 публикации. E-mail: rom2006@gmail.ru

Symbolic marks in Petri nets regarding program analysis*

A.A. Voevoda[1], D.O. Romannikov[2]

[1] *Novosibirsk State Technical University, 20 K. Marx Prospekt, Novosibirsk, 630073, Russian Federation, D. Sc. (Eng.), professor of the automation department. E-mail: ucit@ucit.ru*
[2] *Novosibirsk State Technical University, 20 K. Marx Prospekt, Novosibirsk, 630073, Russian Federation, Ph. D. (Eng.), associate professor of the automation department. E-mail: rom2006@gmail.ru*

In this paper we propose the idea of a new kind of Petri nets. A review of the various approaches to presenting programs, where the emphasis is on the presentation of the program as a model based on Petri nets and its subsequent interpretation. All well-known authors of the model programs based on the concept using Petri nets, suggest that their simulation is performed on a predefined set of variables. Thus, to check on the entire set of variables necessary to perform search of all possible values that would cause an enormous increase in required calculations and actual impossibility to carry out such a test. We propose to use a modified form of Petri nets, where instead of the usual values in labels should be used in a network label containing symbols. Moreover, these characters must be uniquely match the variables in the original program. Of course, this approach complicates the analysis of the interpreted model, but it also makes possible the analysis of the model program in Petri nets on the entire spectrum of variables. The paper presents the process of constructing a Petri net for a simple program that contains conditional statements and assignment statements for a small number of variables, but the proposed principles of the modified Petri nets can be used for large and complex cases. The work ends with conclusions about the opportunities and prospects offered by the ideas and options for further research, which include the development of formal algorithms for constructing network and its analysis.

Keywords: software, testing, input intervals, formal verification, dynamic verification, verification, model checking, software models, graphs, total correctness of programs

DOI: 10.17212/2307-6879-2015-2-80-86

* *Received 12 February 2015.*

REFERENCES

1. Bush W.R., Pincus J.D., Sielaff D.J. A static analyzer for finding dynamic programming errors. *Software: Practice and Experience*, 2000, vol. 30, iss. 7, pp. 775–802. doi: 10.1002/(SICI)1097-024X(200006)30:7<775::AID-SPE309>3.0.CO;2-H

2. Falk H., Marwedel P. *Source code optimization techniques for data flow dominated embedded software*. New York, Springer Science, Business Media, 2004. 226 p. doi: 10.1007/978-1-4020-2829-8

3. Islam S., Krinke J., Binkley D., Harman M. Coherent clusters in source code. *The Journal of Systems and Software*, 2014, vol. 88, pp. 1–24. doi: 10.1016/j.jss.2013.07.040

4. Horwitz S., Reps T., Binkley D. Interprocedural slicing using dependence graphs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1990, vol. 12, iss. 1, pp. 26–60. doi: 10.1145/77606.77608

5. Ferrante J., Ottenstein K.J., Warren J.D. The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1987, vol. 9, iss. 3, pp. 319–349. doi: 10.1145/24039.24041

6. Scholz B., Blieberger J., Fahringer T. Symbolic pointer analysis for detecting memory leaks. *Proceedings of ACM SIGPLAN Workshop on "Partial Evaluation and Semantics-Based Program Manipulation" (PEPM'00)*, Boston, Massachusetts, USA, January 22–23, 2000. New York, ACM Press, 2000, pp. 104–113. doi: 10.1145/328690.328704

7. Fahringer T., Scholz B. Advanced symbolic analysis for compilers. *Lecture Notes in Computer Science*. Vol. 2628. Berlin, Heidelberg, Springer-Verlag, 2003. 136 p. doi: 10.1007/3-540-36614-8

8. Romannikov D.O. *Razrabotka programmnogo obespecheniya s primeneniem UML diagramm i setei Petri dlya sistem upravleniya lokal'nym oborudovaniem*. Diss. kand. tekhn. nauk [Software development using UML diagrams and Petri nets for local control systems equipment. PhD eng. sci. diss.]. Novosibirsk, 2012. 195 p.

9. Korotikov S.V. *Primenenie setei Petri v razrabotke programmnogo obespecheniya tsentrov distantsionnogo kontrolya i upravleniya*. Diss. kand. tekhn. nauk [Application of Petri nets in software development centers, remote monitoring and control. PhD eng. sci. diss.]. Novosibirsk, 2007. 216 p.

10. Voevoda A.A., Romannikov D.O., Zimaev I.V. Primenenie UML diagramm i setei Petri pri razrabotke vstraivaemogo programmnogo obespecheniya [An approach to the using UML and Petri nets for embedded software designing]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Science bulletin of the Novosibirsk state technical university*, 2009, no. 4 (37), pp. 169–174.

11. Korotikov S.V., Voevoda A.A. Primenenie setei Petri v razrabotke programmnogo obespecheniya tsentrov distantsionnogo upravleniya i kontrolya [Using

Petri nets in software development of remote monitoring and control center]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Science bulletin of the Novosibirsk state technical university*, 2007, no. 4 (29), pp. 15–32.

12. Voevoda A.A., Markov A.V. Metodika avtomatizirovannogo proektirovaniya programmnogo obespecheniya funktsionirovaniya slozhnykh sistem na osnove sovmestnogo ispol'zovaniya UML diagramm i setei Petri [Methodology of computer-aided design software of complex systems based on combined use of UML diagrams and PETRI nets]. *Sovremennye tekhnologii. Sistemnyi analiz. Modelirovanie – Modern Technologies. System analysis. Modeling*, 2014, no. 2 (42), pp. 110–115.

13. Voevoda A.A., Markov A.V., Romannikov D.O. Razrabotka programmnogo obespecheniya: proektirovanie s ispol'zovaniem UML diagramm i setei Petri na primere ASU TP vodonapornoi stantsii [Software development: software design using UML diagrams and Petri nets for example automated process control system of pumping station]. *Trudy SPIIRAN – SPIIRAS proceedings*, 2014, iss. 3 (34), pp. 218–231.

14. Clarke E.M., Grumberg O., Peled D. *Model checking*. Cambridge, London, MIT Press, 2001 (Russ. ed.: Klark E., Gramberg O., Peled D. *Verifikatsiya modelei programm: Model checking*. Translated from English. Moscow, MTsNMO Publ., 2002. 416 p.).

15. Karpov Yu.G. *Model Checking. Verifikatsiya parallel'nykh i raspredelennykh programmnykh sistem* [Model Checking. Verification of parallel and distributed software systems]. St. Petersburg, BHV-Petersburg, 2010. 560 p.