ISSN 1814-1196
Научный вестник НГТУ
том 61, № 4, 2015, с. 152–158

http://journals.nstu.ru/vestnik
Science Bulletin of the NSTU
Vol. 61, No. 4, 2015, pp. 152–158

СООБЩЕНИЯ MESSAGES

УДК 62-83: 531.3

Сворачивание циклов в сетях Петри*

А.А. ВОЕВОДА[1], И.Л. РЕВА[2], Д.О. РОМАННИКОВ[3]

[1] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, доктор технических наук, профессор. E-mail: voevoda@ucit.ru*
[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, кандидат технических наук, доцент. E-mail: reva@corp.nstu.ru*
[3] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, кандидат технических наук, доцент. E-mail: rom2006@gmail.com*

Существенным недостатком способов анализа программного обеспечения (ПО), основанных на переборе всех возможных начальных вариантов, является «взрывной» рост числа вариантов анализа (в различных методах этот термин называется по-разному: в проверке моделей – размер тотального множества переходов, в сетях Петри – пространство состояний и т. д.), который приводит к невозможности анализа ПО. Одним из основных источников значительного увеличения числа вариантов для анализа являются конструкции с программными циклами, особенно содержащие условные операторы внутри цикла. Потенциально такие циклы могут приводить к фактически бесконечному числу вариантов для анализа. В работе предлагается метод упрощения анализа ПО, который предполагает преобразование выражений в циклах с целью уменьшения количества вариантов анализа (сокращение размера тотального множества переходов или поиск идентичных участков пространства состояний). Рассматриваемый способ основывается на представлении выражений в цикле в виде функции, зависящей от числа итераций. При этом выражения в цикле преобразовываются в нерекуррентные соотношения, а анализ ПО сводится к решению набора систем нелинейных уравнений или к другим видам анализа полученных выражений. Необходимо заметить, что анализ полученных функций также является достаточно сложной задачей, но ее решение авторам кажется более перспективным, чем решения, основанные на полном переборе. В работе приводятся примеры применения данного метода для преобразования выражений в цикле к функциям от числа итераций, а также примеры анализа программ с помощью анализа полученных функций.

Ключевые слова: программное обеспечение, цепи рекуррентности, рекуррентные соотношения, проверка моделей, сети Петри, пространство состояний, программные циклы, символьный анализ, граф состояния, пространство переходов

DOI: 10.17212/1814-1196-2015-4-152-158

---

* *Статья получена 01 июля 2015 г.*

Работа выполнена при поддержке Министерства образования и науки Российской Федерации, проект № 7.559.2011, гос. рег. номер НИР 01201255056

ВВЕДЕНИЕ

В настоящее время при разработке программного обеспечения (ПО) отсутствие ошибок в основных пользовательских сценариях использования ПО проверяется при помощи методологических средств и методик [1] в лабораторных условиях. Разработка и внедрение формальных инструментов анализа ПО позволит в достаточной мере быстро находить разнообразные программные ошибки, что поспособствует изменению всего процесса разработки ПО.

Задача поиска ошибок в ПО (далее анализа ПО) является актуальной, и существует много различных подходов к ее решению. Например, в ряде публикаций [2–4] были изложены методы, в которых предложена интерпретация программы при помощи сетей Петри, а работы [5, 6] основаны на моделировании всего пространства переходов программы. Оба направления предполагают моделирование программы при всех начальных условиях. В работе [7] показано, что даже для тривиальных программ, например для программы с тремя целочисленными переменными, количество различных начальных вариантов для моделирования может достигать $2^{3\times64}$, что на практике приводит к отказу от решения такой задачи. Кроме того, существует ряд примеров, для которых число начальных вариантов является бесконечным (см. [4, 10]).

В работах [8–11] используются элементы символьных вычислений для анализа ПО. Однако в основном символьные вычисления использовались как основа для последующего статического анализа.

С другой стороны, в работах [12, 13] предлагается использовать различные инструменты, с помощью которых удается выполнять свертку выражений, стоящих в циклах (что эквивалентно частичному свертыванию пространства состояний), и тем самым значительно упрощать их анализ.

В работе на основании рассмотрения двух вышеупомянутых направлений анализа ПО предлагается использование метода, в котором объединены достоинства обоих направлений, а также приводится иллюстрация его применения на примере однопоточной программы.

1. СРАВНЕНИЕ МЕТОДОВ АНАЛИЗА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

На рис. 1 изображена тривиальная программа с пятью переменными и циклом, в котором изменяются переменные a, b, c. Сеть Петри, соответствующая этой программе, приведена на рис. 2. Если использовать методы анализа ПО из [2–4], то согласно им необходимо выполнить моделирование программы при каких-то начальных условиях (согласно [5, 6] необходимо построить тотальное множество переходов, что на практике приводит к идентичной задаче), при этом нет гарантии того, что при других начальных условиях ошибки отсутствуют. Очевидно, что для полной проверки такой программы нужно проанализировать ее на всем наборе входных параметров (или, по крайней мере, на том, который позволяет обнаружить все ошибки в программе), что для четырех параметров означает $2^{4\times64}$ проверок {a, b, c, n}: {0, 0, 0, 0}, {0, 0, 0, 1} и т. д. Несомненно, это является существенным недостатком данных способов.

```
unsigned int a, b, c, n;
for (int i; i < n; ++i) {
        a = 1;
      b = 5*b + 7;
       c = c + b;
          }
  assert(condition());
```

Рис. 1. Пример программы для анализа

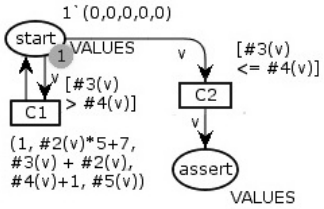

Рис. 2. Сеть Петри для программы на рис. 1

В работах [8–11] для анализа ПО был использован механизм, основанный на представлении рекуррентных соотношений (Chains of Recurrences, далее сокращенно CR) в виде функции f(i) = G(x_0 + i × h) или в сокращенном виде $\{\varphi_0, \odot, h\}$ [14–16], где φ_0 – начальное значение, $\odot \in \{+, *\}$ – либо операция сложения, либо умножения, а h – приращение функции на каждом шаге. Например, для выражения G(x) $= 3x + 1$ форма записи будет иметь вид $\{3x_0 + 1, +, 3h\}$. Для более сложных функций используется следующая форма записи: $f(i) = \{f_0, \odot_0, \{f_1, \odot_1, \{...\}\}\}$. Например, для функции G(x) = e^x^2 форма рекуррентного выражения будет иметь вид $f(i) = \{e^{x^2}, *, e^{2x_0 \times h + h^2}, *, e^{2h^2}\}$ [14].

Несмотря на то что изначально CR рассматривался в качестве инструмента ускорения вычислительных задач, в частности в задачах компиляции исходного кода, в работах [8, 10, 11] он был применен для анализа ПО. Однако стоит заметить, что в этих работах запись рекуррентных соотношений использовалась для определения операторов монотонности* и операторов, ограничивающих значения переменной, на которых и был построен дальнейший статический анализ**.

Рассматриваемая на рис. 1 программа после CR преобразований имеет вид, представленный на рис. 3.

```
unsigned int a, b, c, n
    a(i) = {1, +, 0}
b(i) = {5*b0 + 7, +, 7*5^(h-1)}
c(i) = {c0 + 5*b0 + 7, +, {5*b0 + 7, +,
          7*5^(h-1)}};
  assert(condition());
```

Рис. 3. Преобразованная программа (рис. 1) в выражениях CR

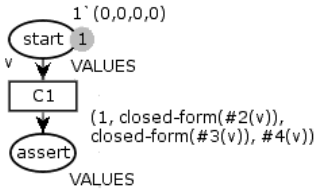

Рис. 4. Сеть Петри для программы на рис. 3

На рис. 4 представлена сеть Петри, в которой цикл преобразован в выражение, зависимое от числа итераций «i» (представлен в виде функций closed-form()). Стоит отметить, что не все функции могут быть представлены

* В работе [8] вводится понятие операторов монотонности и ограничения значения переменной с целью оценки значения переменной в некоторых интервалах. Например, оператор «T» показывает, что переменная больше нуля, «⊥» – значение переменной не определено.

** Анализ программного обеспечения, производимый без реального выполнения исследуемой программы.

в таком виде при помощи CR записи (например, тригонометрические и экспоненциальные функции). После преобразования нет необходимости выполнять все итерации цикла, что приводит к существенному уменьшению числа анализируемых вариантов.

Еще один способ записи рекуррентных соотношений приведен на рис. 5. Данный способ предполагает предварительную обработку выражений под циклом. В данном случае преобразования были выполнены при помощи программного пакета для работы с компьютерной алгеброй Mathematica*.

```
unsigned int a, b, c, n;
a_n = 1;
b_n = ¼*(-7+7*5^n+4*5^n*b0)
c_n = 1/16*(-35+7*5^1+n-28n-20*b0+4*5^1+nb0+16c0)
assert(condition());
```

Рис. 5. Представление программы в виде функции от числа итераций

После выполненного преобразования для анализа ПО необходимо рассмотреть систему уравнений и неравенств. Например, для проверки условия «c_n == 0» система будет иметь вид, показанный на рис. 6. Она была получена на основании представления для выражений c_n (рис. 5) и всех переменных, в него входящих

$$\begin{cases} c\_n = 0; \\ c\_n = \dfrac{1}{16}(-35 + 7 \times 5^{1+n} - 28n - 20 \times b0 + 4 \times 5^{1+n} b0 + 16c0); \\ n > 0. \end{cases}$$

Рис. 6. Система неравенств для проверки условия «*c_n* == 0»

Стоит отметить, что в данном случае нас интересует не решение данной системы, а наличие хотя бы одного набора переменных {n, c0, b0}, при котором выражение не является истинным.

ЗАКЛЮЧЕНИЕ

В ходе анализа способов поиска программного обеспечения, основанных на последовательной проверке всех возможных состояний программы, было выявлено, что их основным недостатком является то, что даже для небольших программ количество необходимых проверок может быть настолько велико, что не может быть выполнено за приемлемое время и приводит к невозможности использования на практике. Также было выявлено, что

* Данные выражения получены следующим способом: для представленной программы (рис. 1) были получены несколько первых значений для переменных на каждой итерации цикла (10 итераций); далее при помощи функции *FindSequenceFunction,* входящей в программный пакет Wolfram Mathematica 10.2, последовательность была преобразована в функцию, зависящую от начальных значений и количества итераций (рис. 4).

наибольшее увеличение числа вариантов анализа происходит в циклах (особенно при неизвестной длине цикла).

Для преодоления этого недостатка в работе предлагается использовать другой способ, который позволяет существенно снижать количество вариантов анализа – представление рекуррентных соотношений в виде функции от значения переменных перед циклом и числа итераций. Для этого могут быть использованы CR выражения или способ, основанный на получении конечного выражения при помощи систем компьютерной алгебры. Далее задача анализа программы сводится к поиску набора значений переменных, при котором проверяемое условие в программе не выполняется. Данная задача также является в достаточной мере сложной, но ее решение представляется более перспективным, чем полный перебор программы при всех начальных условиях. Стоит отметить, что применение данного способа позволяет существенно снизить число вариантов для анализа в программе по сравнению со значением, полученным при полном переборе всех начальных значений переменных.

СПИСОК ЛИТЕРАТУРЫ

1. *Орлов С.А.* Технология разработки программного обеспечения. – СПб.: Питер, 2012. – 609 с.

2. *Романников Д.О.* Разработка программного обеспечения с применением UML диаграмм и сетей Петри для систем управления локальным оборудованием: дис. … канд. техн. наук. – Новосибирск, 2012. – 195 с.

3. *Коротиков С.В.* Применение сетей Петри в разработке программного обеспечения центров дистанционного контроля и управления: дис. … канд. техн. наук. – Новосибирск, 2007. – 216 с.

4. *Марков А.В.* Автоматизация проектирования и анализа программного обеспечения с использованием языка UML и сетей Петри: дис. … канд. техн. наук. – Новосибирск, 2015. – 176 с.

5. *Clarke E.M., Grumberg O., Peled D.* Model checking. – Cambridge: MIT Press, 1999. – 330 p.

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

7. *Воевода А.А., Романников Д.О.* Сравнение методов преобразования программного цикла с использованием символьной нотации // Сборник научных трудов НГТУ. – 2015. – № 1 (79). – С. 65–76.

8. *Engelen R.A. van* The CR# algebra and its application in loop analysis and optimization: technical report TR-041223 / Florida State University. – Florida, 2004. – 13 p.

9. *Scholz B., Blieberger J., Fahringer T.* Symbolic pointer analysis for detecting memory leaks // Proceedings of the 2000 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation (PEPM'00). – New York, 1999. – P. 104–113.

10. *Burgstaller V., Scholz V., Blieberger V.* A symbolic analysis framework for static analysis of imperative programming languages // Journal of Systems and Software. – 2012. – Vol. 85, iss. 6. – P. 1418–1439.

11. A unified framework for nonlinear dependence testing and symbolic analysis / R.A. van Engelen , J. Birch, Y. Shou, B. Walsh, K.A. Gallivan // Supercomputing: Proceedings of the 18th Annual International Conference, (ICS'04), 2004. – New York, 2004. – P. 106–115.

12. *Воевода А.А., Романников Д.О.* Символьные метки в сетях Петри при анализе программ // Сборник научных трудов НГТУ. – 2015. – № 2 (80). – С. 80–86.

13. *Cousot V., Cousot R.* A gentle introduction to formal verification of computer systems by abstract interpretation // Logics and Languages for Reliability and Security / J. Esparza, O. Grumberg, M. Broy, eds. – Washington: IOS Press, 2010. – P. 1–29. – (NATO science for peace and security series. Sub-series D. Information and communication security; vol. 25).

14. *Bachmann O., Wang P.S., Zima E.V.* Chains of recurrences – a method to expedite the evaluation of closed-form functions // Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC'94, Oxford, England, UK, 20–22 July 1994. – New York: ASM, 1994. – P. 242–249.

15. *Kislenkov V., Mitrofanov V., Zima E.* Multidimensional chains of recurrences // Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation, ISSAC'98, Rostock, Germany, 13–15 August 1998. – New York: ASM, 1998. – P. 199–206.

16. *Bachmann O.* Chains of recurrences: PhD diss. of philosophy / Kent State University. – Kent, 1996. – 145 p.

Воевода Александр Александрович, доктор технических наук, профессор Новосибирского государственного технического университета. Основные направления научных исследований: теория автоматического управления, сети Петри. Имеет более 200 публикаций. E-mail: voevoda@ucit.ru

Рева Иван Леонидович, кандидат технических наук, доцент Новосибирского государственного технического университета. Основное направление научных исследований – кодировка сигналов. Имеет более 30 публикаций. E-mail: reva@corp.nstu.ru

Романников Дмитрий Олегович, кандидат технических наук, доцент Новосибирского государственного технического университета. Основное направление научных исследований – верификация. Имеет более 40 публикаций. E-mail: rom2006@gmail.com

Cycle folding in Petri Nets[*]

A.A. VOEVODA[1], *I.L. REVA*[2], *D.O. ROMANNIKOV*[3]

[1] *Novosibirsk State Technical University, 20, K. Marx Prospekt, Novosibirsk, 630073, Russian Federation, D. Sc. (Eng.), professor. E-mail:* voevoda@ucit.ru

[2] *Novosibirsk State Technical University, 20, K. Marx Prospekt, Novosibirsk, 630073, Russian Federation, PhD, associate professor. E-mail: reva@corp.nstu.ru*

[3] *Novosibirsk State Technical University, 20, K. Marx Prospekt, Novosibirsk, 630073, Russian Federation, PhD, associate professor. E-mail: rom2006@gmail.com*

The main drawback of software analysis methods which are based on the enumeration of all possible initial values is an ‘explosive’ growth of the number of analysis options. This term is called differently in different methods. For example, in the model test method it is called the size of a total transition set while in the Petri net method it is referred to as the size of state space. This drawback leads to an inadmissibly long analysis time and thus makes the software analysis impracticable. One of the main sources of increasing the number of analysis options is program loop structures, especially if they contain conditional operators inside loops. In fact, such loops can lead to infinite numbers of analysis options. In the paper a method of a simplified software analysis is considered. The method assumes transformation of expressions in the loops to reduce the number of options for analysis. The method is based on the representation of expressions in the loops as a function that depends on the number of iterations. The expressions in the loops are converted into non-recurrence relations and software analysis is reduced to solving a set of nonlinear equations or analyzing other types of expressions. It should be noted that the analysis of the resulting function is also quite a challenge, but its solution seems more promising than solutions based on an exhaustive search. The paper provides examples of the application of this method to convert expressions in a loop into functions of the number of iterations as well as examples of program analysis through the analysis of the functions obtained.

Keywords: Software; recurrence chains; recurrences, model test; Petri nets; state space; program cycle; symbolic analysis; state graph; transition space

DOI: 10.17212/1814-1196-2015-4-152-158

[*] *Received 01 July 2015.*

The work was supported by the Ministry of education and science of the Russian Federation, project no 7.559.2011, state registration number of scientific research works 01201255056

REFERENCES

1. Orlov S.A. *Tekhnologiya razrabotki programmnogo obespecheniya* [Software engineering]. St. Petersburg, Piter Publ., 2012. 609 p.

2. 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.

3. Korotikov S.V. *Primenenie setei Petri v razrabotke programmnogo obespecheniya tsentrov dis-tantsionnogo 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.

4. Markov A.V. *Avtomatizatsiya proektirovaniya i analiza programmnogo obespecheniya s ispol'zovaniem yazyka UML i setei Petri*. Diss. kand. tekhn. nauk [Computer-aided design and analysis software with UML and Petri nets. PhD eng. sci. diss.]. Novosibirsk, 2015. 176 p.

5. Clarke E.M., Grumberg O., Peled D. *Model Checking*. Cambridge, MIT Press, 1999. 330 p.

6. 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.

7. Voevoda A.A., Romannikov D.O. Sravnenie metodov preobrazovaniya programmnogo tsikla s is-pol'zovaniem simvol'noi notatsii [Comparison of methods for the transformation of the program cycle with the use of symbolic notation]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Transaction of scientific papers of the Novosibirsk state technical university*, 2015, no. 1 (79), pp. 65–76.

8. Engelen R.A van. The CR# algebra and its application in loop analysis and optimization. Technical report TR-041223, Florida State University, 2004. 13 p.

9. Scholz B., Blieberger J., Fahringer T. Symbolic pointer analysis for detecting memory leaks. *Proceedings of the 2000 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation*, New York, 1999, pp. 104–113.

10. Burgstaller V., Scholz V., Blieberger V. A symbolic analysis framework for static analysis of imperative programming languages. *Journal of Systems and Software*, 2012, vol. 85, iss. 6, pp. 1418–1439.

11. Engelen R.A. van, Birch J., Shou Y., Walsh B., Gallivan K.A. A unified framework for nonlinear dependence testing and symbolic analysis. *Supercomputing: Proceedings of the 18th Annual International Conference*, (*ICS'04*), 2004, pp. 106–115.

12. Voevoda A.A., Romannikov D.O. Simvol'nye metki v setyakh Petri pri analize programm [Symbolic tags in Petri nets in the analysis programs]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Transaction of scientific papers of the Novosibirsk state technical university*, 2015, no. 2 (80), pp. 80–86.

13. Cousot V., Cousot R. A gentle introduction to formal verification of computer systems by abstract interpretation. *Logics and Languages for Reliability and Security*. Ed. by J. Esparza, O. Grumberg, M. Broy. Washington, IOS Press, 2010, pp. 1–29.

14. Bachmann O., Wang P.S., Zima E.V. Chains of recurrences – a method to expedite the evaluation of closed-form functions. *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC'94*, Oxford, England, UK, 20–22 July 1994, pp. 242–249.

15. Kislenkov V. Mitrofanov V., Zima E. Multidimensional chains of recurrences. *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation, ISSAC'98*, Rostock, Germany, 13–15 August 1998, pp. 199–206.

16. Bachmann O. *Chains of recurrences.* PhD diss. of philosophy. Kent, Kent State University, 1996. 145 p.