УДК 004.423 + 519.68[1 + 2.1]

Г.Н. ЖОЛТКЕВИЧ, И.Д. ПЕРЕПЕЛИЦА, Ю.В. СОЛЯНИК, М. ТАВИ

Харьковский национальный университет имени В.Н. Каразина, Украина

ОБ ОДНОЙ МОДЕЛИ ПРОГРАММ В ЗАДАЧАХ ФОРМАЛЬНОЙ ВЕРИФИКАЦИИ

Введен класс математических моделей программ, описываемый в терминах языкового каркаса, который базируется на языке охраняемых команд Э. Дейкстры. Определены модели вычислений, названные авторами системами правил перехода. В терминах этих систем описана семантика программ и дана строгая постановка задачи верификации.

Ключевые слова: *программа, модель программы, модель вычисления, синтаксис и семантика программы, задача верификации.*

Введение

Задача повышения гарантоспособности программного обеспечения является одной из актуальных задач современной программной инженерии. В связи с этим возникает необходимость в разработке специальных программных инструментов, предназначенных для доказательства правильности программного обеспечения – пруверов. Однако, хорошо известно, что свойства правильных программ в подавляющем большинстве случаев являются алгоритмически неразрешимыми (см. напр. [1]). В то же время отдельные классы важных свойств могут быть проверены, что демонстрирует метод верификации моделей программ, известный в англоязычной литературе как Model Checking [2].

Начало исследований в этой области было положено работой Р. Флойда [3]. Значительный вклад в развитие теории внесли Ч. Хоар (см. [4]) Э. Дейкстра [5]. В работах [6 – 8] П. Кузо предложил подход к задачам верификации программ, базирующийся на методе абстрагирования, что в определенных случаях позволяет найти безопасный приближенный алгоритм решения задачи. Однако в этих работах не описан метод построения абстракций для программ, позволяющих верифицировать то либо иное свойство. В то же время результаты Д. Ульмана, изложенные в [9], позволяют надеяться, что такой метод существует.

В связи с этим возникает задача построения класса математических моделей программ, описываемых общим языковым каркасом.

1. Логические формулы

Задано счетное множество Φ_0, элементы которого называются атомарными логическими формулами.

Множество логических формул Φ определяется как минимальное множество слов над алфавитом $\Phi_0 \cup \{\to,(,)\}$, удовлетворяющее условиям:

1. $\mathbf{\Phi}_0 \cup \mathbf{P}_0 \cup \{\to,(,),;,\mathrm{if},\mathrm{fi},\mathrm{do},\mathrm{od},:,|\}$;
2. если $\phi \in \Phi_0$, то $\phi \in \Phi$;
3. если $\phi,\psi \in \Phi$, то $(\phi \to \psi) \in \Phi$.

Пусть последовательность $\{\Phi_n \mid n = 1,2,\ldots\}$ построена следующим образом:
для $n \geq 1$:
$\Phi_{n+1} \equiv \Phi_n \cup \{\phi, \text{ãää}(\exists\varphi,\psi \in \Phi_n)(\phi \equiv (\varphi \to \psi))\}$.

Верна следующая теорема, которая характеризует синтаксис логических формул.

Теорема 1. Если $\Phi_\infty = \bigcup_{n\geq 0} \Phi_n$, то $\Phi_\infty = \Phi$.

2. Программы языка $\mathrm{GCL}(\Phi_0, \mathrm{P}_0)$

Множество программ P определяется как минимальное множество слов над алфавитом $\Phi_0 \cup \mathrm{P}_0 \cup \{\to,(,),;,\mathrm{if},\mathrm{fi},\mathrm{do},\mathrm{od},:,|\}$, удовлетворяющее следующим условиям:

1. $\mathrm{skip} \in \mathrm{P}_0$;
2. если $P \in \mathrm{P}_0$, то $P \in \mathrm{P}$;
3. если $P_1,\ldots,P_n \in \mathrm{P}$, то $P_1;\ldots;P_n \in \mathrm{P}$;
4. если $\gamma_1,\ldots,\gamma_n \in \Phi$ и $P_1,\ldots,P_n \in \mathrm{P}$, то $\mathrm{if}\ \gamma_1 : P_1 \mid \ldots \mid \gamma_n : P_n\ \mathrm{fi} \in \mathrm{P}$;
5. если $\gamma_1,\ldots,\gamma_n \in \Phi$ и $P_1,\ldots,P_n \in \mathrm{P}$, то $\mathrm{do}\ \gamma_1 : P_1 \mid \ldots \mid \gamma_n : P_n\ \mathrm{od} \in \mathrm{P}$.

Назовем программу $P \in \mathrm{P}$ блоком, если выполнено одно из трех условий:

1. $P \in \mathrm{P}_0$;
2. $P = \mathrm{if}\ \gamma_1 : P_1 \mid \ldots \mid \gamma_n : P_n\ \mathrm{fi}$, где $\gamma_1,\ldots,\gamma_n \in \Phi$, $P_1,\ldots,P_n \in \mathrm{P}$;

3. $P = \mathrm{do}\,\gamma_1 : P_1 \,|\, \ldots \,|\, \gamma_n : P_n \,\mathrm{od}$,

где $\gamma_1, \ldots, \gamma_n \in \Phi$, $P_1, \ldots, P_n \in \mathrm{P}$.

Аналогично теореме 1 доказывается следующая теорема, которая определяет синтаксис программ.

Теорема 2. Рассмотрим последовательности множеств $\{\mathrm{P_n} \mid \mathrm{n} \geq 0\}$ и $\{\mathrm{B_n} \mid \mathrm{n} \geq 0\}$, определенных рекуррентно равенствами:

$$\mathrm{P_{n+1}} = \mathrm{P_n} \cup \{P_1; \ldots; P_k, \text{ãäå } P_1, \ldots, P_k \in \mathrm{B_n}\};$$

$$\mathrm{B_0} = \mathrm{P_0};$$

$$\mathrm{B_{n+1}} = \mathrm{B_n} \cup \{\text{if } \gamma_1 : P_1 \,|\, \ldots \,|\, \gamma_k : P_k \text{ fi},$$

где

$$\gamma_1, \ldots, \gamma_k \in \Phi,$$

$$P_1, \ldots, P_k \in \mathrm{P_n}\} \cup$$

$$\cup \{\mathrm{do}\,\gamma_1 : \mathrm{P_1} \,|\, \ldots \,|\, \gamma_k : \mathrm{P_k}\ \mathrm{od},$$

где

$$\gamma_1, \ldots, \gamma_k \in \Phi,$$

$$P_1, \ldots, P_k \in \mathrm{P_n}\},$$

тогда множество $\mathrm{P}_\infty = \bigcup_{n \geq 0} \mathrm{P_n}$ совпадает с P, а множество $\mathrm{B}_\infty = \bigcup_{n \geq 0} \mathrm{B_n}$ является множеством всех блоков.

Следствие. Всякая программа P однозначно представляется в виде $B_1; \ldots; B_k$, где все $B_i\,(i = 1, \ldots, k)$ являются блоками.

3. Системы правил переходов какмодели вычислений

В этом разделе будет введена модель вычислений, используемая в дальнейшем для определения семантики языка охраняемых команд.

Пусть, как и прежде, заданы множества Φ_0 – атомарных логических формул и $\mathrm{P_0}$ – элементарных программ. Рассмотрим счетное множество L, элементы которого будем называть локациями. Будем считать, что в множестве L зафиксировано два различных элемента l_* и l^*, которые будем называть входом и выходом соответственно. Построим, как это было описано выше, множество логических формул Φ.

Определение 1.

Элемент $\langle l_1, \gamma, P, l_2 \rangle \in \mathrm{L} \times \Phi \times \mathrm{P_0} \times \mathrm{L}$ назовем правилом перехода, если $l_1 \neq l^*$.

Если $\tau = \langle l_1, \gamma, P, l_2 \rangle$ – правило перехода, то будем использовать следующие обозначения и названия:

1) $l_1 \equiv \mathrm{src}[\tau]$ – источник перехода;

2) $l_2 \equiv \mathrm{trg}[\tau]$ – цель перехода;

3) $\gamma \equiv \mathrm{grd}[\tau]$ – охранное условие перехода;

4) $P \equiv \mathrm{eff}[\tau]$ – эффект перехода.

Множество правил перехода будем обозначать $\mathrm{T}(\Phi_0, \mathrm{P_0}, \mathrm{L})$.

Определение 2. Конечная последовательность правил $\tau_1 \ldots \tau_n$ со свойством: для $i \in \{1, \ldots, n-1\}$ выполняется $\mathrm{trg}(\tau_i) = \mathrm{src}(\tau_{i+1})$, будет называться абстрактным потоком. Если при этом $\mathrm{src}(\tau_1) = l_*$ и $\mathrm{trg}(\tau_n) = l^*$, то поток называется полным абстрактным потоком или абстрактным вычислением.

Определение 3. Конечное множество правил перехода R назовем абстрактной системой правил перехода над $\mathrm{T}(\Phi_0, \mathrm{P_0}, \mathrm{L})$, если для всякого правила $\tau \in \mathrm{R}$ существует полный абстрактный поток $\tau_1 \ldots \tau_n$ со свойствами

1) $\tau_i \in \mathrm{R}$ для всякого $i \in \{1, \ldots, n\}$;

2) $\tau = \tau_i$ для некоторого $i \in \{1, \ldots, n\}$.

Если R – абстрактная система правил перехода над $\mathrm{T}(\Phi_0, \mathrm{P_0}, \mathrm{L})$, то через $\mathrm{L}(\mathrm{R})$ будем обозначать конечное множество локаций, упомянутых в правилах из R.

Определение 4. Пусть R – абстрактная система правил перехода над $\mathrm{T}(\Phi_0, \mathrm{P_0}, \mathrm{L})$. Графом управления системы R назовем ориентированный мультиграф с множеством вершин $\mathrm{L}(\mathrm{R})$ и множеством ребер R. Пара вершин l_1 и l_2 соединяется ребром $\tau \in \mathrm{R}$, если $\mathrm{src}(\tau) = l_1$ и $\mathrm{trg}(\tau) = l_2$.

Граф управления системы R будет обозначаться $\Gamma(\mathrm{R})$.

Утверждение. Для всякой абстрактной системы перехода R над $\mathrm{T}(\Phi_0, \mathrm{P_0}, \mathrm{L})$ граф управления системы $\Gamma(\mathrm{R})$ обладает следующими свойствами:

1) локации l_* и l^* являются его вершинами;

2) не существует ребра исходящего из вершины l^*;

3) всякая вершина $l \in \mathrm{L}(\mathrm{R})$ лежит на пути из l_* в l^*.

Пусть R абстрактная система правил перехода над $\mathrm{T}(\Phi_0, \mathrm{P_0}, \mathrm{L})$, а M – множество. Элементы этого множества будут называться состояниями памяти, а само множество – памятью. Зададим пару отображений $\mu_\Phi : \Phi_0 \to 2^{\mathrm{M}}$ и $\mu_\mathrm{P} : \mathrm{P_0} \to 2^{\mathrm{M} \times \mathrm{M}}$, причем $\mu_\Phi[\bot] = \varnothing$, $\mu_\mathrm{P}[\mathrm{skip}] = \{(\mathrm{u}, \mathrm{u}) \mid \mathrm{u} \in \mathrm{M}\}$.

Отображение μ_Φ продолжим на множество Φ индуктивно:

1) μ_Φ на Φ_0 определено;

2) если μ_Φ определено на Φ_n, то для $\phi \in \Phi_{n+1} - \Phi_n$ определим $\mu_\Phi(\phi)$ следующим образом: $\phi \equiv (\varphi \to \psi)$, где $\varphi, \psi \in \Phi_n$, тогда $\mu_\Phi(\phi) = \complement_M[\mu_\Phi(\varphi)] \cup \mu_\Phi(\psi)$.

Определение 5. Тройка $M = (M, \mu_\Phi, \mu_P)$ называется моделью абстрактной системы правил перехода R над $T(\Phi_0, P_0, L)$, если M – множество, $\mu_\Phi : \Phi \to 2^M$, $\mu_P : P_0 \to 2^{M \times M}$, причем отображение $\mu_\Phi : \Phi \to 2^M$ получено описанным выше продолжением отображения $\mu_\Phi : \Phi_0 \to 2^M$.

Пусть задана абстрактная система правил перехода R над $T(\Phi_0, P_0, L)$ и ее модель $M = (M, \mu_\Phi, \mu_P)$. Для всякой пары локаций $l_1, l_2 \in L(R)$ мы можем определить множество $S[l_1, l_2](R) \subset M \times M$ следующим образом:

$(u, v) \in S[l_1, l_2](R)$ тогда и только тогда, когда существует абстрактный поток $\tau_1 \dots \tau_n$, для которого $src[\tau_1] = l_1$, $trg[\tau_n] = l_2$, и существует последовательность состояний памяти $u_1, \dots, u_{n-1}$ такая, что

1) $u \in \mu_\Phi[grd[\tau_1]]$;

2) $u_i \in \mu_\Phi[grd[\tau_{i+1}]]$ для всех $i = 1, \dots, n-1$;

3) $(u, u_1) \in \mu_P[eff[\tau_1]]$;

4) $(u_{i-1}, u_i) \in \mu_P[eff[\tau_i]]$ для всех $i = 2, \dots, n-1$;

5) $(u_{n-1}, v) \in \mu_P[eff[\tau_n]]$.

Для краткости, ситуацию, описанную выше, будем обозначать $u \xrightarrow{\tau_1} u_1 \xrightarrow{\tau_2} \dots \xrightarrow{\tau_{n-1}} u_{n-1} \xrightarrow{\tau_n} v$ и говорить, что поток $\tau_1 \dots \tau_n$ начинается в локации l_1 в состоянии памяти u и завершается в локации l_2 в состоянии памяти v.

Определение 6.

Множество $S(R) = S[l_*, l^*](R)$ будем называть спецификацией «вход – выход» модели $M = (M, \mu_\Phi, \mu_P)$ абстрактной системы правил перехода R над $T(\Phi_0, P_0, L)$.

Неформально говоря, множество $S(R)$ определяет возможные выходы системы R при заданном входе.

4. Система правил перехода, связанная с программой

В этом разделе с каждой программой $P \in P$ будет связана абстрактная система правил перехода $R(P)$, на основании чего будет определена семантика программы.

Для описания соответствующей процедуры трансляции введем вспомогательное определение обобщенного правила перехода: обобщенным правилом перехода называется четверка $\langle l_1, \gamma, P, l_2 \rangle \in L \times \Phi \times P \times L$ при условии, что $l_1 \neq l^*$.

Пусть $P \in P$.

1) $P \equiv Q_1; \dots; Q_n$ в силу следствия из теоремы 2, где каждая программа Q_i является блоком. Построим $R(P)$ – множество обобщенных правил перехода следующим образом: выберем произвольные различные локации $l_1, \dots, l_{n-1}$, каждая из которых не совпадает ни с l_*, ни с l^* и положим

$$R(P) = \left\{ \langle l_*, \top, Q_1, l_1 \rangle, \left\langle l_{n-1}, \top, Q_n, l^* \right\rangle \right\} \cup \cup \{ \langle l_{i-1}, \top, Q_i, l_i \rangle \mid i = 2, \dots, n-1 \}.$$

2) Пусть $\langle l_1, \gamma, Q, l_2 \rangle \in R(P)$ – обобщенное правило перехода, тогда , если $Q \notin P_0$, то возможны следующие варианты:

a. $Q \equiv B_1; \dots; B_m$, где все B_i блоки, тогда выберем $l'_1, \dots, l'_{m-1} \in L - L(R(P))$, все различные, удалим рассматриваемое правило, добавив правила $\langle l_1, \top, B_1, l'_1 \rangle$, $\langle l'_{i-1}, \top, B_i, l'_i \rangle$ для $i = 2, \dots, m-1$ и $\langle l'_{m-1}, \top, B_m, l_2 \rangle$;

b. $Q \equiv \text{if } \gamma_1 : P_1 \mid \dots \mid \gamma_m : P_m \text{ fi}$, тогда удалим рассматриваемое правило и добавим следующие правила $\langle l_1, \gamma_i, P_i, l_2 \rangle$ для $i = 1, \dots, m$;

c. $Q \equiv \text{do } \gamma_1 : P_1 \mid \dots \mid \gamma_m : P_m \text{ od}$, тогда удалим рассматриваемое правило и добавим следующие правила $\langle l_1, \gamma_i, P_i, l_1 \rangle$ для $i = 1, \dots, m$, а также правило $\langle l_1, \neg(\gamma_1 \vee \dots \vee \gamma_m), \text{skip}, l_2 \rangle$.

3) Если в $R(P)$ нет обобщенных правил перехода, тогда остановим процесс.

Верна следующая теорема.

Теорема 3. Описанная процедура останавливается для любой программы из P и построенное в соответствии с ней множество правил перехода $R(P)$ является системой правил перехода.

Пусть $M = (M, \mu_\Phi, \mu_P)$ – модель системы правил перехода $R(P)$, а $S(P) = S(R(P))$ ее спецификация «вход – выход», тогда $S(P)$ будем называть семантикой программы P в модели M.

Таким образом, мы можем строго поставить задачу верификации программы P: пусть задано отношение S на памяти M, моделирующее требование к связи между входом и выходом, необходимо проверить, что спецификация «вход – выход» программы P не противоречит этому отношению, т. е. $S(P) \subset S$.

Выводы

Подводя итоги изложенному, можно заключить, что в работе: предложен каркас, позволяющий строить широкий класс языков различного назначении, имеющих общую лексику; синтаксис и семантика языков, реализующих описанный каркас могут быть описаны в унифицированных терминах; семантическая модель базируется на асинхронном вычислителе, который является экспертной системой продукционного типа.

Возможными направлениями дальнейших исследований являются: построение языков различного назначения путем специализации каркаса за счет выбора типа памяти и интерпретации базовых программ на ней, например, языка хранимых процедур реляционных баз данных; построение каркаса для объектно-ориентированных языков программирования; вывод уравнений потоков и разработка методов их решения в терминах правил перехода.

Чрезвычайно важным является исследование возможностей абстрагирования свойств программ за счет факторизации модели вычислений.

Литература

1. Чень Ч. Математическая логика и автоматическое доказательство теорем. / Ч. Чень, Р. Ли – М.: «Наука», 1983. – 358 с.

2. Кларк Э. М. Верификация моделей программ: Model Checking. / Э.М. Кларк, О. Грамберг, Д. Пелед – М.: МЦНМО, 2002. – 416 с.

3.Floyd R.W. Assigning meaning to programs / R.W. Floyd // Proc. Symp. Appl. Math. – № 19, 1967. – P. 19-32.

4. Hoar C.A.R. Essays in Computing Science. / C.A.R. Hoar. – New York, London, Toronto, Sydney, Tokyo: Prentice Hall, 1989. – 412 p.

5. Дейкстра Э. Дисциплина программирования. / Э. Дейкстра – М.: Мир, 1978. – 275 с.

6. Cousot P. Inductive Definitions, Semantics and Abstract Interpretation / P. Cousot, R. Cousot. – P. 192.

7. Cousot P. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixed Points / P. Cousot, R. Cousot. – P. 39-62.

8. Cousot P. Verification by Abstract Interpretation / P. Cousot. – P. 12-36.

9.Ахо А., Лам М., Сети Р., Ульман Д. Компиляторы: принципы, технологии и инструменты. / А. Ахо, М. Лам, Р. Сети. – М., Вильямс, 2008. – 1184 с.

Поступила в редакцию 15.03.2009

Рецензент: д-р техн. наук, проф. Б.М. Конорев, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков.

СТОСОВНО ОДНІЄЇ МОДЕЛІ ПРОГРАМ В ЗАДАЧАХ ФОРМАЛЬНОЇ ВЕРИФІКАЦІЇ

Г.М. Жолткевич, І.Д. Перепелиця, Ю.В. Солянік, М. Таві

Введено клас математичних моделей програм, що описується в термінах мовного каркасу. Каркас базується на мові команд, що охороняються, Е. Дейкстри. Визначені моделі обчислень, що названі авторами системами правил переходу. В термінах цих систем описана семантика програм та подана формальна постановка задачі верифікації.

Ключові слова: програма, модель програми, модель обчислення, синтаксис и семантика програми, задача верифікації.

ABOUT ONE PROGRAM MODEL FOR VERIFICATION PROBLEM

G.M. Zholtkevych, I.D. Perepelytsia, Yu.V. Solianik, M. Thawi

A class of program mathematical models is introduced. Models of the class are described by the instrumentality of language framework. E. Djikstra's Guarded Command Language is a base of the framework. Computing models are defined. Authors called the models as transition rules systems. Program semantics and verification problem are described in the term of transition rules systems.

Key words: program, program model, model of calculation, syntax and program semantics, task of verification.

Жолткевич Григорий Николаевич - д-р техн. наук, проф., декан механико-математического факультета, заведующий кафедрой теоретической и прикладной информатики, Харьковский национальный университет им. В.Н. Каразина, Харьков, Украина, e-mail: g.zholtkevych@gmail.com.

Перепелица Иван Дмитриевич - аспирант кафедры теоретической и прикладной информатики, Харьковский национальный университет им. В.Н. Каразина, Харьков, Украина.

Соляник Юрий Владимирович - старший преподаватель кафедры теоретической и прикладной информатики, Харьковский национальный университет им. В.Н. Каразина, Харьков, Украина.

Тави Мезал - аспирант кафедры теоретической и прикладной информатики, Харьковский национальный университет им. В.Н. Каразина, Харьков, Украина.