in_1 — входные значения первой подпрограммы; out_n — выходные значения последней. При этом компилятор проверяет существование композиции подпрограмм. Для увеличения числа возможных композиций подпрограмм предлагается зафиксировать порядок перечисления входных и выходных параметров: сначала описываются параметры-переменные, затем — параметры-логические комплексы, затем — параметры-символьные комплексы. Эта особенность языка будет полезна при создании крупных библиотек на ЛЯПАСе.

ЛИТЕРАТУРА

1. *Агибалов Г. П., Липский В. Б., Панкратова И. А.* О криптографическом расширении и его реализации для русского языка программирования // Прикладная дискретная математика. 2013. № 3. С. 93–104.
2. *Стефанцов Д. А., Томских П. А.* Разработка операционной системы на языке ЛЯПАС // Прикладная дискретная математика. Приложение. 2015. № 8. С. 134–135.
3. Intel 64 and IA-32 Architectures Software Developer Manuals `http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html`

УДК 519.681.2 DOI 10.17223/2226308X/8/50

ОПЕРАЦИОННАЯ СЕМАНТИКА ЛЯПАСА

А. О. Жуковская, Д. А. Стефанцов

Сообщается о разработке операционной семантики ЛЯПАСа. Описываются её возможные применения: доказательство методом абстрактной интерпретации корректности обращения к элементам комплекса и создание верифицирующего транслятора.

Ключевые слова: *операционная семантика, ЛЯПАС, абстрактная интерпретация, верифицирующий транслятор.*

Язык программирования ЛЯПАС разработан в 1960-х годах в Томском государственном университете, использовался в СССР и других странах, в настоящее время возрождается на кафедре защиты информации и криптографии ТГУ с целью разработки доверенного программного обеспечения [1]. В данной работе сообщается об операционной семантике ЛЯПАСа, которая может применяться как минимум для двух целей: для доказательства формальных свойств программ на ЛЯПАСе и для создания верифицирующего транслятора.

Семантика языка программирования — формализация значений конструкций языка построением их математических моделей. Существует несколько вариантов семантик, самые распространённые — операционные и денотационные семантики. Денотационные семантики основаны на абстракции функций и ориентированы на функциональные языки. Операционные семантики основаны на определении состояний абстрактной машины и переходов между ними. Семантические правила представляют собой переход из одного состояния в другое при соблюдении условий. Для ЛЯПАСа и метода абстрактной интерпретации наиболее подходящий вариантом является операционная семантика.

Под программой понимается синтаксически правильная последовательность команд, представленная нумерованным списком. Состоянием программы на ЛЯПАСе является шестёрка объектов (c, var, τ, EL, Q, S), где c — номер текущей команды; var — массив значений переменных; τ — значение внутренней переменной; EL — массив спис-

ков элементов комплексов; Q и S — массивы мощностей и ёмкостей комплексов соответственно. Операции в ЛЯПАСе можно разделить на четыре группы: операции присваивания, логические и арифметические операции, операции перехода, операции над комплексами. Формулы (1) представляют примеры правил для операций из каждой группы. Это операции занесения значения в переменную, дизъюнкции, перехода по нулю и добавления элемента в комплекс:

$$\begin{gathered}
\frac{p[c] =\Rightarrow \alpha}{(c, var, \tau, EL, Q, S) \longrightarrow (c+1, var[\alpha/\tau], \tau, EL, Q, S)}, \\
\frac{p[c] = \vee\gamma,\ \tau_1 = \tau \vee \gamma}{(c, var, \tau, EL, Q, S) \longrightarrow (c+1, var, \gamma, EL, Q, S)}, \\
\frac{p[c] = 0 \to \delta,\ \tau = 0}{(c, var, \tau, EL, Q, S) \longrightarrow (\delta, var, \tau, EL, Q, S)}, \\
\frac{p[c] = @{>}\phi\xi,\ Q(\phi) < S(\phi),\ \xi \leqslant Q(\phi)}{(c, var, \tau, EL, Q, S) \longrightarrow (c+1, var, \tau, EL[\phi/EL(\phi){:}\mathrm{insert}(\xi, \tau)], Q[\phi/Q(\phi)+1], S)}.
\end{gathered} \tag{1}$$

В используемом на данный момент компиляторе имеется недостаток при работе с комплексами: каждый раз при обращении к элементу производится проверка, не выходит ли индекс за границы комплекса. Это значительно замедляет работу программ, но убирать проверку можно только в тех местах, где есть полная уверенность, что выход за границы комплекса не произойдёт. Для того чтобы получить возможность не тратить время на проверку там, где в этом нет необходимости, нужен способ автоматического доказательства корректности обращения к элементам комплекса. Для построения такого способа может быть применён метод абстрактной интерпретации [2]. Суть метода заключается в том, что семантическим правилам в конкретном домене ставятся в соответствие правила некоторого абстрактного домена, который по структуре проще конкретного. Домены и соответствие между ними должны удовлетворять некоторым правилам. Доказательство производится в абстрактном домене, а затем результат интерпретируется в конкретном.

Опираясь на семантические правила, можно не только доказать отдельные утверждения о программах, но и создать верифицирующий транслятор, что важно для написания доверенного программного обеспечения. Для языка C существует верифицирующий транслятор CompCert [3], основанный на применении средства доказательства теорем Coq [4]. Аналогичный транслятор может быть создан в будущем для ЛЯПАСа.

В дальнейшем для реализации поставленных целей следует представить семантические правила в Coq. Для доказательства корректности обращения к элементам комплекса необходимо задать абстрактный домен и соответствие между семантическими правилами и правилами в абстрактном домене. Для создания верифицирующего транслятора следует задать в Coq семантику машинного языка и построить доказательства корректности переходов, используемых транслятором.

ЛИТЕРАТУРА

1. *Агибалов Г. П., Липский В. Б., Панкратова И. А.* О криптографическом расширении и его реализации для Русского языка программирования. // Прикладная дискретная математика. 2013. № 3. С. 93–105.
2. *Besson F., Cachera D., Jensen T., and Pichardie D.* Certified static analysis by abstract interpretation // Foundations of Security Analysis and Design. 2009. No. 5705. P. 223–257.
3. The CompCert page. `http://compcert.inria.fr/`
4. The Coq Proof Assistant page. `https://coq.inria.fr/`