6. Jsoncpp: A C++ library for interacting with JSON. https://github.com/open-source-parsers/jsoncpp. 2016.
7. CompCert. Compilers you can formally trust. http://compcert.inria.fr/. 2016.
8. *Жуковская А. О., Стефанцов Д. А.* Операционная семантика ЛЯПАСа // Прикладная дискретная математика. Приложение. 2015. № 8. С. 131–132.
9. flex: The Fast Lexical Analyzer. http://flex.sourceforge.net/. 2016.
10. GNU Bison. A general-purpose parser generator. https://www.gnu.org/software/bison/. 2016.

УДК 519.714 DOI 10.17223/2226308X/9/50

МЕТОДЫ СИНТЕЗА ФРАГМЕНТОВ ПРЕДИКАТНЫХ ПРОГРАММ[1]

М. С. Чушкин, В. И. Шелехов

Рассматривается задача синтеза фрагментов предикатной программы. Синтезируемая программа определяется в форме композиции подпрограмм, полученных применением правил синтеза. Задача синтеза иллюстрируется на примере эффективной программы вычисления чисел Фибоначчи.

Ключевые слова: *формальная операционная семантика, программный синтез, дедуктивная верификация.*

1. Язык предикатного программирования

Программа на языке P_0 определяется следующей конструкцией:

$$<\textit{имя предиката}>\ (<\textit{аргументы}>:<\textit{результаты}>)\{<\textit{оператор}>\},$$

аргументы и результаты — непересекающиеся наборы имён переменных.

Предположим, что x, y и z обозначают разные непересекающиеся наборы переменных. Набор x может быть пустым, наборы y и z не пусты. Простейшим оператором является *вызов предиката* $A(x : y)$. *Операторами* языка P_0 являются следующие конструкции: $B(x : z); C(z : y)$ — последовательный оператор, $B(x : y)\ ||\ C(x : z)$ — параллельный оператор и $if\ (e)\ B(x : y)\ else\ C(x : y)$ — условный оператор.

Операционную семантику программы $H(x : y)$ определим в виде предиката: $R(H)(x, y) \equiv$ для значения набора x исполнение программы H всегда завершается и существует исполнение, при котором результатом вычисления является значение набора y [1].

Тотальная корректность (далее просто корректность) программы относительно *спецификации* в виде *предусловия* $P(x)$ и *постусловия* $Q(x, y)$ определяется формулой

$$\forall x\ (P(x) \Rightarrow [\forall y\ R(H)(x, y) \Rightarrow Q(x, y)]\ \&\ \exists y\ R(H)(x, y)).$$

2. Задача синтеза

Пусть $Q(x, y)$ — некоторая формула, связывающая между собой аргументы x и результаты y. Целью синтеза является получение (возможно, недетерминированным образом) программы $H(x : y)$, вычисляющей значение результатов y из аргументов x.

Будем говорить, что *синтезом формулы* $Q(x, y)$ является оператор $H(x : y)$ и предусловие $P(x)$ в виде формулы, и писать

$$\lceil Q(x, y) \rceil = H(x : y), \quad \lfloor Q(x, y) \rfloor = P(x),$$

[1]Работа поддержана грантом РФФИ, проект № 16-01-00498.

если существует программа $H(x : y)$, корректная относительно спецификации в виде предусловия $P(x)$ и постусловия $Q(x, y)$. Предусловие при этом описывает контекст, в рамках которого происходит вычисление.

Правила синтеза строятся на базе правил доказательства корректности в системе дедуктивной верификации [2].

Утверждение 1 (Condition Rule). Для некоторой формулы $Q(x, y)$ истинны следующие тождества:

$$\lceil Q(x,y)\rceil = if\ (e(x))\ \lceil e(x) \Rightarrow Q(x,y)\rceil\ else\ \lceil !e(x) \Rightarrow Q(x,y)\rceil,$$
$$\lfloor Q(x,y)\rfloor = e(x)?\lfloor e(x) \Rightarrow Q(x,y)\rfloor : \lfloor !e(x) \Rightarrow Q(x,y)\rfloor.$$

Утверждение 2 (Call Rule). Пусть $Q(x, y)$ — некоторая формула. Если существуют такие формулы $Q'(x, x')$ и $f(x', y)$, что $\forall x, y\ (Q(x, y) \Leftrightarrow \exists x'\ (Q'(x, x')\ \&\ f(x', y)))$, $F(x : y) = \lceil f(x, y)\rceil$ и истинно одно из следующих утверждений:

1) $f(x', y)$ — рекурсивный вызов и $Q'(x, x')\ \&\ m(x') < m(x)$, где m — некоторая функция меры, заданная на аргументах формулы f;
2) $f(x', y)$ — нерекурсивный вызов,

то истинны тождества $\lceil Q(x, y)\rceil = \lceil Q'(x, x')\rceil$; $F(x' : y)$, $\lfloor Q(x, y)\rfloor = \lfloor Q'(x, y)\rfloor$.

3. Пример

Рассмотрим формулу, описывающую числа Фибоначчи:

```
fib(i) := (i = 0 or i = 1) ? i : fib(i- 1) + fib(i - 2);
q(i, r) := r = fib(i);
```

Очевидной программой является реализация функции `fib` в виде условного оператора и двух рекурсивных вызовов. Такая программа будет неэффективна, так как рекурсия не является хвостовой. В предикатном программировании для приведения рекурсии к хвостовому виду используется метод обобщения исходной задачи.

Введём дополнительные параметры-накопители $current = fib(j)$ и $previous = fib(j - 1)$ для некоторого $j \leqslant i$. Формула для обобщённой задачи выглядит следующим образом:

```
g(i, j, current, previous, last) := 1 <= j & j <= i &
  current = fib(j) & previous = fib(j - 1) & last = fib(i);
```

Синтез формулы $q(i, r)$ начинается с применения правила *Condition Rule* с формулой $e(i) \equiv (i = 0 \text{ or } i = 1)$. Истинность посылки правила при этом проверяется с помощью SMT-решателя. Следом применяется правило *Call Rule* с вызовом $g(i, 1, 0, 1, r)$ в качестве вызова формулы. Посылки правила также проверяются с помощью решателя. Результатом синтеза является следующая предикатная программа:

```
main(i: r)
pre true
post q(i, n)
{
    if (i = 0 or i = 1)
        r = i
    else
        G(i, 1, 0, 1: r);
}
```

Синтез формулы $g(i, j, current, previous, last)$ также осуществляется с помощью правил *Condition Rule* и *Call Rule*, условия $i = j$ и формулы $g(i, j + 1, current + previous, current, last)$ для рекурсивного вызова генерируемой программы. Истинность посылок правил позволяет синтезировать следующую программу:

```
G(i, j, current, previous: last)
pre true
post g(i, j, current, previous, last)
{
    if (i = j)
        last = current
    else
        G(i, j + 1, current + previous, current: last);
}
```

Выбор правила на очередном шаге осуществляется методом перебора, указанным в работе [3]; выбор выражения в правилах *Condition Rule* и *Call Rule* — методом перебора, указанным в [4].

Предельная цель проекта — разработать алгоритмы и методы синтеза программы по её спецификации и спецификации обобщённой задачи. Предполагается интегрировать различные методы синтеза функциональных программ, предлагаемых, например, в работе [3].

ЛИТЕРАТУРА

1. *Шелехов В. И.* Семантика языка предикатного программирования // 5-я Всерос. конф. «Знания — Онтологии — Теории». ИМ СО РАН, Новосибирск, 2015. С. 15.
2. *Чушкин М. С.* Методы дедуктивной верификации предикатных программ // Труды 2-й Междунар. конф. «Инструменты и методы анализа программ». Кострома, 2014. С. 205–214.
3. *Jacobs S., Kuncak V., and Suter Ph.* Reductions for synthesis procedures // Verification, Model Checking, and Abstract Interpretation. 2013. LNCS. V. 7337. P. 88–107.
4. *Alur R., Bod'ik R., Juniwal G., et al.* Syntax-guided synthesis // FMCAD, 2013. P. 1–8.