Российская академия наук
Сибирское отделение
Институт систем информатики
имени А. П. Ершова

В. И. Шелехов

МОДЕЛЬ КОРРЕКТНОСТИ ПРОГРАММ НА ЯЗЫКЕ ИСЧИСЛЕНИЯ ВЫЧИСЛИМЫХ ПРЕДИКАТОВ

Препринт
145

Новосибирск 2007

Исчисление вычислимых предикатов [2] является математической моделью класса программ, спецификация которых может быть записана в виде предиката. Программа на языке исчисления есть замкнутый набор рекурсивных определений предикатов. Спецификация каждого определяемого предиката состоит из предусловия и постусловия. Определение корректности программы рассматривается как конкретизация общего закона соответствия спецификации и программы. Дается две системы правил доказательства корректности программ. В первой доказывается постусловие из правой части определения предиката. Во второй для однозначных предикатов реализуется вывод правой части из предусловия и постусловия.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

V. I. Shelekhov

THE MODEL OF PROGRAM CORRECTNESS FOR THE LANGUAGE OF COMPUTABLE PREDICATES

Preprint
145

Novosibirsk 2007

The calculus of computable predicates [2] is a mathematical model of programs whose specification is a predicate. A program on the calculus language is the closed set of recursive definitions of predicates. The specification of each defined predicate consists of the precondition and postcondition. Program correctness definition is considered as the concretization of the general law of the correspondence between a program and its specification. Two rule systems of program correctness proof are introduced. In first rule system, the postcondition is proved from the right part of the predicate definition. In the second rule system, for single-valued predicates, the right part is derived from the precondition and postcondition.

1. ВВЕДЕНИЕ

Программа и ее спецификация находятся в противоречивом единстве. Спецификация первична по отношению к программе. Она необходима для разработки программы. Однако в реальной практике программирования спецификация, оформленная отдельным документом или находящаяся в тексте программы, редко соответствует последней версии программы. Набор свойств, определяющих связь программы и спецификации, приведен в работе [1]. Основным является свойство (закон) соответствия программы своей спецификации: входные и выходные потоки данных исполняемой программы должны удовлетворять спецификации.

Спецификация программы называется *предикатной*[1], если она эксплицируема в виде формулы на языке исчисления предикатов[2]. Для программы с предикатной спецификацией характерна простая схема взаимодействия программы с окружением: ввод входных данных происходит в начале исполнения программы, вывод результирующих данных — в конце, а других взаимодействий с окружением нет. Спецификация определяет функцию, отображающую значения входных данных в значения результатов.

Исчисление вычислимых предикатов [2] является математической моделью для класса программ с предикатной спецификацией и определяет множество всех вычислимых формул исчисления предикатов. Исчисление является формализацией содержательно определенного понятия автоматической вычислимости программы [1]. Исчисление представлено языком ССР (Calculus of Computable Predicates). Язык ССР определяет минимальный базис, с помощью которого можно определить любой чистый язык функционального программирования. На языке ССР можно записать любой алгоритм с предикатной спецификацией [2]. Однако язык ССР неудобен для программирования. Он предназначен для исследования математических свойств программ. Конструкции языка ССР и его формальная семантика (логическая и операционная) описываются в разд. 2.

В разд.3 для программ на языке ССР закон соответствия программы своей спецификации конкретизируется в виде математической модели корректности программы. Обоснована необходимость разделения спецификации на предусловие и постусловие. Предусловие должно полагаться априо-

[1] В работе [1] наряду с предикатной определены процессная и процессорная формы спецификации.

[2] Здесь и далее имеется в виду исчисление предикатов высших порядков.

ри истинным. Выводимость спецификации из программы должна проводиться только для постусловия. Кроме этого, корректность программы включает условие реализуемости программы и постусловия в области предусловия. Понятие корректности является основой для построения системы правил доказательства программ, описанной в разд. 4.

В практике построения программ из математического решения задачи реализуется вывод в обратную сторону: программа выводится из спецификации. Теорема тождества спецификации и программы определяет условия, при которых программа может быть выведена из спецификации. Эти условия используются для построения второй системы доказательства программ на языке ССР.

В заключении рассматриваются возможности распространения предложенной модели корректности программы на язык предикатного программирования [10] и языки функционального программирования. Анализируются связи с другими работами по разным аспектам, в частности по модели корректности программ и использованию логической семантики. Язык ССР может быть использован в качестве ядра для построения языка предикатного программирования [10] и чистых языков функционального программирования.

2. ЯЗЫК ИСЧИСЛЕНИЯ ВЫЧИСЛИМЫХ ПРЕДИКАТОВ

Программа на языке ССР определяет набор вычислимых формул исчисления предикатов. Формулы представлены в форме, ориентированной на исполнение, т.е. в виде программы. Исполнение программы реализуется *абстрактным процессором*. Операционная семантика языка ССР представлена описанием программы абстрактного процессора на метаязыке, базисные операции которого определены в разд. 2.4. Возможен другой способ определения операционной семантики, предполагающий подстановку правой части определения предиката на место исполняемого вхождения (вызова) предиката. Этот способ используется для функциональных языков программирования; например, см. [9].

Семантика языка ССР представлена логической и операционной семантикой. *Логическая семантика* определяет каждую конструкцию языка как некоторую формулу в исчислении предикатов. *Операционная семантика* определяет правила исполнения конструкций.

2.1. Определение предиката

Программа состоит из конечного набора определений предикатов. *Определение предиката* есть конструкция вида:

$$A(x: y) \equiv K(x: y) \tag{2.1}$$

Здесь A обозначает имя предиката; x и y являются наборами переменных: $x = x_1, x_2, ..., x_n$, $y = y_1, y_2, ..., y_m$, причем $n \geq 0$, $m > 0$. Перечисленные переменные попарно различны. Набор x определяет *аргументы* предиката A, набор y — *результаты*. В правой части определения $K(x: y)$ обозначает *оператор*. Операторами являются оператор суперпозиции, параллельный оператор, условный оператор, конструктор предиката и конструктор массива (см. 2.6 – 2.10). Оператор $K(x: y)$ является иной синтаксической формой для соответствующей логической формулы $K^{\sim}(x, y)$. Таким образом,

$$A(x, y) \equiv K^{\sim}(x, y) \tag{2.2}$$

есть правильная формула исчисления предикатов. Правила соответствия между операторами и соответствующими логическими формулами достаточно просты. Далее будем трактовать операторы как соответствующие логические формулы. Для оператора может определяться его истинность. Операторы могут быть использованы в логических формулах.

Соответствие между операторами и соответствующими логическими формулами определяет *логическую семантику* языка ССР.

2.2. Вызов предиката

Конструкцией, используемой в составе правой части определения предиката, является *вызов предиката*:

$$A(z: u) \tag{2.3}$$

Здесь A обозначает имя предиката; z и u являются наборами переменных. Набор z определяет *аргументы вызова* и может быть пустым. Набор u определяет *результаты вызова* и не может быть пустым. Переменные набора u попарно различны и отличны от переменных набора z.

Аргументом вызова может быть также имя предиката, определяемого или базисного. Однако запрещается использование имен ConsPred и ConsArray (см. 2.9, 2.10) в качестве аргументов вызова.

В вызове (2.3) предикат с именем A либо является базисным предикатом языка ССР, либо имеет определение вида (2.1), либо является аргументом определения предиката, в теле которого встречается вызов (2.3).

Вместе с предикатной формой A(z: u), являющейся канонической, будем иногда использовать эквивалентную ей функциональную форму записи u = A(z).

Пусть имеется предикат φ(x), и требуется вычислить логическое значение предиката φ, т.е. x определяет набор аргументов, а набор результатов пуст. В этом случае будем использовать в качестве результата дополнительный параметр b логического типа и записывать предикат в виде φ(x: b)[3]. Переменная b далее будет обозначать логическую переменную.

2.3. Типы данных

Переменная, используемая в определении предиката или в вызове предиката, характеризуется именем и типом. *Тип* определяет множество возможных значений переменной. Тип идентифицируется *именем типа*.

Система типов языка ССР включает примитивные типы, подмножество произвольного типа и структурные типы. Для каждого типа набор операций со значениями типа представлен в виде базисных предикатов.

Примитивными типами являются: логический, целый, вещественный и литерный. Для них соответственно будем использовать имена: BOOL, INT, REAL и CHAR. Набор базисных предикатов для примитивных типов соответствует распространенным арифметическим и логическим операциям. Например, базисные предикаты: +(x, y: z), -(x, y: z), -(x: y), <(x, y: b) соответствуют операциям: z = x + y, z = x - y, y = -x , b = x < y.

Для каждого примитивного типа определены два вида предикатов равенства: =(x: y) и =(x, y: b). Для предиката =(x: y) процессор присваивает переменной y значение переменной x. Для предиката =(x, y: b) процессор определяет значение отношения x = y и присваивает его логической переменной b. Предикат =(d: e) определен также для наборов переменных d и e.

Имеется набор базисных предикатов для задания констант. Предикаты ConsIntZero(: x) и ConsIntOne(: x) определяют целочисленные значения 0 и 1 в качестве значения переменной x.

Далее в определениях типов T, S, X, Y и Z обозначают имена типов.

---

[3] А φ(x) будет соответствовать функциональной форме записи.

Новый тип S как *подмножество типа* T вводится определением:

$$S = SUBSET(T, x, P, d) \tag{2.4}$$

Здесь x — переменная типа T, а d — произвольный, возможно пустой, набор переменных, P — имя предиката. Предикат P(x, d: b) является вычислимым, т.е. должен быть определен на языке ССР. Тип S есть множество истинности предиката P(x, d), т.е. $S = \{x \in T \mid P(x, d)\}$. Переменные набора d являются *параметрами* типа S. Примером подмножества типа является тип натуральных чисел:

$$NAT = SUBSET(INT, x, GE0) = \{x \in INT \mid x \geq 0\},$$

где GE0 есть имя предиката $x \geq 0$. Другим примером является диапазон целых чисел:

$$DIAP(n) = SUBSET(INT, x, IN1\_n, n) = \{x \in INT \mid x \geq 1 \ \& \ x \leq n\},$$

где IN1_n есть имя предиката $x \geq 1 \ \& \ x \leq n$.

Структурный тип определяется в виде композиции других типов, называемых *компонентными* по отношению к структурному. Структурными типами являются: произведение типов (кортеж), объединение типов, массив, множество подмножеств типа и предикатный тип. Базисные предикаты для структурного типа подразделяются на конструкторы, деструкторы и другие операции со значениями типа. *Конструктор* по значениям компонентных типов строит значение структурного типа. *Деструктор* для значения структурного типа определяет соответствующие значения компонент.

Произведение типов

$$Z = X \times Y \tag{2.5}$$

определяет тип Z в виде множества кортежей (x, y), т.е. $Z = \{(x, y) \mid x \in X, y \in Y\}$. Конструктором является предикат ConsStruct(x, y: z), где $x \in X$, $y \in Y$ и $z \in Z$, а деструктором — CompStruct(z: x, y). Для конструктора и деструктора истинно отношение z = (x, y).

Объединение[4] *типов*

$$Z = X + Y \tag{2.6}$$

[4] Размеченное объединение в терминологии Т. Хоара [8].

определяет множество элементов (1, x) и (2, y) для типов X и Y, т.е. Z = {(1, x) | x ∈ X} ∪ {(2, y) | y ∈ Y}. Первая компонента в этих кортежах (1 или 2) является *тегом* значения. Конструктор ConsUnion1(x: z) строит структурное значение z по некоторому x ∈ X, т.е. z = (1, x). Аналогично, для конструктора ConsUnion2(y: z) имеет место z = (2, y). Деструктор CompUnion(z: i, x, y) для z ∈ Z определяет, что либо z = (1, x) и i = 1, либо z = (2, y) и i = 2. Если i = 1, то значение y не определено. Если i = 2, то значение x не определено.

Множество подмножеств

$$Z = SET(X) \tag{2.7}$$

для конечного типа X есть Z = {z | z ⊆ X}. Конструктор ConsSetEmpty(: z) определяет пустое множество в качестве значения переменной z. Конструктор ConsSetElem(x : z) определяет z = {x} для x ∈ X. Предикат CompSet(z, x: b) определяет истинность формулы x ∈ z.

Предикатный тип

$$Z = PRED(D: E) \tag{2.8}$$

есть множество вычислимых предикатов вида φ(d: e) для непересекающихся наборов переменных d и e, причем набор d может быть пустым. Типы переменных определяются соответствующими наборами типов D и E. Таким образом, Z = {φ(d: e) | d ∈ D, e ∈ E, φ ∈ CCP}. Конструктор предиката ConsPred описан в разд. 2.9.

Допустим, набор d не пустой, типы набора D являются конечными, а произвольный предикат φ ∈ Z определен для любого d ∈ D, т.е. ∀φ ∈ Z ∀d ∈ D ∃e φ(d: e). В этом случае предикат φ(d: e) может трактоваться как *массив*. Переменные набора d называются индексами массива, а e — *элементом массива*. Конструктор массива ConsArray описан в разд. 2.10. Деструктор CompArray(φ, d: e) определяет элемент e массива φ для набора индексов d.

Совокупность типов программы определяется типами переменных, используемых в программе. Всякий тип, встречающийся в программе, либо является примитивным, либо вводится одним из определений (2.4)–(2.8). Тип является *рекурсивным*, если определение типа прямо или косвенно, через совокупность определений, использует имя этого типа. Рекурсивный тип определяется решением некоторой системы уравнений вида (2.4)–(2.8),

в которую входит имя данного рекурсивного типа, причем решением является наименьшая неподвижная точка системы уравнений [2].

2.4. Память программы

Память исполняемой программы состоит из набора секций. *Секция памяти* содержит конечный набор переменных вместе с их значениями. Секция создается в начале исполнения вызова предиката, для которого имеется определение в программе. Секция состоит из аргументов, результатов и локальных переменных определения предиката.

Произвольная секция s представляется массивом. Индексами массива являются имена переменных. Пусть a — имя переменной. Тогда s[a] обозначает значение переменной с именем a в секции s. Если x — набор имен переменных, принадлежащих секции s, то s[x] обозначает соответствующий набор значений. Пусть T — тип переменной с именем a, а v — значение типа T. Тогда s[a] := v обозначает оператор присваивания значения v переменной a в секции s. Будем также использовать групповой оператор присваивания s[x] := w, где x — набор имен переменных, w — набор значений, причем списки типов, соответствующие x и w совпадают. Оператор s = newSect(A) создает в памяти новую секцию s, соответствующую определению предиката с именем A. Оператор delSect(s) удаляет из памяти секцию s.

Перечисленные конструкции используются в программе абстрактного процессора, исполняющего программу на языке ССР.

2.5. Исполнение вызова предиката

Пусть имеется вызов предиката (2.3) и соответствующее этому вызову определение предиката (2.1). Рассмотрим состояние исполнения программы перед исполнением вызова A(z: u). В этот момент исполняется определение другого предиката B, содержащее вызов A(z: u) в правой части. Допустим, что q — секция, содержащая переменные предиката B. Исполнение вызова A(z: u) реализуется процедурой runCall(q, A(z: u)), тело которой представлено ниже.

```
s = newSect(A);                                    (2.9)
s[x] := q[z];
runRight(s, K(x: y));
q[u] := s[y];
delSect(s)
```

Таким образом, сначала создается секция s для переменных определения A. Значения аргументов z вызова A(z: u) копируются в аргументы x определения A. Далее, в рамках секции s исполняется правая часть K(x: y) определения A. Результатом этого является вычисление значений набора y. Значения результатов y определения A копируются в результаты u вызова A(z: u). Наконец, секция s удаляется из памяти.

Если предикат A является базисным, то вызов A(z: u) считается элементарным оператором абстрактного процессора, и для его исполнения не требуется исполнять последовательность операторов (2.9). Тем не менее, и в этом случае исполнение вызова A(z: u) будем обозначать через runCall(q, A(z: u)). Исполнение вызова A(z: u) для случая, когда A является переменной предикатного типа, а значением A является массив, определено в разд. 2.10.

Определим *свойство согласованности* Coord(φ) для предиката φ(d: e), где d и e — наборы переменных.

$$\mathrm{Coord}(\varphi) \cong \forall d\ \forall e\ (\varphi(d{:}\ e) \Leftrightarrow \mathrm{RUN}(\varphi, d, e)),$$

где RUN(φ, d, e) обозначает следующее утверждение: существует такое исполнение вызова предиката φ(d: e) для набора d, которое завершается, причем результатом является набор e.

Лемма 2.1. Пусть имеется определение предиката A(x: y) ≡ K(x: y), что соответствует (2.1). Тогда Coord(K) ⇒ Coord(A).

Поскольку определение (2.1) непосредственно не исполняется, а исполняется всегда некоторый вызов предиката A, то здесь Coord(A) относится к произвольному вызову A(z: u) для некоторых наборов z и u.

Доказательство. Пусть истинно Coord(K). Докажем истинность Coord(A). Зафиксируем значения наборов z и u. Пусть истинна A(z: u). Тогда в соответствии с определением (2.1) истинна формула K(z: u). Из этого следует, что исполнение K(z: u) для набора z завершается получением набора u. При завершении исполнения операторов (2.9) справедливы равенства x = z и u = y. Поэтому в последовательности операторов (2.9) существует ис-

полнение K(x: y), завершающееся получением y. Тогда исполнение последовательности (2.9), а значит и исполнение A(z: u), завершается получением значений набора u, что доказывает первую часть свойства Coord(A).

Пусть исполнение A(z: u) для набора z завершается получением набора u. Тогда завершается исполнение соответствующей последовательности (2.9), в частности, исполнение K(x: y) завершается получением y. Поскольку реализуются равенства x = z и u = y, то исполнение K(z: u) завешается вычислением набора u. Из Coord(K) следует, что K(z: u) истинно. Тогда из определения (2.1) следует истинность A(z: u), что доказывает вторую часть свойства Coord(A). □

2.6. Оператор суперпозиции

Оператор суперпозиции является правой частью определения предиката вида "суперпозиция":

$$A(x: y) \equiv B(x: z);\ C(z: y) \tag{2.10}$$

Здесь, x, y и z — произвольные непересекающиеся наборы переменных, причем набор x может быть пустым. Переменные набора z являются *локальными* переменными. Предикаты B и C должны принадлежать языку ССР. Определение (2.10) является программной формой для следующей вычислимой формулы, определяющей композицию суперпозиции:

$$A(x: y) \equiv \exists z\ (B(x: z)\ \&\ C(z: y)) \tag{2.11}$$

Допустим, определение предиката A исполняется в рамках секции s. Секция s состоит из переменных наборов x, y и z. Исполнение правой части определения (2.10) соответствует оператору runRight(s, K(x: y)) в (2.9) и определяется следующим образом:

```
runCall(s, B(x: z));
runCall(s, C(z: y))
```
(2.12)

Таким образом, алгоритм вычисления суперпозиции состоит в последовательном исполнении вызовов предикатов B и C. Значения переменных набора z, вычисленные при исполнении предиката B, используются при вычислении предиката C.

Лемма 2.2. Coord(B) & Coord(C) ⇒ Coord(A).

Доказательство. Пусть истинно Coord(B) и Coord(C). Докажем истинность Coord(A). Пусть истинна A(x: y). Тогда истинна правая часть (2.11). Для некоторого z будут истинны B(x: z) и C(z: y). Из Coord(B) и Coord(C) следует, что исполнение B(x: z) завершается получением набора z, а исполнение C(z: y) — вычислением набора y. Поэтому исполнение операторов (2.12), а значит и A(x: y), завершается вычислением y. Это доказывает первую часть свойства Coord(A).

Допустим, исполнение A(x: y) завершается вычислением y. Докажем истинность A(x: y). Исполнение пары операторов (2.12) завершается вычислением y. В частности, завершается первый оператор runCall(s, B(x: z)). Тогда существует некоторый набор z, являющийся результатом исполнения. Таким образом, для некоторого z исполняются вызовы B(x: z) и C(z: y). Из Coord(B) и Coord(C) следует истинность B(x: z) и C(z: y). Следовательно, истинна правая часть (2.11) и A(x: y). □

2.7. Параллельный оператор

Параллельный оператор является правой частью определения предиката вида “параллельная композиция”:

$$A(x: y, z) \equiv B(x: y) \;||\; C(x: z) \tag{2.13}$$

Здесь, x, y и z — произвольные непересекающиеся наборы переменных, причем набор x может быть пустым. Предикаты B и C должны принадлежать языку ССР. Определение (2.13) является программной формой для вычислимой формулы, определяющей параллельную композицию:

$$A(x: y, z) \equiv B(x: y) \;\&\; C(x: z) \tag{2.14}$$

Допустим, определение предиката A исполняется в рамках секции s. Секция s состоит из переменных наборов x, y и z. Исполнение правой части определения (2.13) соответствует оператору runRight(s, K(x: y)) в (2.9) и определяется следующим образом:

```
runCall(s, B(x: y)) ||                    (2.15)
runCall(s, C(x: z))
```

Алгоритм вычисления параллельного оператора слагается из вычисления вызовов предикатов B и C. Поскольку эти два вычисления между собой не взаимодействуют, они проводятся параллельно.

Лемма 2.3. Coord(B) & Coord(C) ⇒ Coord(A).

2.8. Условный оператор

Условный оператор является правой частью определения предиката вида “альтерация”:

$$A(b, x: y) \equiv \textbf{if } b \textbf{ then } B(x: y) \textbf{ else } C(x: y) \textbf{ end} \qquad (2.16)$$

Здесь, x и y — произвольные непересекающиеся наборы переменных, причем набор x может быть пустым, b — переменная логического типа, не встречающаяся в наборах x и y. Предикаты B и C должны принадлежать языку ССР. Условный оператор является программной формой для вычислимой формулы вида “альтерация”:

$$A(b, x: y) \equiv (b \Rightarrow B(x: y)) \ \& \ (\neg b \Rightarrow C(x: y)) \qquad (2.17)$$

Допустим, определение предиката A исполняется в рамках секции s. Секция s включает переменную b и переменные наборов x и y. Исполнение правой части определения (2.16) соответствует оператору runRight(s, K(x: y)) в (2.9) и определяется следующим образом:

$$\text{if } s[b] \textbf{ then } \text{runCall}(s, B(x: y)) \textbf{ else } \text{runCall}(s, C(x: y)) \textbf{ end} \qquad (2.18)$$

Таким образом, алгоритм исполнения условного оператора зависит от значения переменной b. Если значение b — истинно, исполняется вызов предиката B. В противном случае исполняется вызов предиката C.

Лемма 2.4. Coord(B) & Coord(C) ⇒ Coord(A).

Доказательство. Пусть истинно Coord(B) и Coord(C). Докажем истинность Coord(A). Пусть истинна A(b, x: y). Тогда истинны формулы $b \Rightarrow B(x: y)$ и $\neg b \Rightarrow C(x: y)$. Рассмотрим случай, когда значение b истинно. Тогда истинна формула B(x: y). Из Coord(B) следует, что исполнение вызова B(x: y) завершается получением набора y. Поэтому исполнение оператора (2.18), а значит и A(b, x: y), завершается вычислением y. Это доказывает первую часть свойства Coord(A) для истинного значения b. Доказательство в случае ложного значения b проводится аналогично.

Допустим, исполнение A(b, x: y) завершается вычислением y. Докажем истинность A(b, x: y). Исполнение оператора (2.18) завершается вычислением y. Пусть b истинно. Тогда исполнение вызова B(x: y) завершается получением набора y. Из Coord(B) следует истинность B(x: y). Тогда истинны формулы $b \Rightarrow B(x: y)$ и $\neg b \Rightarrow C(x: y)$, поскольку b истинно. Следовательно, истинна правая часть (2.17) и A(b, x: y). Доказательство истинности A(b, x: y) в случае ложного значения b проводится аналогично. □

2.9. Конструктор предиката

Оператор "конструктор предиката" является базисным предикатом ConsPred в языке ССР и используется для генерации нового предиката как значения предикатного типа (2.8). Предикат ConsPred имеет следующее определение:

$$\text{ConsPred}(x, B: A) \equiv (\forall y)(\forall z)(A(y: z) \equiv B(x, y: z)) \qquad (2.19)$$

Здесь, x, y и z — произвольные непересекающиеся наборы переменных, причем наборы x и y могут быть пустыми. Предикат B должен быть вычислимым, т.е. принадлежать языку ССР. В качестве B нельзя использовать ConsPred или ConsArray. Предикат A является значением предикатного типа PRED(Y: Z), где Y и Z — наборы типов, соответствующие наборам переменных y и z.

Допустим, вызов предиката ConsPred(x, B: A) находится в правой части определения некоторого предиката, исполняемого в рамках секции s. Секция s включает набор x и переменную A предикатного типа. Исполнение вызова ConsPred(x, B: A) реализуется следующим оператором:

$$s[A] := \text{newDef}(s[x], B) \qquad (2.20)$$

Функция newDef строит новое определение предиката:

$$A_x(y: z) \equiv =(x^{\sim}: t);\ B(t, y: z) \qquad (2.21)$$

Здесь, $x^{\sim} = s[x]$ — значение набора x, A_x — новое имя предиката, отличное от имен других предикатов в программе. Имя A_x является значением вызова newDef. Оператор $=(x^{\sim}: t)$ реализует присваивание набора значений $x^{\sim}$ набору локальных переменных t.

Мы не фиксируем, каким образом абстрактный процессор исполняет оператор =(x˜: t). Например, процессор может создавать дополнительные определения, строящие изображения значений x˜. Существует другой способ реализации вызова ConsPred(x, B: A), в котором переменной A присваивается кортеж (s[x], B).

Лемма 2.5. Coord(ConsPred).

Доказательство. Зафиксируем значения $x = x_0$ и $A = A_0$. Пусть истинно ConsPred(x, B: A). Докажем, что исполнение ConsPred(x, B: A) завершается тождественным значением предикатной переменной A, т.е. $A \equiv A_0$. Из истинности ConsPred(x, B: A) следует истинность правой части (2.19). Тогда истинна формула $A_0(y: z) \equiv B(x_0, y: z)$, однозначно определяющая предикат A_0. Рассмотрим исполнение вызова ConsPred(x, B: A), т.е. оператора (2.20). Получаем $A = A_x$, где A_x определяется формулой (2.21). Поскольку $x˜ = s[x] = x_0$, то:

$$A_x(y: z) \equiv =(x_0: t);\ B(t, y: z) \equiv t = x_0\ \&\ B(t, y: z) \equiv B(x_0, y: z) \quad (2.22)$$

Таким образом, $A \equiv A_x \equiv A_0$, что доказывает первую половину леммы.

Допустим, исполнение ConsPred(x, B: A) завершается вычислением $A = A_0$ при $x = x_0$. Докажем истинность ConsPred(x, B: A), т.е. истинность правой части (2.19). Из (2.20) следует $A = A_x$. Из тождества (2.22) следует $A_0(y: z) \equiv B(x_0, y: z)$. Поскольку $A = A_0$ и $x = x_0$, получаем истинность правой части (2.19). □

Пусть имеется вызов A(u: v), где A — переменная предикатного типа. Допустим, что вызов A(u: v) находится в правой части некоторого предиката и q — секция, в которой исполняется определение этого предиката. Исполнение вызова A(u: v) по-прежнему реализуется процедурой runCall(q, A(u: v)), определяемой последовательностью операторов (2.9), в которой первый оператор заменяется на s = newSect(q[A]).

Лемма 2.6. Допустим, что в программе для любого вызова ConsPred(x, B: C) истинно Coord(B). Пусть A — переменная предикатного типа. Тогда истинно Coord(A).

Доказательство. Пусть $A = A_0$ и определение предиката A_0 (вида (2.21)) получено в результате исполнения некоторого вызов ConsPred. Поскольку

B в правой части (2.21) является параметром ConsPred, то истинно Coord(B). Поскольку свойство Coord истинно для оператора =(x˜: t), то по лемме 2.2 оно реализуется также для правой части (2.21). Далее, по лемме 2.1 истинно Coord(A_0), а значит и Coord(A). □

2.10. Конструктор массива

Оператор "конструктор массива" является базисным предикатом ConsArray в языке ССР и используется для генерации нового массива как значения предикатного типа (2.8). Предикат ConsArray имеет следующее определение:

$$\mathsf{ConsArray}(x,\ B\mathbf{:}\ A) \equiv (\forall y)(\forall z)(A(y\mathbf{:}\ z) \equiv B(x,\ y\mathbf{:}\ z)) \qquad (2.23)$$

Здесь, x, y и z — произвольные непересекающиеся наборы переменных, причем набор x может быть пустым. Предикат B должен принадлежать языку ССР. В качестве B нельзя использовать ConsArray или ConsPred. Переменная A имеет предикатный тип PRED(Y: Z), где Y и Z — наборы типов, соответствующие наборам переменных y и z. Типы набора Y являются конечными.

Определение (2.23) тождественно определению (2.19), однако отличается его исполнение. Пусть вызов предиката ConsArray(x, B: A) находится в правой части определения некоторого предиката, исполняемого в рамках секции s. Секция s включает набор x и переменную A, значением которой является массив A˜. Массив A˜ кодирует предикат A следующим образом:

$$A(y\mathbf{:}\ z) \equiv A\tilde{}[y] = z \qquad (2.24)$$

Исполнение вызова ConsArray(x, B: A) реализуется следующими операторами:

```
s[A] = newArray(Y, Z);                                          (2.25)
forAll y∈Y do runCall(s, B(x, y: z)); s[A][y] := z end
```

Первый оператор создает в памяти массив A˜ как значение типа PRED(Y: Z). Массив становится значением переменной A в секции s. Оператор **forAll** реализует полный перебор наборов индексов, принадлежащих набору типов Y, и для каждого набора индексов y исполняет тело оператора **forAll**, причем исполнение тела для разных y может проводиться парал-

лельно. В теле значение z, вычисленное для вызова B(x, y: z) присваивается элементу массива с набором индексов y.

Лемма 2.7. Coord(A).

Доказательство непосредственно следует из соотношения (2.24). □

Лемма 2.8. Допустим, предикат B(x, y: z) определяет однозначную всюду определенную функцию по x и y, т.е. истинно условие:

$$\forall x\ \forall y \in Y\ \exists! z\ B(x, y: z) \tag{2.26}$$

Тогда Coord(B) ⇒ Coord(ConsArray).

Доказательство. Пусть истинно Coord(B). Рассмотрим исполнение вызова оператора **forAll** из (2.25) для некоторого y∈Y. Пусть z — тот единственный набор, для которого истинно B(x, y: z) в соотношении (2.26). Из Coord(B) следует, что исполнение вызова B(x, y: z) завершается получением набора z. Далее будет исполнен оператор s[A][y] := z, в результате чего будет истинно отношение A˜[y] = z, а из (2.24) — истинно A(y: z). В итоге, после исполнения тела **forAll** будет истинно B(x, y: z) & A(y: z). Поскольку z единственно для B(x, y: z) и A(y: z), то будет истинным тождество B(x, y: z) ≡ A(y: z). Дальнейшее доказательство очевидно. □

Допустим, в секции s исполняется вызов предиката A(u: v), где A является переменной предикатного типа, а значением A является массив. Секция s содержит переменную A и наборы u и v. Исполнение вызова A(u: v) реализуется следующим оператором:

$$s[v] := s[A]\ [u] \tag{2.26'}$$

Для исполнения вызова A(u: v) в разд. 2.5 принято обозначение runCall(s, A(u: v)).

2.11. Программа

Программа на языке ССР состоит из конечного замкнутого набора определений предикатов. Для каждого определения правой частью является оператор суперпозиции (2.10), параллельный оператор (2.13) или условный оператор (2.16). Первичными конструкциями, используемыми для построе-

ния операторов, являются вызовы предикатов. В качестве имени вызываемого предиката может быть:

a) имя базисного предиката;
b) имя предиката, имеющего определение в программе;
c) аргумент предиката — имя переменной предикатного типа; в правой части определения предиката встречается данный вызов;
d) имя переменной предикатного типа, являющейся результатом вызова ConsPred или ConsArray, находящегося в правой части того же определения, что и данный вызов.

Имя предиката может использоваться в качестве аргумента другого вызова, в частности, вызова ConsPred или ConsArray. В этом случае имя должно соответствовать одному из условий a) – d).

Набор определений предикатов является *замкнутым*, если для всякого вхождения имени предиката в качестве имени вызываемого предиката или в качестве аргумента другого вызова реализуется одно из условий a) – d). При этом в случае b) предикат должен иметь определение в рассматриваемом наборе определений. Набор определений программы должен быть замкнутым.

Имя определяемого предиката должно быть отлично от имен базисных предикатов, имен других определяемых предикатов, а также от имен типов, используемых в программе. Если в программе используется подмножество типа (2.4), то предикат, определяющий подмножество, должен быть либо базисным, либо иметь определение в программе.

Исполнение программы заключается в исполнении некоторого вызова предиката A(z: u), где A — имя предиката, определяемого в программе, z и u — наборы переменных. Исполнение программы реализуется следующей последовательностью операторов:

```
s = newSect(z, u);                                    (2.27)
input(s[z]);
runCall(s, A(z: u));
output(s[u])
```

Здесь s — секция, состоящая из переменных наборов z и u. Оператор input реализует ввод некоторых значений набора z в секции s. Далее в секции s исполняется вызов предиката A(z: u).

2.12. Рекурсивные определения предикатов

Для определяемых предикатов B и C отношение depend(B, C) обозначает *непосредственную зависимость* B от C, если в правой части определения B имеется вызов предиката C. Предикат B *определяется через* предикат C, def(B, C), если существуют предикаты D_1, D_2, …, D_n ($n > 0$) такие, что $B = D_1$, $C = D_n$ и $depend(D_j, D_{j+1})$ ($j = 1, \ldots, n-1$).

Определение предиката B *рекурсивно*, если истинно отношение def(B, B). Для рекурсивно определяемого предиката B *рекурсивным кольцом* называется набор предикатов $rec(B) = \{C \mid def(B, C) \,\&\, def(C, B)\}$. Очевидно, что если $C \in rec(B)$, то $rec(B) = rec(C)$.

Потенциально возможны более сложные формы рекурсии, поскольку имеются другие формы зависимостей между предикатами. Предикат B *косвенно зависит* от C, если в правой части определения B имеется вызов предиката F(x, C: ...), где C — предикат, имеющий определение в программе. Вызов F(x, C: ...) определяет также зависимость между предикатами F и C, так как предикат C вызывается в правой части определения F. Предполагается, что рекурсия реализуется только через непосредственную зависимость предикатов и не использует других форм зависимостей. Иначе говоря, более сложные формы рекурсии запрещаются. Мы считаем, что в них нет необходимости. При наличии вызова ConsPred(x, B: A) сложной формой рекурсии также является рекурсия для предиката B в случае, когда предикат A вызывается в правой части определения предиката B.

Пусть имеется рекурсивное кольцо предикатов $A_1, A_2, \ldots, A_n$:

$$A_i(x_i: y_i) \equiv K_i(x_i: y_i);\ i = 1...n;\ n > 0, \tag{2.28}$$

где x_i и y_i — наборы переменных, K_i — оператор суперпозиции (2.10), параллельный оператор (2.13) или условный оператор (2.16). В операторах K_i могут встречаться вызовы предикатов $A_1, A_2, \ldots, A_n$, а также вызовы других предикатов, не принадлежащих кольцу и определяемых вне кольца.

Предикаты $A_1, A_2, \ldots, A_n$ являются решением системы уравнений (2.28). Перепишем систему (2.28) в векторной форме:

$$A = K(A), \tag{2.29}$$

где $A = (A_1, A_2, \ldots, A_n)$ — вектор имен предикатов, $K = (K_1, K_2, \ldots, K_n)$ — вектор операторов. При этом $K_i(x_i: y_i)$ рассматривается как $K_i(A_1, A_2, \ldots, A_n)$.

Введем отношение $\sqsubseteq$ на векторах предикатов:

$$B \sqsubseteq C \cong \forall i=1..n\ \forall x_i\ \forall y_i\ (B_i(x_i: y_i) \Rightarrow C_i(x_i: y_i)) \qquad (2.30)$$

Определим вектор $\Phi = (F_1, F_2,..., F_n)$, где $F_1, F_2,..., F_n$ — тотально ложные предикаты, т.е. $\forall i=1..n\ \forall x_i\ \forall y_i\ (\neg F_i(x_i: y_i))$. Вектор Φ является минимальным элементом, т.е. выполняется: $\forall B\ (\Phi \sqsubseteq B)$. Множество векторов предикатов с отношением $\sqsubseteq$ является полной решеткой [2]. Множество предикатов с отношением $\Rightarrow$ также является полной решеткой.

Последовательность векторов предикатов $\{B^m\}_{m\geq 0}$ является *возрастающей цепью*, если $B^0 \sqsubseteq B^1 \sqsubseteq ... \sqsubseteq B^m \sqsubseteq ...$. Для наименьшей верхней грани цепи векторов предикатов $\{B^m\}_{m\geq 0}$ используется обозначение $\cup_{m\geq 0} B^m$. Аналогичным образом определяется возрастающая цепь предикатов и наименьшая верхняя грань цепи предикатов.

Рассмотрим цепь векторов предикатов $\{A^m\}_{m\geq 0}$, определяемую следующим образом:

$$A^0 = \Phi,\ \ A^{m+1} = K(A^m),\ \ m \geq 0 \qquad (2.31)$$

Доказано [2], что решением системы (2.29) является $A = \cup_{m\geq 0} A^m$ — наименьшая неподвижная точка оператора K. Доказательство базируется на монотонности и непрерывности оператора суперпозиции (2.10), параллельного оператора (2.13) и условного оператора (2.16) относительно вхождений вызовов предикатов B и C. При этом установлена немонотонность условного оператора (2.16) относительно вхождения логической переменной b. По этой причине запрещается рекурсия относительно вхождения b в условном операторе.

Рассмотрим подробнее структуру рекурсивного кольца предикатов. Пусть предикат D принадлежит кольцу и имеет определение, в правой части которого вызываются предикаты B и C. Тогда один из предикатов (или оба) принадлежат кольцу. В противном случае предикат D не будет принадлежать кольцу. В соответствии с определением цепи (2.30) $D^0 = F$, где F — тотально ложный предикат. Далее, $D^1 = F$, если D имеет определение в виде оператора суперпозиции или параллельного оператора, поскольку конъюнкция двух вызовов предикатов, из которых один ложный, дает ложный предикат. Если D имеет определение в виде условного оператора, а вызываемые предикаты B и C оба принадлежат кольцу, то также имеем $D^1 = F$. Если во всех условных операторах, используемых в определениях предикатов кольца, оба вызываемых предиката принадлежат кольцу, то решением системы (2.28) является набор тождественно ложных предикатов. Поэтому в дальнейшем будем считать, что в системе (2.28) имеется по

крайней мере один предикат с определением в виде условного оператора, причем в правой части один вызываемый предикат принадлежит кольцу, а другой — нет.

Лемма 2.9. Пусть $\{C^m(x{:}\ y)\}_{m\geq 0}$ является возрастающей цепью предикатов и $C^{\sim} = \cup_{m\geq 0} C^m$. Тогда $\forall x\ \forall y.\ C^{\sim}(x{:}\ y) \Rightarrow \exists k\ C^k(x{:}\ y)$.

Доказательство см. в работе [2]. □

Теорема 2.1. Допустим, что свойство согласованности Coord(φ) реализуется для всякого базисного предиката φ языка ССР. Пусть имеется программа на языке ССР, и ее исполнение реализуется вызовом предиката D(z**:** u). Допустим, что при наличии в программе вызова вида ConsArray(x, B**:** A) предикат B определяет однозначную всюду определенную функцию. Тогда истинно Coord(D).

Доказательство. Рассмотрим случай, когда предикат D является базисным. Тогда истинность Coord(D) следует из первого условия теоремы, кроме случая, когда вызов D(z**:** u) имеет вид ConsArray(x, B**:** A). Поскольку выполняется условие леммы 2.8, то для истинности Coord(ConsArray) достаточно установить истинность Coord(B) для определяемого предиката B, поскольку B отлично от ConsArray (см 2.2), а для других базисных предикатов истинность установлена. Доказательство истинности Coord(B) для определяемого предиката B в свою очередь сводится к установлению истинности Coord(ConsArray) для части программы, содержащей меньшее число[5] определяемых предикатов, что позволяет использовать доказательство по индукции.

Рассмотрим случай, когда программа не содержит рекурсивно определяемых предикатов. Пусть предикат D имеет определение в виде оператора суперпозиции (2.10), параллельного оператора (2.13) или условного оператора (2.16). В соответствии с леммами 2.2, 2.3 и 2.4 достаточно установить свойство согласованности для предикатов B и C, вызываемых в правой части определения. Кроме того, следует доказать свойство согласованности для предикатов, являющимися аргументами вызовов предикатов B и C. Если все эти предикаты — базисные, то для них это свойство установлено. Если какой-то из этих предикатов — определяемый, то его полное замкну-

[5] Здесь используется, что B не может зависеть от A из-за принятых ограничений на сложные формы рекурсии.

тое определение представляется частью программы меньшего размера, что позволяет применить доказательство по индукции.

Пусть имеется рекурсивное кольцо предикатов (2.28). Систему определений (2.28) представим в виде:

$$A = K(A_1,\ A_2, \ldots,\ A_n, B_1, B_2, \ldots, B_s) \qquad (2.32)$$

Здесь параметрами K перечислены все имена предикатов, используемых в правых частях определений (2.28). Предположим, что предикаты $B_1, B_2, \ldots, B_s$ не являются рекурсивно определяемыми. Докажем, что предикаты кольца обладают свойством согласованности. Пусть предикат D(z: u) принадлежит кольцу, т.е. $D = A_j$ для некоторого j. Докажем истинность Coord(D). Зафиксируем значения наборов z и u.

Допустим, D(z: u) истинно и докажем, что исполнение вызова D(z: u) на выбранных значениях набора z завершается вычислением значений набора u. В цепи (2.31) векторов предикатов $\{A^m\}_{m\geq 0}$ рассмотрим компоненту j каждого вектора цепи. Последовательность таких компонент определяет возрастающая цепь векторов $\{D^m\}_{m\geq 0}$ по отношению "$\Rightarrow$", причем $D = \cup_{m\geq 0} D^m$. Из леммы 2.9 следует, что существует k, при котором D^k(z: u) истинно.

Рассмотрим следующую систему определений предикатов:

$$A^0 = \Phi,\ \ A^{m+1} = K(A^m),\ \ 0 \leq m < k \qquad (2.33)$$

Каждая подсистема $A^{m+1} = K(A^m)$ может быть построена из системы определений (2.28) заменой всякого A_i в правой части на имя A_i^{m+1} и всякого A_i в левой части на имя A_i^m. Нетрудно показать, что исполнение вызова D(z: u) в системе (2.28) будет эквивалентно исполнению в системе (2.33) до тех пор, пока исполнение проводится в подсистемах с номерами $m > 0$. Эквивалентность исполнения нарушается при $m = 0$.

Система (2.33) не содержит рекурсивно определяемых предикатов. Предикаты этой системы обладают свойством согласованности, в частности, истинно Coord(D). Поэтому исполнение вызова D(z: u) в системе (2.33) завершается значениями набора u. Остается доказать, что исполнение в системах (2.33) и (2.28) эквивалентно, т.е. исполнение в (2.33) не достигает вызовов предикатов вектора A^0. Предположим противное. Допустим, исполняется вызов F(...) подсистемы $A^1 = K(A^0)$, а в правой части определения предиката F срабатывает вызов предикаты вектора A^0. Поскольку этот вызов ложный, нетрудно показать, что вызов F(...) также будет ложным.

Как следствие, будет ложным Coord(F), что приводит к противоречию. Таким образом, первая часть доказательства истинности Coord(D) закончена.

Предположим, что для фиксированных значений z исполнение вызова D(z: u) в системе (2.28) завершается выбранными значениями набора u. Докажем истинность D(z: u). Процесс исполнения вызова D(z: u) слагается из исполнений вызовов, встречающихся в исполняемых определениях предикатов. Допустим, что суммарное число исполненных вызовов есть k. Для данного k рассмотрим систему (2.33). Исполнение вызова D(z: u) в системе (2.33) тождественно исполнению в системе (2.28), поскольку исполнение в (2.33) не достигает предикатов вектора A^0. Предикаты системы (2.33) обладают свойством согласованности. Поэтому D(z: u) истинно, что доказывает вторую часть истинности Coord(D).

Таким образом, предикаты рекурсивного кольца (2.32) обладают свойством согласованности при условии, что предикаты $B_1, B_2,..., B_s$ не являются рекурсивно определяемыми. Это свойство предикатов кольца можно доказать для более общего случая, когда предикаты $B_1, B_2,..., B_s$ обладают свойствами согласованности. Для этого нужно доказать, что предикаты системы (2.33), также базирующейся на $B_1, B_2,..., B_s$, являются согласованными. Здесь применяется индукция по числу правил.

Рассмотрим произвольную программу на языке ССР. Программа содержит несколько рекурсивных колец предикатов. Пусть кольцо α определяется системой (2.32). Кольцо α *зависит* от кольца β, если один из предикатов $B_1, B_2,..., B_s$ принадлежит другому кольцу β. Рекурсивные кольца образуют граф. Вершинами являются кольца, а дугами — отношения зависимости колец. Допустим, что имеется цикл в графе колец. Нетрудно доказать, что предикаты разных колец, находящихся на цикле, должны принадлежать одному кольцу. Следовательно, граф не имеет циклов и является деревом.

Докажем свойство согласованности для предикатов произвольного рекурсивного кольца α, определяемого системой (2.32). Рассмотрим различные пути от α до листьев дерева колец. Пусть k — максимальная длина пути по этому набору путей. Доказательство проводится индукцией k. При $k = 0$ предикаты $B_1, B_2,..., B_s$ не являются рекурсивно определяемыми. Для этого случая свойство согласованности доказано ранее. Предположим, что свойство согласованности доказано для колец с меньшим значением k, и докажем его для кольца α. Рассмотрим предикат B_j ($j=1,...,s$). Пусть B_j принадлежит рекурсивному кольцу β. Из кольца α имеется дуга в кольцо β. Поэтому максимальная длина пути для β по крайней мере на единицу меньше, чем k. По индуктивному предположению предикаты кольца β об-

ладают свойством согласованности. Таким образом, предикаты $B_1, B_2,..., B_s$ обладают свойством согласованности. Для такого условия свойство согласованности предикатов кольца α также была установлена. □

Доказательство свойств рекурсивно определяемых предикатов далее реализуется с использованием метода математической индукции. Определим подкласс рекурсивно определяемых предикатов, соответствующий этому методу. Среди аргументов предикатов будем выделять *индукционные* переменные, используемые в организации рекурсии. Рассмотрим рекурсивное кольцо предикатов. Пусть в этом кольце используется набор t индукционных переменных. Будем считать, что переменные набора t являются аргументами каждого предиката в рекурсивном кольце. Систему определений (2.28) представим в виде:

$$A_i(t, x_i: y_i) \equiv K_i(t, x_i: y_i);\ i = 1...n;\ n > 0, \qquad (2.34)$$

На множестве значений набора t задан частичный порядок R, удовлетворяющий следующему свойству обрыва бесконечных убывающих цепей[6]. Для всякого непустого подмножества S значений набора t имеется минимальный элемент $r \in S$, т.е. если для $u \in S$ истинно $R(u, r)$, то $u = r$. Данное свойство эквивалентно отсутствию в S бесконечных убывающих цепей. Отметим, что переменные типа REAL не удовлетворяют данному свойству. Пусть T — набор типов, соответствующий набору переменных t. Через U обозначим множество минимальных элементов T по отношению к R.

Допустим, что для предикатов кольца (2.34) требуется доказать некоторое свойство $W(t)$, возможно зависящее и от других переменных. Будем использовать версию математической индукции, являющуюся комбинацией структурной и полной индукции [5–7]. Для доказательства свойства $W(t)$ требуется доказать следующую формулу:

$$\{\forall t.\ (\beta(t) \Rightarrow W(t)) \ \&\ \forall t.\ (\forall u.\ R(u, t) \ \&\ u \neq t \ \&\ W(u) \Rightarrow W(t))\} \Rightarrow \forall t.\ W(t) \qquad (2.35)$$

Здесь предикат $\beta(t)$ определяет *базу индукции*. В нее должны попасть все минимальные элементы, т.е. $U \subseteq \{t \mid \beta(t)\}$. В соответствии с формулой (2.35) метод структурной индукции заключается в следующем. Сначала свойство $W(t)$ доказывается для базы индукции. Далее для произвольного t делается *индукционное предположение*, что свойство $W(u)$ истинно для всех наборов u, меньших t. С использованием индукционного предположе-

[6] Соответствующий английский термин — well-founded relation.

ния доказывается W(t). Если это удается, то в соответствии с методом индукции свойство W(t) является истинным.

Определим ограничения на вызовы предикатов рекурсивного кольца в правых частях системы определений (2.34). Допустим, предикат D принадлежит кольцу и имеется вызов D(u, z: v), где набор u соответствует позициям индукционных переменных. Тогда $u = \alpha(t)$ для некоторой операции α, причем истинно $R(\alpha(t), t)$. Таким образом, для вызова D(u, z: v) становится истинным индукционное предположение, что обеспечивает применимость метода математической индукции.

3. МОДЕЛЬ КОРРЕКТНОСТИ ПРОГРАММЫ

Описание преобразования информации, реализуемого вычислением программы, называется *спецификацией* программы. Спецификация описывает связь между входными и выходными потоками информации программы [1]. Спецификация программы называется *предикатной*, если она эксплицируема в виде формулы на языке исчисления предикатов. Для программы с предикатной спецификацией характерна простая схема взаимодействия программы с окружением: ввод входных данных происходит в начале исполнения программы, вывод результирующих данных — в конце, а других взаимодействий с окружением нет. Спецификация определяет функцию, отображающую значения входных данных в значения результатов.

Спецификация программы на языке ССР состоит из спецификаций определений предикатов, образующих программу. Пусть имеется определение предиката:

$$A(x: y) \equiv K(x: y), \tag{3.1}$$

где K — оператор. В качестве *спецификации предиката* A будем рассматривать некоторую формулу $S_A(x, y)$ исчисления предикатов.

Свойства, определяющие соотношение между спецификацией и программой, определены в работе [1]. Главное свойство: программа должна соответствовать спецификации. Для программы на языке ССР данное свойство формулируется следующим образом: каждое определение предиката должно соответствовать спецификации этого предиката. Это значит, что для определения (3.1) значения наборов переменных x и y после выполнения предиката A(x: y) должны удовлетворять спецификации $S_A(x, y)$. Данное свойство записывается следующим образом:

$$RUN(A, x, y) \Rightarrow S_A(x, y)$$

Согласно теореме 2.1 предикат $A(x{:}\ y)$ вычисляет набор y по набору x тогда и только тогда, когда $A(x{:}\ y)$ истинно. Далее, $A(x{:}\ y)$ эквивалентно $K(x{:}\ y)$. Поэтому соответствие определения предиката и его спецификации может быть определено условием:

$$K(x{:}\ y) \Rightarrow S_A(x, y) \tag{3.2}$$

Для условия (3.2) импликация в обратную сторону в общем случае неверна. Спецификация $S_A(x, y)$ может оказаться многозначной, если рассматривать ее как функцию, отображающую x в y.

В формуле $S_A(x, y)$ можно выделить часть $P_A(x)$, зависящую только от x. Представим спецификацию $S_A(x, y)$ в следующем виде:

$$S_A(x, y) \equiv P_A(x) \,\&\, Q_A(x, y) \tag{3.3}$$

При исполнении определения предиката (3.1) значение формулы $Q_A(x, y)$ определено после исполнения оператора $K(x{:}\ y)$, тогда как значение формулы $P_A(x)$ определено до начала исполнения $K(x{:}\ y)$. Назовем $P_A(x)$ *предусловием* предиката A, а $Q_A(x, y)$ — *постусловием*[7]. Определение предиката со спецификацией в виде предусловия и постусловия будем записывать в следующем виде:

$$A(x{:}\ y) \equiv P_A(x)\ \{K(x{:}\ y)\}\ Q_A(x, y) \tag{3.4}$$

Предусловие $P_A(x)$ не зависит от $K(x{:}\ y)$, поскольку значение $P_A(x)$ определено до исполнения $K(x{:}\ y)$. Поэтому бессмысленно рассматривать доказательство формулы $K(x{:}\ y) \Rightarrow P_A(x)$ как части условия (3.2). В действительности, предусловие считается истинным априори, и определение предиката (3.4) не рассматривается для ложного предусловия. Истинность предусловия должна быть доказана для каждого вызова предиката. В связи с этим правило (3.2) соответствия определения предиката его спецификации должно быть уточнено следующим образом:

$$P_A(x) \,\&\, K(x{:}\ y) \Rightarrow Q_A(x, y) \tag{3.5}$$

[7] Определения предусловия и постусловия согласуются с введенными Хоаром в работе [12].

Спецификация $S_A(x\ y)$ *реализуема* для конкретного x, если $S_A(x\ y)$ истинна для некоторого y, т.е. истинна формула: $\exists y.\ S_A(x\ y)$. Реализуемость спецификации $S_A(x\ y)$ на x эквивалентно реализуемости постусловия $Q_A(x, y)$ для этого x.

Рассмотрим ситуацию, когда для некоторого x истинно предусловие $P_A(x)$, а постусловие $Q_A(x, y)$ не реализуемо для этого x. Поскольку мы предполагаем истинность условия (3.5), то тогда формула $K(x: y)$ будет ложной для всех y. В соответствии с теоремой 2.1 исполнение $K(x: y)$ для данного x не определено. На практике это означает, что исполнение реализуется бесконечно, приводит к аварийной ситуации или дает непредсказуемый результат. Подобная ситуация абсурдна с точки зрения практического применения программы[8]. Отметим, что бесконечное исполнение $K(x: y)$ бессмысленно, поскольку оно не оказывает воздействий на окружение программы — для предикатной спецификации взаимодействия программы с окружением невозможны в середине исполнения программы.

Таким образом, мы приходим к следующему условию *корректности спецификации*: постусловие должно быть реализуемо для всех x, на которых истинно предусловие:

$$P_A(x) \Rightarrow \exists y.\ Q_A(x, y) \qquad (3.6)$$

Формула $H(x, y)$ является *реализуемой для предусловия* $P_A(x)$, если она реализуема для всех x, на которых истинно предусловие. Спецификация является *реализуемой*, если она реализуема для предусловия. Таким образом, спецификация корректна, если она реализуема.

Определим *слабейшее предусловие*[9] $PW_A(x)$ следующей формулой:

$$PW_A(x) \cong \exists y.\ S_A(x, y) \qquad (3.7)$$

Лемма 3.1. Спецификация $S_A(x, y)$ с предусловием $P_A(x)$ реализуема $\Leftrightarrow$ $P_A(x) \Rightarrow PW_A(x)$.

Допустим, что формула $K(x: y)$ нереализуема для некоторого x, однако x удовлетворяет предусловию $P_A(x)$, т.е. истинна формула: $P_A(x)\ \&\ (\forall y)\ \neg K(x: y)$. Тогда для данного x истинно условие (3.5), и значит определение предиката (3.4) соответствует спецификации для x. Поскольку $K(x: y)$ ложно, исполнение $K(x: y)$ не определено для x. Итак, имеем ситуа-

[8] См. модель применения программы в работе [1].

[9] Это понятие введено Э. Дейкстра [3]; E. Hehner использует термин “точное предусловие” [4].

цию, когда спецификация $S_A(x, y)$ истинна, определение предиката соответствует спецификации, а исполнение предиката не определено. Следовательно, необходимым условием корректности определения предиката должна быть реализуемость оператора $K(x: y)$ для предусловия $P_A(x)$, т.е. должна быть истинна формула:

$$P_A(x) \Rightarrow \exists y.\ K(x: y) \qquad (3.8)$$

Приведенные выше требования объединим в понятии корректности определения предиката. Прежде всего, в определении предиката (3.4) предусловие $P_A(x)$ всегда истинно. Определение предиката *корректно*[10], если спецификация реализуема (условие (3.6)), правая часть определения реализуема в области предусловия (условие (3.8)) и определение предиката соответствует спецификации (условие (3.5)).

Требования, составляющие корректность определения предиката, избыточны.

Лемма 3.2. Если правая часть определения предиката реализуема для предусловия и определение предиката соответствует спецификации, то спецификация реализуема.

Следовательно, корректность определения предиката (3.4) может быть представлена как истинность следующей формулы:

$$P_A(x) \Rightarrow (\exists y.\ K(x: y)) \ \&\ (K(x: y) \Rightarrow Q_A(x, y)) \qquad (3.9)$$

Таким образом, общее правило соответствия программы ее спецификации [1] конкретизируется для программы на языке ССР следующим образом. Во-первых, спецификация каждого предиката подразделяется на предусловие и постусловие, причем предусловие считается истинным для определения предиката. Во-вторых, правило (3.5) соответствия предиката его спецификации дополняется условиями реализуемости спецификации и реализуемости правой части определения предиката.

[10] Данное определение согласуется с известным понятием тотальной корректности программы.

4. СИСТЕМА ПРАВИЛ ДОКАЗАТЕЛЬСТВА КОРРЕКТНОСТИ ПРОГРАММЫ

Определим способы доказательства корректности определения предиката по формуле (3.9) для оператора суперпозиции, параллельного оператора или условного оператора в позиции оператора K(x: y) в формуле (3.9). Способы доказательства формулируются в виде правил вывода, истинность которых гарантирует истинность формулы (3.9). При этом предполагается корректность предикатов B и C, вызываемых внутри оператора K(x: y). Правила вывода используют только спецификации предикатов A, B и C. Это позволит проводить доказательство корректности на уровне спецификаций без использования правой части определения предиката.

4.1. Правила для вызова предиката

Пусть имеется вызов B(z: t) внутри оператора K(x: y) в определении (3.4) предиката A. Пусть имеется определение

$$B(u: v) \equiv P_B(u)\ \{\ \dots\ \}\ Q_B(u, v)$$

Пусть определение предиката B является корректным. Тогда истинны следующие формулы:

$$P_B(u) \Rightarrow \exists v.\ Q_B(u, v) \qquad (4.1)$$
$$P_B(u) \Rightarrow \exists v.\ B(u: v) \qquad (4.2)$$
$$P_B(u)\ \&\ B(u: v) \Rightarrow Q_B(u, v) \qquad (4.3)$$

В отличие от формул (3.5) и (3.8) здесь правая часть определения B заменена эквивалентной ей левой частью.

Если B является базисным предикатом, то мы предполагаем наличие спецификации в виде предусловия $P_B(u)$ и постусловия $Q_B(u, v)$, для которых обеспечена истинность формул (4.1)–(4.3). В этом случае мы будем говорить, что определение базисного предиката корректно.

Итак, пусть имеется вызов B(z: t) в правой части определения (3.4) для некоторого предиката A, причем определение предиката B является корректным. Допустим, доказана истинность $P_B(z)$. Тогда в соответствии с (4.1)–(4.3) будут истинны формулы:

$$\exists v.\ Q_B(z, t), \qquad \exists t.\ B(z: t), \qquad B(z: t) \Rightarrow Q_B(z, t) \qquad (4.4)$$

Приведенное утверждение сформулируем в виде следующих правил.

Правило RC1. $P_B(z) \vdash \exists t.\ Q_B(z, t)$
Правило RC2. $P_B(z) \vdash \exists t.\ B(z{:}\ t)$
Правило RC3. $P_B(z) \vdash B(z{:}\ t) \Rightarrow Q_B(z, t)$

Таким образом, для всякого вызова предиката $B(z{:}\ t)$ необходимо доказывать истинность предусловия $P_B(z)$. После такого доказательства может быть применено любое из трех правил. В результате оказываются истинными формулы из правых частей правил, что используется для доказательства корректности определения предиката A, содержащего вызов $B(z{:}\ t)$ в правой части.

4.2. Правила для параллельного оператора

Рассмотрим определение предиката, правой частью которого является параллельный оператор:

$$A(x{:}\ y, z) \equiv P_A(x)\ \{B(x{:}\ y) \mid\mid C(x{:}\ z)\}\ Q_A(x, y, z) \qquad (4.5)$$

Предполагая корректность определений предикатов B и C, определим правила, гарантирующие корректность определения (4.5).

Правило RP1. $P_A(x) \vdash P_B(x)\ \&\ P_C(x)$.
Правило RP2. $Q_B(x, y), Q_C(x, z) \vdash Q_A(x, y, z)$.

Лемма 4.1. Пусть предусловие $P_A(x)$ истинно. Допустим, определения предикатов B и C корректны. Если правила RP1 и RP2 истинны (т.е. правая часть доказуема из левой части для каждого правила), то определение предиката (4.5) корректно.

Доказательство. Для доказательства корректности определения (4.5) в соответствии с формулой (3.9) достаточно доказать реализуемость правой части определения (4.5) и выводимость постусловия $Q_A(x, y, z)$ из правой части. Здесь в качестве правой части рассматривается формула $B(x{:}\ y)\ \&\ C(x{:}\ z)$, определенная в разд. 2.7 как эквивалент $B(x{:}\ y) \mid\mid C(x{:}\ z)$.

Из истинности предусловия $P_A(x)$ по правилу RP1 следует истинность $P_B(x)$ и $P_C(x)$. Далее, по правилу RC2 становятся истинными формулы $\exists y.\ B(x{:}\ y)$ и $\exists z.\ C(x{:}\ z)$. Их конъюнкция определяет реализуемость правой части определения (4.5).

Докажем выводимость постусловия $Q_A(x, y, z)$ из правой части. Допустим, истинна правая часть, т.е. формула $B(x: y)$ & $C(x: z)$. Применим правило RC3 для $P_B(x)$ и $P_C(x)$, истинность которых определена выше. Получаем истинность формул $B(x: y) \Rightarrow Q_B(x, y)$ и $C(x: z) \Rightarrow Q_C(x, z)$. Как следствие, будут истинны $Q_B(x, y)$ и $Q_C(x, z)$. Применяя правило RP2, получаем истинность постусловия $Q_A(x, y, z)$. □

4.3. Правила для оператора суперпозиции

Рассмотрим определение предиката, правой частью которого является оператор суперпозиции:

$$A(x: y) \equiv P_A(x) \{B(x: z); C(z: y)\} Q_A(x, y) \qquad (4.6)$$

Предположим, что определения предикатов B и C корректны. Определим правила, гарантирующие корректность определения (4.6).

Правило RS1. $P_A(x) \vdash P_B(x)$ & $\forall z\,(Q_B(x, z) \Rightarrow P_C(z))$.
Правило RS2. $P_A(x)$ & $\exists z\,(Q_B(x, z)$ & $Q_C(z, y)) \vdash Q_A(x, y)$.

Лемма 4.2. Пусть предусловие $P_A(x)$ истинно. Допустим, определения предикатов B и C корректны. Если правила RS1 и RS2 истинны, то определение предиката (4.6) корректно.

Доказательство. В соответствии с формулой (3.9) достаточно доказать реализуемость правой части определения (4.6) и выводимость постусловия $Q_A(x, y)$ из правой части. Здесь в качестве правой части рассматривается формула $\exists z.\ B(x: z)$ & $C(z: y)$, определенная в разд. 2.6 как эквивалент $B(x: z); C(z: y)$.

Из истинности предусловия $P_A(x)$ по правилу RS1 следует истинность формул $P_B(x)$ и $\forall z\,(Q_B(x, z) \Rightarrow P_C(z))$. Из истинности $P_B(x)$ и правила RC2 следует истинность формулы $\exists z.\ B(x: z)$. Допустим для некоторого z_0 формула $B(x: z_0)$ истинна. Из истинности $P_B(x)$ и правила RC3 следует истинность $B(x: z_0) \Rightarrow Q_B(x, z_0)$. Как следствие, истинно $Q_B(x, z_0)$. Далее, из истинности формулы $\forall z\,(Q_B(x, z) \Rightarrow P_C(z))$ следует истинность $P_C(z_0)$. По правилу RC2 истинна формула $\exists y\ C(z_0: y)$. Далее, истинна конъюнкция $B(x: z_0)$ & $\exists y\ C(z_0: y)$, и затем — формула $\exists y.\ \exists z.\ (B(x: z)$ & $C(z: y))$, т.е. доказана реализуемость правой части определения предиката A.

Докажем выводимость постусловия $Q_A(x, y)$ из правой части. Допустим, истинна правая часть, т.е. формула $\exists z.\ B(x: z)$ & $C(z: y)$. Пусть формула

истинна для некоторого z_1. По правилу RC3 истинна формула $B(x{:}\ z_1) \Rightarrow Q_B(x, z_1)$ и далее — $Q_B(x, z_1)$. Истинность $Q_B(x, z_1)$ и $\forall z\,(Q_B(x, z) \Rightarrow P_C(z))$ влечет истинность $P_C(z_1)$. По правилу RC3 истинно $C(z_1{:}\ y) \Rightarrow Q_C(z_1, y)$. Поскольку $C(z_1{:}\ y)$ истинно, то истинно $Q_C(z_1, y)$. Таким образом, истинна правая часть правила RS2, а значит и левая, т.е. истинно постусловие $Q_A(x, y)$. □

4.4. Правила для условного оператора

Рассмотрим определение предиката, правой частью которого является условный оператор:

$$A(b, x{:}\ y) \equiv P_A(b, x)\ \{\ \textbf{if}\ b\ \textbf{then}\ B(x{:}\ y)\ \textbf{else}\ C(x{:}\ y)\ \textbf{end}\ \}\ Q_A(x, y) \quad (4.7)$$

Легко видеть, что постусловие $Q_A(x, y)$ не может зависеть от логической переменной b. А переменная b обычно зависит от x[11]. Поэтому предусловие можно представить в следующем виде:

$$P_A(b, x) \equiv (b \equiv \varphi(x))\ \&\ \psi(x) \quad (4.8)$$

В соответствии с (4.8) переменная b может быть далее заменена на $\varphi(x)$, а собственно предусловием является $\psi(x)$. Предположим, что определения предикатов B и C корректны. Определим правила, гарантирующие корректность определения (4.7).

Правило RA1. $\psi(x) \vdash (\varphi(x) \Rightarrow P_B(x))\ \&\ (\neg\varphi(x) \Rightarrow P_C(x))$.
Правило RA2. $\psi(x)\ \&\ (\varphi(x) \Rightarrow Q_B(x, y))\ \&\ (\neg\varphi(x) \Rightarrow Q_C(x, y)) \vdash Q_A(x, y)$.

Лемма 4.3. Пусть предусловие $P_A(b, x)$ истинно, т.е. переменная b тождественна $\varphi(x)$ и истинно $\psi(x)$. Допустим, определения предикатов B и C корректны. Если правила RA1 и RA2 истинны, то определение предиката (4.7) корректно.

Доказательство. В соответствии с формулой (3.9) достаточно доказать реализуемость правой части определения (4.7) и выводимость постусловия $Q_A(x, y)$ из правой части. Здесь в качестве правой части рассматривается формула

$$(b \Rightarrow B(x{:}\ y))\ \&\ (\neg b \Rightarrow C(x{:}\ y)), \quad (4.9)$$

определенная в разд. 2.8 как эквивалент

[11] b может также зависеть от других переменных. Однако зависимость от них не является принципиальной, и мы не будем их указывать.

if b **then** B(x: y) **else** C(x: y) **end**.

Из истинности предусловия $\psi(x)$ по правилу RA1 следует истинность формул $(\varphi(x) \Rightarrow P_B(x))$ и $(\neg\varphi(x) \Rightarrow P_C(x))$. Допустим, что $\varphi(x)$ истинна. Тогда будет истинна $P_B(x)$. По правилу RC2 истинна формула $\exists y.\ B(x: y)$. Далее будет истинной формула $\exists y.\ (\varphi(x) \Rightarrow B(x: y))$. Из истинности $\varphi(x)$ следует истинность формулы $\neg\varphi(x) \Rightarrow C(x: y)$, и следовательно, формулы $\exists y.\ (\varphi(x) \Rightarrow B(x: y)) \ \&\ (\neg\varphi(x) \Rightarrow C(x: y))$. Это обеспечивает реализуемость формулы (4.9) в случае истинности $\varphi(x)$. Реализуемость (4.9) в случае ложности $\varphi(x)$ доказывается аналогичным образом, начиная с установления истинности $P_C(x)$.

Докажем выводимость постусловия $Q_A(x, y)$ из правой части. Допустим, истинна правая часть, т.е. формула (4.9). Заменим b на $\varphi(x)$. Пусть $\varphi(x)$ истинна. Тогда истинна $B(x: y)$. По правилу RA1 истинна $P_B(x)$. По правилу RC3 истинна формула $B(x: y) \Rightarrow Q_B(x, y)$, а значит и $Q_B(x, y)$. Далее, истинна формула $\varphi(x) \Rightarrow Q_B(x, y)$. Поскольку $\varphi(x)$ истинна, то истинна формула $\neg\varphi(x) \Rightarrow Q_C(x, y)$. Таким образом, истинна правая часть правила RA2. Тогда истинна левая часть правила, т.е. истинно постусловие $Q_A(x, y)$. Доказательство истинности постусловия $Q_A(x, y)$ для случая, когда $\varphi(x)$ ложна, проводится аналогично. □

4.5. Правила для рекурсивного кольца предикатов

Пусть имеется система определений для рекурсивного кольца предикатов $A_1, A_2, \ldots, A_n$:

$$A_j(t, x_j: y_j) \equiv P_j(t, x_j)\{K_j(t, x_j: y_j)\}\ Q_j(t, x_j, y_j);\ j = 1\ldots n;\ n > 0 \qquad (4.10)$$

Здесь P_j и Q_j — предусловие и постусловие для предиката A_j, t — набор индукционных переменных. На значениях набора t определен частичный порядок R, удовлетворяющий свойству обрыва бесконечных убывающих цепей. Пусть имеется предикат $\beta(t)$, определяющий базу индукции, причем всякий минимальный элемент удовлетворяет предикату β. Корректность определений рекурсивного кольца представлена формулой W(t) как корректность всех определений кольца:

$$W(t) \equiv \mathop{\&}_{j=1}^{n} \{\ P_j(t, x_j) \Rightarrow (\exists y_j.\ K_j(t, x_j: y_j)) \ \&\ (K_i(t, x_j: y_j) \Rightarrow Q_j(t, x_j, y_j))\ \} \qquad (4.11)$$

Определим правила доказательства корректности определений кольца (4.10) по методу структурной индукции.

Правило RR1. $\beta(t) \vdash W(t)$.
Правило RR2. $\forall u.\ R(u, t)\ \&\ u \neq t\ \&\ W(u) \vdash W(t)$.

Лемма 4.4. Если правила RR1 и RR2 истинны, то определения кольца (4.10) корректны.

Доказательство. Конъюнкция правил RR1 и RR2 дает формулу (2.35) доказательства методом структурной индукции. □

В качестве примера рассмотрим программу на языке ССР для вычисления факториала. Введем предикат F(n: f), обозначающий отношение "f = n!":

$$F(n{:}\ f) \equiv f = n! \equiv (n = 0 \Rightarrow f = 1)\ \&\ (n > 0 \Rightarrow f = n * F(n - 1)) \quad (4.12)$$

Программа является непосредственной реализацией данного определения F(n: f) и представлена ниже определениями предикатов A0, A1, A2, A3, A4 и A5.

```
A0(n: f) ≡          n ≥ 0                                          (4.13)
                    {A1(: c0, c1); A2(n, c0, c1: f)}
                    f = F(n)
A1(: c0, c1 ) ≡     {ConsIntZero(: c0) || ConsIntOne(: c1)}
                    c0 = 0 & c1 = 1
A2(n, c0, c1: f) ≡  n ≥ 0 & c0 = 0 & c1 = 1
                    {=(n, c0: b); A3(n, c1, b: f)}
                    f = F(n)
A3(n, c1, b: f) ≡   n ≥ 0 & c1 = 1 & (b ≡ n = 0)
                    { if b then =(c1: f) else A4(n, c1: f) end }
                    f = F(n)
A4(n, c1: f) ≡      n > 0 & c1 = 1
                    {-(n, c1: n1); A5(n, n1: f)}
                    f = F(n)
A5(n, n1: f) ≡      n > 0 & n1 = n - 1
                    {A0(n1: f1); *(n, f1: f)}
                    f = F(n)
```

Предикаты A0, A2, A3, A4 и A5 образуют рекурсивное кольцо. Переменная n является индуктивной. В качестве отношения порядка R исполь-

зуется отношение "≤" на натуральных числах. Базой индукции является $\beta(n) \equiv n = 0$.

Доказательство корректности определений кольца будем проводить по правилам RR1 и RR2. Пусть истинна посылка правила RR1, т.е. истинно $n = 0$. Требуется доказать $W(0)$, т.е. корректность определений для случая $n = 0$. Докажем корректность определения A3, т.е. истинность формулы (3.9). Пусть истинно предусловие A3. Получим $b =$ **true**. В соответствии с (2.17) условный оператор в определении A3 эквивалентен оператору $=(c1: f)$, откуда истинно $f = 1$. Из определения (4.12) получаем $F(0) = 1$. Таким образом, постусловие $f = F(n)$ выводимо из правой части. Реализуемость правой части также обеспечена. Корректность определения A3 для $n = 0$ доказана. Поскольку истинно $A3(0, c1,$ **true:** $1)$, нетрудно доказать корректность определения A2 для $n = 0$, а затем и определения A0. Корректность определений A4 и A5 для $n = 0$ обеспечена ввиду ложности предусловия.

Пусть истинна посылка правила RR2, т.е. истинно $\forall u.\ u < n \ \&\ W(u)$, в частности, обеспечивается корректность определения A0 для $u < n$. Требуется доказать $W(n)$. Докажем корректность определения A5. Пусть истинно предусловие, т.е. $n > 0$ и $n1 = n - 1$. Поскольку $n1 \geq 0$, то истинно предусловие вызова $A0(n1: f1)$. Определение A0 корректно для $u = n1$. По правилу RC3 истинно $f1 = F(n1)$. Это обеспечивает реализуемость правой части A5. Пусть правая часть истинна. Тогда $f = n * f1 = n * F(n - 1) = F(n)$; последнее равенство следует из (4.12). Корректность определения A5 доказана. Далее доказывается корректность определения A4, используя корректность определения A5, затем корректность определения A3 и т.д. □

5. СИСТЕМА ПРАВИЛ ВЫВОДА ПРОГРАММЫ ИЗ СПЕЦИФИКАЦИИ

Понятие корректности определения предиката, введенное в разд. 3, состоит из трех условий. Главным из них является условие (3.5) соответствия определения предиката спецификации, согласно которому постусловие должно быть доказано из правой части определения предиката. Однако в практике построения программ из математического решения задачи, как правило, реализуется вывод в обратную сторону: программа выводится из спецификации. Нашей целью является определения условий, при которых такой вывод был бы возможен, и построение соответствующих правил до-

казательства корректности программы. Необходимым условием является однозначность спецификации.

5.1. Однозначность предикатов

Предикат A(x: y) является *однозначным* в области X, если он определяет функцию, отображающую значения набора x в значения набора y. Должно быть истинным следующее условие:

$$\forall x \in X\ \forall y_1, y_2.\ A(x: y_1)\ \&\ A(x: y_2) \Rightarrow y_1 = y_2 \qquad (5.1)$$

Отметим, что однозначный предикат в некоторой области X не обязательно является реализуемым в этой области.

Теорема 5.1. Допустим, всякий базисный предикат языка ССР, кроме ConsPred и ConsArray, является однозначным на всем множестве значений аргументов. Тогда любой предикат языка ССР является однозначным.

Доказательство. Рассмотрим произвольное определение предиката A(x: y) ≡ K(x: y) и предположим, что предикаты, вызываемые в операторе K, являются однозначными. Кроме того, если в наборе x имеются переменные предикатного типа, то будем считать, что предикаты, являющиеся значениями таких переменных, являются однозначными. Нетрудно доказать, что предикат A является однозначным, если он имеет определение в виде оператора суперпозиции (2.10), параллельного оператора (2.13) или условного оператора (2.16).

Базисные предикаты ConsPred и ConsArray не являются однозначными. В двух разных исполнениях вызова ConsPred(x, B: A) (при совпадающих x и B) в качестве значения переменной A будут получены разные имена. Однако при этом предикаты, обозначаемые разными именами, будут совпадать, что в итоге обеспечивает однозначность вызовов предиката A при условии, что предикат B является однозначным. Аналогично, будет однозначным предикат, порождаемый вызовом предиката ConsArray.

Предикат A, являющийся переменной предикатного типа, либо является аргументом определения предиката, либо получен вызовом конструктора ConsPred(x, B: A) или ConsArray(x, B: A). Если предикат A является аргументом определения некоторого предиката F, то однозначность A определяется однозначностью всех предикатов подставляемых на место аргумента A в разных вызовах предиката F. В некотором вызове предиката F на место A может быть подставлен либо предикат, полученный конструктором, либо

аргумент другого определения. В последнем случае необходимо прослеживать вызовы предиката, в определении которого встретился вызов F. В конечном счете, для предиката A можно определить полный список предикатов, порождаемых конструкторами и подставляемых на место аргумента A. Предикат A будет однозначным, если однозначны предикаты, являющиеся аргументами конструкторов построенного списка.

Таким образом, для каждого конструктора ConsPred(x, B: A) или ConsArray(x, B: A) достаточно установить однозначность предиката B. Для предиката A определим *граф иерархии*. Корневая вершина A соединена дугой с B. Предикат B также может быть результатом другого конструктора. В этом случае вершина B соединена с предикатом, являющимся аргументом второго конструктора. Кроме того, B может быть аргументом определения предиката. Тогда для B можно определить список вызовов конструкторов, которыми порождается B. В графе B соединяется дугами с предикатами — аргументами вызовов конструкторов из списка. Листьями в графе иерархии являются предикаты, которые не порождаются конструкторами, т.е. базисные или явно определяемые. Граф является деревом, поскольку предикат, аргумент конструктора, не может зависеть от результата конструктора — такой вид рекурсии не допускается в разд. 2.12. Индукцией по длине пути в дереве иерархии однозначность предиката A сводится к однозначности базисных предикатов и явно определяемых предикатов.

Допустим, предикат A имеет определение в программе, не содержащей рекурсивно определяемых предикатов. Однозначность предиката A доказывается индукцией по числу определений в программе.

Пусть имеется рекурсивное кольцо предикатов (2.32). Предикаты $A_1, A_2,\ldots, A_n$ принадлежат рекурсивному кольцу. Предикаты $B_1, B_2,\ldots, B_s$ используются в определениях предикатов кольца, но не принадлежат этому кольцу. Предположим, что предикаты $B_1, B_2,\ldots, B_s$ не содержат рекурсивно определяемых предикатов. Докажем, что предикаты кольца являются однозначными. Пусть предикат D(z: u) принадлежит кольцу, т.е. $D = A_j$ для некоторого j. Зафиксируем значения набора z. Пусть истинны $D(z: u_1)$ и $D(z: u_2)$. Докажем, что $u_1 = u_2$. В цепи (2.31) векторов предикатов $\{A^m\}_{m\geq 0}$ рассмотрим компоненту j каждого вектора цепи. Последовательность таких компонент определяет возрастающая цепь векторов $\{D^m\}_{m\geq 0}$ по отношению "$\Rightarrow$", причем $D = \cup_{m\geq 0} D^m$. Из леммы 2.9 следует, что существуют l и h, для которых $D^l(z: u_1)$ и $D^h(z: u_2)$ истинны. Пусть $k = \max(l, h)$. Тогда истинны $D^k(z: u_1)$ и $D^k(z: u_2)$. В разд. 2.12 введена система определений (2.33), решение которой является также решением исходной системы (2.33). По-

скольку система (2.33) не содержит рекурсивно определяемых предикатов, то предикаты этой системы являются однозначными. Поэтому $u_1 = u_2$.

Оставшаяся часть доказательства реализуется по той же схеме, которая применяется при доказательстве теоремы 2.1. Доказательство проводится индукцией при движении по дереву рекурсивных колец, рассматриваемому для произвольной программы на языке ССР. □

В определении языка ССР мы не фиксировали конкретный набор базисных предикатов, полагая, что он естественным образом покрывает стандартные операции, обычно используемые в программах. Тем не менее, набор базисных предикатов в принципе произвольный и может содержать неоднозначные предикаты, например, для операции извлечения очередного случайного числа. Если программа содержит неоднозначные базисные предикаты, то предлагаемая ниже система правил доказательства оказывается неприменимой для программы.

Спецификация предиката A, представленная предусловием $P_A(x)$ и постусловием $Q_A(x, y)$, является *однозначной*, если постусловие является однозначным в области предусловия, т.е.:

$$\forall x.\ P_A(x) \Rightarrow (\forall y_1,y_2.\ Q_A(x, y_1)\ \&\ Q_A(x, y_2) \Rightarrow y_1 = y_2) \qquad (5.2)$$

Отметим, что однозначная спецификация не обязательно является реализуемой.

5.2. Теорема тождества спецификации и программы

Теорема тождества спецификации и программы определяет условия, при которых программа может быть выведена из спецификации. При доказательстве теоремы используется следующая лемма.

Лемма 5.0. Допустим, предикат $D(x: y)$ является однозначным в области X, а предикат $Z(x, y)$ реализуем в области X. Пусть истинна формула $Z(x, y) \Rightarrow D(x: y)$. Тогда истинна следующая формула $\forall x \in X.\ D(x: y) \Rightarrow Z(x, y)$.

Доказательство. Пусть истинно $D(x: y)$. для некоторого $x \in X$. Докажем истинность $Z(x, y)$. Поскольку предикат Z реализуем, то существует некоторый y', для которого истинно $Z(x, y')$. Из $Z(x, y) \Rightarrow D(x: y)$ получаем истинность $D(x: y')$. Поскольку истинны $D(x: y)$ и $D(x: y')$, то из однозначности D следует $y = y'$. Тогда истинно $Z(x, y)$. □

Теорема 5.2 тождества спецификации и программы. Рассмотрим произвольное определение предиката:

$$A(x: y) \equiv P_A(x)\ \{K(x: y)\}\ Q_A(x, y) \qquad (5.3)^{12}$$

Допустим, оператор $K(x: y)$ является однозначным в области истинности предусловия $P_A(x)$. Спецификация предиката A является реализуемой. Правая часть определения предиката выводима из спецификации, т.е.:

$$P_A(x)\ \&\ Q_A(x, y) \Rightarrow K(x: y) \qquad (5.4)$$

Тогда определение предиката A является корректным.

Доказательство. Для доказательства корректности определения предиката A достаточно доказать истинность формулы (3.9), т.е.:

$$P_A(x) \Rightarrow (\exists y.\ K(x: y))\ \&\ (K(x: y) \Rightarrow Q_A(x, y))$$

Допустим, предусловие $P_A(x)$ истинно.

Докажем реализуемость правой части, т.е. истинность $\exists y.\ K(x: y)$. Поскольку спецификация предиката A реализуема, то истинна формула $\exists y.\ Q_A(x, y)$. Пусть эта формула истинна для некоторого y'. Тогда из (5.4) истинна $K(x: y')$ и, следовательно, $\exists y.\ Q_A(x, y)$.

Докажем выводимость постусловия из правой части, т.е. истинность формулы $K(x: y) \Rightarrow Q_A(x, y)$. Поскольку истинно $P_A(x)$ и формула (5.4), то истинна формула $Q_A(x, y) \Rightarrow K(x: y)$. В соответствии с леммой 5.0 истинна формула $K(x: y) \Rightarrow Q_A(x, y)$, поскольку для предиката $Q_A(x, y)$ и оператора $K(x: y)$ выполняются условия этой леммы. □

Лемма 5.1. В условиях теоремы 5.2 истинна следующая формула:

$$P_A(x) \Rightarrow (K(x: y) \equiv Q_A(x, y)) \qquad (5.5)$$

Доказательство. Допустим, предусловие $P_A(x)$ истинно. Истинность формулы $K(x: y) \Rightarrow Q_A(x, y)$ установлена при доказательстве теоремы 5.2. Обратная импликация следует из (5.4). □

Как следствие леммы 5.1 спецификация предиката A оказывается однозначной.

[12] Эта формула совпадает с (3.4).

Система правил доказательства корректности программ, приведенная в разд. 4 базировалась на формуле (3.9). Предлагаемая ниже система правил базируется на теореме 5.2, в соответствии с которой для реализуемой спецификации и при условии однозначности используемых базисных предикатов требуется доказать истинность формулы (5.4). Отметим, что однозначность спецификации не требуется. Однако для неоднозначной спецификации не удастся доказать истинность формулы (5.4).

Лемма 5.2. Допустим, определение предиката (5.3) является корректным, а спецификации предиката A — однозначной. Тогда истинна формула (5.4), т.е. правая часть определения выводима из спецификации.

Доказательство. Допустим, истинны $P_A(x)$ и $Q_A(x, y)$. Докажем истинность $K(x: y)$. Из корректности определения A по формуле (3.9) следует истинность формулы $K(x: y) \Rightarrow Q_A(x, y)$, а также реализуемость $K(x: y)$. Пусть истинно $K(x: y')$ для некоторого y'. Тогда истинна $Q_A(x, y')$, а из однозначности Q_A следует $y = y'$. Следовательно, истинно $K(x: y)$. □

5.3. Правила для параллельного оператора

Рассмотрим определение предиката, правой частью которого является параллельный оператор:

$$A(x: y, z) \equiv P_A(x)\ \{B(x: y) \parallel C(x: z)\}\ Q_A(x, y, z) \qquad (5.6)^{13}$$

Предполагая корректность определений предикатов B и C, однозначность самих предикатов и их спецификаций определим правила, гарантирующие корректность определения (5.6).

Правило LP1. $P_A(x) \vdash P_B(x)\ \&\ P_C(x)$.
Правило LP2. $P_A(x)\ \&\ Q_A(x, y, z) \vdash Q_B(x, y)$.
Правило LP3. $P_A(x)\ \&\ Q_A(x, y, z) \vdash Q_C(x, z)$.

Лемма 5.3. Допустим, спецификация предиката A реализуема, определения предикатов B и C корректны, а их спецификации и правые части — однозначны. Если правила LP1, LP2 и LP3 истинны (т.е. правая часть доказуема из левой части для каждого правила), то определение предиката (5.6) корректно.

Доказательство. Поскольку правые части предикатов B и C — однозначны, то и сами предикаты — однозначны. Оператор $B(x: y) \parallel C(x: z)$ также оп-

[13] Формула совпадает с (4.5).

ределяет однозначный предикат. Поэтому выполняются условия теоремы 5.2 для предиката A, и для доказательства леммы достаточно доказать истинность формулы:

$$P_A(x) \;\&\; Q_A(x, y, z) \Rightarrow B(x{:}\ y) \;\&\; C(x{:}\ z) \qquad (5.7)$$

Подформула $B(x{:}\ y) \;\&\; C(x{:}\ z)$ определена в разд. 2.7 как эквивалент $B(x{:}\ y) \;||\; C(x{:}\ z)$.

Пусть истинны $P_A(x)$ и $Q_A(x, y, z)$. Докажем истинность $B(x{:}\ y)$ и $C(x{:}\ z)$. Из истинности предусловия $P_A(x)$ по правилу LP1 следует истинность $P_B(x)$ и $P_C(x)$. По правилам LP2 и LP3 становятся истинными $Q_B(x, y)$ и $Q_C(x, z)$. Для предикатов B и C выполняются условия леммы 5.2. Поэтому истинны формулы:

$$P_B(x) \;\&\; Q_B(x, y) \Rightarrow B(x{:}\ y)$$
$$P_C(x) \;\&\; Q_C(x, z) \Rightarrow C(x{:}\ z)$$

Поскольку посылка этих формул истинны, то истинны $B(x{:}\ y)$ и $C(x{:}\ z)$. □

5.4. Правила для оператора суперпозиции

Рассмотрим определение предиката, правой частью которого является оператор суперпозиции:

$$A(x{:}\ y) \equiv P_A(x)\ \{B(x{:}\ z);\ C(z{:}\ y)\}\ Q_A(x, y) \qquad (5.8)^{14}$$

Предположим, что определения предикатов B и C корректны, предикаты и их спецификации однозначны. Определим правила, гарантирующие корректность определения (5.8).

Правило LS1. $P_A(x) \vdash P_B(x)$.
Правило LS2. $P_A(x) \;\&\; Q_A(x, y) \;\&\; Q_B(x, z) \vdash P_C(z) \;\&\; Q_C(z, y)$.

Лемма 5.4. Допустим, спецификация предиката A реализуема, определения предикатов B и C корректны, а их спецификации и правые части — однозначны. Если правила LS1 и LS2 истинны, то определение предиката (5.8) корректно.

Доказательство. Поскольку правые части предикатов B и C — однозначны, то и сами предикаты — однозначны. Оператор $B(x{:}\ z);\ C(z{:}\ y)$ также опре-

[14] Формула совпадает с (4.6).

деляет однозначный предикат. Поэтому выполняются условия теоремы 5.2 для предиката A, и для доказательства леммы достаточно доказать истинность формулы:

$$P_A(x) \,\&\, Q_A(x, y) \Rightarrow \exists z.\ (B(x{:}\ z) \,\&\, C(z{:}\ y)) \qquad (5.9)$$

Подформула $\exists z.\ (B(x{:}\ z) \,\&\, C(z{:}\ y))$ определена в разд. 2.6 как эквивалент $B(x{:}\ z);\ C(z{:}\ y)$.

Пусть истинны $P_A(x)$ и $Q_A(x, y)$. Докажем истинность $\exists z.\ (B(x{:}\ z) \,\&\, C(z{:}\ y))$. Из истинности предусловия $P_A(x)$ по правилу LS1 следует истинность $P_B(x)$. Из корректности B по правилу RC2 следует истинность формулы $\exists z.\ B(x{:}\ z)$. Допустим для некоторого z_0 истинно $B(x{:}\ z_0)$. Для предиката B истинны условия леммы 5.1. Поэтому истинно $Q_B(x, z_0)$. В соответствии с правилом LS2 истинна формула $P_C(z_0) \,\&\, Q_C(z_0, y)$. В соответствии с леммой 5.2 истинна формула

$$P_C(z) \,\&\, Q_C(z, y) \Rightarrow C(z{:}\ y)$$

Тогда истинна $C(z_0{:}\ y)$. В итоге, будет истинна формула $\exists z.\ (B(x{:}\ z) \,\&\, C(z{:}\ y))$. □

5.5. Правила для условного оператора

Рассмотрим определение предиката, правой частью которого является условный оператор. Дадим повторно определения (4.7) и (4.8):

$$A(b, x{:}\ y) \equiv P_A(b, x)\ \{\ \textbf{if}\ b\ \textbf{then}\ B(x{:}\ y)\ \textbf{else}\ C(x{:}\ y)\ \textbf{end}\ \}\ Q_A(x, y) \qquad (5.10)$$

$$P_A(b, x) \equiv (b \equiv \varphi(x)) \,\&\, \psi(x) \qquad (5.11)$$

Предположим, что определения предикатов B и C корректны, предикаты и их спецификации являются однозначными. Определим правила, гарантирующие корректность определения (5.10).

Правило LA1. $\psi(x) \,\&\, Q_A(x, y) \,\&\, \varphi(x) \vdash P_B(x) \,\&\, Q_B(x, y).$

Правило LA2. $\psi(x) \,\&\, Q_A(x, y) \,\&\, \neg\varphi(x) \vdash P_C(x) \,\&\, Q_C(x, y).$

Лемма 5.5. Допустим, спецификация предиката A реализуема, определения предикатов B и C корректны, а их спецификации и правые части — однозначны. Если правила LA1 и LA2 истинны, то определение предиката (5.10) корректно.

Доказательство. Поскольку правые части предикатов B и C — однозначны, то и сами предикаты — однозначны. Условный оператор в правой части (5.10) также определяет однозначный предикат. Поэтому выполняются условия теоремы 5.2 для предиката A, и для доказательства леммы достаточно доказать истинность формулы:

$$(b \equiv \varphi(x)) \,\&\, \psi(x) \,\&\, Q_A(x, y) \Rightarrow (b \Rightarrow B(x{:}\ y)) \,\&\, (\neg b \Rightarrow C(x{:}\ y))$$

Подформула $(b \Rightarrow B(x{:}\ y)) \,\&\, (\neg b \Rightarrow C(x{:}\ y))$ определена в разд. 2.8 как эквивалент **if** b **then** B(x: y) **else** C(x: y) **end**.

Пусть истинны $P_A(x)$ и $Q_A(x, y)$. Докажем истинность формулы $b \Rightarrow B(x{:}\ y)$. Пусть истинно b. Докажем истинность $B(x{:}\ y)$. Можно применить правило LA1, поскольку истинны $\psi(x)$, $Q_A(x, y)$ и $\varphi(x)$. Получим истинность формулы $P_B(x) \,\&\, Q_B(x, y)$. В соответствии с леммой 5.2 истинна формула

$$P_B(x) \,\&\, Q_B(x, y) \Rightarrow B(x{:}\ y)$$

Поскольку истинна посылка, то истинно $B(x{:}\ y)$. Следовательно, доказана истинность формулы $b \Rightarrow B(x{:}\ y)$. Истинность формулы $\neg b \Rightarrow C(x{:}\ y)$ доказывается аналогично. □

5.6. Правила для рекурсивного кольца предикатов

Пусть имеется система определений для рекурсивного кольца предикатов $A_1, A_2, \ldots, A_n$[15]:

$$A_j(t, x_j{:}\ y_j) \equiv P_j(t, x_j)\{K_i(t, x_j{:}\ y_j)\}\, Q_j(t, x_j, y_j);\ j = 1\ldots n;\ n > 0 \qquad (5.12)$$

Здесь P_j и Q_j — предусловие и постусловие для предиката A_j, t — набор индукционных переменных. На значениях набора t определен частичный порядок R, удовлетворяющий свойству обрыва бесконечных убывающих цепей. Пусть имеется предикат $\beta(t)$, определяющий базу индукции, причем всякий минимальный элемент удовлетворяет предикату β.

Выводимость программы рекурсивного кольца (5.12) из спецификаций представим формулой V(t):

[15] Формула совпадает с (4.10).

$$V(t) \equiv \mathop{\&}_{j=1}^{n} \{ (P_j(t, x_j) \,\&\, Q_j(t, x_j, y_j) \Rightarrow K_i(t, x_j{:}\, y_j)) \}$$

Определим правила доказательства корректности определений кольца (5.12) по методу структурной индукции.

Правило LR1. $\beta(t) \vdash V(t)$.
Правило LR2. $\forall u.\ R(u, t) \,\&\, u \neq t \,\&\, V(u) \vdash V(t)$.

Лемма 5.6. Допустим, для каждого предиката A_j рекурсивного кольца его спецификация (в виде предусловия P_j и постусловия Q_j) является реализуемой, а правая часть K_i является однозначной. Если правила LR1 и LR2 истинны, то определения кольца (5.12) корректны.

Доказательство. Конъюнкция правил LR1 и LR2 дает формулу (2.35) для доказательства истинности $V(t)$ методом структурной индукции. Истинность $V(t)$ есть истинность формулы (5.4) для всех предикатов рекурсивного кольца. Таким образом, все условия теоремы 5.2 удовлетворены, и поэтому все определения предикатов рекурсивного кольца являются корректными. □

В качестве примера дадим доказательство корректности программы вычисления факториала (4.13). Докажем корректность определений рекурсивного кольца, состоящего из предикатов A0, A2, A3, A4 и A5. Переменная n является индуктивной. В качестве отношения порядка R используется отношение "≤" на натуральных числах. Базой индукции является $\beta(n) \equiv n = 0$.

Доказательство. Поскольку факториал $F(n)$ определен для всех $n \geq 0$, то спецификация всех предикатов кольца является реализуемой. Правые части определений кольца являются однозначными. Это следует из теоремы 5.1 и однозначности используемых базисных предикатов. В соответствии с леммой 5.6 необходимо доказать истинность правил LR1 и LR2. Пусть истинна посылка правила LR1, т.е. истинно $n = 0$. Требуется доказать $V(0)$, т.е. корректность определений для случая $n = 0$. Докажем сначала истинность конъюнкта $V(0)$ для предиката A3. Пусть истинно предусловие и постусловие A3, т.е. истинна формула:

$$n \geq 0 \,\&\, c1 = 1 \,\&\, (b \equiv n = 0) \,\&\, f = F(n). \qquad (5.13)$$

Докажем истинность условного оператора в правой части определения A3. Поскольку n = 0, то b = **true**. В соответствии с (2.17) достаточно доказать истинность оператора =(c1: f), что следует из (5.13) и F(0) = 1. Далее, используя истинность A3(0, c1, **true:** 1), доказывается истинность V(0) для предиката A2, а затем и для A0.

Пусть истинна посылка правила LR2, т.е. истинно индукционное предположение: ∀u. u < n & V(u). Требуется доказать V(n). Докажем сначала истинность V(n) для определения A5. Пусть истинно предусловие и постусловие A5, т.е. истинна формула:

n > 0 & n1 = n - 1 & f = F(n). (5.14)

Требуется доказать истинность правой части определения A5, т.е. истинность формулы:

A0(n1: f1); ∗(n, f1: f) (5.15)

Поскольку n1 < n , то в соответствии с индукционным предположением истинна формула

n1 ≥ 0 & f1 = F(n1) ⇒ A0(n1: f1) (5.16)

Поскольку n > 0, то истинно n1 ≥ 0. Далее, в соответствие с леммой 5.1 из реализуемости F и однозначности A0 следует истинность тождества f1 = F(n1) ≡ A0(n1: f1). Тогда (5.15) эквивалентна формуле f1 = F(n1) & f = n ∗ f1 или f = n ∗ F(n - 1), истинность которой следует из f = F(n) и определения (4.12) для F. Для данного n в соответствие с леммой 5.1 истинна формула:

n > 0 & n1 = n - 1 ⇒ (A5(n, n1: f) ≡ f = F(n)). (5.17)

Формула (5.17) используется далее для доказательства выводимости правой части определения A4 из спецификации. Далее это используется для доказательства выводимости из спецификации правой части определения A3 и т.д. □

ЗАКЛЮЧЕНИЕ

Общее правило соответствия программы ее спецификации [1] конкретизируется для программы на языке ССР в виде следующей модели корректности программы. Спецификация каждого предиката подразделяется на

предусловие и постусловие, причем предусловие считается истинным для определения предиката. Понятия предусловия и постусловия соответствуют введенным Хоаром [12]. Постусловие должно выводится из правой части определения предиката (условие (3.5)). Кроме того, постусловие и правая часть определения предиката должны быть реализуемы в области истинности предусловия. Суммарно корректность определения предиката представлена формулой (3.9).

Модель корректности программы, сформулированная для программ на языке CPP, распространяется и на язык предикатного программирования [10] и может быть адаптирована для языков функционального программирования. Отметим, что данная модель верна только для класса программ с предикатной спецификацией и не годится для программ с процессной спецификацией [1], описывающей эффект исполнения программы в виде системы параллельных взаимодействующих процессов.

Существует много работ по доказательству свойств функциональных программ. Однако язык, на котором формулируются свойства, не является частью языка функционального программирования и никак с ним не интегрирован. Исключением является язык ML, содержащий в себе язык алгебраических спецификаций. Корректность функций ограничивается здесь заданием предусловия и постусловия функций. Практически все работы по доказательству корректности программ (например, [14 - 16]) ориентированы на доказательство свойств классов, отражающих разные аспекты объектной ориентированности. Модель корректности в виде предусловия и постусловия функций реализуется также в языках спецификаций VDM [11], Z [17], B [18] и др.

Модель корректности программы, представленная в разд. 3, использует основное свойство логической семантики: каждый оператор в языке CCP является формулой исчисления предикатов. Логическая семантика свойственна также языкам логического программирования. Основополагающей для логического программирования считается работа Р. Ковальски [13], в которой язык исчисления предикатов рассматривается как язык программирования. При этом вычисление реализуется там посредством логического вывода в комбинации с перебором. В нашей статье исчисление предикатов также рассматривается как язык программирования. Однако вычисление трактуется более узко — только для случаев явной вычислимости в соответствии с понятием автоматической вычислимости.

Логическая семантика используется также для языка ***предикативного*** программирования [4], разработанного Э. Хехнером. Программа получается как результат последовательного уточнения (refinement) спецификации

программы. Уточнения реализуются применением системы правил логического вывода. Таким образом, программа является корректной по построению. При этом корректность определяется традиционно как конъюнкция предусловия и постусловия.

Логическая семантика может быть определена для любого чистого языка функционального программирования. Обычно для функциональных языков используется денотационная семантика.

Язык ССР совершенно не годится для написания программ. Он используется для изучения математических свойств программ. Язык ССР может быть использован в качестве ядра при построении языков программирования. Язык предикатного программирования [10], а также любой чистый язык функционального программирования может быть построен из языка ССР в виде иерархической системы обозначений. Построение каждой новой конструкции языка из базисных конструкций ядра должно сопровождаться построением формальной семантики конструкции, т.е. построением логической и операционной семантики, а также системы правил доказательства корректности, используя семантику базисных конструкций.

СПИСОК ЛИТЕРАТУРЫ

1. Шелехов В.И. Анализ общего понятия программы // Методы предикатного программирования. — Новосибирск, 2006. — Вып.2. — С. 7–16.
2. Шелехов В.И. Исчисление вычислимых предикатов. — Новосибирск, 2007. — 24с. — (Препр. / ИСИ СО РАН; № 143).
3. Дейкстра Э. Дисциплина программирования. Пер. с англ. М.: Мир. — 1978. — 272 с.
4. Hehner E.C.R. A Practical Theory of Programming, second edition. — 2004. — http://www.cs.toronto.edu/~hehner/aPToP/
5. Hopcroft J.E., Motwani R., Ullman J.D. Introduction to Automata Theory, Languages, and Computation (2nd Edition). — Addison Wesley, 2001. — 521p.
6. Кнут Д. Искусство программирования для ЭВМ. Т. 1. — М.: Мир, 1976. — С. 38–50.
7. Mathematical induction. — http://en.wikipedia.org/wiki/Mathematical_induction.
8. Хоар К. О структурной организации данных // Структурное программирование. — М.: Мир, 1975. — С.98–197.
9. Backus J. Can programming be liberated from the von Neumann style? A. Functional Style and Its Algebra of Programs // Communs ACM. — 1978. — Vol. 21, N 8. — P. 613–641.

10. Шелехов В.И. Введение в предикатное программирование. — Новосибирск, 2002. — 82с. — (Препр. / ИСИ СО РАН; № 100).
11. Jones C. B. Systematic Software Development using VDM. — Prentice Hall, 1990. — 361 p.
12. Hoare C. A. R. An axiomatic basis for computer programming // Communs ACM. — 1969. — Vol. 12, N 10. — P.576–585.
13. Kowalski R. Predicate Logic as Programming Language // IFIP Congress. — Stockholm, North Holland Publishing Co,1974. — P. 569–574.
14. Sannella D., Tarlecki A. Towards Formal Development of Programs from Algebraic Specifications: Model-Theoretic Foundations. — ICALP, 1992. — P. 656-671.
15. Sannella D., Tarlecki A. Toward Formal Development of ML Programs: Foundations and Methodology (Extended Abstract) // LNCS. — 1989. — Vol. 352. — P. 375 – 389 (LNCS).
16. Sannella D. Formal program development in Extended ML for the working programmer. Proc. 3rd BCS/FACS Workshop on Refinement, Hursley Park, 1990 // Springer Workshops in Computing. — 1991. — P. 99-130.
17. Abrial J.-R., Schuman S.A., Meyer B. Specification Language // On the Construction of Programs. — 1980. — P. 343-410.
18. Abrial J.-R The B-Book: Assigning Programs to Meanings. — Cambridge University Press, 1996. — 779 p.

В.И. Шелехов

МОДЕЛЬ КОРРЕКТНОСТИ ПРОГРАММ НА ЯЗЫКЕ ИСЧИСЛЕНИЯ ВЫЧИСЛИМЫХ ПРЕДИКАТОВ

Препринт
145

Рукопись поступила в редакцию 22.10.07
Рецензент И. С. Ануреев
Редактор Т. М. Бульонкова

Подписано в печать 28.12.07
Формат бумаги 60 × 84 1/16 Объем 2.9 уч.-изд.л., 3.1 п.л.
Тираж 60 экз.

Центр оперативной печати «Оригинал 2»
г.Бердск, ул. Островского, 55, оф. 02, тел. (383) 214-45-35