ТЕОРИЯ ВЫЧИСЛИМОСТИ И ЯЗЫКИ СПЕЦИФИКАЦИЙ
(Вычислительные системы)
1991 год Выпуск 139

УДК 681.3.06

СЕМАНТИКА ЯЗЫКА ПРОГРАММИРОВАНИЯ ФОЛИ

В.В.Зубенко, А.В.Протасов

Язык Фоли является языком функционального программирования, предназначенным для обработки символьной информации [1]. Он наследует черты таких функциональных языков, как Лисп и Форт, откуда и происходит название Фоли (ФОртЛИсп). От языка Лисп он унаследовал типы данных, от языка Форт - постфиксный синтаксис функций и использование стека при вычислении выражений.

В данной работе приводится формальное описание синтаксиса и семантики языка Фоли. С семантической точки зрения, Фоли трактуется как специальная прикладная алгебра [2]. Описание следует принципу отделения синтаксиса от семантики и является семантико-синтаксическим [3].

Введем обозначения: Z - множество целых чисел с обычными арифметическими, логическими функциями и функциями сравнения; "ложь", "истина" кодируются соответственно $\emptyset$, -1; P - множество указателей с операциями прибавления и вычитания целого числа, указатели изображаются как целые; O - множество объектов, на которых определены операции равенства и неравенства, объекты изображаются идентификаторами; ASCII - множество символов - есть операция CHAR, дающая символ по номеру: CHAR:Z → ASCII.

Определим множество атомов и данных: ATOM = Z ∪ O, DATA = = ATOM ∪ P.

Множество LIST - это множество конечных упорядоченных последовательностей, называемых списками, вида: $(X_1 \ldots X_n)$, $X_i \in$ $\in$ ATOM ∪ LIST.

Множество SEQ(X) - это множество конечных упорядоченных последовательностей вида: $(X_1,\ldots,X_n)$, $X_i \in X$.

В дальнейшем будут нужны некоторые вспомогательные функции на последовательностях.

Функция TOP дает первый элемент последовательности:

$$\text{TOP: } SEQ(X) \setminus \{\emptyset\} \to X,$$

$$\text{TOP}((X_1,X_2,\ldots,X_n)) = X_1.$$

Функция POP дает последовательность без первого элемента:

$$\text{POP: } SEQ(X) \setminus \{\emptyset\} \to SEQ(X),$$

$$\text{POP}((X_1,X_2,\ldots,X_n)) = (X_2,\ldots,X_n).$$

Функция PUSH добавляет элемент в начало последовательности:

$$\text{PUSH: } SEQ(X) \times X \to SEQ(X),$$

$$\text{PUSH}((X_1,\ldots,X_n),X) = (X,X_1,\ldots,X_n).$$

Функция APPEND добавляет элемент в конец последовательности:

$$\text{APPEND: } SEQ(X) \times X \to SEQ(X),$$

$$\text{APPEND}((X_1,\ldots,X_n)X) = (X_1,\ldots,X_n,X).$$

Определим операцию REP, которая изменяет график функции:

$$\text{REP: } (X \to Y) \times X \times Y \to (X \to Y),$$

$$\text{REP}(F,X,Y) = F \setminus \{(X,F(X))\} \cup \{(X,Y)\}.$$

Язык Фоли состоит из двух непересекающихся частей - базового Фоли и надстройки. Базовый Фоли определяется в терминах теории множеств, а надстройка полностью выражается через функции и функционалы базового Фоли, т.е. используется техника раскрутки. Рассмотрим функции и функционалы базового Фоли. Функции отображают некоторое состояние памяти в другое состояние

памяти. Состояние памяти S - это отображение с областью опре - деления:

$$DOM(S) = \{STACK, BACK, VARS, HEAP, FREE, INPUT, OUTPUT\} \subset O,$$

где элементы области определения соответствуют: STACK - стеку данных, BACK - стеку возвратов, VARS - множеству переменных, HEAP - куче, FREE - указателю на свободную часть кучи, INPUT - входной последовательности, OUTPUT - выходной последователь - ности.

Область значения состояния определяется следующим обра - зом:

$S(STACK) \in SEQ(DATA)$, $S(BACK) \in SEQ(DATA)$, $S(VARS): O \to DATA$,

$S(HEAP): P \to DATA$, $S(FREE) \in P$, $S(INPUT) \in SEQ(ATOM \cup LIST)$,

$S(OUTPUT) \in SEQ(ATOM \cup ASCII)$.

Целое число рассматривается как функция, которая заносит соответствующее число на стек:

$$F(X) = REP(X, STACK, PUSH(X(STACK), F)),\ F \in Z.$$

Бинарные арифметические, логические функции и функции сравнения работают с двумя верхними элементами стека:

$$F(X) = REP(X, STACK, PUSH(POP(POP(X(STACK))),$$
$$F(TOP(POP(X(STACK))), TOP(X(STACK))))),$$

$$F \in \{+, -, *, /, \%, <, >, <=, >=, =, <>, \wedge, !\}.$$

Перечисленные функции имеют обычную математическую семан- тику (% - остаток от деления, ! - дизъюнкция).

Функция # - это логическое отрицание:

$$\#(X) = REP(X, STACK, PUSH(POP(X(STACK)), \#(TOP(X(STACK))))).$$

Функция ' является параметрической. Если параметр - атом, то она заносит его в стек. Если параметр - список, то функция оп-

ределяется через функцию >< надстройки, которая от функции '
не зависит:

$'_y(X) = REP(X,STACK,PUSH(X(STACK),Y))$, $Y \in ATOM$,

$'_{(X_1 \ldots X_N)}(X) = ><('_{X_1}(\ldots'_{X_N}(NIL(X))\ldots))$, $(X_1 \ldots X_N) \in LIST$.

Функция NIL заносит в стек объект NIL:

$NIL(X) = REP(X,STACK,PUSH(X(STACK),NIL))$, $NIL \in O$.

Функция CAR дает содержимое ячейки кучи:

$CAR(X) = REP(X,STACK,PUSH(POP(X(STACK)),$

$X(HEAP)(TOP(X(STACK)))))$.

Функция] вычитает из указателя некоторое число:

$](X) = REP(X,STACK,PUSH(POP(POP(X(STACK))),$

$-(TOP(POP(X(STACK))),\ TOP(X(STACK)))))$.

Функция <+ меняет содержимое ячейки кучи:

$<+(X) = REP(REP(X,HEAP,REP(X(HEAP),\ TOP(X(STACK)),$

$TOP(POP(X(STACK)))),STACK,POP(POP(X(STACK))))$.

Функция NUMBER проверяет, является ли вершина стека чис -
лом:

$NUMBER(X) = REP(X,STACK,PUSH(POP(X(STACK)),$

$TOP(X(STACK)) \in Z))$.

Функция ATOM проверяет, является ли вершина стека атомом:

$ATOM(X) = REP(X,STACK,PUSH(POP(X(STACK)),$

$TOP(X(STACK)) \in ATOM))$.

Объект, применяемый к состоянию, рассматривается как пе-
ременная. При этом значение переменной заносится в стек:

$F(X) = REP(X,STACK,PUSH(X(STACK),\ X(VARS)(F)))$, $F \in O$.

Параметрическая функция -> присваивает параметру значение:

$$->_F(X) = REP(REP(X,VARS,REP(X(VARS),F,TOP(X(STACK)))),$$
$$STACK,POP(X(STACK))),\ F \in O.$$

Функция >- дает значение переменной:

$$>-(X) = REP(X,STACK,PUSH(POP(X(STACK)),$$
$$X(VARS)(TOP(X(STACK))))).$$

Функция <- присваивает переменной новое значение:

$$<-(X) = REP(REP(X,VARS,REP(X(VARS),\ TOP(X(STACK)),$$
$$TOP(POP(X(STACK))))),\ STACK,POP(POP(X(STACK)))),$$

Функция NEW сохраняет значения переменных из списка и присваивает им новые значения. Значения сохраняются во вспомогательном стеке:

$$NEW_{(X_1 \ldots X_N)}(X) = SAVE_{X_N}(\ \ldots\ SAVE_{X_1}(X)\ \ldots\ ),$$

$$(X_1 \ldots X_N) \in LIST,\ X_I \in O.$$

Вспомогательная функция SAVE сохраняет значение переменной и присваивает ей новое значение:

$$SAVE_F(X) = REP(REP(REP(X,BACK,PUSH(X(BACK),X(VARS)(F))),$$
$$VARS,REP(X(VARS),F,TOP(X(STACK)))),STACK,$$
$$POP(X(STACK))),\ F \in O.$$

Функция OLD восстанавливает старые значения переменных:

$$OLD_{(X_1 \ldots X_N)}(X) = REST_{X_N}(\ \ldots\ REST_{X_1}(X)\ \ldots\ ),$$

$$(X_1 \ldots X_N) \in LIST,\ \ X_I \in O.$$

Вспомогательная функция REST восстанавливает старое значение переменной:

$REST_F(X) = REP(REP(X,VARS,REP(X(VARS),F,TOP(X(BACK)))),$

$BACK,POP(X(BACK))),\ F \in 0.$

Функция & дает указатель на первую свободную ячейку кучи:

$\&(X) = REP(X,STACK,PUSH(X(STACK),X(FREE))).$

Функция ->& изменяет адрес первой свободной ячейки кучи:

$->\&(X) = REP(REP(X,FREE,TOP(X(STACK))),STACK,POP(X(STACK))).$

Функция SPACE заносит в стек объект SP, который изображается пробелом:

$SPACE(X) = REP(X,STACK,PUSH(X(STACK),SP)),\ SP \in 0.$

Функция <<A выводит атом:

$<<A(X) = REP(REP(X,OUTPUT,APPEND(X(OUTPUT,TOP(X(STACK)))),$

$STACK,POP(X(STACK))).$

Функция <<C выводит символ:

$<<C(X) = REP(REP(X,OUTPUT,APPEND(X(OUTPUT),$

$CHAR(TOP(X(STACK))))),\ STACK,POP(X(STACK))).$

Функция ; осуществляет перевод строки:

$$;(X) = <<C(13(X)).$$

Функция >> вводит данное:

$>>(X) = '_{TOP(X(INPUT))}(REP(X,INPUT,POP(X(INPUT)))).$

Тождественная функция I вспомогательная:

$$I(X) = NEW_{()}(X) = X.$$

Вспомогательная функция DROP снимает данное со стека:

$$DROP(X) = REP(X,STACK,POP(X(STACK))).$$

Кроме вышеописанных функций, базовый Фоли содержит некоторые функционалы. Функционал умножения · обеспечивает последовательное применение функций $F_1 \cdot F_2 (X) = F_2 (F_1 (X))$.

Функционал IF - это ветвление с несколькими альтернативами:

$$IF(X_1,Y_1, \ldots ,X_N,Y_N) = ?(X_1,Y_1,?(\ldots,?(X_N,Y_N,I) \ldots)).$$

Функционал ? не входит в базовый Фоли. Он обеспечивает условное применение функции:

$$?(F,F_1,F_2) = F_1(DROP(F(X))), \text{если } TOP(F(X)(STACK)) \neq 0,$$

$$?(F,F_1,F_2) = F_2(DROP(F(X))), \text{ если } TOP(F(X)(STACK)) = 0.$$

В Фоли-программах используются системы равенств вида: $F_1 = F_1, \ldots, F_N = F_N$, где $F_I \in O$, F_I - функции, полученные из Фоли-функций и объектов F_I с помощью функционалов; F_I выступают в роли функциональных параметров. Значение функции F_I определяется следующим образом: $F_I(X) = F_I(X)$.

Базовый Фоли содержит функционал LOOP, который обеспечивает циклирование с несколькими альтернативами. Функция EXIT производит выход из цикла:

$$LOOP(X_1,Y_2, \ldots ,X_I,Y_I \cdot EXIT, \ldots ,X_N,Y_N)(X) = F(X),$$

$$F = IF(X_1,Y_1 \cdot F, \ldots ,X_I,Y_I, \ldots ,X_N,Y_N \cdot F),$$

где $F \in O$, и больше нигде в программе не используется.

Функция STOP всюду не определена и останавливает программу $DOM(STOP) = \emptyset$.

Фоли-программа - это последовательность равенств $F_1 = F_1, \ldots ,F_N = F_N$ и функций $EXPR_1, \ldots ,EXPR_M$, где $EXPR_I$ построены из базовых функций, функций надстройки и параметров F_I с помощью функционалов. Аппликация последовательности функций эквивалентна аппликации функции $EXPR_1 \cdot << \cdot \ldots \cdot << \cdot EXPR_M \cdot <<$.

Значение программы - это значение этой функции в исходном состоянии. Исходное состояние X задается следующим образом:

$X(STACK) = \emptyset$, $X(BACK) = \emptyset$. $\forall F \in O\ X(VARS)(F) = NIL$,

$\forall F \in P\ X(HEAP)(F) = NIL$, $X(FREE) = 0$, $X(OUTPUT) = \emptyset$,

X(INPUT) - произвольное, задаваемое пользователем.

Покажем теперь, как по абстрактному синтаксису Фоли-программы построить ее конкретный синтаксис. Определим функцию SYN, которая отображает множество функций Фоли (FOLI) во множество строк в алфавите ASCII: $SYN: FOLL \to ASCII^*$,

$SYN('_X) = 'X$,

$SYN(->_X) = -> X$,

$SYN(NEW_{(X_1 \ldots X_N)}) = NEW(X_1 \ldots X_N)$,

$SYN(OLD_{(X_1 \ldots X_N)}) = OLD(X_1 \ldots X_N)$,

$SYN(F_1 \cdot F_2) = SYN(F_1)\ SYN(F_2)$,

$SYN(IF(X_1,Y_1, \ldots ,X_N,Y_N)) =$

$= IF(SYN(X_1)\ ?(SYN(Y_1)) \ldots SYN(X_N)\ ?(SYN(Y_N)))$,

$SYN(LOOP(X_1,Y_1, \ldots ,X_N,Y_N)) =$

$= IF(LOOP\ SYN(X_1)\ ?(SYN(Y_1)) \ldots SYN(X_N)\ ?(SYN(Y_N))END)$.

В остальных случаях SYN отображает функцию в ее абстрактное имя.

Распространим функцию SYN на множество Фоли-программ (PROG):

$SYN: PROG \to ASCII^*$,

$SYN(F_1 = F_1, \ldots ,F_N = F_N, EXPR_1, \ldots ,EXPR_M) =$

= (DEF F_1 SYN(F_1)) ... (DEF F_1 SYN(F_N)) (SYN($EXPR_1$)) ...

... (SYN($EXPR_M$)).

Ниже приводятся функции Фоли-надстройки. Они определяются через функции и функционалы базового Фоли.

Функция CDR смещает указатель на единицу:

```
(DEF CDR 1 ]).
```

Функция NULL проверяет, содержит ли ячейка кучи NIL:

```
(DEF NULL CAR NIL = ).
```

Функция T - это тождественно истинная функция:

```
(DEF T -1).
```

Функция $ разворачивает список на стеке:

```
(DEF $ NEW(X)
 IF(X NULL ?(NIL)
    T ?(X CDR  $ X CAR))
 OLD(X)).
```

Функция $- разворачивает список на стеке без NIL:

```
(DEF $-  NEW(X)
 IF(X NULL ?( )
    T ?(X CDR  $- X CAR))
 OLD(X)).
```

Функция $+ соединяет два списка

```
(DEF $+ NEW(Y X) Y $ X $- >< OLD(X Y)).
```

Функция > < формирует список из отдельных элементов:

```
(DEF >< 1 NEW(X)
 IF(X NIL <> ?(>< 1)) X &  <+ & -1  ] -> &
```

```
OLD(X))

(DEF >< >< 1 & CDR).
```

Функция << выводит атом или список с переводом строки:

```
(DEF  <<1 NEW (X)

 IF(X ATOM ?(X  <<A)

    T ?(40  <<C

        IF(LOOP X NULL # ?(X CAR  <<1 SPACE  <<A CDR -> X)
           END)

        41  <<C))

 OLD(X))

(DEF << <<<1 ; ).
```

Таким образом, язык Фоли описан почти полностью, за исключением некоторых функций ввода-вывода и функции вычисления списка, которая требует синтаксисо-семантического описания языка.

Литература

1. ЗУБЕНКО В.В., ПРОТАСОВ А.В. Фоли - язык функционального программирования //Методы представления знаний и доказательное программирование: Сб.науч .тр. ИК АН УССР. -Киев. - 1990.-С.46-54.

2. ZUBENKO V. Applied Program Algebras. - Bericht Nr.8504, Okt. 1985, CAU Kiel.-S.52.

3. РЕДЬКО В.Н. Основания композиционного программирования //Программирование. - 1979. -№3. -С. 3-12.

Поступила в ред.-изд.отд.

15 апреля 1990 года