УДК 681.142.001.2

ЯЗЫК ДЛЯ ОПИСАНИЯ АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ ЦВМ

С.В. Пискунов, С.Н. Сергеев, Б.А. Сидристый

Язык для описания алгоритмов функционирования ЦВМ должен служить одновременно языком как для моделирования, так и для автоматического проектирования. В этом случае в процессе проектирования не нужно описывать функционирование устройства на разных языках для проведения разных работ. Единое описание будет пригодно для всех целей, что в значительной мере уменьшит число ошибок, сократит сроки разработки и облегчит обмен информацией между проектировщиками.

Задачи, которые возникают в процессе проектирования, требуют от языка разных средств. Эти средства можно разделить на три группы:

1. Средства для описания алгоритмов функционирования отдельных устройств ЦВМ с целью их автоматического проектирования.

2. Совокупность средств для детального описания функций, имена которых используются при описании алгоритмов.

3. Совокупность средств алгоритмического описания работы внешних устройств ЦВМ для моделирования проектируемых ЦВМ.

Полное описание проектируемого устройства использует все указанные средства, и оно может служить исходной информацией для моделирования. В данной работе рассматривается та часть полного языка описания ЦВМ (называемая в дальнейшем Ф-языком), которая включает в себя средства описания алгоритмов функционирования устройств ЦВМ с целью их проектирования. С развитием системы автоматического проектирования ЦВМ представленный здесь Ф-язык будет расширяться, пока не перерастет в средство, отвечающее всем необходимым требованиям. Основные конструкции Ф-языка были предложены в докладе [1], но изложенное там представ-

ление Ф-языка было недостаточно наглядным для человека. В настоящей работе сделана попытка устранить этот недостаток. При описании языка используется система обозначений Бэкуса, дополненная двумя символами } и {. Пример использования этих символов: вместо записи вида $A ::= ab|ac$ применяется $A ::= a\{b|c\}$.

I. Элементарные конструкции

I.I. Основные символы.

<основной символ> ::= <буква> | <цифра> | <разделитель> | <скобка> | <признак оператора> | <признак массива> | <пробел> .

В качестве букв используются все большие буквы русского и латинского алфавитов, в качестве цифр - арабские цифры от 0 до 7.

<разделитель> ::= ⊗ | НЕ | ⊕ | ТО | ИЛИ | , | ; | + | ⊽ | - | : | → | КОНЕЦ МАССИВА.

<скобка> ::= [|] | (|).

<признак оператора> ::= ЕСЛИ | НЕПРЕРЫВНАЯ ПРОВЕРКА | НА .

<признак массива> ::= РЕГИСТРЫ | ВХОДНЫЕ ПЕРЕМЕННЫЕ | ВЫХОДНЫЕ ПЕРЕМЕННЫЕ | ПЕРЕМЕННЫЕ | АЛГОРИТМ | МЕТКИ ВЫХОДНЫХ ОПЕРАТОРОВ | СПИСОК ПЕРЕИМЕНОВАНИЙ ФУНКЦИЙ | СПИСОК ВНЕШНИХ МЕТОК.

пробел ::= ␣ .

I.2. Элементы имен переменных и функций, имена переменных и функций.

<имя переменной> ::= <последовательность слов> <указатель разрядности переменной> .

<имя функции> ::= <последовательность слов> <номер функции> .

<имя стандартной функции> ::= ⊽ | ⊗ | ⊕ | НЕ | = .

<последовательность слов> ::= <слово> | <последовательность слов> <последовательность пробелов> <слово> .

<указатель разрядности переменной> ::= <пусто> | (<натуральное число> + <натуральное число>) | (<натуральное число>).

<номер функции> ::= <пусто> | - <натуральное число> .

<последовательность пробелов> ::= <пусто> | <последовательность пробелов> <пробел> .

<натуральное число> ::= <цифра> | <натуральное число> <цифра> .

<слово> ::= <буква> | <цифра> | <слово> {<буква> | <цифра>} .

Примеры и семантика. Значениями переменных в Ф-языке являются n -разрядные двоичные коды (n =I,2,...). Указатель разрядности переменной определяет разряды переменной, которые используются в данный момент. Нумерация разрядов переменной начинается с 0. Примеры имен переменных: P1(0+45), ОМЕГА, RZG (7+I0), 56, P(5). Запись P(5) означает то же самое, что и P(5+5).

Функции в Ф-языке могут иметь n входных переменных (n =1, 2,...). Областью значений функции являются k -разрядные двоичные коды (k =I,2,...). В имени функции может присутствовать номер функции. Такая ситуация наблюдается в тех случаях, если необходимо указать, что две одинаковые функции в разных операторах при проектировании схемы устройства должны быть реализованы разными элементарными схемами. Символы ⊽ , ⊗ , ⊕ , НЕ обозначают соответственно поразрядную дизъюнкцию двух кодов, поразрядную конъюнкцию, поразрядное сложение по модулю 2 и поразрядное отрицание кода. Функция = предназначена для сравнения двух кодов. Эта функция может принимать два значения: I0 и 0I. Функция = принимает значение I0, если коды равны, и 0I - в противном случае.

Примеры имен функций: СЛОЖИТЬ, АНАЛИЗ I, ДЕШИФРАЦИЯ - 2.

2. Операторы

<оператор> ::= <непомеченный оператор> | <метка> : <непомеченный оператор> | <оператор условного перехода с непрерывной проверкой> .

<непомеченный оператор> ::= <оператор пересылки> | <оператор безусловного перехода> | <оператор условного перехода> | <оператор конца непрерывной проверки> | <оператор СТОП> .

<метка> ::= <последовательность слов> .

2.I. Элементы операторов.

<список входных переменных оператора> ::= [<последовательность переменных> <имя переменной>] .

<последовательность переменных> ::= <пусто> | <последовательность переменных> <имя переменной>,.

<список выходных переменных операторов> ::= <последовательность переменных> <имя переменной> .

<список меток> ::= <последовательность меток> <метка> .

<последовательность меток> ::= <последовательность меток> <метка> ИЛИ | <пусто> .

2.2. Оператор пересылки.

<оператор пересылки> ::= {<имя функции> <список входных пе

менных оператора>| НЕ <имя переменной> | <имя переменной> {⊙|⊗|⊕|=} <имя переменной> | <имя переменной>} → <список выходных переменных оператора>| 0 → <список выходных переменных оператора> .

Примеры и семантика. СЛОЖИТЬ [P1, P2]→P3 (0÷44); НЕ РИН (0÷5)→ P1 (0÷5), P4; РИН→ P4; 5→ ОМЕГА; I ⊕ A2→ A3; 0 → АЛЬФА.

Выполнение оператора пересылки представляет собой присвоение каждой выходной переменной значения функции оператора от входных переменных. Разрядность выхода функции и разрядности всех его выходных переменных должны совпадать. Особого упоминания заслуживает оператор 0→ <список выходных переменных оператора> . Совокупность символов 0 → всегда обозначает функцию очистки регистра.

2.3. Оператор безусловного перехода.

<оператор безусловного перехода> ::= НА <метка> .

2.4. Оператор условного перехода.

<оператор условного перехода> ::= ЕСЛИ {<имя функции> <список входных переменных оператора>| НЕ <имя переменной>| <имя переменной>{⊙|⊗|⊕|=}<имя переменной> | <имя переменной>} ТО список меток

Примеры и семантика. ЕСЛИ АНАЛИЗ НА O[P1] ТО СДВИГ ИЛИ A2; ЕСЛИ P2=5 ТО A6 ИЛИ A10; ЕСЛИ P6 (0÷3) ТО Б1 ИЛИ Б2 ИЛИ C1 ИЛИ C10; ЕСЛИ T(0) ТО A5 ИЛИ A6.

В списке меток оператора условного перехода столько меток, сколько разрядов имеет выход функции оператора. Если значение функции оператора от его входных переменных равно $(x_0, x_1, \ldots, x_{K-1})$, то всегда должно выполняться условие: $\sum_{j=0}^{K-1} x_j = 1$.Если после выполнения оператора условного перехода j -й разряд выхода функции равен I, то в алгоритме реализуется переход по метке, стоящей в списке оператора на j -м месте. Нумерация меток в списке меток проводится слева направо и начинается с 0. Например, после выполнения оператора ЕСЛИ P6 (0÷3) ТО Б1 ИЛИ Б2 ИЛИ C1 ИЛИ C10; происходит переход по j -й метке в списке меток оператора, если j -й разряд P6 равен I. В операторе ЕСЛИ T(0) ТО A5 ИЛИ A6; происходит переход по метке A5,если T(0)=I, и по A6, если T(0)=0.

2.5. Операторы условного перехода с непрерывной проверкой (УНП) и конца непрерывной проверки.

<оператор условного перехода с непрерывной проверкой> ::= <метка> : НЕПРЕРЫВНАЯ ПРОВЕРКА <оператор условного перехода>.

<оператор конца непрерывной проверки> ::= НЕПРЕРЫВНАЯ ПРОВЕРКА <метка> .

Семантика. На значение выхода функции в операторе УНП накладывается такое же ограничение, как и в операторе условного перехода. Правила выполнения операторов УНП и конца непрерывной проверки описываются в п. I4.9.

2.6. Оператор СТОП.

<оператор СТОП> ::= СТОП.

Семантика. Оператор СТОП предназначен для отметки конца выполнения алгоритма.

3. Массив переменных

<массив переменных> ::= <пусто> |ПЕРЕМЕННЫЕ <последовательность переменных> <имя переменной> КОНЕЦ МАССИВА.

Примеры и семантика. ПЕРЕМЕННЫЕ P1(0÷I0), A5 (0), АЛЬФА (0÷I5) КОНЕЦ МАССИВА.

В массиве переменных перечисляются все переменные алгоритма с указанием их разрядности. Если переменная в алгоритме употребляется с указанием полной разрядности, то в массиве переменных эту переменную можно не указывать.

4. Массив частей переменных

<массив частей переменных> ::= <пусто> | ЧАСТИ ПЕРЕМЕННЫХ <последовательность пар переменных> <пара переменных> КОНЕЦ МАССИВА.

<последовательность пар переменных> ::= <пусто> | <последовательность пар переменных> <пара переменных>,.

<пара переменных> ::= <имя переменной> = <имя переменной> .

Примеры и семантика. ЧАСТИ ПЕРЕМЕННЫХ РИН=P1(5÷I0), КОП=РК(0÷6) КОНЕЦ МАССИВА.

В тех случаях, когда часть разрядов какой-либо переменной несет самостоятельную функциональную нагрузку, человек может эту часть называть самостоятельным именем,инфо

мацию об этом он должен занести в массив частей переменных. Например, если в алгоритме, в котором фигурирует приведенный в примере массив частей переменных, встретится запись РИН (I÷2), то это означает, что речь идет о части разрядов переменной РI с 6 по 7 разряд.

5. Массив регистров

⟨массив регистров⟩ ::= ⟨пусто⟩| РЕГИСТРЫ ⟨последовательность имен переменных⟩ ⟨имя переменной⟩ КОНЕЦ МАССИВА.

Семантика. В массиве регистров перечисляются переменные, которым при проектировании схемы по данному алгоритму должны быть поставлены в соответствие регистры.

6. Массив переименований функций

⟨массив переименований функций⟩ ::= ⟨пусто⟩| СПИСОК ПЕРЕИМЕНОВАНИЙ ФУНКЦИЙ ⟨список переименований функций⟩ КОНЕЦ МАССИВА.
⟨список переименований функций⟩ ::= ⟨пара переименования⟩|
⟨список переименований функций⟩ ; ⟨пара переименования⟩ .
⟨пара переименования⟩ :: = ⟨имя функции⟩{[⟨последовательность переменных⟩ ⟨имя переменной⟩]| пусто} = ⟨имя функции⟩[⟨последовательность переменных⟩ ⟨имя переменной⟩]
{ ⟨пусто⟩ | ⟨номер части функции⟩} .
⟨номер части функции⟩ ::= ЧАСТЬ ⟨натуральное число⟩

Примеры и семантика. СПИСОК ПЕРЕИМЕНОВАНИЙ ФУНКЦИЙ Б=Ь [5,А,РI (0÷5)] ; Л [А,Д] = Г[6,А,Д] ЧАСТЬ 5 КОНЕЦ МАССИВА.

В тех случаях, когда какая-либо функция употребляется в алгоритме довольно часто так, что часть входных переменных $\{p_1, \ldots, p_k\}$ этой функции в разных операторах остается одинаковой, бывает удобно во всех таких вхождениях этой функции не писать переменные $p_1, \ldots, p_k$, а имя функции заменить другим именем, мнемонический смысл которого лучше отражает функциональное преобразование, чем имя функции с полным набором выходных переменных. Например, в алгоритме используется функция сложения двух 10-разрядных переменных. Причем эта функция имеет три входные переменные, две 10-разрядные и одну одноразрядную – F. Если $F = 0$, то функция реализует сложение с переносом, если $F = 1$, то поразрядное сложение по модулю 2. В массиве переименований

функций можно записать две такие пары переименования:
СЛ ПО МОД [А,В] = СЛОЖЕНИЕ [А,Б,I] ;
СЛОЖ П [А,В]= СЛОЖЕНИЕ [А,Б,О] ,
где переменные I и О имеют постоянные значения I и О соответственно.

Кроме причины, описанной выше, переименование функции вызывается еще и тем, что некоторые функции могут быть частями других. Понятие части функции появляется вследствие того, что в процессе выбора множества функций, которые используются при описании алгоритма функционирования проектируемого устройства, каждой функции ставится в соответствие схема функционального преобразователя, которая её реализует. При этом некоторым функциям может ставиться в соответствие не полная схема, а ее часть. Например, в пятиразрядной схеме логического сложения могут в некоторых случаях использоваться только первые три разряда для логического сложения двух трехразрядных кодов, в схеме дешифратора в некоторых случаях может использоваться только часть выходов и т.п. В таких ситуациях все варианты использования данного функционального преобразователя частями перенумеровываются (нумерация начинается с I), а в массиве переименований функций для имени функции, которая реализуется данной частью функционального преобразователя, указывается информация об имени и номере этой части полной функции, реализуемой полной схемой.

7. Массив входных переменных

⟨массив входных переменных⟩ ::= ВХОДНЫЕ ПЕРЕМЕННЫЕ СИГНАЛ ЗАПУСКА АЛГОРИТМА ⟨имя переменной⟩{ ⟨пусто⟩ | , ⟨последовательность имен переменных или констант⟩ { ⟨имя переменной⟩ | ⟨имя константы⟩}} КОНЕЦ МАССИВА.
⟨последовательность имен переменных или констант⟩ ::=⟨пусто⟩|
⟨последовательность имен переменных или констант⟩ { ⟨имя переменной⟩ | ⟨имя константы⟩ } , .
⟨имя константы⟩ ::= ⟨имя переменной⟩ = ⟨двоичное число⟩ .
⟨двоичное число⟩ ::= 0|I| ⟨двоичное число⟩ {0 | I} .

Примеры и семантика. ВХОДНЫЕ ПЕРЕМЕННЫЕ СИГНАЛ ЗАПУСКА АЛГОРИТМА Т1, Р4, К5÷76, 1÷I,56÷56 КОНЕЦ МАССИВА. В массиве входных переменных алгоритма перечисляются все входные переменные алгоритма и указываются значения констант, имена которых ис-

пользуются в алгоритме.

8. Массив операторов

⟨массив операторов⟩ ::= АЛГОРИТМ ⟨список операторов⟩ ⟨оператор СТОП⟩ КОНЕЦ МАССИВА.
⟨список операторов⟩ ::= ⟨оператор⟩ ; | ⟨список операторов⟩ ⟨оператор⟩ ; .

Семантика. Оператор СТОП встречается в массиве операторов только один раз и должен стоять в конце массива. Ни одна пара операторов не может быть помечена одинаковыми метками.

9. Массив выходных переменных

⟨массив выходных переменных⟩ ::= ⟨пусто⟩ | ВЫХОДНЫЕ ПЕРЕМЕННЫЕ ⟨последовательность переменных⟩ ⟨имя переменной⟩ КОНЕЦ МАССИВА.

Семантика. В массиве выходных переменных указываются все выходные переменные алгоритма, значения которых используются вне алгоритма.

IO. Массив временны́х ограничений

⟨массив временны́х ограничений⟩ ::=⟨пусто⟩ |ВРЕМЕННЫЕ ОГРАНИЧЕНИЯ ⟨последовательность временны́х ограничений⟩ ⟨временное ограничение⟩ КОНЕЦ МАССИВА.
⟨последовательность временны́х ограничений⟩ ::= ⟨пусто⟩| ⟨последовательность временны́х ограничений⟩ ⟨временное ограничение⟩ , .
⟨временно́е ограничение⟩ ::= ⟨метка⟩ - ⟨метка⟩ > ⟨натуральное число⟩ | ⟨метка⟩ = ⟨метка⟩ | ⟨метка⟩ - ⟨метка⟩ < ⟨натуральное число⟩ .

Примеры и семантика. ВРЕМЕННЫ́Е ОГРАНИЧЕНИЯ А55 - Б > IO , ЧТЕНИЕ - ЗАПИСЬ < I5,
А - Б > 5, С - Д < 6, АЛЬФА=К КОНЕЦ МАССИВА.
Временны́е ограничения $x-y>\ell$, $x-y<\ell$ и $x=y$ интерпретируются соответственно следующими выражениями: $\tau(x)-\tau(y)\geqslant\ell$, $\tau(x)-\tau(y)\leqslant\ell$, $\tau(x)=\tau(y)$, где ℓ - натуральное число, а $\tau(x)$ - момент времени, в который начинает выполняться оператор, помеченный меткой x . В массиве временны́х ограничений человек указывает информацию о временны́х соотношениях между операторами алгоритма, если таковые имеются.

ПРИМЕЧАНИЕ. Если операторы a и b находятся во временно́м ограничении типа = или одновременно в ограничениях типа > и < при ℓ = 0, то в записи алгоритма эти операторы должны следовать непосредственно один за другим.

II. Массив меток выходных операторов блоков

⟨массив меток выходных операторов блоков⟩ ::= ⟨пусто⟩| МЕТКИ ВЫХОДНЫХ ОПЕРАТОРОВ ⟨список меток⟩ КОНЕЦ МАССИВА.
⟨список меток⟩ ::= ⟨метка⟩ | ⟨список меток⟩ , ⟨метка⟩ .

Семантика. В массиве меток выходных операторов блоков указываются метки операторов, которые являются выходными операторами блоков алгоритма, если человек сам проводит разбиение алгоритма на блоки (при проектировании каждому блоку ставится в соответствие отдельная микропрограмма).

I2. Массив внешних меток

⟨массив внешних меток⟩ ::= ⟨пусто⟩ | ВНЕШНИЕ МЕТКИ ⟨список меток⟩ КОНЕЦ МАССИВА.

В массиве внешних меток указываются входные и выходные метки алгоритма. Входными метками алгоритма помечаются операторы алгоритма, выходные метки могут входить в операторы безусловного перехода.

I3. Алгоритм

⟨алгоритм⟩ ::= ⟨массив переменных⟩ ⟨массив частей переменных⟩ ⟨массив регистров⟩ ⟨массив переименований функций⟩ ⟨массив входных переменных⟩ ⟨массив операторов⟩ ⟨массив выходных переменных⟩ ⟨массив временны́х ограничений⟩ ⟨массив меток выходных операторов⟩ ⟨массив внешних меток⟩ .

I4. Правила выполнения операторов алгоритма, представленного на Ф-языке

Определим на множестве M всех операторов алгоритма бинарные отношения β^j $(j = 0,1,\ldots)$.

Для операторов $a, b \in M$ выполняется $a \beta^0 b$, если выполняется одно из следующих условий:

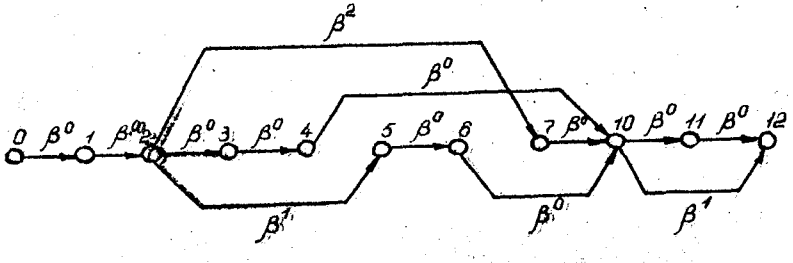
I4.I. a – оператор перехода (условного, безусловного или условного с непрерывной проверкой), и в списке меток оператора a на нулевом месте стоит метка оператора b (в операторе безусловного перехода будем считать, что метка оператора стоит на нулевом месте).

I4.2. a – оператор, не являющийся оператором перехода, и в записи алгоритма непосредственно следует за b.

Для $a, b \in M$ выполняется $a \beta^j b$ $(j \neq 0)$, если:

I4.3. a – оператор условного перехода или условного перехода с непрерывной проверкой, и в списке меток оператора a на j-м месте стоит метка оператора b.

Для иллюстрации определенных выше отношений на рисунке изображен граф, вершины которого соответствуют номерам операторов приведенной ниже последовательности операторов, а каждая дуга с надписью β^j указывает на то, что соединенные этой дугой операторы находятся в отношении β^j.




```
ОПЕР ЛОГИЧ: 0——ОМЕГА ; 0——РI ; ЕСЛИ ШИНЫ ДШО (I+3)
     ТО ОТРИЦАНИЕ ИЛИ ДИЗЪЮНКЦИЯ ИЛИ КОНЪЮНКЦИЯ;
ОТРИЦАНИЕ: НЕ Р2——РI; НА КОНЕЦ ОПЕР ЛОГИЧ;
ДИЗЪЮНКЦИЯ: Р2 Ⓥ Р3 ——РI; НА КОНЕЦ ОПЕР ЛОГИЧ;
КОНЪЮНКЦИЯ: Р2 Ⓧ Р3 ——РI;
КОНЕЦ ОПЕР
```



```
ЛОГИЧ  :  ЕСЛИ АНАЛИЗ НА 0 [РI] ТО Д1 ИЛИ Д2;
Д1     :  I —— ОМЕГА;
Д2     :  НА ЧТ КОМАНДЫ.
```

В дальнейшем нам понадобятся отношения β_1 и β, которые определяются следующим образом.

Для $a, b \in M$ выполняется $a \beta_1 b$, если:

I4.4 выполняется $a \beta^j b$ для некоторого j.

Для $a, b \in M$ выполняется $a \beta b$, если выполняется одно из условий I4.5 – I4.7.

I4.5 a есть b;

I4.6 выполняется: $a \beta_1 b$;

I4.7 выполняется: $\exists x_1, \ldots, x_k [(a \beta_1 x_1) \wedge (x_1 \beta_1 x_2) \wedge \ldots \wedge (x_k \beta_1 b)]$

Из приведенного выше примера в отношении β_1 находится та пара операторов, которой соответствует на графе пара вершин, соединенных дугой. Если на графе между двумя вершинами существует путь, то операторы, соответствующие этим вершинам, находятся в отношении β.

Перейдем к определению системы правил R_0 выполнения операторов алгоритма функционирования, предотавленного на Ф-языке. При этом в некоторых случаях для записи условий выполнения операторов будем использовать язык узкого исчисления предикатов. Предикаты будут изображаться или большой буквой латинского алфавита с перечислением переменных, от которых они зависят (например, $R(a, b, \tau)$), или выражениями вида (R), где R – некоторое условие. Если R выполняется, то (R) принимает значение I, если не выполняется, то (R) принимает значение 0.

I4.8. Выполнение операторов алгоритма функционирования происходит в дискретном времени. Текущее значение каждой переменной сохраняется во времени до тех пор, пока этой переменной не будет присвоено новое значение. При выполнении алгоритма значения его внешних – выходных и входных – переменных с течением времени могут меняться. Для обеспечения заданной реакции алгоритма на эти изменения на выполнение его операторов накладываются временные ограничения описанных выше трех типов. Выполнение любого оператора алгоритма занимает единицу времени.

Алгоритм функционирования во время выполнения переходит из одного внутреннего состояния в другое. Под внутренним состоянием алгоритма, в котором он находится в некоторый момент, здесь понимается совокупность операторов, выполняющихся в дан-

ный момент совместно с совокупностью значений всех его переменных, кроме входных. Если оператор a входит в совокупность операторов, определяющую состояние S, то будем говорить, что a принадлежит состоянию S. При выполнении алгоритма отсчитывается два времени: внешнее t и внутреннее τ.

I4.9. Операторы условного перехода с непрерывной проверкой и конца непрерывной проверки введены в Ф-язык для описания таких внешних воздействий на алгоритм, которые могут иметь место только при выполнении определенных условий. Пусть оператор b – оператор условного перехода с непрерывной проверкой, а d –оператор конца непрерывной проверки, в котором указана метка оператора b. При реализации перехода в операторе b по одной из меток начинают выполняться операторы, соответствующие этому переходу. Если же переход был по метке, стоящей в списке меток b на нулевом месте, то условие оператора b продолжает непрерывно проверяться, пока не встретится оператор d или пока в процессе этой проверки не изменится условие. В этом случае происходит переход в соответствии с изменившимся условием. Определим понятие области непрерывной проверки оператора b. Множество операторов $a_0 \cup \{a\}$, для которых выполняется $\exists d\,[(a_0 \beta a) \wedge (a \beta d)]$, где d – оператор конца непрерывной проверки, соответствующий оператору условного перехода с непрерывной проверкой b, а a_0 – оператор, для которого выполняется $b \beta^0 a_0$, называется областью непрерывной проверки оператора b. Если в операторе b реализуется переход при выполнении одного из операторов его области непрерывной проверки, то результат этого оператора считается неопределенным.

I4.I0. В множестве внешних переменных алгоритма функционирования выделяется одноразрядная переменная, называемая сигналом запуска алгоритма. Если в некоторый момент t внешнего времени сигнал запуска алгоритма принимает значение 0, то алгоритм в течение последующего времени остается в том состоянии, в котором он находился в момент t, а отсчет внутреннего времени прекращается. После того, как сигнал запуска алгоритма принимает единичное значение, начинается выполнение операторов алгоритма с того состояния, в котором он находился, при этом возобновляется отсчет внутреннего времени.

I4.II. После выполнения оператора СТОП сигнал запуска алгоритма принимает нулевое значение, и алгоритм переходит в состояние (a_0, Σ), где a_0 – первый оператор в записи алгоритма, а Σ – совокупность значений внутренних и выходных переменных алгоритма, которая имела место при выполнении оператора СТОП.

I4.I2. Как уже отмечалось в п. I4.8, при выполнении алгоритма функционирования он переходит из одного внутреннего состояния в другое. Состояние алгоритма может характеризоваться одним или несколькими операторами. Последний случай имеет место только тогда, когда операторы связаны временным ограничением типа $=$ или одновременно временными ограничениями типов $>$ и $<$ при $\ell = 0$.

Пусть алгоритм функционирования в момент $\tau - 1$ внутреннего времени находится в состоянии S_1. В момент времени τ алгоритм будет находиться в состоянии S_2, если выполняется условие I4.I2 а) или I4.I2д) или одновременно выполняются условия I4.I2 б) – I4.I2 г), в противном случае алгоритм остается в состоянии S_1.

I4.I2. а) $\exists a\,[(a \in S_2) \wedge (n_a \in z) \wedge B(n_a, \tau)]$,
здесь n_a есть метка оператора a, z – множество внешних меток алгоритма, $B(n_a, \tau) = 1$, если в момент времени τ по внешним для алгоритма причинам требуется переход по внешней метке n_a.

I4.I2. б) $\exists a, b\{(b \in S_1) \wedge (a \in S_2) \wedge [P_1(b, a) \vee P_2(b, a)]\}$,
здесь $P_1(b, a)$ есть $\exists j\,[(b \in M_1) \wedge (b \beta^j a) \wedge (b \to j)]$,
где M_1 – множество операторов переходов алгоритма, а $(b \to j) = 1$, если после выполнения b требуется переход к оператору, стоящему в списке меток оператора b на j-м месте. $P_2(b, a)$ есть $(b \in M \setminus M_1) \wedge (b \beta^0 a)$.

I4.I2. в) Любой оператор $a \in S_2$ можно выполнять согласно временным ограничениям, наложенным на выполнение операторов алгоритма.

I4.I2. г) $\neg\, \exists b\,[(b \notin S_2) \wedge (n_b \in z) \wedge B(n_b, \tau)]$.

I4.I2. д) $\exists a, b, c\,[(a \in S_2) \wedge (b \in S_1) \wedge (c \in M_2) \wedge (c \beta^j a) \wedge (c \to j) \wedge (j \neq 0)]$.

Здесь M_2 – множество всех операторов условного перехода с непрерывной проверкой, M_c – область непрерывной проверки оператора c.

Отметим, что в силу примечания к п. I0 для любого состояния S_1 алгоритма одновременно не может найтись более одного состояния, в которое алгоритм может перейти в соответствии с условиями I4.I3 а) – I4.I3 д).

При реализации перехода по внешней метке результат работы

операторов, во время выполнения которых произошел переход, считаются неопределенным.

14.13. После выполнения оператора безусловного перехода, в запись которого входит выходная метка алгоритма, выполняется оператор, следующий в записи алгоритма за данным оператором. В то же время выполнение этого оператора является причиной перехода по внешней метке для того алгоритма, для которого данная метка является входной.

Литература

1. Б.А. СИДРИСТЫЙ. Получение временной диаграммы работы узла вычислительного устройства с оптимизацией по времени и оборудованию. Семинар "Теория автоматов", Киев, 1969, вып. 5.

Поступила в редакцию
18. IX. 1968г.