6429

Экз. чит. зала

СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
Дубна

11 - 6429

С.Х.Бычваров

ПРАВИЛО УМОЛЧАНИЯ И ЕГО ИСПОЛЬЗОВАНИЕ В ЯЗЫКЕ ПРОГРАМИРОВАНИЯ PL/1

ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И АВТОМАТИЗАЦИИ

1972

11 - 6429

С.Х.Бычваров

ПРАВИЛО УМОЛЧАНИЯ И ЕГО ИСПОЛЬЗОВАНИЕ В ЯЗЫКЕ ПРОГРАМИРОВАНИЯ PL/1

I. Введение

Принцип умолчания определяется в /I/ как соглашение, при помощи которого "всюду, где язык предоставляет несколько возможностей, а программист не указал никакой, компилятор применяет интерпретацию умолчания, т.е. подразумевается некоторая из возможностей, предусмотренная в языке на этот случай. В качестве таких подразумеваемых для каждой конструкции возможностей в языке выбраны те, которые вероятнее всего потребуются программисту".

В данной работе будет рассмотрено применение принципа умолчания при получении полной совокупности описателей.

Процесс получения полной совокупности описателей при помощи принципа умолчания будем называть процессом умолчания.

Фактически процесс умолчания является процессом расширения.

В известных автору сообщениях об языке PL/I, в частности, в /I,2,3,4/, при описании процесса умолчания существует ряд недостатков:

I.I. Противоречивые утверждения. К примеру, приведем несколько пар противоречивых утверждений:

I.I.I. Утверждения I.I.I.I и I.I.I.2, вместе взятые, несовместимы с методом присваивания описателей в языке PL /I.

I.I.I.I. В параграфе " EXTERNAL and INTERNAL" (Assumptions:) на стр. 68 утверждается, что описатель INTERNAL (EXTERNAL) добавляется по умолчанию для имен входов внутренних (внешних) процедур (этот и следующие примеры взяты из /4/).

I.I.I.2. В параграфе " Label Prefixes " на стр. 43 утверждается, что идентификатор, который является именем входа внутренней (внешней) процедуры, считается явно объявленным с описателем INTERNAL (EXTERNAL).

I.I.2. Утверждения I.I.2.I и I.I.2.2., вместе взятые, несовместимы с альтернативностью описателей EXTERNAL и INTERNAL.

I.I.2.I. В параграфе " Standard Default Rules " (Condition:) на стр. 48 утверждается, что описатель INTERNAL добавляется по умолчанию для Condition .

I.I.2.2. В параграфе " EXTERNAL and INTERNAL " (Assumptions:) на стр. 68 утверждается, что описатель EXTERNAL добавляется по умолчанию для имен ситуации (conditions), определяемых программистом.

I.I.3. Утверждения I.I.3.I и I.I.3.2, вместе взятые, несовместимы с альтернативностью описателей FIXED и FLOAT .

I.I.3.I. Из параграфа " Standard Default Rules " (arithmetic data :) на стр. 48 вытекает, что к описателям BINARY , number-of-digits и scale-factor добавляется по умолчанию описатель FLOAT .

I.I.3.2. Из параграфа" FIXED and FLOAT " (Assumptions:) на стр. 69 вытекает, что к описателям BINARY , number-of-digits и scale-factor добавляется, по умолчанию, описатель FIXED.

I.2. Неполная информация. Приведем два примера:

I.2.I. Нигде не сказано, какие правила умолчания применяются при получении полной совокупности описателей для значения, возвращаемого функцией, если хотя бы один описатель был явно объявлен.

I.2.2. В параграфе " ENTRY " (IO.) на стр. 64 сообщается, что к описателю ENTRY добавляются по умолчанию VARIABLE, если он явно объявлен хотя бы с одним из описателей ALIGNED , AUTOMATIC, BASED, CONTROLLED, DEFINED, Dimension, INITIAL, Parameter, SECONDARY, STATIC, UNALIGNED.

Этот список неполный. К нему необходимо еще добавить описатель CONNECTED.

I.3. Дублирование. Приведем два примера:

I.3.I. К описателям ENTRY , Parameter и CONTROLLED добавляется описатель INTERNAL согласно двум правилам умолчания из параграфа "Standard Default Rules" (ENTRY:и parameter:) на стр. 48.

I.3.2. Идентификатор, являющийся приставкой перед оператором PROCEDURE или ENTRY , считается явно объявленным и получает описатель ENTRY . Этот идентификатор получает еще описатель Constant в результате того же самого явного объявления, а также согласно одному правилу умолчания из параграфа "Standard Default Rules " (ENTRY:) на стр. 48.

I.4. Правила описания процесса умолчания разбросаны по всему сообщению. Например, часть правил умолчания для получения описателей области действия и способа размещения переменных первого уровня находится в параграфе " Standard Default Rules " (All level-*one* variables :) на стр. 48, а часть - в параграфе " EXTERNAL and INTERNAL " (Assumptions:)) на стр. 68.

I.5. Нет четкого выделения всех правил, которые описывают процесс умолчания. Приведем два примера:

I.5.I. Исходя не только из правил умолчания, но также и из общих правил (General rules:) параграфа "COMPLEX and REAL " (I.), на стр. 60, мы получаем, что описатель REAL добавляется по умолчанию к описателю PICTURE-N (т.е. описатель PICTURE для числовых данных).

I.5.2. Только в общем правиле параграфа "ENTRY " (3.) на стр. 64 говорится, что описатель ENTRY добавляется по умолчанию, если уже присутствует хотя бы один из описателей REDUCIBLE , IRREDUCIBLE и RETURNS .

I.6. При описании процесса умолчания нет единого подхода. Он описан при помощи нескольких десятков правил умолчания.

Теоретико-множественное описание процесса умолчания, при котором описатели рассматриваются как элементы множеств, является одним из возможных путей устранения указанных недостатков. При таком подходе процесс умолчания можно описать при помощи одного простого правила, которое в дальнейшем, для краткости, будем называть: " Правило умолчания".

Предложенное правило умолчания является важным и полезным для стандартизации и реализации языка PL/1 . Оно предлагается для включения в стандартный язык PL/1 и рекомендуется для использования в качестве исходного пункта при реализации процесса умолчания в трансляторах.

2. Определение правила умолчания

Пусть заданы

2.I. Множество A конечное

2.2. Множества $A_1, A_2, \ldots, A_n$ ($n \geqslant 1$) такие, что

2.2.I. $\bigcup_{i=1}^{n} A_i = A$

2.2.2. $A_i \neq A_j$, если $i \neq j$ ($i, j = 1,2,\ldots, n$).

2.3. Множество B такое, что $B \subset A$.

Множества $A_1, A_2, \ldots, A_n$ ($n \geqslant 1$) назовем результирующими.

Предполагаем, что результирующие множества упорядочены между собой. Для определенности будем считать, что они упорядочены по возрастающим индексам, т.е.в том порядке, как они выписаны.

Правило умолчания. Если существует минимальное i, для которого $B \subset A_i$, то B расширяется до A_i.

3. Необходимое и достаточное условие применимости правила умолчания

Пусть заданы 2.I., 2.2. и

3.I. Множества $B_1, B_2, \ldots, B_m$ ($m \geq 1$) такие, что для любого i ($1 \leq i \leq m$) существует j ($1 \leq j \leq n$) (вообще не единственное), для которого выполнено включение $B_i \subset A_j$.

Рассмотрим функцию, которая любому Bi ($1 \leq i \leq m$) поставит в соответствие некое A_j ($1 \leq j \leq n$), для которого $B_i \subset A_j$. Такую функцию будем называть функцией расширения.

Так как процесс умолчания является процессом расширения, то любой процесс умолчания можно описать подходящей функцией расширения.

Функцию расширения можно задать совокупностью пар множеств. Левая часть любой пары есть аргумент, а правая – значение функции для этого аргумента.

Для удобства i-ую пару будем обозначать (α_i, β_i) $i = 1,2,\ldots, m$.

Очевидно, существует функция расширения, для которой не существует упорядоченных множеств A_i ($i = 1,2,\ldots, n$) таких, что ее можно задать при помощи правила умолчания.

Например, если $A_1 = (a_1, a_2, a_3, a_4)$, т.е. множество A_1 состоит из элементов a_1, a_2, a_3 и a_4, $A_2 = (a_1, a_2, a_3)$, $B_1 = (a_1, a_3)$, $B_2 = (a_1, a_2)$, $\alpha_1 = B_1$, $\alpha_2 = B_2$, $\beta_1 = A_1$ и $\beta_2 = A_2$, то для функции расширения (α_i, β_i) $i = 1,2$ не существует упорядоченных A_1 и A_2 таких, что ее можно задать при помощи правила умолчания.

Пара (α_i, β_i) предшествует паре (α_j, β_j) или, что то же самое, пара (α_j, β_j) следует за парой (α_i, β_i)

$((\alpha_i, \beta_i) \rightarrow (\alpha_j, \beta_j)$ или $(\alpha_j, \beta_j) \leftarrow (\alpha_i, \beta_i))$, если

3.2.I. $\beta_i \neq \beta_j$ и
3.2.2 $\alpha_i \subset \beta_j$.

Мы говорим, что пары $(\alpha_{i_1}, \beta_{i_1}), (\alpha_{i_2}, \beta_{i_2}), \ldots, (\alpha_{i_k}, \beta_{i_k})$ образуют цикл, если

3.3. $(\alpha_{i_1}, \beta_{i_1}) \to (\alpha_{i_2}, \beta_{i_2}) \to \dots \to (\alpha_{i_k}, \beta_{i_k}) \to (\alpha_{i_1}, \beta_{i_1})$,

где $2 \le k \le m$; $1 \le i_1, i_2, \dots, i_k \le m$; $i_p \neq i_q$ если $p \neq q$.

Теорема. Для того чтобы функцию расширения получить при помощи правила умолчания, необходимо и достаточно потребовать отсутствия циклов в совокупности пар, задающих эту функцию.

Необходимость

В самом деле, если имеет место 3.3., то при упорядочении множеств для получения функции расширения при помощи правила умолчания, необходимо β_{i_1} расположить выше β_{i_2} и так далее: $\beta_{i_{k-1}}$ расположить выше β_{i_k}, β_{i_k} – выше β_{i_1}. В частности, необходимо β_{i_1} расположить выше β_{i_k} и β_{i_k} – выше β_{i_1}. Получим противоречивые утверждения. Следовательно, если функция расширения получается при помощи правила умолчания, то не имеет места 3.3.

Достаточность

Рассмотрим совокупность пар, задающих функцию расширения. Исходя из этой совокупности и из всех реляций предшествования (следования), которые существуют между ее элементами, построим направленный граф Γ_1 такой, что:

3.4.1. Между вершинами графа Γ_1 и элементами совокупности пар существует взаимно-однозначное соответствие. Обозначаем $V(\alpha_i, \beta_i)$ вершину графа Γ_1, соответствующую элементу (α_i, β_i) из совокупности пар.

3.4.2. Две вершины графа Γ_1 связаны ориентированным ребром тогда и только тогда, когда между соответствующими элементами из совокупности пар существует реляция предшествования (следования). Точнее, если $(\alpha_i, \beta_i) \to (\alpha_j, \beta_j)$, то соответствующее ребро направлено от вершины $V(\alpha_i, \beta_i)$ к вершине $V(\alpha_j, \beta_j)$ $(V(\alpha_i, \beta_i) \to V(\alpha_j, \beta_j))$, и наоборот, если

$V(\alpha_i, \beta_i) \longrightarrow V(\alpha_j, \beta_j)$, то $(\alpha_i, \beta_i) \longrightarrow (\alpha_j, \beta_j)$.

Из 3.4.I и 3.4.2 следует, что совокупность пар и граф Γ_I эквивалентны по отношению циклов*, т.е. между циклами в совокупности пар и в графе Γ_I можно установить взаимно-однозначное соответствие.

Вершины $V(\alpha_i, \beta_i)$ и $V(\alpha_j, \beta_j)$ графа Γ_I называем подобными, если $\beta_i = \beta_j$ и $i \neq j$.

Подобные вершины графа Γ_I обладают следующими свойствами:

Пусть $V(\alpha_i, \beta_i)$ и $V(\alpha_j, \beta_j)$ - произвольные подобные вершины. Тогда, если

$$V(\alpha_p, \beta_p) \longrightarrow V(\alpha_i, \beta_i) \quad \text{,то} \quad V(\alpha_p, \beta_p) \longrightarrow V(\alpha_j, \beta_j)$$

и наоборот, если

$$V(\alpha_q, \beta_q) \longrightarrow V(\alpha_j, \beta_j) \quad \text{,то} \quad V(\alpha_q, \beta_q) \longrightarrow V(\alpha_i, \beta_i).$$

В самом деле, из $V(\alpha_p, \beta_p) \longrightarrow V(\alpha_i, \beta_i)$ следует $(\alpha_p, \beta_p) \longrightarrow (\alpha_i, \beta_i)$, т.е. $\alpha_p \subset \beta_i$, и,следовательно, $\alpha_p \subset \beta_j$, т.е. $(\alpha_p, \beta_p) \rightarrow (\alpha_j, \beta_j)$, отсюда $V(\alpha_p, \beta_p) \longrightarrow V(\alpha_j, \beta_j)$. Второе утверждение доказывается аналогично.

Граф Γ_I преобразуем в направленный граф Γ_2. Обозначим i -ую вершину графа Γ_2 с $V(A_i)$. Между Γ_I и Γ_2 имеют место следующие соотношения:

3.5.I. Все подобные вершины графа Γ_I сужаются в одной вершине графа Γ_2, т.е. всем подобным вершинам $V(\alpha_{i_1}, \beta_{i_1})$, $V(\alpha_{i_2}, \beta_{i_2})$,.. $V(\alpha_{i_k}, \beta_{i_k})$ $(1 \leq k \leq m)$ ставится в соответствие вершина $V(A_i)$, где $A_i = \beta_{i_1} = \ldots = \beta_{i_k}$

---

* Мы говорим, что вершины $V_{i_1}, V_{i_2}, \ldots, V_{i_k}$ направленного графа образуют цикл, если $V_{i_1} \rightarrow V_{i_2} \rightarrow \cdots V_{i_k} \rightarrow V_{i_1}$, где $i_p \neq i_q$, если $p \neq q$.

3.5.2. Если существуют i и j такие, что

а) $V(\alpha i, \beta i) \longrightarrow V(\alpha_j, \beta_j)$,

б) $\beta i = A_K$ и $\beta_j = Ae$,

то $V(A_K) \longrightarrow V(Ae)$.

Из описанного способа преобразования графа Γ_I в граф Γ_2 и из свойства подобных вершин графа Γ_I следует, что граф Γ_I имеет циклы тогда и только тогда, когда имеет циклы граф Γ_2. Следовательно, из отсутствия циклов в графе Γ_I следует их отсутствие и в графе Γ_2.

Рассмотрим граф Γ_2. Если вершина $V(A_j)$ изолирована (т.е. нет ребер, ориентированных от этой вершины или к ней), то множество A_j можно расположить где угодно при упорядочении результирующих множеств.

Рассмотрим направленный граф Γ_2', полученный из Γ_2 путем отбрасывания всех изолированных вершин. В графе Γ_2' существует хоть одна вершина, все инцидентные ребра которой ориентированы либо только от нее, либо только к ней. На самом деле, если такой вершины нет, то в любой вершине графа Γ_2' входят и из нее выходят ребра. Следовательно, имеет место

$$V(Ai_1) \longrightarrow V(A_{i_2}) \longrightarrow V(A_{i_n}) \longrightarrow V(A_{i_{n+1}}),$$

где $1 \le i_1, i_2, \ldots, i_n, i_{n+1} \le n$.

Так как среди индексов $i_1, i_2, \ldots, i_n, i_{n+1}$ могут быть не более чем n различных между собой, то существует минимальное К ($K \le n+1$) такое, что $i_1 = i_K$, и тогда будем иметь:

$$V(A_{i_1}) \longrightarrow V(A_{i_2}) \longrightarrow \cdots \longrightarrow V(A_{i_{K-1}}) \longrightarrow V(Ai_1),$$

т.е. граф Γ_2 имеет цикл. Пришли к противоречию.

Пусть вершина $V(A_{j_1})$ такая, что все ребра выходят из нее. Тогда множество A_{j_1} занимает очередное место (сверху вниз) при упорядочении результирующих множеств. Если вершина $V(A_{j_1})$ такая, что все ребра входят в нее, то множество A_{j_1} занимает очередное

место (снизу вверх) при упорядочении результирующих множеств.

Рассмотрим граф Γ_2^2, полученный из Γ_2^1 путем отбрасывания вершины $V(A_{j_1})$, и все входящие в нее (или все выходящие из нее) ребра. В графе Γ_2^2 существует вершина $V(A_{j_2})$, все инцидентные ребра которой ориентированы либо только от нее, либо только к ней. Следовательно, для множества A_{j_2} можно найти подходящее место при упорядочении результирующих множеств.

Продолжим этот процесс до получения графа Γ_2^k, который содержит только вершину $V(A_{j_k})$. Множество A_{j_k} занимает очередное место сверху вниз (или, что то же самое, занимает очередное место снизу вверх) при упорядочении результирующих множеств.

Доказательство достаточности дает нам идею построения алгоритма проверки выполнения необходимого и достаточного условия применимости правила умолчания и упорядочения результирующих множеств так, что процесс умолчания можно задать при помощи правила умолчания.

4. Описание результирующих множеств

Результирующие множества описываем набором формул.

4.I. Синтаксис формулы

Любая формула состоит из левой части, знака равенства и правой части. Левая часть есть целое число. Правая часть состоит из одной или нескольких компонент, разделенных между собой символом $\cup$. Любая компонента есть либо идентификатор, либо целое число, либо заключенные в фигурных скобках две или больше единиц, разделенных между собой вертикальной чертой, либо заключенные в квадратных скобках одна или несколько единиц, разделенных между собой вертикальной чертой. Единица есть либо целое число, либо идентификатор.

4.2. Семантика формулы

Идентификатор обозначает элемент множества A.

Целое число обозначает переменную, которая принимает значения--подмножества A.

Символ $\cup$ имеет обычный смысл из теории множеств.

Заключение в фигурные скобки определяет фиктивную переменную, совокупность допустимых значений которой есть объединение допустимых значений всех заключенных в этих скобках единиц (допустимым значением идентификатора является подмножество A, содержащее только один элемент, обозначенный этим идентификатором).

Заключение в квадратные скобки имеет тот же смысл, что и заключение в фигурные скобки, с той разницей, что совокупность допустимых значений заключенного в квадратных скобках обязательно содержит пустое подмножество A.

Любая формула определяет совокупность допустимых значений своей левой части.

Результирующие множества определяются как совокупность допустимых значений некоторой переменной.

Отметим, что при помощи описанных формул можно определить любую совокупность множеств.

5. Упорядочение результирующих множеств

Обозначаем:

α_i – компонента

β_i – единица

$\varkappa_i$ – идентификатор

Правая часть любой формулы определяет дерево левой части. Дерево правой части формулы получается применением следующих правил:

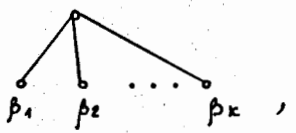
5.I. $\alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_K = (\ldots((\alpha_1 \cup \alpha_2) \cup \alpha_3) \cup \ldots \cup \alpha_K) \quad (K \geqslant 2)$

Результат операции $\cup$ является деревом, полученным путем прикрепления к любой терминальной вершине дерева первого операнда, дерева второго операнда.

Замечание: расстановка круглых скобок не имеет значения из-за ассоциативности операции $\cup$.

5.2. Дерево компоненты $\{\beta_1 | \beta_2 | \dots | \beta_k\}$ $(k \geq 2)$

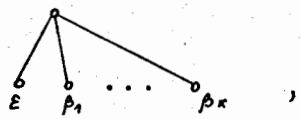
имеет вид:



где вершина β_i является головной вершиной дерева β_i .

5.3. Дерево компоненты $[\beta_1 | \beta_2 | \dots | \beta_k]$ $(k \geq 1)$

имеет вид:



где вершина β_i является головной вершиной дерева β_i , а вершина ε – терминальная.

5.4. Дерево идентификатора $\mathcal{H}_i$ имеет вид:

где идентификатор $\mathcal{H}_i$ есть идентификатор ребра.

Рассмотрим путь в дереве некоторой переменной (целого числа). Все элементы (идентификаторы) ребер по этому пути задают A_i .

Обход всех путей дерева слева направо задает упорядочение результирующих множеств.

Отметим, что набор формул не только описывает результирующие множества, но и задает их упорядочения.

6. Описание результирующих множеств для языка PL/1

I = {AUTOMATIC |BASED| CONTROLLED| STATIC }

2 = [POSITION] ∪ DEFINED

3 = INTERNAL ∪ {1|2}

4 = EXTERNAL ∪ { STATIC|CONTROLLED}

5 = [SECONDARY] ∪ { 3|4}

6 = Parameter ∪ INTERNAL ∪ [CONNECTED|CONTROLLED]

7 = Major-structure ∪ VARIABLE ∪ [Dimenjion] ∪
[ALIGNED|UNALIGED] ∪ {5|6}

8 = Minor-structure ∪ VARIABLE ∪ [Dimendion] ∪
[ALIGNED|UNALIGNED] ∪ INTERNAL

9 = Scale-factor ∪ FIXED

I0 = {ALIGNED|UNALIGNED} ∪ { DECIMAL|BINARY} ∪
Number-of-digits ∪ { FLOAT|9}

II = {UNALIGNED|ALIGNED}∪ PICTURE-N

I2 = {REAL|COMPLEX}∪ { 10|11}

I3 = { CHARACTER|BIT } ∪ [VARYING] ∪ Length

I4 = { UNALIGNED|ALIGNED} ∪ { 13|PICTURE-C }

I5 = INTERNAL ∪ [INITIAL]

I6 = 12 ∪ 15

I7 = 14 ∪ 15

I8 = ENTRY ∪ { IRREDUCIBLE|REDUCIBLE } ∪ [OPTIONS] ∪
RETURNS

I9 = 18 ∪ 15

20 = {TASK|EVENT|FILE|LABEL|POINTER|OFFSET }∪ 15

2I = Size ∪ AREA

22 = 21 ∪ 15

23 = {ALIGNED|UNALIGNED} ∪ {19|20|22}

24 = Base-element ∪ VARIABLE ∪ [Dimension] ∪ {16|17|23}

25 = 1 ∪ [INITIAL]

26 = [SECONDARY] ∪ INTERNAL ∪ {25|2}

27 = [SECONDARY] ∪ EXTERNAL ∪ {STATIC|CONTROLLED} ∪
[INITIAL]

28 = CONTROLLED ∪ [INITIAL]

29 = Parameter ∪ INTERNAL ∪ [CONNECTED|28]

30 = {AUTOMATIC|BASED|CONTROLLED} ∪ [INITIAL]

31 = [SECONDARY] ∪ INTERNAL ∪ {30|STATIC|2}

32 = [SECONDARY] ∪ EXTERNAL ∪ {STATIC|28}

33 = {12|14} ∪ {26|27|29}

34 = {ALIGNED|UNALIGNED} ∪ {TASK|EVENT|FILE|21} ∪
{26|27|29}

35 = {ALIGNED|UNALIGNED} ∪ {LABEL|POINTER|OFFSET|18} ∪
{31|32|29}

36 = VARIABLE ∪ [Dimension] ∪ {34|35}

37 = Constant ∪ {REAL|COMPLEX} ∪ ALIGNED ∪
{DECIMAL|BINERY} ∪ Number-of-digits ∪ Scale-factor ∪
{FIXED|FLOAT} ∪ {DECIMAL-interpreted|BINARY-interpreted}

38 = Constant ∪ UNALIGNED ∪ {CHARACTER|BIT} ∪
Length

39 = RETURNS ∪ EXTERNAL

40 = [RETURNS] ∪ INTERNAL

41 = Constant ∪ ALIGNED ∪ ENTRY ∪
{IRREDUCIBLE|REDUCIBLE} ∪ [OPTIONS] ∪ {39|40}

42 = [PRINT] ∪ OUTPUT

43 = STREAM ∪ {INPUT|42}

44 = BITSTEAM ∪ { INPUT|OUTPUT}

45 = INPUT ∪ [KEYED] ∪ [BACKWARDS]

46 = OUTPUT ∪ [KEYED]

47 = UPDATE ∪ [KEYED]

48 = SEQUENTIAL ∪ {BUFFERED|UNBUFFERED} ∪ {45|46|47}

49 = INPUT ∪ KEYED

50 = OUTPUT ∪ KEYED

51 = UPDATE ∪ [EXCLUSIVE] ∪ KEYED

52 = DIRECT ∪ { UNBUFFERED|BUFFERED} ∪ {49|50|51}

53 = INPUT ∪ [KEYED]

54 = TRANSIENT ∪ { BUFFERED|UNBUFFERED} ∪ {53|46}

55 = RECORD ∪ { 48|52|54}

56 = Constant ∪ ALIGNED ∪ FILE ∪ { EXTERNAL|INTERNAL} ∪
[ENVIRONMENT] ∪ {43|44|55}

57 = {ALIGNED|UNALIGNED} ∪ { TASK|EVENT|FILE|POINTER|OFFSET|21|18}

58 = RETURNS( ) ∪ {12|14|57}

59 = Constant ∪ ALIGNED ∪ LABEL ∪ INTERNAL

60 = Constant ∪ ALIGNED ∪ GENERIC ∪ INTERNAL

61 = Constant ∪ ALIGNED ∪ BUILTIN ∪ INTERNAL

62 = Constant ∪ ALIGNED ∪ Condition ∪ EXTERNAL

63 = {64|56|41|36|7|8|24|58|37|38|59|60|61|62}

64 = VARIABLE ∪ [Dimension] ∪ 33

Результирующие множества определяются как совокупность допустимых значений переменной (целого числа) 63. Дерево переменной 63 задает упорядочение результирующих множеств.

7. Заключение

Элементы множества A были выделены на основании определенных конструкций языка PL/I. Из всех 69 элементов множества A 64 были выделены на основании описателей языка PL/1 . Только элементы base-element, BINARY-interpreted, DECIMAL-interpreted, Major-structure и Minor-structure выделены на основании других конструкций. Это выделение подчинялось цели получения модели принципа умолчания.

Если элемент обозначен идентификатором, который содержит только большие буквы, то он основывается на описателе, который явно объявляется конструкцией, содержащей этот идентификатор в качестве служебного слова. Таких элементов в множестве A 53.

В предлагаемой модели принципа умолчания не отражено влияние первой буквы идентификатора на получение полной совокупности описателей, так как непосредственное включение этого влияния приводит к невыполнению необходимого и достаточного условия применимости правила умолчания.

Описатель PICTURE для числовых данных и описатель PICTURE для данных типа символьных строк вызывают разные процессы умолчания. Поэтому в модели на основании описателя PICTURE выделены два элемента: PICTURE-N (для числовых данных) и PICTURE-C (для данных типа символьных строк).

В результате конструирования результирующих множеств были устранены все нелогичности описания процесса умолчания. Предложенные изменения процесса умолчания не меняют его по существу. Они сохраняют сложность процесса умолчания и создают удобства для его описания при помощи правила умолчания.

Множества B_i в языке PL/I являются частью всех возможных подмножеств A_i ($I \leq i \leq 4224$). Следовательно, применение правила умолчания не может быть единственным средством для обнаружения синтаксических ошибок при задании множеств B_i .

Иногда можно путем выделения нескольких элементов на основании одного описателя добиться более полного синтаксического контроля объявления этого описателя. В этом случае процесс умолчания не меняется по существу, но из-за увеличения результирующих множеств он становится менее эффективно реализуемым на ЭВМ. Поэтому мы отказались от выделения нескольких элементов на основании описателя INITIAL .

Распределение результирующих множеств по количеству их элементов следующее:

Количество элементов	Процент результирующих множеств	Количество элементов	Процент результирующих множеств
3	0,4	8	I8,0
4	I,8	9	2I,4
5	6,I	I0	I7,3
6	II,8	II	7,3
7	I4,7	I2	I,2

Из таблицы видно, что часто будут встречаться результирующие множества из 8,9 и I0 элементов (56,7%), а редко – из 3,4 и I2 элементов (3,4%). Следовательно, если мы обязаны явно задавать полную совокупность описателей, то в более чем половине случаев мы должны писать 8,9 и I0 описателей для каждого идентификатора (или,в среднем,8,I6 описателей) (в предположении равномерного использования всех результирующих множеств). В частности,

чтобы преодолеть такое неудобство практического использования языка PL/1, в нем был применен процесс умолчания. Применимость процесса умолчания принципиально возможна, если не все подмножества результирующих множеств являются результирующими множествами. В языке PL/I число подмножеств результирующих множеств, которые не являются результирующими множествами, приблизительно в 35 раз больше числа результирующих множеств.

Формулы, синтаксис которых описан в 4.I, отражают структуру представления результирующих множеств в памяти ЭВМ. Эти формулы можно обобщить с целью получения их более компактной записи. Если разрешить вложение символа $\cup$ в квадратные и фигурные скобки, то число формул уменьшится от 63 до 50. Если вдобавок разрешить вложение квадратных и фигурных скобок с любой глубиной, то все формулы могут быть редуцированы в одну.

Например, из формул 9,I0,II и I2 мы получаем формулу:

$$I2 = \{\text{REAL}|\text{COMPLEX}\} \cup \{\{\text{ALIGNED}|\text{UNALIGNED}\} \cup \{\text{DECIMAL}|\text{BINARY}\} \cup \text{Number-of-digits} \cup \{\text{FLOAT}|\text{Scale-factor} \cup \text{FIXED}\} | \{\text{UNALIGNED}|\text{ALIGNED}\} \cup \text{PICTURE-N}\}.$$

Автор искренне благодарен Н.Н.Говоруну и В.П.Ширикову за содействие, Е.А.Жоголеву за полезные обсуждения и И.Б.Бычваровой за техническую помощь при осуществлении данной работы.

ЛИТЕРАТУРА

I. Универсальный язык программирования PL/I. (Перевод с английского под редакцией В.М.Курочкина. Москва, I968).

2. IBM SYSTEM/360 Operating System PL/1 Language Specifications Form C 28-6571-4, IBM, 1965.

3. Charles Philip Lecht, The Programmer's PL/1, A Complete Reference, McGRAW-HILL Book Company, 1968.

4. PL/1 Language Specifications Order Number GY 33-6003-2 (Major Revision June 1970) IBM 1970.

Рукопись поступила в издательский отдел
I6 мая I972 г.