Д. Ю. Булычев

СОПОСТАВЛЕНИЕ С ОБРАЗЦОМ И ИДЕНТИФИКАЦИЯ СТРУКТУР В ЯЗЫКЕ PL/I

Идентификация структур в языке PL/I [1] является одной из самых неприятных задач его семантического анализатора. Основная ее сложность происходит оттого, что указания промежуточных подструктур и полей при обращении к элементам структурного значения могут отсутствовать. Таким образом, выборка из структуры фактически определяет образец, которому должен соответствовать единственный элемент контекста. Идентификация структур, следовательно, сводится к решению задачи сопоставления с образцом. В работе [2] содержится изложение основных трудностей при идентификации, а также приведен эвристический алгоритм.

Данная статья содержит решение задачи идентификации применением общего подхода сопоставления с образцом для деревьев. Раздел 1 содержит формализацию, которая представляется подходящей: контекст рассматривается как лес, а квалификатор выборки (агрегат) — как образец, которому должен сопоставляться точно один элемент контекста. Раздел 2 описывает некоторый подход к сопоставлению с образцом, являющийся вариантом системы переписывания деревьев [3–5]. В разделе 3 этот подход использован для решения данной задачи, что позволило получить обоснование его корректности и сформулировать оценку сложности. Раздел 4 содержит описание реализации базового алгоритма идентификации, а раздел 5 — его расширение для целей видозависимого анализа. Наконец, в разделе 6 приведены результаты сравнения предложенного решения с существующей реализацией.

Автор выражает признательность Ю.А. Губанову, А.Н. Иванову, Д.В. Кознову, О.М. Смирновой и А.А. Терехову за обсуждение данной работы и ценные замечания.

1. Формальная постановка

Описание PL/I [1] не содержит формального определения однозначности идентификации элементов структур, оперируя многочисленными примерами. Некоторую определеность вносит то, что, согласно описанию, точная идентификация обладает приоритетом по сравнению со всеми остальными. Это значит, что если в дереве структуры существует единственный путь от корня, который в точности соответствует выборке, то в качестве результата идентификации принимается именно он. Таким образом, первой задачей оказывается построение подходящей формализации.

Элемент данных, имеющий структурный тип, может быть рассмотрен как дерево, которое отражает относительную вложенность подструктур. Поскольку подструктуры и поля снабжены идентификаторами, это дерево будет помеченным. Не умаляя общности, можно считать, что задан некоторый универсальный алфавит пометок M, который на практике может быть получен коллекционированием всех идентификаторов, встретившихся в качестве имен структур, подструктур или полей. В этом помеченном дереве каждому пути μ соответствует слово $w_\mu \in M^*$, которое получается последовательным выписыванием пометок его вершин.

Агрегатом назовем любое слово в алфавите M. В программах ему отвечает (частичное) указание полей при выборке из структуры. В дальнейшем будем обозначать агрегаты $I_1 I_2 ... I_n$, где $I_j \in M$.

Пусть задан элемент данных σ и агрегат $I_1 I_2 ... I_n$. Рассмотрим регулярное выражение $\pi = \{\_*\} I_1 \{\_*\} ... \{\_*\} I_n \{\_*\}$, в котором значок «_» обозначает произвольный символ M.[1] Рассмотрим в σ множество P всех путей от корня, соответствующие слова которых распознаются регулярным выражением π.[2] Будем говорить, что

- дерево σ *соответствует* $I_1 I_2 ... I_n$, если в P существует наименьший относительно включения элемент;

[1] Разумеется, это можно было бы выписать явно, но из соображения экономии места мы так поступать не будем.

[2] Точнее, распознаются автоматом, соответствующим π. В дальнейшем мы не будем различать регулярное выражение и автомат, а также агрегат и соответствующее ему регулярное выражение, разрешая неоднозначность по контексту.

- дерево σ *точно соответствует* $I_1I_2...I_n$, если в P существует единственный элемент, который распознается выражением $I_1I_2...I_n\{\_*\}$;
- идентификация $I_1I_2...I_n$ однозначна в дереве σ, если σ соответствует или точно соответствует $I_1I_2...I_n$.

Это определение хорошо согласуется с интуитивным представлением о сущности идентификации структур. Во-первых, если в дереве есть произвольный (но не содержащий более одного сына любой вершины) путь, распознаваемый выражением π (что, вообще говоря, требуется для выборки), то дополнение этого пути до пути от корня также распознается выражением π в силу того, что π допускает некоторые слова вместе с любыми их префиксами. Кроме того, именно путь от корня содержательно соответствует понятию «выборка», так как он описывает полную последовательность подструктур, содержащих искомый элемент данных. Наличие среди таких путей единственного, точно соответствующего агрегату, означает возможность однозначной идентификации. Если же такого элемента нет, но множество содержит наименьший элемент (т. е. самый короткий путь, являющийся подпутем любого другого), то идентификация также может быть выполнена однозначно в силу его очевидной единственности.

Например, идентификация агрегата AC в дереве a на рис. 1 не может быть выполнена однозначно, так как существуют два несравнимых пути от корня, соответствующих образцу $\{\_*\}A\{\_*\}C\{\_*\}$. В то же время такая идентификация возможна для дерева b, поскольку среди всех подходящих путей существует минимальный. Наконец, в дереве c не существует минимального пути, отвечающего образцу ABC, но зато есть единственный путь, который соответствует ему точно, — следовательно, идентификация также однозначна.

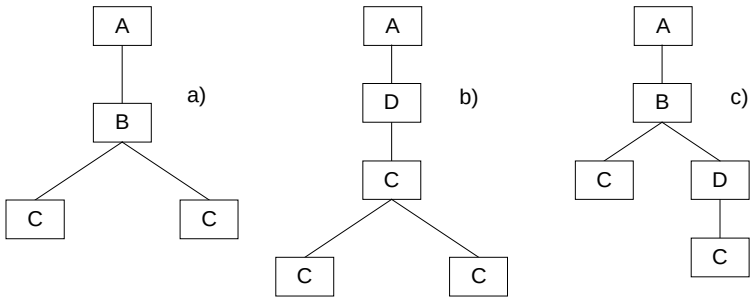

Рис. 1

Тривиальным решением задачи идентификации, таким образом, является построение для данного агрегата конечного автомата, соответствующего регулярному выражению π, проверка его применимости ко всем путям от корня σ с последующим поиском минимального или точно соответствующего. Более совершенное решение может быть получено применением подхода восходящего переписывания деревьев (bottom-up tree rewriting).

2. Восходящее переписывание деревьев

Здесь мы рассмотрим некоторую процедуру, которая необходима для формализации и обоснования корректности предлагаемого решения. Эта процедура является вариантом реализации систем восходящего переписывания деревьев (Bottom-Up Rewrite System, BURS), которые широко применяются в различных задачах, таких, как генерация оптимального кода, видозависимый анализ и другие [3–5]. Нам в данном случае требуется некоторый стандартный механизм разметки, формализация которого была бы наиболее простой. С этой точки зрения грамматики восходящего переписывания выглядят наиболее привлекательно.

Атрибутная грамматика восходящего переписывания деревьев — это система

G = (T, N, Ψ, A, R), где

- T — алфавит терминалов;
- N — алфавит нетерминалов;
- Ψ — множество (возможно бесконечное) атрибутов;

- $A = \{\alpha^{n,m} \mid \alpha^{n,m} : \Psi^n \times N^m \to \Psi\}$ — алфавит преобразователей атрибутов;
- R — множество правил вида $K[\alpha^{n,m}(\omega_1,...,\omega_n;K_1,...,K_m)] \to \rho$, где

  $\alpha^{n,m}$ — некоторый преобразователь атрибутов;
  K — выводимый данным правилом нетерминал;
  ρ — образец, имеющий следующий вид:
  а) $\rho = a$, $a \in T$, множества нетерминалов ρ_N и атрибутов ρ_Ψ образца ρ пусты;
  б) $\rho = a(K_1(\omega_1),...,K_n(\omega_n))$, $a \in T$, $\rho_N = \{K_i \mid K_i \in N\}$ — множество нетерминалов ρ, $\rho_\Psi = \{\omega_i \mid \omega_i \in \Psi\}$ — множество атрибутов ρ.

В приведенной формализации грамматика не может содержать цепных правил.

Пусть есть дерево с вершинами, помеченными символами из T. Будем говорить, что вершина дерева x соответствует образцу ρ, если

- $\rho = a$, $a \in T$, вершина x — лист с пометкой a;
- $\rho = a(K_1(\omega_1),...,K_n(\omega_n))$, $a \in T$, $K_i \in N$, $\omega_i \in \Psi$, x — вершина с сыновьями $y_1,...,y_n$, x помечена символом a, y_i имеет пометку K_i;

Пусть $r = K[\alpha^{n,m}(\omega_1,...,\omega_n;K_1,...,K_m)] \to \rho$ — правило, x — вершина, которая соответствует ρ. Тогда правило r определяет разметку x в следующем смысле: x помечается нетерминалом K с атрибутом $\alpha^{n,m}(\omega_1,...,\omega_n;K_1,...,K_m)$, где $\omega_i \in \rho_\Psi$, $K_i \in \rho_N$.

Далее мы будем рассматривать только грамматики, которые удовлетворяют следующему ограничению: если для вершины x применимы правила $K_1[\alpha^{k,l}(\omega_1,...,\omega_k;K_1,...,K_l)] \to \rho_1$ и $K_2[\beta^{m,n}(\varpi_1,...,\varpi_m;K_1,...,K_n)] \to \rho_2$, то $\alpha^{k,l}(\omega_1,...,\omega_k;K_1,...,K_l) = \beta^{m,n}(\varpi_1,...,\varpi_m;K_1,...,K_n)$. Иными словами, такие грамматики приписывают вершинам дерева атрибуты, значения которых не зависят от порядка применения правил.

Неформально говоря, правило грамматики определяет процедуру вычисления некоторого свойства в вершине дерева в зависимости от ее пометки и свойств, вычисленных для ее сыновей. Это свойство представляется множеством нетерминалов, каждый из которых выведен применением соответствующего правила и снабжен атрибутом, вычисленным по этому правилу. Таким образом, информация распространяется в дереве снизу вверх: пометка вершины зависит только от ее собственной терминальной пометки и нетерминальных пометок ее сыновей.

Поскольку образцы грамматики — это деревья, разметка вершины дерева τ зависит только от разметки ее поддеревьев и не зависит от разметки вышележащих или соседних вершин, что дает возможность построить эффективный алгоритм ее нахождения. Ниже приведена схема такого алгоритма, который известен как алгоритм восходящего переписывания деревьев:

```
label (v:vertex, G:grammar)
{
  for ∀y∈v.sons do label (y, G);
    (* пометить поддеревья *)
  for ∀(N[α(ω1,...,ωn;K1,...,Km)]→ρ)∈G.R
    (* применить правила   *)
  do
    if match (v, ρ) then
     v.labels ∪:= N[α(ω1,...,ωn;K1,...,Km)]
     where ωi∈ρΨ, Ki∈ρN;
}
```

Сложность алгоритма переписывания зависит от структуры грамматики. Приведенный алгоритм в худшем случае применяет каждое правило к каждому узлу не более одного раза. Действительно, применимость правила целиком определяется начальной терминальной пометкой вершины и нетерминальной разметкой ее сыновей. Таким образом, появившаяся в данной вершине x нетерминальная пометка не влияет на возможность дальнейшего применения правил к x (при наличии цепных правил это не так), что дает $O(|R| \times |V|)$ применений обычного сопоставления с образцом в правой части правила и вычислений атрибутов (здесь $|V|$ — количество вершин дерева).

3. Решение задачи идентификации

Как следует из формализации, для решения поставленной задачи необходимо для данного агрегата $I_1I_2...I_n$ уметь распознавать свойства соответствия и точного соответствия ему дерева структуры.

Поскольку исходное дерево структуры σ может иметь любую степень, преобразуем его в дерево σ', имеющее степень не выше двух. Для этого «разбавим» его следующим образом:

1. Для каждой вершины x в σ с наследниками $y_1,...,y_k$, $k>2$, преобразуем фрагмент $x(y_1,...,y_k)$ в $x(z_1(z_2(...z_{k-1}(y_1,y_2)...),y_k))$, где z_i — новые вершины со специальной пометкой $\varepsilon \notin M$.
2. Для каждой вершины x в σ с наследниками y_1, y_2 и пометкой из $\{I_1,I_2,...,I_n\}$ преобразуем фрагмент $x(y_1,y_2)$ в $x(z(y_1, y_2))$, где z — новая вершина со специальной пометкой $\varepsilon \notin M$.

Очевидно, что если вершина дерева σ' помечена символом из $\{I_1,I_2,...,I_n\}$, то ее степень не более единицы. На рис. 2 показано исходное дерево, а на рис. 3 — дерево после применения описанного преобразования. Символом (*) отмечены вставленные вершины.

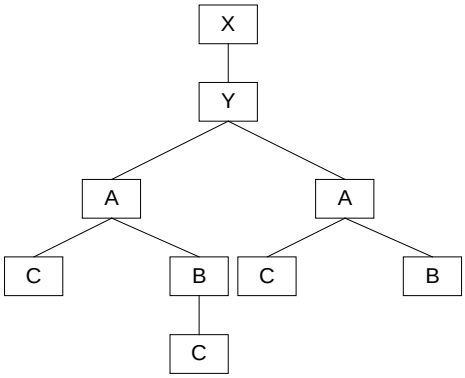

Рис. 2

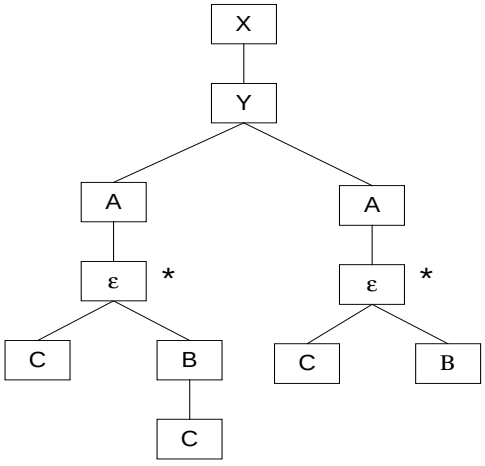

Рис. 3

Пусть есть некоторый путь μ от корня σ. Легко видеть, что преобразование однозначно сопоставляет ему путь μ' в σ', отличающийся от μ только вставкой вершин с пометкой ε. Кроме того, преобразование не меняет отношение вложенности путей. Таким образом, дерево σ' соответствует $I_1I_2...I_n$ тогда и только тогда, когда этим свойством обладает σ. Если же при рассмотрении пометок вершин не учитывать символ ε, то это становится верным и для отношения точного соответствия. Следовательно, приведенное преобразование корректно в рамках данной задачи. Далее мы будем считать, что дерево уже представлено в этой специальной форме и среди пометок его вершин выделен символ ε, которым помечены только вставленные вершины.

Построим атрибутную грамматику переписывания, такую, что определяемая ею разметка может быть интерпретирована как результат точного сопоставления дерева σ агрегату $I_1I_2...I_n$.

Рассмотрим следующую грамматику:

$G_1 = (N, T, R, \Psi)$
$T = M \cup \{\varepsilon\}$,
$N = \{K\}$,
$\Psi = 2^{\{N_1,...,N_n\}\times\{1,\infty\}}$,
$R = \{$

$$K[\{(N_n,1)\}] \to J,\ J=I_n, \quad (1)$$

$$K[\varnothing] \to x,\ x \neq I_n, \quad (2)$$

$$K[\varnothing] \to x(K[\omega]),\ x \notin \{I_1,I_2,...,I_n\}, \quad (3)$$

$$K[\varnothing] \to x(K[\omega_1],K[\omega_2]),\ x \notin \{I_1,I_2,...,I_n\},\ x \neq \varepsilon \quad (4)$$

$$K[\gamma(\omega_1, \omega_1)] \to \varepsilon(K[\omega_1], K[\omega_2]), \quad (5)$$

$$K[\delta(\omega; J)] \to J(K[\omega]),\ J \in \{I_1,...,I_n\} \quad (6)$$

$\}$,

где

$$\gamma(\omega_1, \omega_2) =$$
$$\{(N_i, k+m) : \exists(N_i, k) \in \omega_1 \ \&\ \exists(N_i, m) \in \omega_2\} \cup$$
$$\{(N_i, k) : \exists(N_i, k) \in \omega_1 \ \&\ \forall m(N_i, m) \notin \omega_2\} \cup$$
$$\{(N_i, m) : \forall k(N_i, k) \notin \omega_1 \ \&\ \exists(N_i, m) \in \omega_2\}$$

и операция + определена следующим образом:

+	1	∞
1	∞	∞
∞	∞	∞

$$\delta(\omega; J) = \{(N_{k-1}, m) : (N_k, m) \in \omega \ \&\ J = I_{k-1}\} \cup \{(N_n, 1) : J = I_n\}$$

Множество правил данной грамматики разбивается таким образом на шесть схем, каждая из которых призвана описать разметку одного из возможных вариантов устройства текущей вершины. Поскольку каждая вершина помечена ровно одним терминалом, только одно правило из соответствующей схемы может быть использовано для ее разметки. Следовательно, данная грамматика размечает дерево однозначно.

Неформально, единственный нетерминал грамматики просто отражает тот факт, что данная вершина уже размечена. Атрибут же представляет собой множество пар, в которых первый член описывает некоторый подобразец (т. е. постфикс $I_kI_{k+1}...I_n$), а второй — число путей (один или много) от данной вершины, которые точно соответствуют этому подобразцу. Преобразователь γ объединяет атрибуты сыновей ε-вершины, а преобразователь δ достраивает атрибут в зависимости от терминальной пометки текущей вершины.

Более строго, имеет место следующее утверждение:

Утверждение 1. Пусть вершина x при разметке в соответствии с G_1 получила пометку $K[\omega]$. Тогда $(N_k, m) \in \omega$ тогда и только тогда, когда в поддереве с корнем x существуют m путей, которые при неучете ε-вершин точно соответствуют $I_kI_{k+1}...I_n$.

Доказательство. Индукция по высоте дерева.

В случае тривиального дерева применимы только правила (1) или (2); кроме того, очевидно, что пометка единственной вершины отлична от ε. Применение правила (1) сопоставляет вершине пометку $K[\{N_n, 1\}]$, что соответствует утверждению, так как вершина с пометкой $J=I_n$ является единственным путем, точно соответствующим подобразцу I_n. Применение правила (2) также дает нужную разметку, поскольку в рассматриваемом дереве ни один путь не соответстует точно никакому подобразцу. Осталось заметить, что к данной вершине всегда применимо только одно из этих правил и что оно обязательно будет применено при разметке в соответствии с приведенным алгоритмом.

Далее символом T_r будем обозначать дерево с корнем r.

Пусть имеется дерево T_r произвольной высоты. Рассмотрим его корень r и предположим, что после разметки r получил пометку $K[\omega]$. Как уже отмечалось, к каждой вершине может быть применено только одно правило, и оно будет обязательно применено описанным алгоритмом. Таким образом, дальнейшие рассуждения сводятся к перебору случаев применения различных правил к r.

Случай 1. Применяемое правило — (3). Заметим, что r не может быть помечена символом ε, поскольку ее степень равна единице. Но в этом случае r вносит посторонний символ во все пути T_r, и, таким образом, ни один из них не может точно соответствовать ни одному подобразцу.

Случай 2. Применимое правило — (4). Аналогично.

Случай 3. Применимое правило — (5). Обозначим через T_x и T_y поддеревья r. По индукционному предположению, пара (N_k,m) содержится в атрибуте $T_x(T_y)$ тогда и только тогда, когда в $T_x(T_y)$ существует m путей от корня, точно соответствующих подобразцу $I_kI_{k+1}...I_n$ за неучетом ε-вершин. Для любого такого пути μ путь $r\mu$ также обладает этим свойством, поскольку r помечена символом ε. Таким образом, множество путей в T_r, точно соответствующих некоторому подобразцу $I_kI_{k+1}...I_n$ за вычетом ε-вершин, есть объединение множеств таких путей в T_x и T_y. Но преобразователь γ как раз и вычисляет атрибут, значение которого есть объединение в этом смысле атрибутов T_x и T_y.

Случай 4. Применимое правило — (6). Обозначим через T_x единственное поддерево r. Тогда для x имеет место индукционное предположение. Пусть пара (N_k,m) содержится в атрибуте x. Если r помечена символом I_{k-1}, то в T_r существует ровно m путей, точно соответствующих подобразцу $I_{k-1}I_k...I_n$ — они получаются приписыванием r в начало каждого пути T_x, точно соответствующего $I_k...I_n$. Если же r помечена символом I_n, то она сама есть единственный путь в T_r, точно соответствующий подобразцу I_n. Поскольку символы I_j не обязаны быть попарно различными, эти множества следует объединить, что и делает преобразователь δ. Утверждение доказано.

Следствие 1. Дерево σ точно соответствует агрегату $I_1...I_n$ тогда и только тогда, когда переписывание, порождаемое G_1, сопоставляет корню преобразованного дерева σ' пометку $K[\omega]$, где $(N_1,1)\in\omega$.

Для доказательства достаточно заметить, что корень σ' совпадает с корнем σ, а отношение точного соответствия σ равноценно отношению точного соответствия σ' за неучетом ε-вершин.

На рис. 4 изображена разметка, полученная для данного дерева применением правил грамматики G_1, построенной для агрегата ABC. В скобках указаны номера правил. Поскольку корень дерева получил пометку $K[\omega]$ и $(N_1,1)\in\omega$, рассматриваемое дерево точно соответствует агрегату.

Перейдем ко второй задаче. Рассмотрим соответствующее агрегату $I_1I_2...I_n$ регулярное выражение $\pi = \{\_^*\}\ I_1\ \{\_^*\}\ I_2\ \{\_^*\}\ ...\ I_{n-1}\ \{\_^*\}\ I_n\ \{\_^*\}$.

Построим по нему атрибутную грамматику G_2 следующего вида:

$G_2 = (N,T,R,\Psi)$

$T = M\cup\{\varepsilon\}$,

$N = \{N_1,...,N_n,N_{n+1}\}$,

$\Psi = 2^{\{N1,...,Nn\}}$,

$R = \{$

$N_n[\{N_n\}]\to J,\ J=I_n,$ (1)

$N_{n+1}[\varnothing]\to x,\ x\neq I_n,$ (2)

$N_i[\omega]\to x(N_i[\omega]),1\le i\le n+1,\ x\notin \{I_1,I_2,...,I_n\},$ (3)

$N_{\min\{i,k\}}[\varphi(\omega_1,\omega_2,N_i,N_k)]\to x(N_i[\omega_1],N_k[\omega_2]),$ (4)
$1\le i,k\le n+1,\ x\notin \{I_1,I_2,...,I_n\},$

$N_{m-1}[\psi(\omega,J)]\to J(N_m[\omega]),\ J=I_{m-1},$ (5)

$N_m[\psi(\omega,J)]\to J(N_m[\omega]),J\in \{I_1,I_2,...,I_n\},\ J\neq I_{m-1}$ (6)

$\}$,

где

$$\varphi(\omega_1, \omega_2, N_i, N_j) = \begin{cases} \omega_1 \setminus \{N_j, ..., N_n\}, i < j, \\ \omega_2 \setminus \{N_i, ..., N_n\}, i \geq j, \end{cases}$$

$$\psi(\omega, J) = \omega \cup \{N_k : J = I_k \,\&\, N_{k+1} \in \omega\} \cup \{N_n : J = I_n\} \setminus \{N_k : J = I_k \,\&\, N_k \in \omega \,\&\, N_{k+1} \notin \omega\}.$$

Неформально, нетерминал грамматики описывает теперь максимальный подобразец, которому соответствует хотя бы один путь в поддереве с корнем в данной вершине. Атрибут же описывает множество минимальных путей, соответствующих некоторому подобразцу.

Заметим снова, что к данной вершине может быть применено только одно правило. Следовательно, данная грамматика также сопоставляет дереву однозначную разметку. Из этого же вытекает, что любая вершина будет помечена ровно одним нетерминалом.

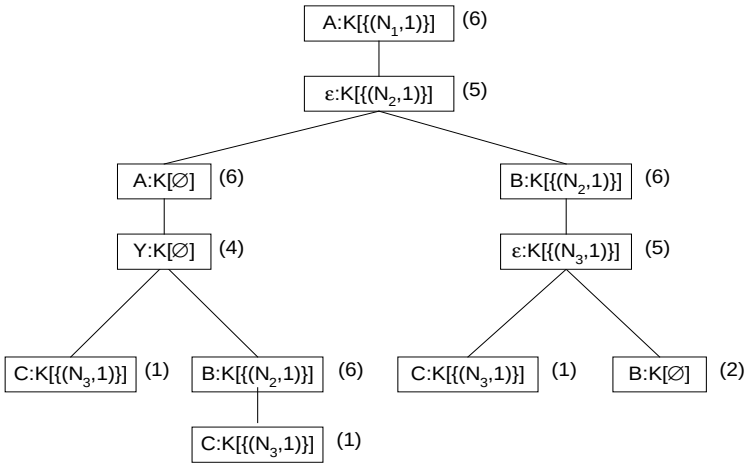

Рис. 4

Утверждение 2. Поддерево дерева σ' соответствует подобразцу $I_kI_{k+1}...I_n$ в том и только том случае, когда переписывание в соответствии с G_2 сопоставляет его корню разметку $N_m[\omega]$ и $N_k \in \omega$. При этом $I_mI_{m+1}...I_n$ есть максимальный подобразец, которому соответствует хотя бы один путь от корня этого поддерева.

Доказательство. Доказательство будем вести индукцией по высоте переписываемого поддерева. При этом T_r опять будет обозначать поддерево с корнем r.

Для дерева высоты 0 (состоящего из одного листа) единственными применимыми правилами являются (1) и (2), причем эти применения взаимоисключаемые. Таким образом, возможной пометкой корня является либо $N_n[\{N_n\}]$ (если он помечен символом I_n и, следовательно, является минимальным путем, соответствующим I_n), либо $N_{n+1}[\varnothing]$. В последнем случае дерево не соответствует никакому подобразцу $I_1...I_n$. Поскольку при переписывании применяются все применимые правила, база доказана.

Пусть теперь мы рассматриваем разметку дерева высотой $h>0$. В силу изложенных выше причин возможны только следующие варианты:

3. Корень r имеет пометку $J \in \{I_1, I_2, ..., I_n\}$. Тогда у него ровно один наследник x и единственными применимыми правилами являются (5) или (6), причем эти применения взаимоисключаемые, поскольку, как было замечено ранее, каждая вершина (в том числе и x) помечается ровно одним нетерминалом. Пусть x имеет пометку $N_m[\omega]$.

Случай 1: $J=I_{m-1}$, применимым правилом является (5). По индукционному предположению, в дереве T_x существует хотя бы один путь μ, который соответствует подобразцу $I_mI_{m+1}...I_n$ и нет пути, соответствующего большему подобразцу. Но тогда максимальный подобразец, которому соответствует хотя бы один путь в T_r, есть $I_{m-1}I_m...I_n$, поскольку таким путем является путь $r\mu$.

Рассмотрим теперь значение атрибута. Пусть атрибут x есть ϖ. Если в T_x существует минимальный путь μ, соответствующий подобразцу $I_{k+1}...I_n$, то, по индукционному предположению, $N_{K+1} \in \varpi$. Если $J=I_k$, то путь $r\mu$ будем минимальным путем в T_r, соответствующим подобразцу $I_k...I_n$, и, следовательно, $N_k \in \psi(\varpi, J)$. Если $N_k \in \varpi$, $J=I_k$ и $N_{k+1} \notin \varpi$,

то в поддереве T_r не может быть минимального пути, соответствующего $I_k...I_n$, так как в T_x нет минимального пути, соответствующего $I_{k+1}...I_n$. Для таких k $N_k \notin \psi(\varpi, J)$. Остальные элементы ϖ наследуются атрибутом r, так как ее терминальная пометка никак не влияет на множество минимальных путей, описываемых ими. Наконец, если $J=I_n$, то r есть минимальный путь в T_r, сооветствующий I_n. Следовательно,

$$\psi(\omega, J) = \omega \cup \{N_k : J = I_k \,\&\, N_{k+1} \in \omega\} \cup \{N_n : J = I_n\} \setminus \{N_k : J = I_k \,\&\, N_k \in \omega \,\&\, N_{k+1} \notin \omega\}.$$

Случай 2: $J \neq I_{m-1}$, применимым правилом является (6). Поскольку в T_x существует путь μ, соответствующий максимальному подобразцу $I_m...I_n$, то $r\mu$ будет путем в T_r, соответствующим этому же подобразцу, причем никакой другой путь в T_r не будет соответствовать большему подобразцу. Наконец, по уже изложенным выше соображениям, значение атрибута $\psi(\omega,J)$ будет правильно описывать множество минимальных путей в T_r.

4. Корень r имеет пометку, отличную от $I_1,...,I_n$, и единственного сына x с пометкой $N_m[\omega]$. Поскольку добавление нового узла с такой пометкой никак не влияет на множество интересующих нас путей, а правило (3) приписывает r пометку x, поддерево T_r соответствует утверждению.

5. Корень r имеет пометку, отличную от $I_1,...,I_n$, и сыновей x и y с пометками $N_i[\omega_1]$ и $N_k[\omega_2]$ соответственно. Поскольку множество путей от корня в T_r есть множество путей в T_x и T_y с добавленной в начало вершиной r, то максимальный подобразец, которому соответствует хоть один путь в T_r, есть максимальный из максимальных подобразцов для T_x и T_y. Таким образом, нетерминальная пометка r $N_r = \begin{cases} N_i, i < k \\ N_k, i \geq k \end{cases} = N_{\min\{i,k\}}$, что и приписывается правилом (4). Наконец, рассмотрим значение атрибута $\varphi(\omega_1, \omega_2, N_i, N_k)$. Не умаляя общности, будем считать, что $i \geq k$. Тогда $\varphi(\omega_1, \omega_2, N_i, N_k) = \omega_2 \setminus \{N_i,...,N_n\}$. Но для $t \geq i$ в T_r существуют как минимум два пути, которые соответствуют подобразцу $I_t...I_n$ — как минимум один в T_x (так как x помечена N_i и, по индукционному предположению, в T_x есть хотя бы один путь, соответствующий $I_i...I_n$, а значит и $I_t...I_n$) и как минимум один в T_y (по тем же соображениям). При этом ни один из них не является подпутем другого, так как они имеют лишь одну общую вершину r. Таким образом, в T_r не существует минимальных путей, соответствующих подобразцам $I_t...I_n$ при $t \geq i$. Что же касается минимальных путей, соответствующих подобразцам $I_t...I_n$ для $k \leq t < i$, то их множество в дереве T_r такое же, как и в дереве T_y, поскольку каждый такой путь получается из соответствующего пути $I_t...I_n$ приписыванием вершины r и никакой путь T_x не соответствует ни одному такому образцу. Случай $i < k$ рассматривается аналогично.

Наконец, как было отмечено ранее, каждое применимое правило действительно будет применено алгоритмом переписывания. Утверждение доказано.

Из утверждения 2 следует, что если вершина дерева получила пометку $N_i[\{N_{k_1},..., N_{k_m}\}]$, то $i \leq \min\{k_1,..,k_m\}$. Таким образом, можно сформулировать:

Следствие 2. Дерево σ соответствует образцу $\pi = \{\_*\}I_1\{\_*\}...\{\_*\}I_n\{\_*\}$ тогда и только тогда, когда переписывание, определяемое грамматикой G_2, сопоставляет корню σ' пометку $N_1[\omega]$ и $N_1 \in \omega$.

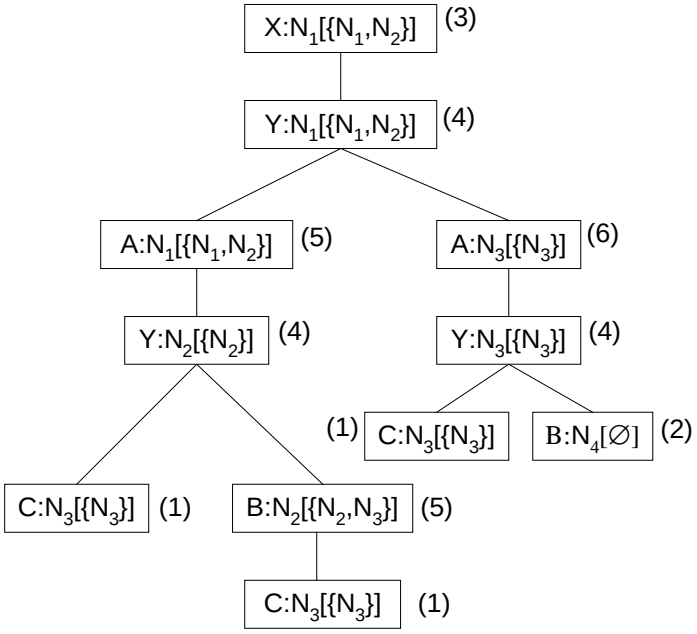

Рис. 5

На рис. 5 показана разметка, определяемая грамматикой G_2 для агрегата ABC. Здесь числа в скобках указывают номера правил, применением которых получен нетерминал. В данном случае корень дерева имеет в качестве элемента атрибута нетерминал N_1, следовательно, сопоставление прошло успешно.

4. Алгоритм идентификации

Предложенное в предыдущем разделе решение задачи идентификации опирается на некоторое предварительное преобразование исходного дерева структуры. Это преобразование может существенно увеличить его размер и внести соответствующие поправки в оценку сложности. Кроме того, привлеченные преобразователи атрибутов содержат теоретико-множественные операции, что также не способствует простоте реализации. Наконец, предложенные грамматики имеют очень большое число правил, что ведет к неэффективности применения алгоритма переписывания напрямую. Здесь мы покажем, как из описанного абстрактного решения получить простую реализацию.

Если в исходном дереве присутствовала вершина $x(y_1,...,y_k)$ $(x \notin \{I_1,...,I_n\} \ \& \ k>2$ или $x \in \{I_1,...,I_n\} \ \& \ k>1)$, то в преобразованном дереве ей соответствует поддерево $x(\varepsilon(\varepsilon(...\varepsilon(y_1,y_2)...),y_k))$. Но тогда единственным правилом G_1, применимым при разметке новых ε-вершин, является правило (5), причем применяться оно будет только снизу вверх. Аналогичное верно и для грамматики G_2 — здесь ε–вершины могут быть размечены только правилом (4) и тоже только снизу вверх.

Определим функцию join: $(N, \Psi) \times (N, \Psi) \to (N, \Psi)$ следующим образом:

$$\text{join}((N_i, \omega_1), (N_k, \omega_2)) = (N_{min\ \{i,k\}}, \varphi(\omega_1, \omega_2, N_i, N_k)).$$

Эта функция по определению описывает пометку вершины, полученную применением правила (4) грамматики G_2. Таким образом, единственный наследник x при применении правил G_2 будет помечен нетерминалом $N_{\min\{i_1,...,i_k\}}$ с атрибутом $\varphi(\omega_1, \varphi(\omega_2,...), N_{i_1}, \min\{i_1,...,i_1\})$, где $N_{i_1},..., N_{i_k}$ — нетерминалы, метящие $y_1,...,y_k$; $\omega_1,..,\omega_k$ — их атрибуты[3]. Но тот же самый эффект дает следующее применение функции join:

---

[3] Здесь мы пользуется ассоциативностью и коммутативностью функции min. Мало того, можно доказать, что функция join в целом ассоциативна и коммутативна (что и следовало ожидать исходя из общих соображений). Для этого достаточно показать, что преобразователь φ обладает следующими свойствами: $\varphi(\omega_1, \omega_2, i, k) = \varphi(\omega_2, \omega_1, k, i)$; $\varphi(\omega_1, \varphi(\omega_2, \omega_3, i, j), k, \min\{i,j\}) = \varphi(\varphi(\omega_1, \omega_2, k, i), \omega_3, \min\{k,i\}, j)$. Преобразователь γ также очевидно коммутативен и ассоциативен.

$$\mathrm{join}(\mathrm{join}(...\mathrm{join}(\mathrm{join}((N_{i_1},\omega_1),(N_{i_2},\omega_2)),(N_{i_3},\omega_3))...),(N_{i_k},\omega_k)) \quad (*)$$

Аналогично, атрибут единственного наследника x при переписывании в соответствии с G_1 будет равен

$$\gamma(\gamma(...\gamma(\gamma(\omega_1,\omega_2),\omega_3))...),\omega_k)\,. \qquad (**)$$

Из этого следует, что нет необходимости в явном виде выполнять преобразование дерева — достаточно просто посчитать (*) или (**) для всех сыновей данной вершины и затем применить нужное правило.

Наконец, очевидно, что нет необходимости также хранить в явном виде все правила грамматик. Достаточно просто разобрать варианты конфигураций текущей вершины и применить единственно подходящую схему правил.

Алгоритм разметки в соответствии с G_1 будет тогда выглядеть так. Поскольку грамматика обладает единственным нетерминалом, который всегда приписывается вершине при разметке, его представление в алгоритме излишне. Легко видеть, что атрибут может быть представлен вектором целых чисел длиной n, обозначим его exacts. Для $1 \le i \le n$ exacts[i] содержит число путей в текущем поддереве, которые точно соответствуют агрегату $I_i...I_n$. Осталось аккуратно разобрать случаи:

```
label₁ (v)
{
  if v.sons=∅ then
    v.exacts[n] :=
    if v.mark = Iₙ
      then [0,...,1]          - по правилу (1)
      else [0,...,0]          - по правилу (2)
  else begin
    for x∈{v.sons(1),...,v.sons(k)} do label₁ (x);

    e := v.sons(1).exacts;
    for x∈{v.sons(2),...,v.sons(k)}  - по правилу (5)
      do e := join₁ (e, x.exacts);

    found := false;

    for k=n,n-1,...,1 do       - попытка применения
      if v.mark = Iₖ then      - правил (3, 4, 6)
      begin
      found := true;           - применяется (6)
        if k=n then e[n] := 1;
        else begin
          e[k] := if e[k+1] = 1 then 1 else 0;
          e[k+1] := 0
        end
      end;

      v.exacts :=
        if found
        then e                 - по правилу (6)
        else [0,...,0]         - по правилам (3, 4)
  end
```

Функция join_1, в свою очередь, выглядит так:

```
join₁ (e, eₖ)
{
    for i=1,2,...,n do e[i] +:= eₖ[i];

    return e;
}
```

Здесь мы использовали обычные целые числа и операцию обычного сложения вместо приведенного в описании G_1 множества $\{1,\infty\}$ со специальной операцией. В пределах возможных размеров задачи такое сведение представляется разумным.

Опишем теперь алгоритм переписывания для G_2. Будем представлять нетерминалы их номерами, а атрибуты (множества нетерминалов) — битовыми шкалами. Тогда

```
type
    label_t = {nonterm:int; attr: [1..n] bit}

label2 (v)
{
  if v.sons=∅ then
    v.label :=
      if v.mark = In
        then {nonterm=n; attr=[0,...,1]}
                                        - по правилу (1)
        else {nonterm=n+1; attr=[0,...,0]}
                                        - по правилу (2)
  else begin
    for x∈{v.sons(1),...,v.sons(k)} do label2 (x);
    l := v.sons(1).label;

    for x∈{v.sons(2),...,v.sons(k)}
                                        - по правилу (4)
      do l := join2 (l, x.label);
      v.label := l;
      for k=n,n-1,...,1 do
                                - по правилам (5,6)
        if v.mark = Ik then
        begin
          if k=n then begin
            v.label.attr[n]:=true;
            v.label.nonterm:=min(v.label.nonterm, n);
          end
          else begin
            if l.attr[k+1]
              then v.label.attr[k]:= true;
            if l.attr[k] and not l.attr[k+1]
              then v.label.attr[k] := false;
            if (l.attr[k+1]&&(l.nonterm > k))
            ||(l.nonterm = k+1)
              then l.nonterm := k
          end
        end;
        v.label.nonterm := l.nonterm
                                - по правилу (3)
     end
}
```

Осталось описать алгоритм вычисления функции $join_2$:

```
join2 (l, lk)
{
    if l.nonterm < lk.nonterm
           then begin
             for i=lk.nonterm to n do l.attr[i] = 0;
               return l
           end
           else begin
             for i=l.nonterm to n do lk.attr[i] = 0;
             return lk
           end
}
```

Теперь задача идентификации может быть решена применением описанных функций. Действительно, если после применения $label_1$ к корню дерева r r.exacts(1)=1, то, в соответствии

со Следствием 1, в дереве есть единственный путь, который точно соответствует $I_1...I_n$. В противном случае можно применить функцию $label_2$. Если после этого r.nonterm=1 и r.attr[1]=1, то, в соответствии со Следствием 2, в дереве существует минимальный путь, соответствующий $I_1...I_n$.

Вычисления функций $label_1$ и $label_2$, очевидно, можно произвести одновременно — они имеют одинаковый поток управления и не зависят друг от друга. Таким образом, возникает функция label, которая строит для дерева разметку, являющуюся объединением разметок, порождаемых G_1 и G_2.

Легко видеть, что применение правил, реализуемое этой функцией для вершины x, требует времени $O(\deg(x)\times n)$, где $\deg(x)$ — степень x. Таким образом, разметка всего дерева σ потребует $O\left(n\times\sum_{x\in v(\sigma)}\deg(x)\right)$, где $v(\sigma)$ — множество вершин σ. Но так как $\sum_{x\in v(\sigma)}\deg(x)=|v(\sigma)|-1$, окончательной оценкой будет $O(v(\sigma)\times n)$. Поскольку идентификация требует применения этого алгоритма к каждому элементу контекста, можно сделать вывод: данная реализация обеспечивает идентификацию агрегата $I_1I_2....I_n$ в контексте $\{\sigma_1,..,\sigma_k\}$ за время $O\left(n\times\sum_i v(\sigma_i)\right)$.

5. Применение алгоритма идентификации для видозависимого анализа

Описанный алгоритм позволяет осуществить проверку однозначности идентификации, т. е. ответить на вопрос, соответствует ли данному агрегату единственный элемент контекста. Однако этого недостаточно для целей идентификации как таковой — необходимо дополнительно определить тип элемента контекста, который соответствует данному агрегату, и его положение внутри структуры. Эти задачи могут быть тривиально решены простой модификацией сформулированного выше подхода.

Действительно, удачное сопоставление дерева агрегату означает наличие минимального (или точного) пути от корня дерева, который соответствует агрегату. Последовательность пометок вершин этого пути и есть полная квалификация выборки, а последняя его вершина определяет тип этой выборки. Например, можно считать, что вершины дерева дополнительно нагружены информацией об их типе. Тогда тип выборки — это тип последней вершины (если она лист) или тип ее поддерева (в противном случае). Поскольку конечная вершина найденного пути целиком его определяет, достаточно вычислить только ее. Предлагаемая ниже модификация позволяет это сделать.

Утверждение 3. Пусть вершина x точно соответствует подобразцу $I_k...I_n$, пометка x, определяемая G_1, есть $K[\omega]$, $N_k\in\omega$. Тогда атрибут ω вычислен по правилам (1), (5) или (6), примененным к вершинам, составляющим единственный точно соответствующий $I_k...I_n$ путь от x. Доказательство аналогично доказательству Утверждения 1.

Обозначим через $x{:}T$ вершину x, помеченную терминалом T. Рассмотрим следующую грамматику[4]:

$G'_1 = (N,T,R,\Psi)$
$T = M\cup\{\varepsilon\}$,
$N = \{K\}$,
$\Psi = 2^{\{N_1,...,N_n\}\times V\times\{1,2,...\}}$,
$R = \{$

$K[\{(N_n,x,1)\}]\to x{:}J,\ J=I_n,$ (1)

$K[\varnothing]\to x,\ x\neq I_n,$ (2)

$K[\varnothing]\to x(K[\omega]),\ x\notin \{I_1,I_2,...,I_n\},$ (3)

$K[\varnothing]\to x(K[\omega_1],K[\omega_2]),\ x\notin \{I_1,I_2,...,I_n\},\ x\neq\varepsilon$ (4)

[4] Строго говоря, грамматика G'_1 не удовлетворяет предложенной формализации, поскольку значения преобразователей атрибутов зависят от самих вершин дерева (а не только от их терминальных пометок). Однако это несоответствие несущественно в том смысле, что легкая модификация формализма исправляет этот недостаток и оставляет все результаты на месте.

$$K[\gamma(\omega_{1,}\ \omega_1)]\rightarrow\varepsilon(K[\omega_1],\ K[\omega_2]), \quad (5)$$

$$K[\delta(\omega;\ x;\ J)]\rightarrow x{:}J(K[\omega]),\ J\in\{I_1,...,I_n\} \quad (6)$$

},

где

V — множество вершин дерева,

$$\gamma(\omega_1,\omega_2) =$$
$$\{(N_i,x,k+m):\exists(N_i,x,k)\in\omega_1\ \&\ \exists(N_i,y,m)\in\omega_2\}\cup$$
$$\{(N_i,x,k):\exists(N_i,x,k)\in\omega_1\ \&\ \forall m\forall y(N_i,y,m)\notin\omega_2\}\cup$$
$$\{(N_i,x,m):\forall k\forall y(N_i,y,k)\notin\omega_1\ \&\ \exists(N_i,x,m)\in\omega_2\},$$

$$\delta(\omega;x;J)=\{(N_{k-1},y,m):(N_k,y,m)\in\omega\ \&\ J=I_{k-1}\}\cup\{(N_n,x,l):J=I_n\}.$$

Утверждение 4. Вершина x при разметке в соответствии с G'_1 получает пометку $K[\omega]$, $(N_k,y,I)\in\omega$, тогда и только тогда, когда поддерево T_x точно соответствует $I_k...I_n$, а y есть конечная вершина единственного точно соответствующего пути от x. Доказательство — по индукции с применением предыдущего утверждения.

Неформально говоря, правила грамматики G'_1 «протаскивают» последнюю вершину подозрительного на соответствие пути через остальные вершины дерева, если их добавление в путь не меняет его свойство соответствия подобразцу. Аналогичным образом может быть модифицирована грамматика G_2. Ее выписывание и модификация окончательных алгоритмов оставляется читателю.

При реализации могут быть предприняты шаги по дополнительному отсечению ненужных применений правил или всей процедуры идентификации в целом. Например, заведомо не имеет смысла сопоставлять с образцом деревья, высота которых меньше длины образца или среди пометок вершин которых нет какого-либо его символа. Кроме того, можно модифицировать атрибутную грамматику таким образом, чтобы разметка некоторого поддерева прекращалась при определенных условиях, — например, если вершина, помеченная символом I_k, лежит выше k-го уровня дерева.

6. Сравнение с существующей реализацией

Описанный алгоритм был сравнен с реализацией идентификации структур в одном из наиболее авторитетных компиляторов — IBM Visual Age PL/I v.2.0. Необходимо отметить, что данный компилятор изначально накладывает некоторые ограничения на входной язык — так, он не допускает описания одноименных подструктур или полей на одном и том же уровне (согласно [1] это не запрещено). Ниже приведена таблица, содержащая тесты и результаты, полученные при попытках идентификации с помощью Visual Age PL/I и предложенного здесь алгоритма.

№	Структура	Агрегат	Visual Age PL/I	Данный алгоритм
1	DCL 1 A, 2 B, 3 X, 4 A, 5 B, 6 X FIXED DEC, 4 B, 5 C FIXED DEC;	ABC	Ok	Ok
2	DCL 1 A, 2 B, 3 C, 4 A, 5 B, 6 X FIXED DEC, 4 B, 5 C FIXED DEC;	ABC	Ok	Ok

3	DCL 1 A, 2 B, 3 C, 4 B, 5 C FIXED DEC;	ABC	Ok	Ok
4	DCL 1 A, 2 B, 3 X, 4 A, 5 B, 6 C FIXED DEC, 4 B, 5 C FIXED DEC;	ABC	Fail	Fail
5	DCL 1 A, 2 X, 3 B FIXED DEC, 2 Y, 3 B, 4 C FIXED DEC, 2 B, 3 C, 4 D FIXED DEC;	ABC	Ok	Ok
6	DCL 1 A, 2 X, 3 B FIXED DEC, 2 Y, 3 B, 4 C FIXED DEC, 2 Z, 3 B, 4 C, 5 D FIXED DEC;	ABC	Fail	Fail
7	DCL 1 X, 2 A, 3 Y FIXED DEC, 3 B, 4 Z FIXED DEC, 4 C, 5 C, 6 B, 7 C FIXED DEC, 5 B, 6 C FIXED DEC, 6 B, 7 C FIXED DEC;	ABC	Fail	Ok
8	DCL 1 A, 2 B, 3 C, 4 D FIXED DEC, 2 C, 3 D FIXED DEC;	AD	Fail	Fail

Таким образом, алгоритм, реализованный в Visual Age PL/I, совместим с описанным снизу вверх — принимаемые им случаи принимаются и нашим, отвергаемые нашим — отвергаются Visual Age PL/I.

Указатель литературы

1. OS and DOS PL/1 Language Reference Manual. 2nd ed., Sept. 1994, IBM Corp., 322 p.
2. **Береснева А.В., Терехов А.А.** Анализ языка PL/I в системе RescueWare // Наст. сб. С. 268–277.
3. **Aho A.V., Ganapathi M., Tjiang S.W.K.** Code generation using tree matching and dynamic programming // ACM Transactions on Programming Languages and Systems. October 1989. Vol. 11. N 4. P. 491–516.
4. **Pelegrí-Llopart E., Graham S.L.** Optimal code generation for expression trees: an application of BURS Theory // Proceedings of the Conference on Principles of Programming Languages. 1988. P. 294–308.
5. **Proebsting T.A.** BURS automata generation // ACM Transactions on Programming Languages and Systems. May 1995. Vol. 17. N 3. P. 461–486.