5.2 Ações Semânticas para a Construção de Tabelas de Símbolos

Esta seção apresenta esquemas de tradução que reconhecem declarações de variáveis e geram tabelas de símbolos. Inicialmente, é apresentado um esquema que gera tabelas de símbolos para programas monolíticos, isto é, formados por um único bloco. Posteriormente, esse esquema é estendido para permitir a geração de tabelas para programas bloco-estruturados com procedimentos aninhados.

EXEMPLO 5.5 Esquema de tradução para gerar tabela de símbolos para bloco unitário.

O esquema de tradução a seguir constrói a tabela de símbolos para uma lista de declarações.

P	→	M D	
M	→	ε	{ desloc = 0 }
D	→	D ; D	
D	→	id : T	{ adSimb (id.nome, T.tipo, desloc); desloc = desloc + T.tam }
T	→	int	{ T.tipo = int; T.tam = 4 }
T	→	real	{ T.tipo = real; T.tam = 8 }
T	→	array [num] of T1	{ T.tipo = matriz (num.val, T1.tipo); T.tam = num.val * T1.tam }
T	→	^ T1	{ T.tipo = ponteiro (T1.tipo); T.tam = 4 }

O não–teminal M é empilhado logo no início do processo para atribuir o valor zero à variável *desloc*. Essa variável contém o próximo endereço disponível na área de dados do procedimento. Observe que *desloc* não é atributo de símbolo da gramática. A rotina *adSimb* adiciona um novo identificador à Tabela de Símbolos, com seu tipo e endereço. O atributo *tam* contém o tamanho em bytes para os diferentes tipos de dados aceitos nas declarações.

Observe que a redução M←ε poderia ser realizada em qualquer momento do processo de construção da tabela, pois sempre se tem a palavra vazia no topo da pilha. Contudo, ao programar o analisador, deve-se ter em mente que essa redução só deve ser efetuada no início da análise. Esse artifício de introduzir na gramática do esquema de tradução um não terminal artificial (caso de M) para forçar uma ação especial é uma técnica comumente usada.

Scanned with CamScanner

A Figura 5.6 mostra a árvore de derivação e as ações semânticas para a declaração **a: int ; b: real**.

P	→	M D	
M	→	ε	{ desloc = 0 }
D	→	D ; D	
D	→	id : T	{ adSimb (id.nome, T.tipo, desloc); desloc = desloc + T.tam }
T	→	int	{ T.tipo = int; T.tam = 4 }
T	→	real	{ T.tipo = real; T.tam = 8 }
T	→	array [num] of T1	{ T.tipo = matriz (num.val, T1.tipo); T.tam = num.val * T1.tam }
T	→	^ T1	{ T.tipo = ponteiro (T1.tipo); T.tam = 4 }

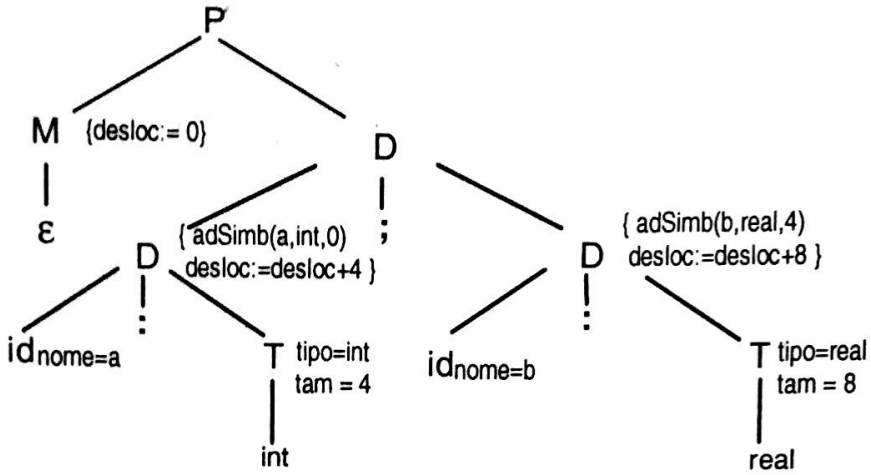

Figura 5.6 Árvore de derivação para a declaração **a: int; b: real** .

O esquema de tradução a seguir estende o esquema anterior, incluindo uma regra de produção para a declaração de procedimentos. Isso permitirá a geração de tabelas de símbolos para procedimentos embutidos.

Scanned with CamScanner