1. COMPILADORES – CONCEITOS BÁSICOS

4. TRADUÇÃO DIRIGIDA POR SINTAXE

- Uma definição dirigida por sintaxe é uma generalização de uma gramática livre do contexto.
- Cada símbolo da gramática tem um conjunto de atributos associados, particionados em dois subconjuntos denominados **ATRIBUTOS SINTETIZADOS E HERDADOS**.
- Um atributo pode representar uma palavra, um número, um tipo ou uma posição na memória, por exemplo.
- O valor de um atributo na árvore de derivação é definido por uma regra semântica associada à produção utilizada naquele nodo.
- O valor de um atributo sintetizado é computado a partir dos atributos dos filhos daquele nodo.
- Um atributo herdado é computado a partir dos atributos dos irmãos ou dos pais daquele nodo.
- Regras semânticas estabelecem dependências entre atributos que são representadas por um grafo. A partir do *grafo de dependências*, é derivada uma ordem de avaliação para as regras semânticas.
- Uma árvore de derivação que apresenta os valores dos atributos de cada nodo é denominada árvore de derivação "decorada".

4.1. Formato de uma Definição Dirigida por Sintaxe

- Em uma definição dirigida por sintaxe, cada produção da gramática $A \rightarrow \alpha$ tem associada um conjunto de regras semânticas no formato $b := f(c_1, c_2,\ldots,c_k)$, onde f é uma função e:

1. b é um atributo sintetizado de A e $c_1, c_2,\ldots,c_k$ são atributos pertencentes aos símbolos da gramática da produção, ou

2. *b* é um atributo herdado de um dos símbolos da gramática do lado direito da produção, e $c_1, c_2,...,c_k$ são atributos pertencentes aos símbolos da gramática da produção.

- Uma gramática de atributos é uma definição dirigida por sintaxe na qual as funções nas regras semânticas não têm efeitos colaterais.

- Assume-se que terminais tenham apenas atributos sintetizados, assim como o símbolo inicial da gramática. Um exemplo de definição dirigida por sintaxe é apresentado na Tabela 4.1:

Tabela 4.1. Definição dirigida por sintaxe de uma calculadora simples.

PRODUÇÃO	**REGRAS SEMÂNTICAS**
L → E n	printf("%i", E.val);
E → E_1 + T	E.val = E_1.val + T.val;
E → T	E.val = T.val;
T → T_1 * F	T.val = T_1.val * F.val;
T → F	T.val = F.val;
F → (E)	F.val = E.val;
F → dígito	F.val = dígito.lexval;

- token *dígito* tem um atributo sintetizado *lexval* cujo valor é fornecido pelo analisador léxico.

4.2. Atributos Sintetizados

- Uma definição que utiliza apenas atributos sintetizados é denominada de *definição-S de atributos*.
- Uma árvore de sintaxe para uma definição deste tipo pode ser *anotada* através da avaliação das regras semânticas para os atributos em cada nodo, de forma *bottom-up*, das folhas para a raiz.

- A Figura 4.1 apresenta um exemplo de árvore de derivação anotada para o reconhecimento e cálculo da sentença de entrada *3*5+4n*.

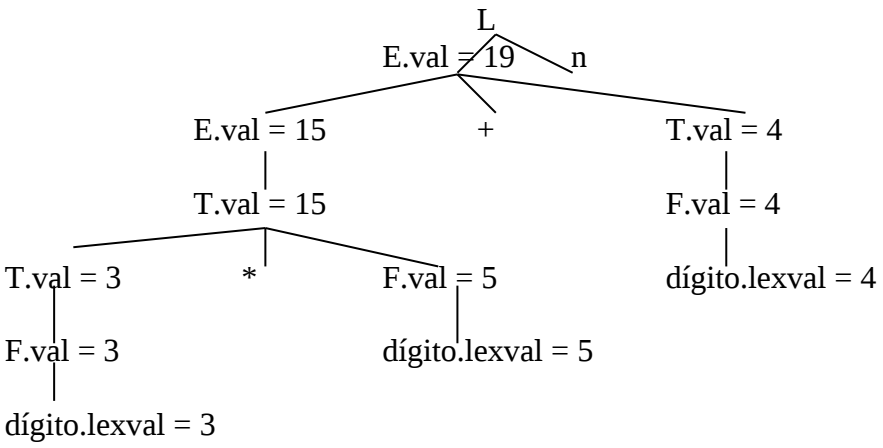

Figura 4.1. Árvore de derivação anotada para o reconhecimento da sentença 3*5+4n.

4.3. Atributos Herdados

- Um atributo herdado é definido a partir dos atributos dos nodos irmãos ou pais.
- Adequado para expressar a dependência de uma estrutura da linguagem de programação no contexto em que ela aparece.
- Um exemplo de definição dirigida por sintaxe utilizando atributos herdados é apresentada na Tabela 4.2:

Tabela 4.2. Definição dirigida por sintaxe com o atributo herdado *L.in*.

PRODUÇÃO	REGRAS SEMÂNTICAS
D → T L	L.in = T.type;
T → int	T.type = integer;

T → real	T.type = real;
L → L_1,id	L_1.in = L.in; addType(id.entry,L.in);
L → id	addType(id.entry,L.in);

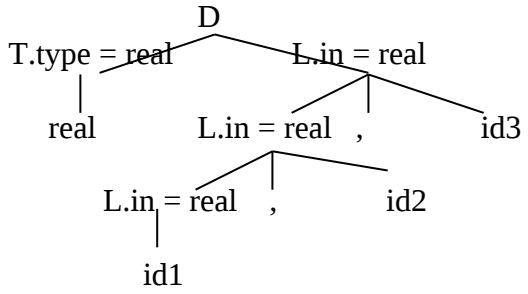

Figura 4.2. Árvore de derivação com o atributo herdado *in* em cada nodo *L*.

4.4. Grafo de Dependências

- Se um atributo *b* em um nodo da árvore de derivação depende do atributo *c*, então a regra semântica para *b* naquele nodo deve ser avaliada depois da regra semântica que define *c*.
- As interdependências entre os atributos herdados e sintetizados nos nodos da árvore de derivação pode ser apresentados por um grafo direcionado denominado *grafo de dependências*.
- Antes de construir um grafo de dependência para uma árvore de sintaxe, deve-se expressar cada regra semântica no formato $b = f(c_1, c_2,..., c_k)$, introduzindo um atributo sintetizado *b* para cada regra semântica, que consiste em uma chamada de procedimento.
- O grafo tem um nodo para cada atributo e um arco do nodo *b* para o nodo *c*, se o atributo *b* depende do atributo *c*. O seguinte algoritmo pode ser seguido para a construção do grafo de dependências:

Para cada nodo n da árvore de derivação faça
Para cada atributo a do símbolo da gramática em n faça
construa um nodo no grafo de dependências para a;

Para cada nodo n da árvore de derivação faça
Para cada regra semântica b = f(c_1, c_2,..., c_k) associada a uma produção usada em n faça
Para i = 1 até k faça construa um arco do nodo c_i para o nodo b.

- Por exemplo, considerando que a regra semântica *A.a = f(X.x,Y.y)* esteja associada à produção *A* → XY, de forma que o atributo sintetizado A.a dependa dos atributos X.x e Y.y. Se esta produção é usada em uma árvore de derivação, haverá três nodos A.a, X.x e Y.y no grafo de dependências com um arco de X.x para A.a e de Y.y para A.a.

- Se a produção A → XY tivesse uma regra semântica X.i = g(A.a,Y.y) associada, haveria um arco de A.a para X.i e de Y.y para X.i. A seguir é apresentado um exemplo de grafo de dependência para a produção E → E_1 + E_2, que tem a seguinte regra semântica associada: E.val = E_1.val + E_2.val.

- grafo de dependência resultante tem três nodos, representando os atributos sintetizados E.val, E_1.val e E_2.val. O arco de E.val para E_1.val mostra que E.val depende de E_1.val. As linhas pontilhadas representam a árvore de derivaão e não fazem parte do grafo de dependências.

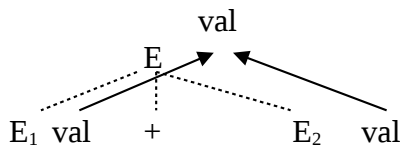

Figura 4.3. E.val é sintetizado a partir de E_1.val e E_2.val.

 EXERCÍCIO: construir o grafo de dependência para a árvore de derivação apresentada na Figura 4.2.

4.5. Ordem de Avaliação

- Uma ordenação topológica ou um grafo acíclico direcionado é qualquer ordenação m_1, m_2, ...,m_k de nodos do grafo, tal que os arcos que saem de nodos anteriores (numeração

menor), na ordenação, chegam em nodos posteriores; ou seja, se $m_i \rightarrow m_j$ é um arco de m_i para m_j, então m_i aparece antes de m_j na ordenação.

- Qualquer ordenação topológica de um grafo de dependências fornece uma ordem válida na qual as regras semânticas associadas aos nodos na árvore de derivação podem ser avaliados. Ou seja, na ordem topológica, os atributos aparecem após aos atributos de que eles dependem.

- Inicialmente a árvore de sintaxe é construída (pelo analisador sintático). Depois o grafo de dependências. A partir da ordenação topológica, obtém-se a ordem de avaliação para as regras semânticas. Por exemplo, considerando a Figura 4.4, verifica-se que que a ordenação topológica é obtida escrevendo-se os números dos nodos em ordem crescente. Desta ordenação, obtém-se o seguinte programa, onde a_n é o atributo associado com o nodo numerado n, no grafo de dependências:

a_4 = *real;*
$a_5 = a_4$;
addType(id_3.*entry*,a_5);
$a_7 = a_5$;
addType(id_2.*entry*,a_7);
$a_9 = a_7$;
addType(id_1.*entry*,a_9);

- A avaliação destas regras semânticas armazena o tipo *real* na entrada da tabela de símbolos para cada identificador.

- Os seguintes métodos foram propostos para a avaliação das regras semânticas:

1. BASEADOS NA ANÁLISE SINTÁTICA: em tempo de compilação, estes métodos obtêm uma ordem de avaliação a partir da ordem topológica do grafo de dependências. Estes métodos

falham na obtenção da ordem da avaliação somente se o grafo considerado possui um ciclo.

2. BASEADOS EM REGRAS: em tempo de construção do compilador, as regras semânticas associadas com as produções são analisadas, manualmente, ou por uma ferramenta específica. Para cada produção, a ordem na qual os atributos associados são avaliados é então determinada.

3. ÓBVIOS: uma ordem de avaliação é escolhida sem considerar as regras semânticas. Por exemplo, se a tradução é feita durante a análise sintática, a ordem de avaliação é forçada pelo método sintático, independentemente das regras semânticas. Uma avaliação óbvia restringe a classe de definições baseadas na sintaxe que podem ser implementadas.

4.6. Construção de Árvores de Sintaxe

- Uma árvore de sintaxe é uma forma condensada da árvore de derivação, na qual somente os operandos aparecem nas folhas, enquanto que os operadores aparecem em nodos interiores da árvore. Cadeias simples de produções, tais como A → B e B → C, por exemplo.

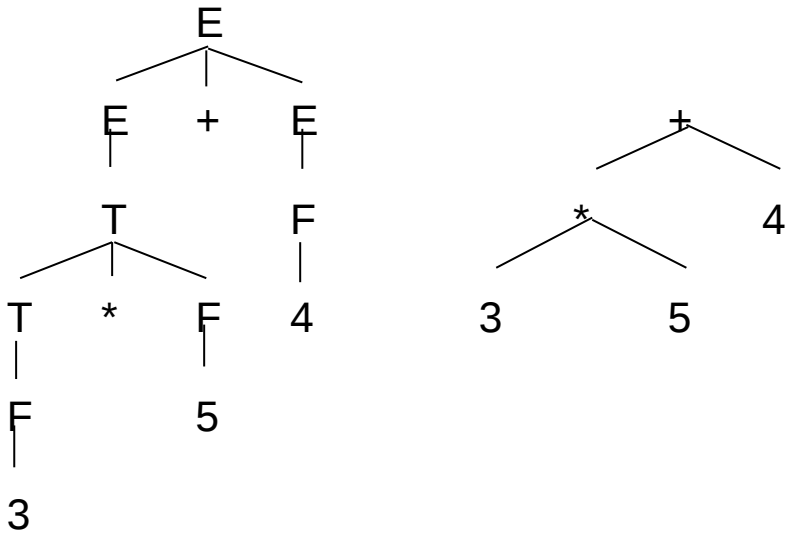

Figura 4.5. Árvores de derivação e de sintaxe para a sentença *3*5+4*.

- seguinte esquema de tradução pode ser utilizado para a construção de árvores de sintaxe para expressões aritméticas simplificadas (Tabela 4.3):

Tabela 4.3. Esquema de tradução para construção de árvores de sintaxe para expressões aritméticas simplificadas.

PRODUÇÃO	**REGRAS SEMÂNTICAS**
E → E1 + T	E.ptr = geranodo("+", E1.ptr,T.ptr);
E → E1- T	E.ptr = geranodo("-", E1.ptr,T.ptr);
E → T	E.ptr = T.ptr;
T → (E)	T.ptr = E.ptr;
T → id	T.ptr = gerafolha(id,id.índice);
T → num	T.ptr = gerafolha(num,num.val);

- O atributo sintetizado *ptr* conserva os ponteiros retornados pelas chamadas das funções. As funções que geram os nodos de operadores binários e folhas de operandos são as seguintes:

- *geranodo(op,esq,dir)*: cria um nodo de operador com rótulo op e dois campos contendo ponteiros para *esq* e *dir*.
- *gerafolha(id,índice)*: cria um nodo de identificador com rótulo *id* e o campo *índice* contendo um ponteiro para a entrada do identificador na tabela de símbolos.
- *gerafolha(num,val)*: cria um nodo de número com rótulo *num* e campo *val* contendo o valor do número.

EXERCÍCIO: Construa uma árvore de derivação anotada, mostrando a construção de uma árvore de sintaxe, para a expressão *a – 4 + c*. Faça a construção no sentido *bottom-up*.

4.7. Avaliação Bottom-up de Esquemas de Tradução S-atribuídos

Atributos sintetizados podem ser avaliados por um analisador redutivo a medida em que a cadeia de entrada é reconhecida. Os valores dos atributos sintetizados são conservados, asociados aos símbolos da gramática, em sua própria pilha. Sempre que ocorre uma redução, são computados os valores dos novos atributos sintetizados, a partir dos atributos que estão na pilha, associados aos símbolos do lado direito da produção que está sendo reduzida.

São usados campos adicionais na pilha do analisador para armazenar os valores dos atributos sintetizados. Por exemplo, considerando o seguinte esquema de tradução:

$A \rightarrow X\ Y\ Z$ $\qquad\qquad$ $A.a = f(X.x,Y.y,Z.z);$

tem-se o seguinte conteúdo na pilha (Tabela 4.4):

Tabela 4.4. Pilha para análise sintática de gramática com esquema de tradução.

PILHA	
ANÁLISE	**ATRIBUTOS**
Z	Z.z
Y	Y.y
X	X.x
...	...

Antes que XYZ seja reduzido para A, os valores dos atributos Z.z, Y.y e X.x estão armazenados na pilha, na coluna

Atributos, nas posições [topo], [topo-1] e [topo-2], respectivamente. Se um símbolo não tem atributos, então a entrada correspondente na pilha é vazia. Após a redução, topo é decrementado de 2, A é armazenado em *Análise[topo]* e o valor do atributo sintetizado *A.a* é colocado em *Atributos[topo]*.

A gramática definida na Tabela 4.1 pode ter seus atributos sintetizados avaliados por um analisador empilha-reduz durante uma análise *bottom-up*. Para isso, deve-se modificar o analisador para executar as instruções abaixo ao fazer as reduções apropriadas. O código foi obtido a partir das regras semânticas substituindo-se cada atributo por uma posição no vetor *Atrib*.

L → E =	printf("%i",Atrib[topo]);
E → E1 + T	Atrib[ntopo] = Atrib[topo-2] + Atrib[topo];
E → T	
T → T1 * F	Atrib[ntopo] = Atrib[topo-2] * Atrib[topo];
T → F	
F → (E)	Atrib[ntopo] = Atrib[topo-1];
F → dígito	Atrib[topo] = dígito.lexval;

Quando o analisador reconhece um dígito, o token *dígito* é empilhado em *Análise[topo]* e seu atributo é armazenado em *Atrib[topo]*. O código não mostra como as variáveis *topo* e *ntopo* são operadas. Quando uma produção com *n* símbolos do lado direito é reduzida, o valor de *ntopo* é atualizado para *topo-n+1*. Após a execução de cada instrução, topo é atualizado para *ntopo*. A tabela abaixo mostra a seqüência de movimentos realizados com a sentença *3*5+4=*.

Tabela 4.5. Seqüência de movimentos realizados com a sentença *3*5+4*.

ENTRADA	ANÁLISE	ATRIBUTOS	PRODUÇÃO
3*5+4=	-	-	
*5+4=	3	3	
*5+4=	F	3	F → dígito
*5+4=	T	3	T → F
5+4=	T*	3 -	
+4=	T*5	3 – 5	
+4=	T*F	3 – 5	F → dígito
+4=	T	15	T → T*F
+4=	E	15	E → T
4=	E+	15 -	
=	E+4	15 – 4	
=	E+T	15 – 4	F → dígito
=	E+T	15 – 4	T → F
=	E	19	E → E + T
	E=	19	
	L	19	L → E=

Na implementação, as instruções são executadas sempre antes de uma redução. Os esquemas de tradução considerados na próxima seção provêm uma notação para intercalar ações com anáise sintática.

4.8. Esquemas de Tradução L-atribuídos

Quando a tradução ocorre durante a análise, a ordem de avaliação de atributos está amarrada à ordem pela qual os nodos da árvore de derivação são criados pelo método de análise. Uma ordem natural, que caracteriza vários métodos de tradução *top-down* e *bottom-up* é chamada *depth-first order*, definida pelo seguinte algoritmo:

DEPTH_FIRST(n: nodo);
Para cada filho m de n, da esquerda para a direita,
    avaliar os atributos herdados de m;
    DEPTH_FIRST(m);
Avaliar os atributos sintetizados de n.

Os atributos dos esquemas L-atribuídos devem ser avaliados sempre na ordem *depth-first*. Este tipo de esquema baseia-se nas gramáticas LL(1). Um esquema é L-atribuído se cada atributo herdado de Xj, 1<j<n, no lado direito de A → X1 X2 ... Xn, depende somente:

a) dos atributos dos símbolos X1, X2, ..., Xj-1 à esquerda de Xj na produção e
b) dos atributos herados de A.

Todo esquema S-atribuído é também L-atribuído, há que as restrições *a* e *b* aplicam-se somente a atributos herdados. O esquema apresentado abaixo não é L-atribuído, já que o atributo herdado Q.i depende do atributo R.s, cujo não-terminal associado R está a direita de Q.

```
A → LM          { L.i = I(A.i);
                  M.i = m(L.s);
                  A.s = f(M.s);
A → QR            R.i = r(A.i);
                  Q.i = q(R.s);
                  A.s = f(Q.s);
```

O exemplo apresentado a seguir é de um esquema de tradução que mapeia expressões infixadas em pós-fixadas:

E → TR
R → op T {printf(“%c”,op.val); } R
| ε
T → num {printf(“%i”,num.val); }

A Figura 4.6 apresenta a árvore de derivação para a expressão *9 – 5 + 2*, com as ações semânticas agregadas aos nodos apropriados. Ações semânticas são tratadas como símbolos terminais. Quando executadas na ordem *depth-first*, imprimem a expressão *9 5 – 2* +.

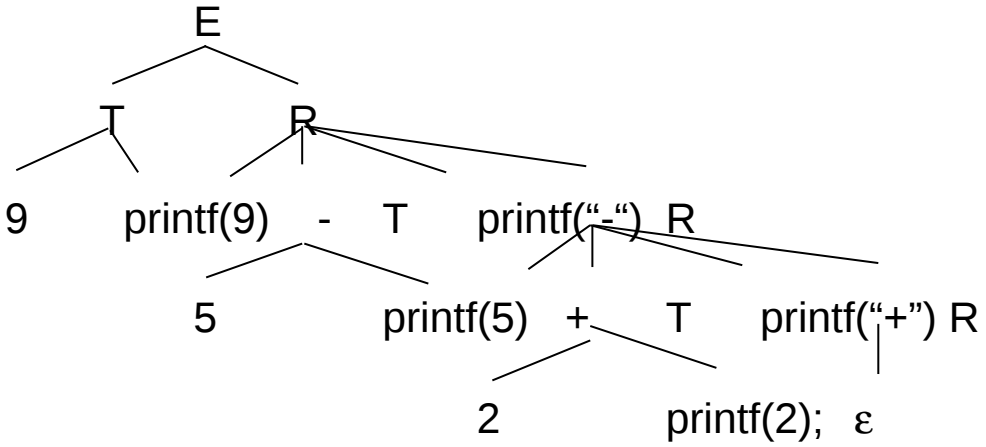

Figura 4.6. Árvore de derivação para a expressão *9 – 5 + 2*.

Quando um esquema de tradução é definido, deve-se assegurar que os valores dos atributos estejam disponíveis sempre que ações se refiram a eles. No caso mais simples, quando são usados apenas atributos sintetizados, pode-se construir o esquema de tradução colocando-se as ações que computam os valores dos atributos no fim do lado direito da produção associada. Se existem ambos, atributos herdados e sintetizados, deve-se proceder da seguinte forma:

1. Um atributo herdado para um símbolo no lado direito de uma produção deve ser computado em uma ação antes deste símbolo.

2. Uma ação não deve se referir a um atributo sintetizado de um símbolo à direita desta ação.
3. Um atributo sintetizado associado ao não-terminal do lado esquerdo da produção deve ser computado somente depois que todos os atributos que ele referencia tenham sido computados. A ação que computa tais atributos pode, em geral, ser especificada no final da produção.

O seguinte esquema de tradução não satisfaz o primeiro dos requisitos acima:

S → A1 A2	{ A1.h = 1; A2.h = 2; }
A → a	{printf(“%c”,A.h); }

O atributo herdado A.h na segunda produção ainda não está definido quando uma tentativa de imprimir seu valor é feita durante uma travessia *depth-first* da árvore de derivação para a entrada *aa*. A travessia inicia em S e visita as sub-árvores A1 e A2 antes que os valores A1.h e A2.h sejam atribuídos.