Compiladores

Sintático

José Luis Seixas Junior

Índice

- Análise Sintática;
- Análise Sintática Descendente;
- Análise Sintática Ascendente;

Análise Sintática

- Segundo bloco componente dos compiladores;
- Caracteriza-se como o mais importante na maioria dos compiladores:
  - Compiladores orientados à sintaxe;
  - Controle de organização das atividades;

Análise Sintática

- Principal função do analisador:
  - Promover a análise da sequência com que os átomos componentes do texto-fonte se apresentam;
  - Base gramatical da linguagem-fonte com sequenciamento da síntese da árvore da sintaxe;
  - Composição gramatical;

Análise Sintática

- Outras funções:
  - Identificação das sentenças;
  - Detecção de erros de sintaxe;
  - Montagem da árvore abstrata da sentença;

Análise Sintática

- Adicionais:
  - Recuperação e correção de erros;
  - Comando da ativação e do modo de operação do léxico;
    - Para compiladores orientados à sintaxe;
  - Ativação (orientação) de rotinas de análise semântica e de síntese do código objeto;

Análise Sintática

- Existem diferentes métodos de análise sintática;
- Variam de acordo com:
  - Eficiência;
  - Simplicidade de implementação;
  - Aplicabilidade às linguagens de programação;

Análise Sintática

- Comunmente usados:
  - Ascendente: Construção da árvore sintática começa pelas folhas;
  - Descendente: Construção da árvore se inicia pela raiz;
    - Com retrocesso: Tentativa de derivação pelos primeiros comandos da gramática;
    - Sem retrocesso: Derivação com a análise futura da estrutura gramatical a ser utilizada;

Descendente

- Nos algoritmos de análise sintática descendente, a construção da árvore de derivação começa pela sua raiz e procede na direção das folhas. Quando todas as folhas têm rótulos que são terminais, a fronteira da árvore deve coincidir com a cadeia dada.

Descendente

- Este método, apesar de potencialmente muito ineficiente, foi usado em algumas implementações pioneiras de sistemas automáticos para gerar analisadores sintáticos e ainda é usado para algumas aplicações especiais. O funcionamento básico do método pode ser descrito da seguinte maneira:

Descendente

- Passo 1:
  - Adota-se a cadeia dada como o valor inicial de α, e uma folha cujo rótulo é o símbolo inicial S da gramática, como o valor da árvore D. Esta folha é adotada também como a folha corrente.

Descendente

- Passo 2:
  - Seja X o rótulo da folha corrente. Se X não é um terminal, então escolhe-se uma produção da forma $X ::= X_1X_2...X_n$ e substitui-se, na árvore D, a folha corrente por uma árvore cuja raiz tem rótulo X e cujos descendentes diretos são folhas $X_1,X_2,...X_n$.
  - A folha de rótulo X_1 torna-se a nova folha corrente, e o passo 2 é repetido.

Descendente

- Passo 2:
  - Se X é um símbolo terminal, e α = Xβ para alguma cadeia β, então adota-se β como o novo valor de α; a folha que segue a folha corrente em D, quando D é percorrida da esquerda para a direita, é adotada como a nova folha corrente, e o passo 2 é repetido.

Descendente

- Passo 2:
  - Se X é um terminal, e o primeiro símbolo de α não é X (ou α = λ), então deve-se retroceder (isto é, restaurar os valores de α, D e da folha corrente) à última configuração em que foi feita a escolha de uma produção, adotando-se uma outra alternativa para o não-terminal da folha corrente.
  - Caso não haja mais alternativas, repete-se o retrocesso até que seja encontrada, e volta-se a repetir o passo 2.

Descendente

- Passo 2:
  - Finalmente, se o algoritmo avança além da última folha de D, mas $\alpha \neq \lambda$, então deve-se retroceder como no caso anterior.

Descendente

- Passo 3:
  - A análise termina quando o algoritmo avança além da última folha de D e α = λ. Neste caso, a cadeia dada é uma sentença da linguagem.

Descendente

- Passo 3:
  - Se o algoritmo é forçado a retroceder à configuração inicial do passo 1 depois de esgotar todas as alternativas para o símbolo inicial da gramática, então a cadeia dada não pertence à linguagem.

Descendente

- Considere a gramática:

  E::= T+E / T

  T::= F*T / F

  F::= a / b / (E)

- Sentença: a*b

Análise Sintática Ascendente

- Nos algoritmos de análise sintática ascendente, a construção da árvore de derivação começa pelas folhas da árvore e procede na direção da raiz;
  - Bottom-up.

Ascendente

- Caso a árvore obtida cuja raiz é o símbolo inicial da gramática, tendo as folhas a cadeia dada, então a cadeia pertence a linguagem:
  - Sentença aceita (sem erros);
  - Obtida a árvore de derivação;
- Caso contrário:
  - Sentença não é aceita;

Ascendente

- Passo 1:
  - Adota-se como valor inicial uma cadeia;

Ascendente

- Passo 2:
  - Decomponha a cadeia de forma que $\alpha = \beta X_1 X_2 \ldots X_n \gamma$ e exista uma produção $X ::= X_1 X_2 \ldots X_n$.
  - Caso isto seja possível, adota-se a nova cadeia $\alpha = \beta X \gamma$, associando-se com esta ocorrência do não terminal X uma árvore cuja raiz tem rótulo X e cujas subárvores diretas são as árvores que estavam associadas com as ocorrências de $X_1 X_2 \ldots X_n$;

Ascendente

- Passo 3:
  - Passo 2 repetido até que α seja o símbolo inicial da gramática.

Ascendente

- Problemas:
  - Identificação da parte da cadeia que deve ser analisada;
  - Identificação da produção que deve ser usada na redução;

Ascendente

- Frase:
  - Seja G uma gramática, e suponhamos que $S =>^* \alpha A \gamma$ e $A =>^+ \beta$. Dizemos então que β é uma frase da forma sentencial αβγ para o não-terminal A (é fácil ver que $S =>^+ \alpha\beta\gamma$). Em termos da árvore de derivação para a forma sentencial αβγ, β é a fronteira de uma subárvore cuja raiz tem o rótulo A e que não é uma folha.
- Frase Simples:
  - Caso se tenha $A => \beta$ (derivação direta), então β é chamada de frase simples.

Precedência Simples

- A parte que deve ser reduzida é a parte $Y_k \ldots Y_m$ que está mais à esquerda com $Y_{k-1} <. Y_k = Y_{k+1} = \ldots = Y_m .> Y_{m+1}$
- Caso k = 1 ou m=n, então Y_{k-1} ou Y_{m+1} não existem.

Precedência Simples

- As relações <. , = e .> são disjuntas, isto é, não existe nenhum par de símbolos que pertence a mais de uma destas relações;
- G não possui duas produções da forma A::=γ e B::=γ
- Toda gramática de Precedência Simples é não ambígua.

Precedência Simples

- Dizemos que X <. Y se existe uma forma sentencial direita $\alpha=\beta XY\gamma w$ tal que $Y\gamma$ é um redutendo de α.
- Dizemos que X = Y se existe uma forma sentencial direita $\alpha=\beta\gamma XY\delta w$ tal que $\gamma XY\delta$ é um redutendo de α.
- Dizemos que X .> Y se existe uma forma sentencial direita $\alpha=\beta\gamma XYw$ tal que γX é um redutendo de α.

Precedência Simples

- Gramática:

  S ::= aSb / A
  A ::= BC / c
  B ::= (
  C ::= A)

	S	A	B	C	a	b	c	(	)
S						=			
A						·>			=
B		<·	<·	=			<·	<·	
C						·>			·>
a	=	<·	<·		<·		<·	<·	
b						·>			
c						·>			·>
(							·>	·>	
)						·>			·>

Precedência Simples

Passo	**Forma Sentencial**	**Redutendo**	**Redução p/**
1	a<·a<·(·>c) b b	(	B
2	a<·a<·B<·c·>) b b	c	A
3	a<·a<·B<·A=)·>b b	A)	C
4	a<·a<·B=C·>b b	BC	A
5	a<·a<·A·>b b	A	S
6	a<·a=·S=b·>b	aSb	S
7	a=S=b	aSb	S
8	S		

Precedência de Operadores

Precedência	Operador	Comentários	Associatividade
1	++ -- !	Operadores Unários	R
2	* / %	Multi., Divi., Resto	L
3	+ -	Add, Sub.	L
4	<< >>	Shift	L
5	< > <= >=	Relacionais	L
6	== !=	Igualdade	L
7	&	Bit/Loginal AND	L
8	^	XOR (ou exclusivo)	L
9	|	Bit/Logical OR	L
10	&&	AND	L
11	||	OR	L
12	?:	Condicional	R
13	= op=	Atribuição	R

R = Direita L = Esquerda