Análise Sintática Descendente Preditiva (Tabular)

O processo de análise em questão é também denominado LL(k) da esquerda para a direita ("Left to Right") através da derivação mais à esquerda ("Leftmost Derivation") com *lookahead* de k *tokens*. Ou seja, o texto é lido da esquerda para a direita e usa a derivação mais a esquerda, lendo k *tokens* de *lookahead*, para construção da árvore sintática de forma descendente.

O modelo de análise usa um autômato de pilha baseado em uma tabela de decisão construída para a linguagem através de uma gramática da linguagem.

Este analisador usa o algoritmo abaixo:

(**x** representa topo da pilha; **a** *token* da cadeia de entrada)
entrada: sentença a ser analisada
terminada com "$" (terminador)
saída: indicação de erro ou de aceitação

Início:
acaba = false
insere $ na pilha
insere S na pilha (não terminal)
enquanto não acaba faça
Se x = a =$ então
aceita a sentença; acaba ⇐ true
Senão
Se x = a ≠ $ então
desempilha
avança cursor (pegar o próximo token)
Senão
Se x ≠ a e x é não terminal então
regra = tab [x,a]; ($x \rightarrow x_1,x_2,x_3,x_n...$)
desempilha
empilha lado direito ; ($x_n...x_2,x_1$)
Senão
erro ("terminal não esperado")
Fim se
Fim se
Fim se
Fim enquanto
Fim

Exemplo: Gramática para expressões lógicas
$E \rightarrow E \vee T \mid T$
$T \rightarrow T \& F \mid F$
$F \rightarrow \sim F \mid id$
Não terminais E, F, T (inicial E)
$E \rightarrow TE'$

E’ →vTE’ | ε
T → FT’
T’ → &FT’ | ε
F → ~F | id

	terminais				
Não terminais	id	v	&	~	$
E	E → TE'			E → TE’	
E'		E' →vTE'			E' = ε
T	T → FT’			T → FT’	
T'		T’→ε	T’→& FT’		T’→ε
F	F →id			F → ~F	

Fases	Pilha	Cadeia	Regra
1	E$	id v id & id $	E →TE’
2	TE'$	id v id & id $	T → FT’
3	FT'E'$	id v id & id $	F → id
4	idT'E'$	id v id & id $	Avança
5	T'E'$	V id & id $	T’ → ε
6	E'$	V id & id $	E’ → vTE’
7	vTE'$	V id & id $	Avança
8	TE'$	id & id $	T → FT’
9	FT’E'$	id & id $	F → id
10	idT'E’$	id & id $	Avança
11	T'E'$	& id $	T’ → & FT’
12	&FT'E'$	& id $	Avança
13	FT'E'$	id $	F → id
14	idT'E$	id $	Avança
15	T'E$	$	T’ → ε
16	E'$	$	E’ → ε
17	$	$	**Aceita**

Construção da Tabela

A construção da tabela é baseada nos conjuntos **First** e **Follow** para cada não terminal. Os conjuntos First indicam quais são os terminais que cada não terminal pode gerar. Enquanto que os conjuntos Follow indicam quais são os terminais que podem aparecer após o reconhecimento de cada não-terminal.

First(A) é definido a partir do lado direito das regras de A:

A → aα

- Se a é terminal então $\{a\} \subseteq$ First (A);
- Se a é não terminal $\Rightarrow$ First (a) $\subseteq$ First (A);
- Se aα = ε então ε $\in$ First (A);

Exemplo
$F \rightarrow \sim F \mid id$
First(F) = {~, id}
$T \rightarrow FT' \Rightarrow$ First (T) = First (F) = {~, id}

Follow (A) é definido a partir das regras nas quais A está no lado direito.

1. se A é o não-terminal inicial então $\$ \in$ Follow (A);
2. se $X \rightarrow \alpha A \beta \Rightarrow$ First $(\beta) \subseteq$ Follow (A);
3. se $X \rightarrow \alpha A$ ou $X \rightarrow \alpha A \beta$ e $\varepsilon \in$ First $(\beta) \Rightarrow$ Follow (X) $\subseteq$ Follow (A);

Exemplo:

$E \rightarrow TE'$
$E' \rightarrow \vee TE' \mid \varepsilon$
$\Rightarrow$ First(E') = {$\vee$, ε}
Follow (T) = {$, $\vee$}

Algoritmo para Construção

Entrada: Gramática Simplificada
Saída: Tabela Construída
Início
Para todo $A \in N$ faça
Calcule First (A)
Para todo $a \in$ First(A)
Tab[A, a] = regra que gerou a
Fim para
Se $\varepsilon \in$ First(a) então
Cálculo do Follow (A)
Para cada $a \in$ Follow(A) Faça
Tab [A, a] = $A \rightarrow \varepsilon$
Fim para
Fim se
Fim para
Fim

Cálculo de First e Follow dos não-terminais:

First (E) = First (T) = First (F)= {~, id};
First (E') = {$\vee$, ε} $\Rightarrow$ Follow (E') = {$} (regra 3);
First (T') = {&, ε}$\Rightarrow$ Follow (T') = {$\vee$, $} (regra 3);

Follow (E) = {$} (regra 1);
Follow (T) = {v, $}

	Id	v	&	~	$
E	E → TE'			E → TE'	
E'		E' → vTE'			E' = ε
T	T → FT'			T → FT'	
T'		T'→ε	T'→& FT'		T'→ε
F	F →id			F →~F	

Esta tabela é montada para o reconhecimento descendente supondo-se que a linguagem é LL(1).

$A \rightarrow \alpha_1 \mid \alpha_2$

- First $(\alpha_1) \cap$ First $(\alpha_2) = \varnothing$

Para ser LL(1)

- Se First (α_1) inclui ε então $\varepsilon \notin$ First (α_2)
- Somente uma das regras pode incluir ε
- Se First (A) inclui ε então (suponha que First (α_1)inclui ε)
- Follow $(\alpha_1) \cap$ First $(\alpha_2) = \varnothing$
- Follow (A) $\cap$ First (A) $= \varnothing$

Exemplo: Gramática para **IF**

S → if C then S S' | a
S' → else S | ε
C → b

A gramática é ambígua, observe a cadeia abaixo:

if b then if b then a else a

Há duas árvores de derivação possíveis a partir da primeira regra porque S' será substituído por ε em algum dos ramos da árvore, construa as árvores!