



Eliminação de Retrocessos e da Recursão Esquerda

A maneira mais simples de evitar retrocessos é fazer com que o algoritmo sempre tome a decisão correta quanto à produção a ser aplicada. Uma classe muito simples de gramáticas para as quais isto pode ser feito é obtida impondo-se as restrições [KOWALTOWSKI, 83]:

1. Toda produção é da forma $A ::= X\alpha$, onde X é um terminal e α pertence ao V^*
2. Se $A ::= X_1\alpha_1 / X_2\alpha_2 / \dots X_n\alpha_n$ são todas as alternativas para o não-terminal A , então os terminais X_i são todos distintos entre si.



Eliminação de Retrocessos e da Recursão Esquerda

É suficiente fazer uma pequena modificação no algoritmo anterior. Fazemos com que a escolha da produção seja baseada no primeiro símbolo da cadeia α . Obviamente só há no máximo uma alternativa que deverá ser escolhida. Caso não haja nenhuma, ou seja, α não começa com algum dos símbolos X_i que correspondem ao não-terminal A da folha corrente, então a cadeia não é uma sentença.



Eliminação de Retrocessos e da Recursão Esquerda

Exemplo: Considere a gramática:

$E ::= a/b/+EE/*EE$ (a qual satisfaz as restrições)

Na figura da página 54 do KOWALTOWSKI (83) encontram-se os passos da análise da sentença $+a*ba$. Foram omitidos os passos correspondentes à comparação do primeiro terminal de cada alternativa com o primeiro símbolo da cadeia corrente, uma vez que esta comparação já foi feita para escolher a própria alternativa.

Cabe ressaltar que as restrições impostas acima são muito severas para serem viáveis na prática.



LL(1)

Definição: Seja X um símbolo de uma gramática

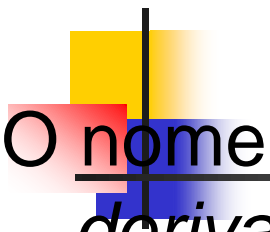
$$G=(V_N, V_T, P, S)$$

$$\psi(X)=\{Y \in V_T, / X \rightarrow Y\alpha, \alpha \in V^*\}$$

se G tem produções da forma:

$A \rightarrow x_1\alpha_1 / x_2\alpha_2 / \dots x_n\alpha_n$ $X_i \in V$ e $\psi(X)$ são disjuntos
dois a dois, então G é LL(1). Obs: X_i podem ser
terminais ou não-terminais.

LL(1)



O nome LL(1) - *Left-to-right parsing producing Leftmost derivation* - vem do fato de se poder analisar uma cadeia da esquerda para a direita, produzindo uma derivação esquerda verificando apenas um símbolo da cadeia de entrada para decidir qual é a produção a ser aplicada. A definição pode ser generalizada para LL(K), $k \geq 0$. Para tais gramáticas, podem-se obter analisadores descendentes sem retrocesso, se a escolha da produção a ser aplicada for baseada nos k primeiros símbolos (se existirem) da cadeia corrente* (KOWALTOWSKI, 83).

* Em geral, podem ser incluídas nessa classe de gramáticas com produções da forma $A \rightarrow \lambda$

LL(1)



Exemplo:
 $S \rightarrow AS / BA$
 $A \rightarrow aB / C$
 $B \rightarrow bA / d$
 $C \rightarrow c$

	ψ_P	ψ_{P^*}	ψ
S	A,B	S,A,B,a, C,b,d,c	a,b,d,c
A	a,C	A,a,C,c	a,c
B	b,d	B,b,d	b,d
C	c	C,c	c

$S \rightarrow AS / BA$

$A \rightarrow aB / C$

$B \rightarrow bA / d$

$\psi(A) = \{a, c\}$

$\psi(a) = \{a\}$

$\psi(b) = \{b\}$

$\psi(B) = \{b, d\}$

$\psi(C) = \{c\}$

$\psi(d) = \{d\}$

$\psi(A) \cap \psi(B) = \emptyset$

$\psi(a) \cap \psi(C) = \emptyset$

$\psi(b) \cap \psi(d) = \emptyset$

Portanto, G é LL(1)



LL(1)

Na figura da pag.55 do KOWALTOWSKI (83) podemos ver os passos da análise para a sentença abcdad.

Propriedade Importante: Toda gramática que pertence à classe LL(1) é não ambígua (KOWALTOWSKI,83).

- Nenhuma gramática ambígua ou recursiva à esquerda pode ser LL(1).

Métodos LL de análise sintática detectam erros tão cedo quanto possível. Possuem a *propriedade do prefixo viável*, significando que detectam que um erro ocorreu tão logo tenham examinado um prefixo da entrada que não seja o de qualquer cadeia da linguagem (AHO, SETHI & ULLMAN, 86).



LL(K)

Exemplo de gramática que não é LL(1):

$S ::= aAB / aBA$

$A ::= b / cS$

$B ::= d / eS.$ Nesta G:

- a primeira produção faz com que não seja do tipo LL(1);
- é fácil verificar, que uma cadeia derivada de A começa sempre com os símbolos b ou c , e uma cadeia derivada de B , com os símbolos d ou e . Com isso, a escolha das produções para o não-terminal S pode se basear no segundo símbolo da cadeia corrente.