Compiladores

Gramáticas

José Luis Seixas Junior

Índice

- Eliminação de Recursão Esquerda;
- Analisadores Sintáticos LL(*k*);
- Analisadores Sintáticos LR(*k*);
- Ambiguidade;

Eliminação de Recursão Esquerda

- Para evitar retrocessos ou recursões infinitas.
  - Tomar a decisão correta de construção da árvore.
- A gramática deve conter restrições;

Eliminação de Recursão Esquerda

- Toda produção da forma $A ::= X\alpha$, onde X é um terminal e α pertence ao V^*;
- Se $A ::= X_1\alpha_1$ / $X_2\alpha_2$ / ... $X_n\alpha_n$ são todas as alternativas para o não-terminal A, então os terminais X_i são todos distintos entre si.
- Ressalta-se que as restrições são severas para serem viáveis na prática.

Eliminação de Recursão Esquerda

- Assim a escolha da produção seja baseada no primeiro símbolo da cadeia α.
- Obviamente, assim, há no máximo uma alternativa que deverá ser escolhida para qualquer X_i a ser lido;
- Caso não haja nenhuma produção que se inicie com o terminal corrente X da folha corrente A, então a cadeia não é uma sentença pertencente à linguagem.

Analisadores Sintáticos LL(*k*)

- O nome LL(1) vem de *Left-to-right, parsing producting 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.

Analisadores Sintáticos LL(k)

- A definição pode ser generalizada para LL(k), $k >= 0$.
- Para tais gramáticas, podemos ter analisadores descendentes sem retrocesso, se a escolha da produção a ser aplicada for baseada nos k primeiros símbolos (se existirem) da cadeia corrente.

Analisadores Sintáticos LL(k)

- $A \rightarrow X_1\alpha_1 \,/\, X_2\alpha_2 \,/\, \ldots\, X_n\alpha_n$
- $X_i \in V$ e $\psi(X)$ são disjuntos dois a dois.
  - $\psi(X) = \{Y \in V_T \,/\, X \rightarrow Y\alpha,\ \alpha \in V^*\}$
- Então G é LL(1).
- Obs: X_i podem ser terminais ou não-terminais.

Analisadores Sintáticos LL(*k*)

- Propriedade Importante:
  - Toda gramática que pertence à classe LL(1) é não ambígua;
  - Assim, nenhuma gramática ambígua ou recursiva à esquerda pode ser LL(1).
- Métodos LL de análise sintática detectam erros tão cedo quanto possível.

Analisadores Sintáticos LL(*k*)

- Possuem a propriedade do *prefixo viável,* significando que detectam que um erro ocorreu tão logo tenham examinando um prefixo da entrada que não seja o de qualquer cadeia da linguagem.

Exemplo

- S ::= aAB / aBA

  A ::= b / cS

  B ::= d / eS

- Esta gramática NÃO é LL(1). Pois a primeira produção não é suficiente para determinar a produção de derivação.

Exemplo

- S ::= aAB / aBA

  A ::= b / cS

  B ::= d / eS

- Toda cadeia que parta da subcadeia iniciando em A começa sempre com os símbolos *b* ou *c* e a partir de B começam com *d* ou *e*.

Exemplo

- S ::= aAB / aBA

  A ::= b / cS

  B ::= d / eS
- Portanto, a escolha das produções para o não-terminal S pode se basear no segundo símbolo da cadeia corrente.

Analisadores Sintáticos LL(*k*)

- Eliminação de Recursão Esquerda:
- E ::= T + E / T
  T ::= F * T / F
  F ::= a / b / (E)
- A partir do não-terminal T podemos derivar cadeias terminais arbitrariamente compridas.

Analisadores Sintáticos LL(*k*)

- Consequentemente, nenhum número *k* finito de símbolos iniciais da cadeia corrente será suficiente para decidir, em geral, qual das duas alternativas para o não-terminal E.
- Analogamente para T.

Analisadores Sintáticos LL(*k*)

- Para resolver o problema, adicionamos produções com lados direito nulos.

- E::= T E'

  E'::= + E / λ

  T::= F T'

  T'::= * T / λ

  F::= a / b / (E)

Analisadores Sintáticos LL(*k*)

- Problema:
  - Aumento no comprimento das derivações, que será refletido num número maior de operações para realizar a análise.

Analisadores Sintáticos LL(k)

- Suponha que algumas alternativas para o não-terminal A têm a forma $A ::= \beta\gamma_1 \,/\, \beta\gamma_2 \,/\, ... \,/\, \beta\gamma_n$ com $\beta \neq \lambda$.
- Pode-se "fatorar" estas produções escrevendo $A ::= \beta\ (\gamma_1 \,/\, \gamma_2 \,/\, ... \,/\, \gamma_n)$. Caso $\gamma_i = \lambda$ para algum i, coloca-se esta alternativa em último lugar, isto é, $\gamma_n = \lambda$.
- Assim, $E ::= T + E \,/\, T$ pode ser reescrita como $E ::= T\ (\ + E \,/\, \lambda)$.

Analisadores Sintáticos LL(*k*)

- Considere a gramática:
- E ::= T + E / T - E / T

  T ::= F * T / F \ T / F

  F ::= a / b / (E)
- Que pode ser reescrita como:

  E ::= T (+ E / - E / λ)

  T ::= F (* T / \ T / λ)

  F ::= a / b / (E)

Analisadores Sintáticos LR(*k*)

- Pilha de pilhas:
  - Empilha uma pilha com relação <.;
  - Empilha um símbolo com relação =;
  - Desempilha e reduz com .>;
- Para analises LR(0) e LR(1):
  - Necessita de alocação de estados que interpretam reduções ou deslocamentos;

Analisadores Sintáticos LR(*k*)

Passo	**Forma Sentencial**	**Redutendo**	**Redu-ção 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		

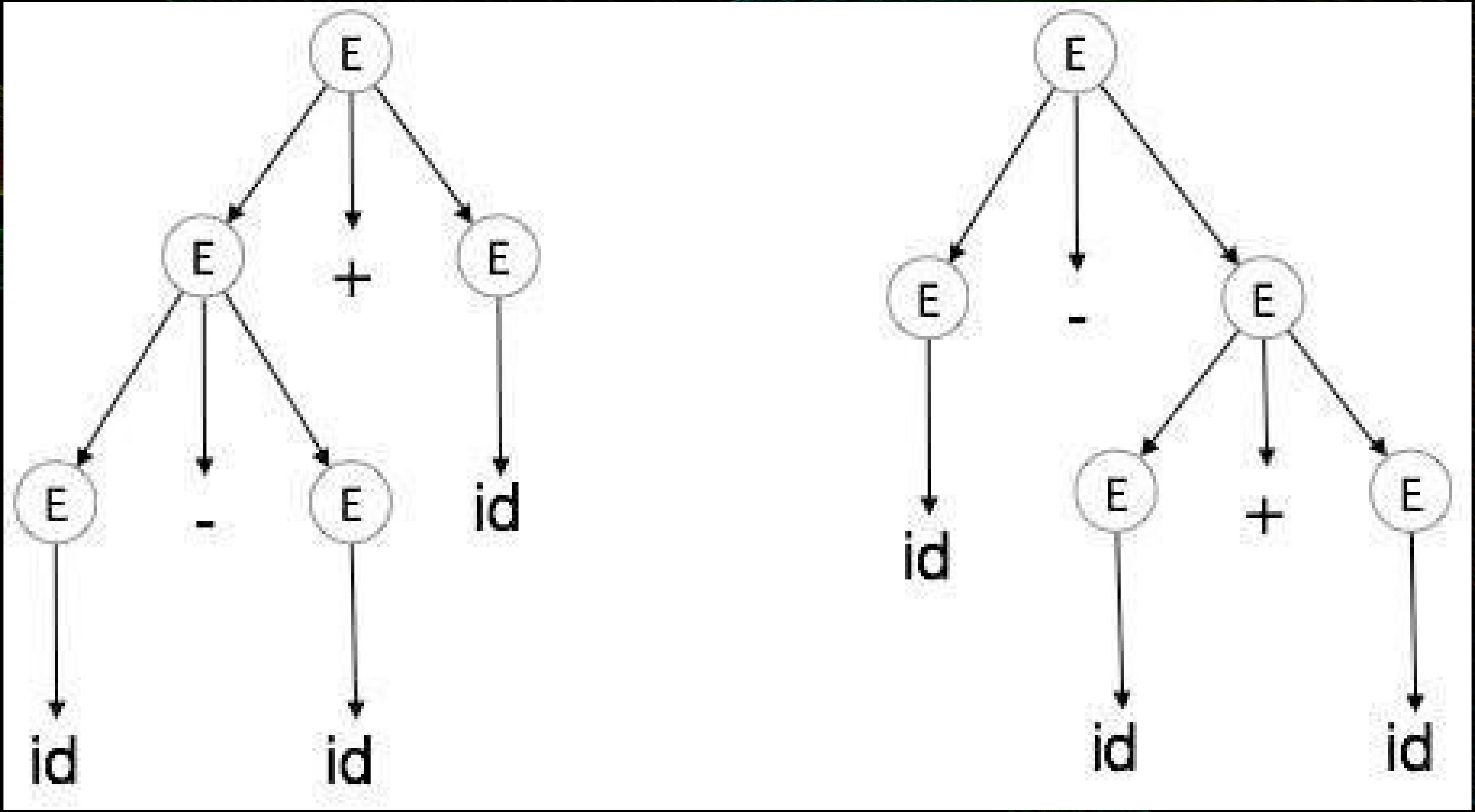
Ambiguidade

- Uma gramática é dita ambigua, caso exista uma sentença α, que pode ser aceita com duas formas de derivação diferentes;
- Assim, se existem duas formas de derivação é possível construir duas árvores de derivação diferentes para a mesma sentença.

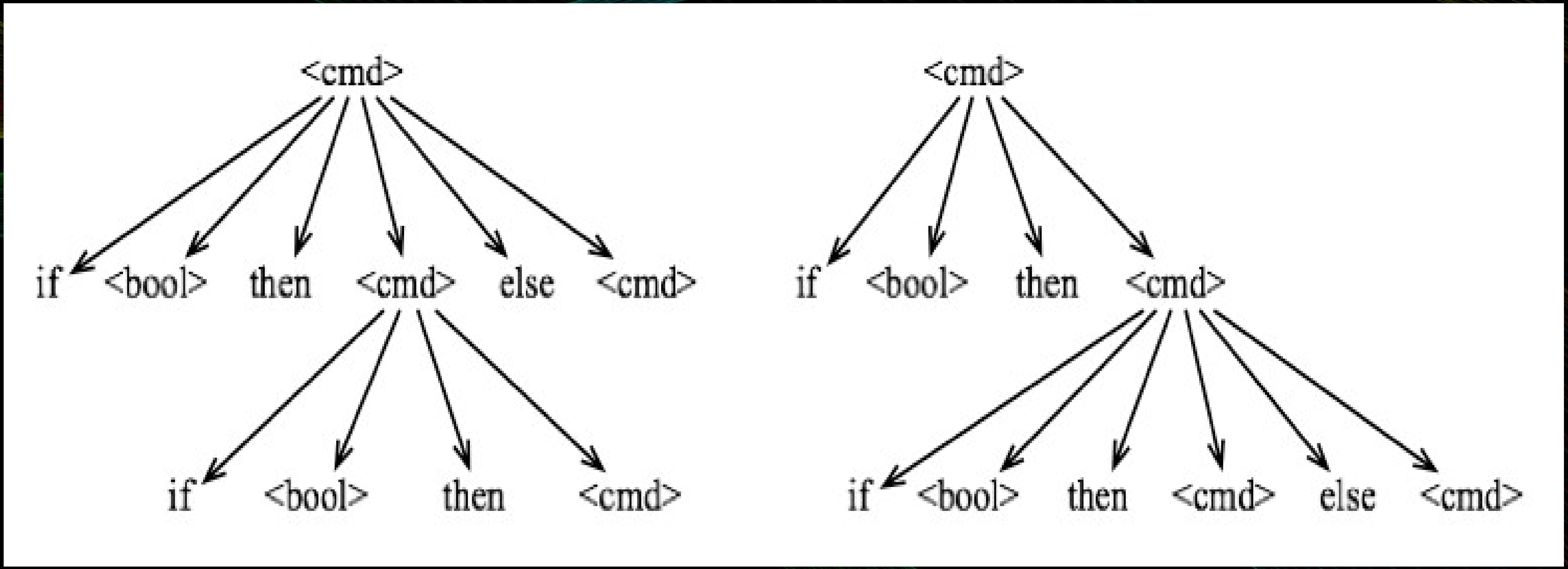
Ambiguidade

- Gramática:
  - E ::= a / b / E+E / E*E / (E)
- Sentença:
  - (a+b)*a
- Derivações:
  - $E \rightarrow E{*}E \rightarrow (E){*}E \rightarrow (E+E){*}E \rightarrow (a+E){*}E \rightarrow (a+b){*}E \rightarrow (a+b){*}a$
  - $E \rightarrow E{*}E \rightarrow E{*}a \rightarrow (E){*}a \rightarrow (E+E){*}a \rightarrow (E+b){*}a \rightarrow (a+b){*}a$

Ambiguidade



Ambiguidade



Ambiguidade

- Remoção:
  - Eliminação de Recursão à Esquerda;
    - Garante uma gramática não ambígua.
  - Criação de não terminais para funcionalidades diferentes;

Ambiguidade

- Nem sempre é possível eliminar uma ambiguidade:
  - Uma linguagem é dita inerentemente ambígua se qualquer gramática que a gere for ambígua;
- É muito difícil garantir que uma gramática é não ambígua à priori;
- Para ser ambígua basta uma única sentença que cause duas árvores;

Ambiguidade

- Considere a gramática:
  - E ::= E+T / E-T / +T / -T / T
  - T ::= T*F / T\F / F
  - F ::= a / b / (E) / E
- Sentença:
  - -a+b*a-a