Compiladores

Prof. Marc Antonio Vieira de Queiroz

Ciência da Computação - UNIFIL
LAB 7
marc.queiroz@unifil.br

05/06/2013

Roteiro I

Análise Sintática Descendente I

Constrói a [arvore de derivação para a cadeia de entrada de cima para baixo, ou seja, da raiz para as folhas, criando os nós da árvore em pré-ordem (busca em profundidade). Neste processo, a análise descendente pode ser vista como o método que produz uma derivação mais à esquerda.

duas formas principais

análise sintática descendente recursiva que pode exigir o retrocesso para encontrar a produção correta

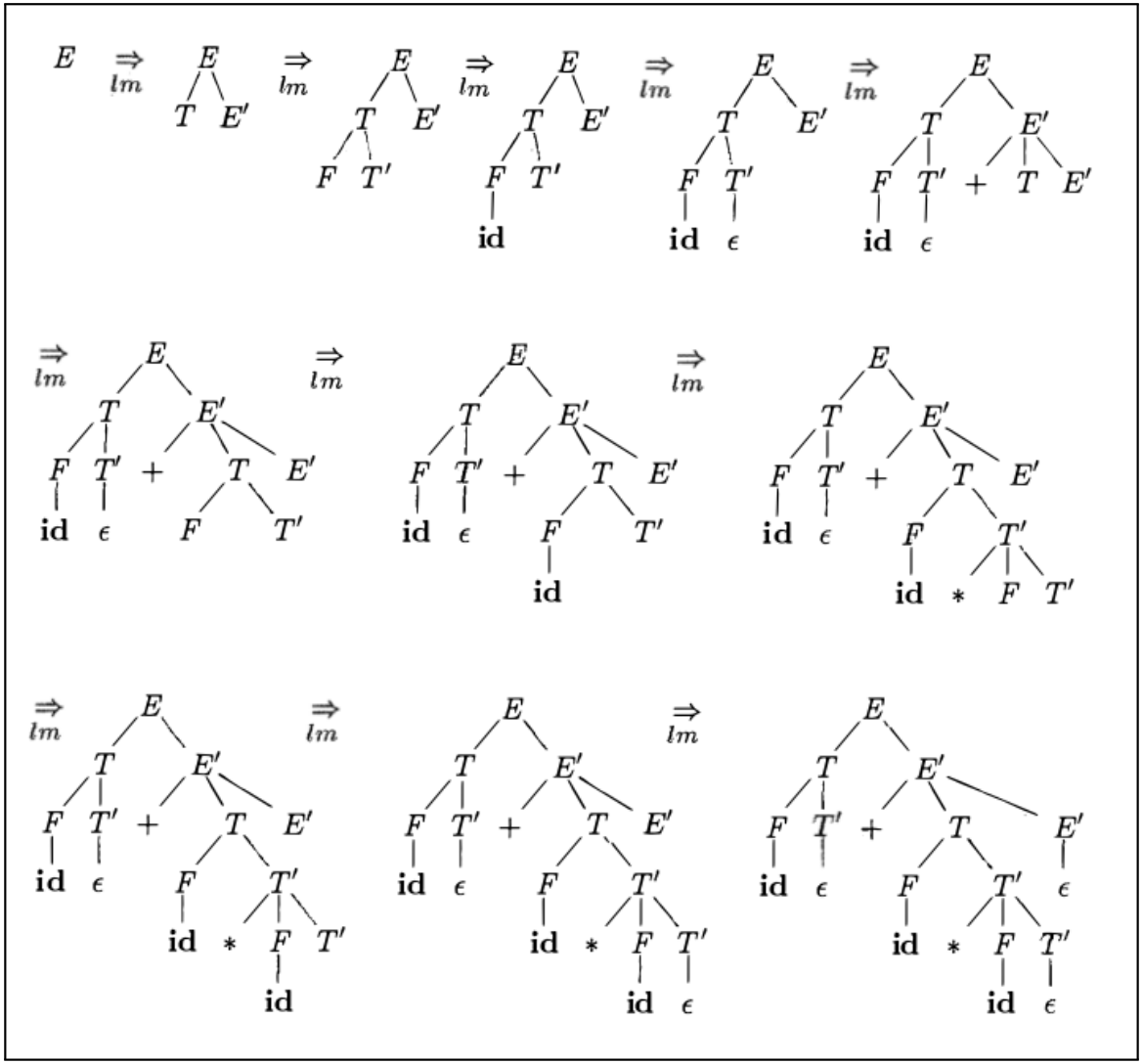
método de reconhecimento sintático preditivo, um caso especial da análise descendente recursiva em que nenhum retrocesso é necessário

Exemplo de gramática I

$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathbf{id}
\end{array}
\tag{4.28}
$$

Figura: 1

árvore de derivação id + id *id



Análise Sintática de descida recursiva I

consiste em um conjunto de procedimentos, um para cada não-terminal da gramática. A execução começa com a ativação do procedimento referente ao símbolo inicial da gramática, que pára e anuncia sucesso se o seu corpo conseguir expandir toda a cadeia de entrada.

pode exigir retrocesso; ou seja; pode demandar voltar atr[as no reconhecimento, fazendo repetidas leituras sobre a entrada

0.void A()

1. Escolha uma produção-A, $A \Rightarrow X_1 X_2 ... X_n$
2. for (i=1 até n)
3. if (X_i é um não-terminal)
4. ativa procedimento $X_i()$;
5. else if (X_i é igual ao símbolo de entrada a)
6. avança a entrada para o próximo símbolo

Análise Sintática de descida recursiva II

7. else /* tratamento de erro */
8.
9.

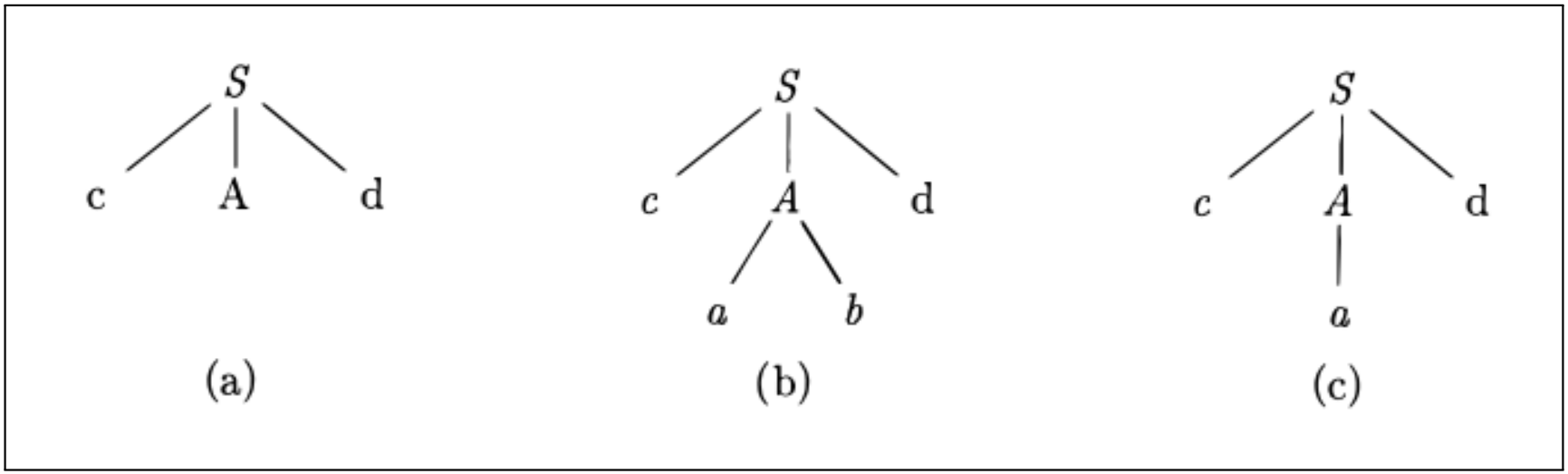
Considere a gramática

$$\begin{array}{lcl} S & \rightarrow & c\ A\ d \\ A & \rightarrow & a\ b \mid a \end{array}$$

determine os passos de uma análise sintática descendente para a cadeia de entrada w=cad usando retrocesso.

para permitir retrocesso, não podemos escolher uma única produção-A na linha(1) do algorítmo base; devemos tentar cada uma das várias alternativas da produção-A em alguma ordem.

um erro na linha(7) não é uma falha definitiva, apenas sugere que precisamos retornar a linha 01 e tentar outra produção. Só quando não houver mais produções a serem tentadas é que declara-se um erro na entrada.


S
c
A
d
(a)
S
c
A
d
a
b
(b)
S
c
A
d
a
(c)

Exercícios

Passos com retrocesso para:
a) id+id
b) (id*id)+id

$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathbf{id}
\end{array}
\tag{4.28}
$$

First e Follow I

A construção de analisadores descentes e ascendentes é auxiliada por duas funções, First e Follow, associadas a uma gramática G.

Durante a análise descendente, elas permitem escolher qual a produção aplicar, com base no próximo símbolo de entrada.

a) FIRST(α)

Define-se FIRST(α), se α é uma foram sentencial (sequência de símbolos da gramática), então FIRST(α) é o conjunto de terminais que iniciam formas sentenciais derivadas a partir de α. Se $\alpha \stackrel{*}{\Rightarrow} \epsilon$, então a palavra vazia também faz parte do conunto.

b) FOLLOW(A)

A função FOLLOW é definida para símbolos não-terminais. Sendo A um não terminal, FOLLOW(A) é o conjunto de terminais **a** que podem aparecer imediatamente à direita de A em alguma forma sentencial. Isto é, o conjunto de terminais **a**, tal que existe uma derivação da forma $S \Rightarrow \alpha A a \beta$ para α e β quaisquer.

Processo para calcular First de todos os símbolos da gramática aplique as seguintes regras até que não haja mais terminais ou ϵ que possam ser acrescentados a algum dos conjuntos First.

1. Se X é um símbolo terminal, então:
FIRST(X)=X
2. Se $X \rightarrow \epsilon$ é uma produção, então adicione ϵ a FIRST(X).
3. Se X é um símbolo não-terminal e $X \Rightarrow Y_1 Y_2 ... Y_k$ é uma produção para algum $k \geq 1$, então acrescente **a** a FIRST(X), se para algum i, **a** estiver em FIRST(Y_i), e ϵ estiver em todos os FIRST(Y_1)...FIRST(Y_{i-1}), ou seja, $Y_1 ... Y_{i-1} \Rightarrow \epsilon$

Para calcular FOLLOW(X) de todos os não-terminais A, as seguintes regras devem ser aplicadas até que nada mais possa ser acrescentado a nenhum dos conjuntos FOLLOW:

1. Se X é o símbolo inicial da gramática coloque \$ em FOLLOW(X), onde \$ é o marcador de fim da entrada
2. Se houver uma produção $A \rightarrow \alpha X \beta$, entao tudo em FIRST($\beta$) exceto ϵ está em FOLLOW(X)
3. Se houver uma produção $A \rightarrow \alpha X$, ou uma produção $A \rightarrow \alpha X \beta$, onde o FIRST($\beta$) contém ϵ, então inclua o FOLLOW(A) em FOLLOW(X).

Exemplo

Considere a gramática abaixo e encontre os conjuntos FIRST e FOLLOW.

$$
\begin{aligned}
E &\rightarrow T\,E' \\
E' &\rightarrow \vee\, T\,E' \mid \varepsilon \\
T &\rightarrow F\,T' \\
T' &\rightarrow \&\; F\,T' \mid \varepsilon \\
F &\rightarrow \neg\, F \mid \mathrm{id}
\end{aligned}
$$