Análise Sintática

Tipos de Analisadores Gramáticais: PARSERS

* Analisadores TOP-DOWN:
  Árvore de derivação começa pela raiz indo para as folhas => Análise Descendente
  Tipo LL(1): Left to right / Leftmost derivation / 1 symbol each time - lookahead
  - Recursivo com Retrocesso (Backtracking)
  - Recursivo Preditivo
  - Preditivo Tabular (não recursivo - pilha + tabela)

* Analisadores BOTTON-UP:
  Árvore de derivação começa pelas folhas indo para a raiz => Análise Ascendente
  Shift / Reduce <= Análise Redutiva
  - LR(k) => i) SLR (simple)
  ii) LR canônicos
  iii) LALR (lookahead LR)
  Tipo LALR(1) => Yacc / Bison - Left to right / Rightmost derivation / 1 each time

* Recuperação de Erros
* Tradução dirigida pela sintaxe

1

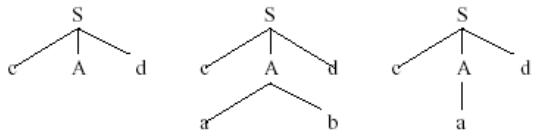
Análise Sintática

* Analisadores Gramaticais **TOP-DOWN**:
  Árvore de derivação começa pela raiz indo para as folhas => Análise Descendente
  Tipo LL(1): Left to right / Leftmost derivation / 1 symbol each time - lookahead
  1. Recursivo com Retrocesso (Backtracking)
  2. Recursivo Preditivo
  3. Preditivo Tabular (não recursivo - pilha + tabela)

Exemplo:

$S \rightarrow cAd$
$A \rightarrow ab$
$\quad / \ a$

Sentença: **cad**



2

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

Exemplo...

OCW MIT – OpenCourseWare
Web: http://www.ocw.mit.edu/

6.035 - Computer Language Engineering Fall 2002
Lecture 5: Top Down Parsing

Web: http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-035Computer-Language-EngineeringFall2002/CourseHome/

Material de apoio...

- Implementação de Linguagens de Programação: Compiladores
  Ana Price e Simão Toscani – Pags. 38 à 53
- Compiladores : Princípios, Técnicas e Ferramentas
  Aho, Sethi, Ullman – Pags. 81-86

3

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

1. *Analisador Recursivo com Retrocesso*

Gramática G = ({<S>,<L>}, {a, ;, [,], $}, P, <S>)

P = { <S> => a | [<L>]

<L> => <S> ; <L> | <S> }

Exemplos de sentenças:

"a$" , "[a]$" , "[a;[a;a]]$"

Analisador:

`$=Fim da entrada`

```
Begin /* Analisador */
  token:=LETOKEN;
  if S then if token=´$´
            then write (´SUCESSO´)
            else write (´ERRO´)
       else write (´ERRO´)
End

Function S
  if token=´a´
  then { token:=LETOKEN; return true }
  else if token=´[´
       then {
              token:=LETOKEN
              if L
              then if token=´]´
                   then { token:=LETOKEN; return true }
                   else return false
              else return false
            }
       else return false
```

```
Function L
  MARCA_PONTO
  if S
  then  if token=´;´
        then {
               token:=LETOKEN
               if L
               then return true
               else return false
             }
        else {
               RETROCEDE
               if S
               then return true
               else return false
             }
  else return false
```

4

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

1. *Analisador Recursivo com Retrocesso*

Gramática G = ({<S>,<L>}, {a, ;, [,], $}, P, <S>)

P = { <S> => a | [<L>]

<L> => <S> ; <L> | <S> }

Exemplos de sentenças:

"a$" , "[a]$" , "[a;[a;a]]$"

Analisador:

```
Begin /* Analisador */
  token:=LETOKEN;
  if S then if token=´$´
            then write (´SUCESSO´)
            else write (´ERRO´)
       else write (´ERRO´)
End

Function S
  if token=´a´
  then { token:=LETOKEN; return true }
  else if token=´[´
       then {
              token:=LETOKEN
              if L
              then if token=´]´
                   then { token:=LETOKEN; return true }
                   else return false
              else return false
            }
       else return false
```

```
Function L
  if S
  then  if token=´;´
        then {
               token:=LETOKEN
               if L
               then return true
               else return false
             }
        else return true
  else return false

>> Sem Back-tracking <<
```


Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

2. *Analisador Recursivo Preditivo*

Gramática:

COMANDO => if EXPR then COMANDO |
while EXPR do COMANDO |
repeat LISTA until EXPR |
id := EXPR

As seguintes regras definem a função FIRST(β), que identifica o conjunto de símbolos terminais que iniciam sentenças deriváveis a partir da forma sentencial β:

1. Se β =*=> ε, então ε é um elemento de FIRST(β).
2. Se β =*=> *a*δ, então *a* é um elemento de FIRST(β).

Gramática Versão Alternativa:

COMANDO => CONDICIONAL |
ITERATIVO |
ATRIBUIÇÃO

CONDICIONAL => if EXPR then COMANDO

ITERATIVO => repeat LISTA until EXPR |
while EXPR do COMANDO

ATRIBUIÇÃO => id := EXPR

Sendo *a* um símbolo terminal e δ uma forma sentencial qualquer, podendo ser vazia.

Para a gramática apresentada ao lado:

FIRST(CONDICIONAL) = { if }
FIRST(ITERATIVO) = { while, repeat }
FIRST(ATRIBUIÇÃO) = { id }

Dado um símbolo não-terminal A, definido por várias alternativas que não iniciam por terminais, a implementação de um analisador recursivo preditivo para A exige que os conjuntos FIRST para os não terminais que iniciam as várias alternativas de produção sejam disjuntos.

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

2. *Analisador Recursivo Preditivo*

Gramática:

COMANDO => if EXPR then COMANDO |
while EXPR do COMANDO |
repeat LISTA until EXPR |
id := EXPR

Gramática Versão Alternativa:

COMANDO => CONDICIONAL |
ITERATIVO |
ATRIBUIÇÃO

CONDICIONAL => if EXPR then COMANDO

ITERATIVO => repeat LISTA until EXPR |
while EXPR do COMANDO

ATRIBUIÇÃO => id := EXPR

```
Analisador PREDITIVO

Function COMANDO
Begin
   if token = 'if'
   then  if CONDICIONAL
         then return true
         else return false
   else  if token = 'while' or
            token = 'repeat'
         then if INTERATIVO
              then return true
              else return false
         else if token = 'id'
              then  if ATRIBUIÇÃO
                    then return true
                    else return false
              else  return false
End
```

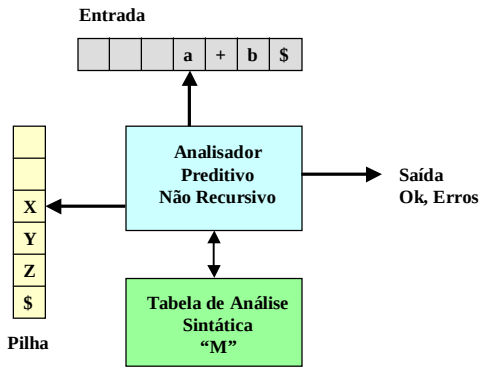
FIRST(CONDICIONAL) = { if }
FIRST(ITERATIVO) = { while, repeat }
FIRST(ATRIBUIÇÃO) = { id }

7

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

3. *Analisador Preditivo Tabular*



Analisador Preditivo Não-Recursivo:

Composto por:

- **Uma "fita de entrada"**
- **Uma pilha auxiliar**
- **Uma tabela de derivação preditiva (Tabela Sintática)**

Funcionamento:

Similar ao Analisador Preditivo Recursivo. Necessita que seja gerada previamente a Tabela Sintática.

É possível construir um analisador preditivo não-recursivo mantendo explicitamente uma pilha, ao invés de implicitamente através de chamadas recursivas

8

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

3. *Analisador Preditivo Tabular*

Análise Sintática:

O analisador é controlado por um programa que considera X, o símbolo no topo da pilha, e *a* é um símbolo terminal da entrada. Estes dois símbolos determinam a ação do analisador. Há três possibilidades então:

1. **Se X = *a* = $, o analisador encerra o reconhecimento, com sucesso.**
2. **Se X = *a* ≠ $, o analisador desempilha X da pilha e avança o ponteiro de entrada ao próximo símbolo de entrada.**
3. **Se X é um não-terminal, o programa consulta a entrada M[X,*a*] da tabela de derivação M. Esta entrada pode ser uma produção de X ou um erro. Se, por exemplo, M[X,*a*] = { X⇒UVW }, o analisador substitui X no topo da pilha por WVU (com U no topo).**

Como saída, assume-se que o analisador apenas imprime a produção utilizada. Se M[X,a] = erro, o analisador chama a rotina de recuperação de erro.

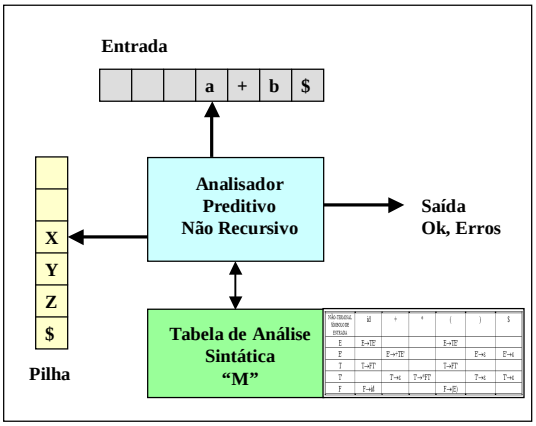


Tabela de Análise Sintática:

É uma matriz M com *n* linhas e *t+1* colunas, onde *n* é o número de símbolos não-terminais, e *t* é o número de símbolos terminais (a coluna extra corresponde ao Símbolo $)

9

Análise Top Down

* Analisadores **TOP-DOWN**:

3. *Analisador Preditivo Tabular*

Regras de Produção

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

Terminais:
{ id, +, *, (,) }

Não-Terminais:
{ E, E´, T, T´, F }

Entrada:
Id + Id * Id

Algoritmo:
Price & Toscani, pg.47

PILHA	ENTRADA	SAÍDA
$E	id+id*id$	
$E'T	id+id*id$	E → TE'
$E'T'F	id+id*id$	T → FT'
$E'T'id	id+id*id$	F → id
$E'T'	+id*id$	
$E'	+id*id$	T' → ε
$ET+	+id*id$	E' → +TE'
$ET	id*id$	
$ETF	id*id$	T → FT'
$ETid	id*id$	F → id
$ET'	*id$	
$ETF*	*id$	T' → *FT'
$ETF	id$	
$ETid	id$	F → id
$ET'	$	
$E'	$	T' → ε
$	$	E' → ε

NÃO-TERMINAL / SÍMBOLO DE ENTRADA	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Gramática LL(1)

10

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

3. *Analisador Preditivo Tabular*

Regras de Produção

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

NÃO-TERMINAL / SÍMBOLO DE ENTRADA	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Criando a Tabela de Análise Sintática...

A tabela de análise sintática (derivação preditiva) é feita utilizando duas funções associadas à gramática G: FIRST e FOLLOW.

FIRST: Se α é um símbolo da gramática, FIRST(α) é o conjunto dos terminais que iniciam as palavras derivadas por α. Sendo que: i) o FIRST de um terminal *a*, FIRST(*a*), é o próprio terminal *a*; ii) o FIRST de um símbolo *a* que deriva o ε (vazio), inclui o símbolo ε.

Determinando o conjunto FIRST:

1. **Se *a* é um terminal, então FIRST(*a*) é {*a*}.**
2. **Se X ⇒ ε é uma produção, então acrescente ε ao FIRST(X).**
3. **Se X é um não-terminal e X ⇒ $Y_1Y_2...Y_k$ é uma produção, então insira *a* em FIRST(X) se para algum Y_j, *a* está em FIRST(Y_j), e insira ε, caso este esteja em todos os conjuntos FIRST(Y_1), ..., FIRST(Y_{i-1}); ou seja, $Y_1,..., Y_{i-1}$ derivam, mesmo que indiretamente, ε. Se ε está no conjunto FIRST(Y_j), então e deve ser acrescentado ao conjunto FIRST(X).**

11

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

3. *Analisador Preditivo Tabular*

Regras de Produção

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

NÃO-TERMINAL / SÍMBOLO DE ENTRADA	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Criando a Tabela de Análise Sintática...

A tabela de análise sintática (derivação preditiva) é feita utilizando duas funções associadas à gramática G: FIRST e FOLLOW.

FOLLOW: a função FOLLOW(A), para um não-terminal A, é o conjunto dos terminais *a* que aparecem imediatamente no lado direito de A, em alguma forma de derivação, mesmo que indiretamente (passando por várias produções). Se A for o símbolo mais à direita de alguma derivação, então $ pertence ao FOLLOW(A).

Determinando o conjunto FOLLOW:

1. **O símbolo $ pertence ao conjunto FOLLOW(S), onde S é o símbolo inicial da gramática e $ é o marcador de final de entrada.**
2. **Se há uma produção A ⇒ αXβ, então todos os terminais de FIRST(β), com exceção de ε, fazem parte de FOLLOW(X).**
3. **Se há uma produção A ⇒ αX, ou uma produção A ⇒ αXβ, onde FIRST(β) contém ε, então todos os terminais que pertencerem a FOLLOW(A) pertencem também a FOLLOW(X).**

12

Análise Sintática – Top Down Parsing

* Analisadores Gramaticais **TOP-DOWN**:

3. *Analisador Preditivo Tabular*

Regras de Produção

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

NÃO-TERMINAL / SÍMBOLO DE ENTRADA	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

FIRST e FOLLOW para as regras de produção acima...

FIRST(E) = FIRST(T) = FIRST(F) = { (, id }
FIRST(E') = {+,ε}
FIRST(T) = FIRST(F) = { (, id }
FIRST(T') = {*, ε}
FIRST(F) = { (, id }

FOLLOW(E) = FOLLOW(E') = {), $}
FOLLOW(T) = FOLLOW(T') = {+,), $}
FOLLOW(F) = {+, *,), $}

A tabela de análise preditiva é obtida assim:

1. **Para cada produção A ⇒ α da gramática, siga os passos 2 e 3.**
2. **Para cada terminal *a* pertencente ao conjunto FIRST(α), acrescente A ⇒ α na posição M[A,*a*] da tabela.**
3. **Se ε pertence a FIRST(α), acrescente A ⇒ α a M[A,*b*] para cada terminal *b* em FOLLOW(A). Se ε pertence a FIRST(α) e $ pertence a FOLLOW(A), acrescente A ⇒ α na posição M[A,$].**
4. **Cada entrada não definida da tabela determina um estado de erro.**

13

Análise Sintática – Top Down Parsing

3. *Analisador Preditivo Tabular*

Regras de Produção

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

NÃO-TERMINAL / SÍMBOLO DE ENTRADA	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

FIRST e FOLLOW para as regras de produção acima...

E ⇒ TE' ∴ FIRST(TE') = { (, id }
Adicionar E ⇒ TE' em M[E,(] ; M[E,id]
E' ⇒ +TE' ∴ FIRST(+TE') = { + }
Adicionar E ⇒ +TE' em M[E',+]
E' ⇒ ε ∴ FOLLOW(E') = { $,) }
Adicionar E' ⇒ ε em M[E',$] ; M[E',)]
T ⇒ FT' ∴ FIRST(FT') = { (, id }
Adicionar T ⇒ FT' em M[T,(] ; M[T,id]
T' ⇒ *FT' ∴ FIRST(*FT) = { * }
Adicionar T' ⇒ *FT' em M[T',*]
T' ⇒ ε ∴ FOLLOW(T') = { +,),$ }
Adicionar T' ⇒ ε em M[T',+] ; M[T',)] ; M[T',$]
F ⇒ (E) ∴ FIRST((E)) = { (}
Adicionar F ⇒(E) em M[F,(]
F ⇒ id ∴ FIRST(id) = { id }
Adicionar F ⇒ id em M[F,id]

A tabela de análise preditiva é obtida assim:

1. **Para cada produção A ⇒ α da gramática, siga os passos 2 e 3.**
2. **Para cada terminal *a* pertencente ao conjunto FIRST(α), acrescente A ⇒ α na posição M[A,*a*] da tabela.**
3. **Se ε pertence a FIRST(α), acrescente A ⇒ α a M[A,*b*] para cada terminal *b* em FOLLOW(A). Se ε pertence a FIRST(α) e $ pertence a FOLLOW(A), acrescente A ⇒ α na posição M[A,$].**
4. **Cada entrada não definida da tabela determina um estado de erro.**

14

That´s all foks!

Análise Sintática – Top Down Parsing

Próximo assunto...

Análise Sintática – Botton Up Parsing

COMPILERS:
May the Force be with you...