Análise Sintática
(Cap. 04)
Análise Sintática Descendente

Análise Sintática

- Análise sintática descendente
  - Constrói a árvore de derivação de cima para baixo, da raíz para as folhas, criando os nós da árvore em pré-ordem
  - Produz uma derivação mais à esquerda para uma cadeia de entrada
  - Exercício: mostre a árvore sintática descendente para a cadeia id+id*id considerando a gramática:
    - E-> T E'
    - E'-> +TE' | ε
    - T-> FT'
    - T'-> *FT' | ε
    - F-> (E) | id

Análise Sintática

- Análise Sintática Descendente Recursiva (ASDR)
  - Um método de ASDR consiste em um conjunto de procedimentos, um para cada não-terminal da gramática
  - A execução começa com o (procedimento do símbolo) não-terminal que representa o símbolo inicial da gramática
  - A execução avança caracter por caracter a medida que os símbolos são reconhecidos. A execução indica um erro quando o símbolo não é reconhecido

Análise Sintática

- Análise Sintática Descendente Recursiva (ASDR)
  - Exemplo:

```
void A(){
1) escolha uma produção-A, A-> X1| X2 | ... | Xk
2) para (cada i de 1  até k)
3)     se (Xi é um não-terminal)
4)         invocar procedimento Xi()
5)     senão se (Xi==a) //a é o símbolo de entrada atual
6)         avance na entrada para o próximo símbolo
7)     senão /* erro */
}
```

Análise Sintática

- Observe que gramáticas recursivas à esquerda não funcionam com o analisador sintático descendente recursvivo
  - Exemplo: expr -> expr ’+’ termo | termo

```
int expr(void) {
    return (expr () && require(token('+'))  && termo()
                || termo());
}
```

  - Observe o **loop infinito** em expr()

Análise Sintática

- Análise Sintática Descendente Recursiva (ASDR)
  - Exemplo: aplicar ASDR com a entrada w=cad e a gramática:
    - S -> cAd
    - A -> ab | a

Análise Sintática

- Durante a análise descendente, as variáveis First e Follow nos ajudam a escolher qual produção deve ser usada, com base no próximo símbolo de entrada
- Dada uma entrada **a** e o não-terminal **A**, deve-se saber qual das produções alternativas A -> B1|B2|B3|... deriva a seqüência que inicia por **a**
- Calcula-se então os conjuntos de primeiros símbolos produzidos por todas as alternativas na gramática, ou seja, os conjuntos **FIRST**
- Durante a recuperação de erros, os valores retornados por **FOLLOW** podem ser usados como tokens de sincronismo

Análise Sintática

- **First(A)**, onde **A** é qualquer cadeia de símbolos da gramática, é o conjunto de símbolos terminais que iniciam as cadeias derivadas de **A**
- Cálculo de FIRST(X):
  - Se X é terminal, então First(X)={X}
  - Se X é não-terminal e X->Y1Y2...Yk é produção para algum k>=1, então acrescente **a** a First(X) se, para algum i, **a** estiver em First(Yi) e **epsilon** estiver em First(Y1)...First(Yi-1). Se epsilon está em First(Yj) para todo j=1,2,...,k então adicione **epsilon** a First(X)
  - Se X->epsilon é uma produção, então acrescente epsilon a First(X)

Análise Sintática

- Exemplo: A partir da gramática a seguir:
- COMAND -> COND | ITER | ATRIB
- COND -> if EXPR then COMAND
- ITER -> repeat LISTA until EXPR | while EXPR do COMAND
- ATRIB-> id := EXPR
- Calcule:
  - First(ATRIB): {id}
  - First(ITER): {repeat, while}
  - First(COND): {if}
  - First(COMAND): {if, repeat, while, id}

Análise Sintática

- **Follow(A)**, para o não-terminal **A**, é o conjunto de terminais **a** que podem aparecer imediamente à direita de **A** em alguma forma sentencial. Se **A** é o não-terminal mais a direita em alguma forma setencial, então **$** (símbolo marcador de final de sentença/arquivo) estará em Follow(A)
- Para calcular Follow:
  - Coloque $ em Follow(S) se S é o símbolo inicial da gramática
  - Se houver uma produção A->aBC, então tudo em First(C), exceto epsilon, está em Follow(B)
  - Se há uma produção A->aB, ou uma produção A->aBC, onde o First(C) contém epsilon, então inclua o Follow(A) em Follow(B)

Análise Sintática

- Exemplo: A partir da gramática a seguir:
- S -> cAd
- A -> b | a
- Calcule:
  - First(S): {c}
  - First(A):{b, a}
  - Follow(S): {$}
  - Follow(A): {d}

Análise Sintática

- Exercício:
  - Calcule os conjuntos Follow e First para a seguinte gramática:
    - E-> T E'
    - E' -> + T E' | epsilon
    - T -> F T'
    - T' -> * F T' | epsilon
    - F -> (E) | id

Análise Sintática

- Gramáticas LL(1)
  - Gramáticas que utilizam ASDR
  - O primeiro L refere-se à maneira como os caracteres de entrada são lidos -> da esquerda para a direita
  - O segundo L indica que há uma derivação mais à esquerda
  - O “1” indica que 1 símbolo de entrada é visto na entrada
  - Uma gramática G é LL(1) sse qualquer regra A->B|C são duas produções distintas de G tal que:

    1) Dado um terminal a, ambos B e C não derivam uma mesma string que inicia com a

    2)No máximo um entre B e C podem derivar a string vazia

    3)Se C->epsilon em zero ou mais passos, então B não deriva qualquer string iniciando com um terminal em Follow(A). O mesmo caso ocorre para B->epsilon em zero ou mais passos

Análise Sintática

- Analisadores sintáticos preditivos podem ser construídos para gramáticas LL(1) se podemos selecionar a produção adequada a partir do símbolo de entrada atual
- O algoritmo a seguir obtém informações dos conjuntos First e Follow e armazena em uma tabela de predição, na forma M[A,a], onde **M** é o nome da tabela, **A** é o símbolo não-terminal e **a** é o símbolo terminal
- Idéia básica: A produção **A->B** é escolhida se o próximo símbolo de entrada **a** está em First(**B**)
  - Obs: Uma complicação surge quando **B->epsilon** ou, mais comumemente, **B->epsilon** em zero ou mais passos. Nesse caso, escolhemos **A->B** se o símbolo de entrada atual está em **Follow(A)** ou se $ foi alcançado e $ está em **Follow(A)**

Análise Sintática

- **Algoritmo: construção da tabela do analisador preditivo**
- Entrada: Gramática G
- Saída: Tabela M
- Para cada produção A->B da gramática:
  - Para cada terminal **a** em **First(A)**, adicionar **A->B** em **M[A,a]**
  - Se epsilon está em **First(B)**, então, para cada terminal **b** em **Follow(A)**, adicionar **A->B** em **M[A,b]**. Se epsilon, está em **First(B)** e **$** está em **Follow(A)**, adicionar **A->B** em **M[A,$]** também.
  - Se, após realizar os passos anteriores, não existe produção para **M[A,a]** então, indicar um **erro** (entrada vazia na tabela)

Análise Sintática

- **Exemplo:** Considere a gramática:

S -> cAd

A -> b | a

e seus conjuntos First e Follow:

First(S): {c}; First(A):{b, a}; Follow(S): {$}; Follow(A): {d}

Não-terminal	c	d	a	b
S	S->cAd			
A			A->a	A->b

Análise Sintática

- **Exercício:** Dada a expressão: id+id*id

e a gramática:

E->T E'

E'-> + T E' | epsilon

T-> F T'

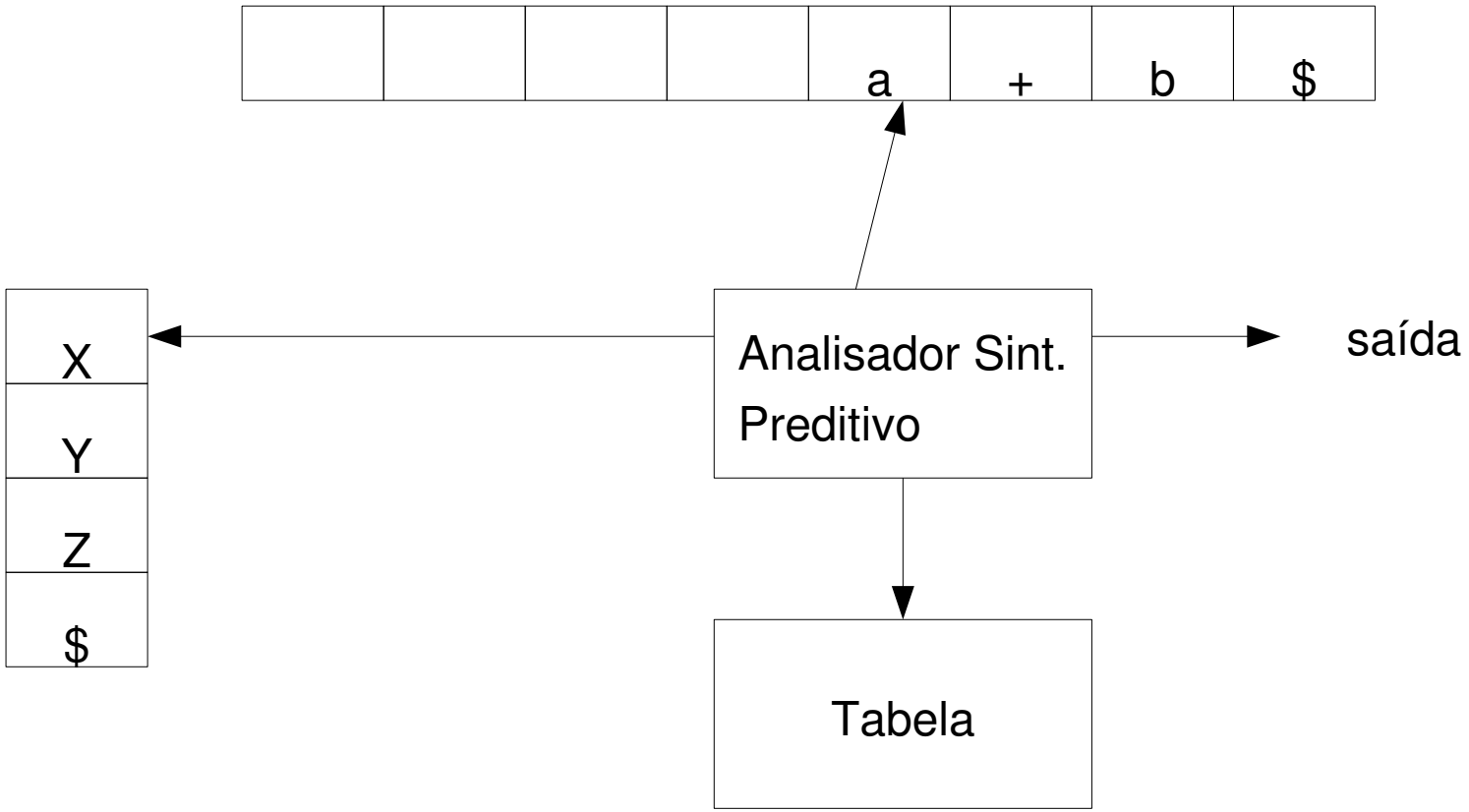
T'-> * F T' | epsilon

F-> (E) | id

Construa a tabela do analisador sintático preditivo

Análise Sintática

- Analisador Sintático preditivo não-recursivo
  - É um AS que mantém, explicitamente, uma pilha de símbolos não-terminais. Nesse caso, o AS imita uma derivação à esquerda



Análise Sintática

- Analisador Sintático preditivo não-recursivo

  - Algoritmo:

    - Entrada: string w e tabela M para gramática G
    - Saída: derivação à esquerda de w ou indicação de erro

```
definir ip como ponteiro para o 1o. Símbolo de w
definir X como ponteiro para o topo da pilha
while (X->topo!=$)
    if (X->topo==a) //a é o símbolo apontado por ip
       retirar X->topo da pilha e avançar ip
   else if (X->topo==TERMINAL) error();
   else if (M[X->topo,a]==ERRO) error();
   else if (M[X->topo,a]==X->Y1, Y2, ..., Yk){
       retirar X->topo da pilha;
       colocar Yk, ..., Y2, Y1 na pilha
   }
   definir X como ponteiro para o topo da pilha
```