

# Compiladores

## Análise Sintática

### Análise Sintática LL(1)

Prof. Dr. Luiz Eduardo G. Martins  
(adaptado por Profa Dra Ana Carolina Lorena)  
UNIFESP

# Análise Sintática Descendente Recursiva

$S \rightarrow \text{BEGIN } S \text{ L}$   
 $S \rightarrow \text{IF } E \text{ THEN } S \text{ ELSE } S$   
 $S \rightarrow \text{PRINT ID}$

```
void S()
{
    switch (tok)
    {
        case BEGIN: eat (BEGIN); S(); L(); break;
        case IF: eat (IF); E(); eat (THEN); S(); eat (ELSE); S(); break;
        case PRINT: eat (PRINT); eat(ID); break;
        default: ERRO();
    }
}
```

- Diferenciação das escolhas das regras baseada nos símbolos terminais
- Nem sempre a GLC apresentará regras nesse “formato”

# Análise Sintática Descendente Recursiva

- Analisadores descendentes recursivos só podem ser construídos para gramáticas em que o primeiro símbolo terminal de cada regra fornece informações suficientes para escolher a produção a ser utilizada
- **Problemas** nessa abordagem:
  - $A \rightarrow \alpha \mid \beta \mid \dots$ 
    - $\alpha$  e  $\beta$  iniciarem com símbolos não-terminais
  - $A \rightarrow \epsilon$ 
    - O não-terminal  $A$  pode desaparecer, precisamos saber quais marcas podem suceder  $A$  legalmente

# Análise Sintática LL(1)

- Solução para esses problemas
  - Gramática LL(1)
    - Primeiro L: a cadeia de entrada (sentença de *tokens*) é obtida da esquerda para a direita (*left to right*)
    - Segundo L: adota-se derivação mais à esquerda (*leftmost derivation*)
    - Número (1): indica o número de símbolos de entrada (terminais) necessários p/ decidir qual produção será escolhida (*lookahead*)
  - A gramática LL(1) leva à implementação do *parser* LL(1)

# Análise Sintática LL(1)

- *O parser* LL(1) utiliza uma pilha explícita em vez de ativações recursivas
- Portanto, implementa uma análise sintática descendente, não mais recursiva, chamada *análise sintática LL(1)*

# Análise Sintática LL(1)

- Modelo básico da análise sintática LL(1)

– Considere o exemplo:

$S \rightarrow ( S ) S \mid \epsilon$  (GLC que gera cadeias de parênteses balanceados)

	Pilha de Análise Sintática	Entrada	Ação
1	\$ S	( ) \$	$S \rightarrow ( S ) S$ [substitui]
2	\$ S ) S (	( ) \$	[casamento]
3	\$ S ) S	) \$	$S \rightarrow \epsilon$ [substitui]
4	\$ S )	) \$	[casamento]
5	\$ S	\$	$S \rightarrow \epsilon$ [substitui]
6	\$	\$	[aceita]

É incluído o símbolo \$ para indicar a base da pilha e o final da entrada

# Análise Sintática LL(1)

- Modelo básico da análise sintática LL(1)

– Considere o exemplo:

$S \rightarrow ( S ) S \mid \epsilon$  (GLC que gera cadeias de parênteses balanceados)

	Pilha de Análise Sintática	Entrada	Ação
1	\$ S	( ) \$	$S \rightarrow ( S ) S$ [substitui]
2	\$ S ) S (	( ) \$	[casamento]
3	\$ S ) S	) \$	$S \rightarrow \epsilon$ [substitui]
4	\$ S )	) \$	[casamento]
5	\$ S	\$	$S \rightarrow \epsilon$ [substitui]
6	\$	\$	[aceita]

No começo o símbolo inicial da gramática é colocado na pilha

# Análise Sintática LL(1)

- Modelo básico da análise sintática LL(1)

– Considere o exemplo:

$S \rightarrow ( S ) S \mid \epsilon$  (GLC que gera cadeias de parênteses balanceados)

	Pilha de Análise Sintática	Entrada	Ação
1	\$ S	( ) \$	$S \rightarrow ( S ) S$ [substitui]
2	\$ S ) S (	( ) \$	[casamento]
3	\$ S ) S	) \$	$S \rightarrow \epsilon$ [substitui]
4	\$ S )	) \$	[casamento]
5	\$ S	\$	$S \rightarrow \epsilon$ [substitui]
6	\$	\$	[aceita]

Analizador, a cada passo: substitui um não-terminal no topo da pilha por uma de suas regras **ou** casa uma marca no topo da pilha com a entrada seguinte



# Análise Sintática LL(1)

- Modelo básico da análise sintática LL(1)

– Considere o exemplo:

$S \rightarrow ( S ) S \mid \epsilon$  (GLC que gera cadeias de parênteses balanceados)

	Pilha de Análise Sintática	Entrada	Ação
1	\$ S	( ) \$	$S \rightarrow ( S ) S$ [substitui]
2	\$ S ) S (	( ) \$	[casamento]
3	\$ S ) S	) \$	$S \rightarrow \epsilon$ [substitui]
4	\$ S )	) \$	[casamento]
5	\$ S	\$	$S \rightarrow \epsilon$ [substitui]
6	\$	\$	[aceita]

Quando a pilha e a entrada estiverem vazias, então a análise chegou ao final, sem identificação de erros (**aceita** então a cadeia)

# Análise Sintática LL(1)

- Modelo básico da análise sintática LL(1)

– Considere o exemplo:

$S \rightarrow ( S ) S \mid \epsilon$  (GLC que gera cadeias de parênteses balanceados)

	Pilha de Análise Sintática	Entrada	Ação
1	\$ S	( ) \$	$S \rightarrow ( S ) S$ [substitui]
2	\$ S ) S (	( ) \$	[casamento]
3	\$ S ) S	) \$	$S \rightarrow \epsilon$ [substitui]
4	\$ S )	) \$	[casamento]
5	\$ S	\$	$S \rightarrow \epsilon$ [substitui]

Tabela corresponde a passos de derivação à esquerda de ( )

$S \Rightarrow (S)S$  [ $S \rightarrow (S)S$ ]

$\Rightarrow ()S$  [ $S \rightarrow \epsilon$ ]

$\Rightarrow ()$  [ $S \rightarrow \epsilon$ ]

# Análise Sintática LL(1)

- Técnica de implementação do *parser* para gramática LL(1):
  - Cria-se uma tabela preditiva  $M[N,T]$  para auxiliar na construção do *parser*
  - $N$  é o conjunto de símbolos não-terminais da gramática
  - $T$  é o conjunto de símbolos terminais da gramática
  - Suponha que tenhamos o não-terminal  $A$  e o terminal  $a$ 
    - Na posição  $A \times a$  da tabela devemos colocar qual é a produção que devemos utilizar se, ao tentarmos reconhecer  $A$ , encontramos na entrada o símbolo  $a$

# Análise Sintática LL(1)

- Regras para preencher  $M[N,T]$ :
  - Se  $A \rightarrow \alpha$  e existe derivação  $\alpha \Rightarrow^* a\beta$ , então  $M[A,a] = A \rightarrow \alpha$ 
    - Se  $a$  está na entrada, queremos selecionar uma regra que produza  $a$
  - Se  $A \rightarrow \alpha$  e existem derivações  $\alpha \Rightarrow^* \varepsilon$  e  $S\$ \Rightarrow^* \beta A a \gamma$ , então  $M[A,a] = A \rightarrow \alpha$ 
    - Se  $A$  derivar  $\varepsilon$  e se  $a$  sucede  $A$  em uma derivação, queremos selecionar que  $A$  desapareça

# Análise Sintática LL(1)

- Exemplo de tabela preditiva

$S \rightarrow aSAb$

$S \rightarrow bAa$

$A \rightarrow bAb$

$A \rightarrow c$

Não-Terminal	Terminal		
	a	b	c
S	$S \rightarrow aSAb$	$S \rightarrow bAa$	
A		$A \rightarrow bAb$	$A \rightarrow c$

- As posições vazias na tabela indicam que esse terminal não pode aparecer no início da regra de produção (são os erros sintáticos)

# Análise Sintática LL(1)

- Uma gramática é **LL(1)** se a tabela de análise sintática associada tiver no máximo uma produção em cada célula
  - Não pode ser ambígua

Se tivermos mais do que uma produção em alguma posição da tabela, então a GLC não pode ser usada na construção de um *parser* LL(1)

A gramática não é do tipo LL(1)

# Análise Sintática LL(1)

- Algoritmo análise sintática LL(1):

Coloca símbolo de começo no topo da pilha

```
while topo da pilha  $\neq$  $ and próxima marca  $\neq$  $ do
    if topo da pilha for terminal a and próxima marca for a
    then // casamento
        Retira topo da pilha
        Avança entrada
    else if topo da pilha for não-terminal A and
        Próxima marca for terminal a and
         $M[A,a]$  contiver produção  $A \rightarrow X_1X_2\dots X_n$ 
    then // substitui
        Retira topo da pilha
        For  $i = n$  downto 1 do
            Coloca  $X_i$  na pilha
        else erro
if topo da pilha = $ and próxima marca = $
then aceita
else erro
```

# Análise Sintática LL(1)

- Para que o *parser* LL(1) seja implementado com sucesso, é necessário garantir:
  - Que as regras de produção da GLC não apresentem recursão à esquerda
    - ✓ Técnica da remoção de recursão à esquerda
  - Que a disposição dos símbolos terminais nas regras da GLC permitam a escolha de uma única produção
    - ✓ Técnica da fatoração à esquerda



# Análise Sintática LL(1)

- Recursão à Esquerda

- Ocorre quando, p/ algum não-terminal  $B$ , temos  $B \rightarrow B\alpha$  ( $\alpha$  significa uma cadeia de terminais ou não-terminais)
- Intuitivamente é fácil perceber que o *parser* poderia entrar em uma recursão infinita, ao tentar reconhecer  $B$  sem consumir nenhum *token* de entrada

# Análise Sintática LL(1)

- Recursão à Esquerda

—Pode ser de dois tipos

- Direta (ou imediata):

$$B \rightarrow B\beta$$

- Indireta:

$$B \rightarrow A\alpha$$

$$A \rightarrow B\beta$$

# Análise Sintática LL(1)

- Recursão à Esquerda

- Remoção da recursão à esquerda direta (reescrevendo a gramática em notação BNF)

- 1) Dividir as produções de B em 2 subconjuntos:

- $N = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ , das produções que não possuem recursão à esquerda
- $R = \{B\beta_1, B\beta_2, \dots, B\beta_m\}$ , das produções que possuem recursão à esquerda

- 2) Eliminar as produções de R da gramática

# Análise Sintática LL(1)

- Recursão à Esquerda

– Remoção da recursão à esquerda direta

3) Adicionar à gramática as seguintes produções:

$B \rightarrow \alpha_1 B'$  ( $\alpha_i$  representa a parte sentencial das regras sem recursão à esquerda)

$B \rightarrow \alpha_2 B'$

...

$B \rightarrow \alpha_n B'$

onde  $B'$  é um novo símbolo não-terminal

# Análise Sintática LL(1)

- Recursão à Esquerda

- Remoção da recursão à esquerda direta

4) Adicionar à gramática as novas produções p/ B'

$B' \rightarrow \beta_1$

$B' \rightarrow \beta_2$

...

$B' \rightarrow \beta_m$

$\beta_i$  representa a parte sentencial das regras com recursão à esquerda, após o símbolo não-terminal recursivo

$B' \rightarrow \beta_1 B'$

$B' \rightarrow \beta_2 B'$

...

$B' \rightarrow \beta_m B'$

# Análise Sintática LL(1)

**Exemplo:** Utilizar o procedimento descrito p/ eliminar a recursão à esquerda direta nas produções da seguinte gramática:

$$E \rightarrow E + T$$

$$E \rightarrow T$$

$$T \rightarrow \text{NUM}$$

- $\alpha$   $\beta$
- 1)  $N = \{E \rightarrow T\}$ ,  $R = \{E \rightarrow E + T\}$
- 2)  ~~$R = \{E \rightarrow E + T\}$~~
- 3)  $E \rightarrow T E'$
- 4)  $E' \rightarrow +T \mid +T E'$

Gramática Modificada

$$E \rightarrow T E' \mid T$$

$$E' \rightarrow +T E' \mid +T$$

$$T \rightarrow \text{NUM}$$

# Análise Sintática LL(1)

## Remoção da recursão à esquerda

### Gramática Original

$E \rightarrow E + T$   
 $E \rightarrow T$   
 $T \rightarrow \text{NUM}$

5 + 8 + 2

$E \Rightarrow E + T$   
 $\Rightarrow E + T + T$   
 $\Rightarrow T + T + T$

...

$\Rightarrow \text{NUM} + \text{NUM} + \text{NUM}$  OK

### Gramática Modificada

$E \rightarrow TE' \mid T$   
 $E' \rightarrow +TE' \mid +T$   
 $T \rightarrow \text{NUM}$

5 + 8 + 2

$E \Rightarrow TE'$   
 $\Rightarrow \text{NUM } E'$   
 $\Rightarrow \text{NUM} + TE'$   
 $\Rightarrow \text{NUM} + \text{NUM } E'$   
 $\Rightarrow \text{NUM} + \text{NUM} + T$   
 $\Rightarrow \text{NUM} + \text{NUM} + \text{NUM}$  OK

# Análise Sintática LL(1)

**Exercício 1:** Utilizar o procedimento descrito p/ eliminar a recursão à esquerda nas produções da gramática:

$\text{expressão} \rightarrow \text{expressão} + \text{termo}$

$\text{expressão} \rightarrow \text{expressão} - \text{termo}$

$\text{expressão} \rightarrow \text{termo}$

$\text{termo} \rightarrow \text{termo} * \text{fator}$

$\text{termo} \rightarrow \text{termo} / \text{fator}$

$\text{termo} \rightarrow \text{fator}$

$\text{fator} \rightarrow (\text{expressão})$

$\text{fator} \rightarrow \text{ID}$

$\text{fator} \rightarrow \text{NUM}$



# Análise Sintática LL(1)

- **Fatoração à Esquerda**

- Quando duas ou mais escolhas de regras compartilham uma cadeia de prefixo comum

- $A \rightarrow \alpha\beta | \alpha\gamma$

- Considere a seguinte gramática

- $S \rightarrow \text{if } E \text{ then } S \text{ else } S$

- $S \rightarrow \text{if } E \text{ then } S$

- ...



*Problema: o mesmo terminal – “if” – inicia a produção S.  
Logo, não há como escolher a produção correta, indicando que  
essa não é uma gramática LL(1).*

# Análise Sintática LL(1)

- Fatoração à Esquerda

- Esse problema pode ser resolvido utilizando-se o procedimento chamado **FATORAÇÃO À ESQUERDA**
- A fatoração à esquerda consiste em modificar as produções de um não-terminal de modo a adiar a decisão sobre qual produção utilizar, até que tenham sido lidos *tokens* suficientes para isso
- Basicamente, deve ser identificado o **maior prefixo comum** das produções a serem fatoradas, e criar novas produções para completar a produção original

# Análise Sintática LL(1)

- Fatoração à Esquerda

- $A \rightarrow \alpha\beta | \alpha\gamma$

- É reescrita como:

- $A \rightarrow \alpha A'$

- $A' \rightarrow \beta | \gamma$

Precisamos garantir que  $\alpha$  seja a cadeia mais longa compartilhada do lado direito

# Análise Sintática LL(1)

- Fatoração à Esquerda

$S \rightarrow \text{if } E \text{ then } S \text{ else } S$

$S \rightarrow \text{if } E \text{ then } S$

- A produção  $S$  acima, após a fatoração à esquerda ficaria assim:

$S \rightarrow \text{if } E \text{ then } S X$

$X \rightarrow \text{else } S$

$X \rightarrow \epsilon$

*prefixo comum: if E then S*

*X: parte variável, pode ser else S, ou vazio*

*X é um não-terminal anulável*

# Análise Sintática LL(1)

Aplicar o procedimento de fatoração à esquerda para a gramática a seguir:

$\text{decl-seq} \rightarrow \text{decl} ; \text{decl-seq} \mid \text{decl}$   
 $\text{decl} \rightarrow s$

## **Gramática Modificada**

$\text{decl-seq} \rightarrow \text{decl decl-seq}'$   
 $\text{decl-seq}' \rightarrow ; \text{decl-seq} \mid \varepsilon$

# Análise Sintática LL(1)

Aplicar o procedimento de fatoração à esquerda para a gramática a seguir:

declaração  $\rightarrow$  identificador := exp | identificador ( exp-lista ) | outra

## **Gramática Modificada**

declaração  $\rightarrow$  identificador declaração' | outra

declaração'  $\rightarrow$  :=exp | ( exp-lista )

# Análise Sintática LL(1)

**Exercício 2:** Aplicar o procedimento de fatoração à esquerda para a gramática a seguir:

expressão  $\rightarrow$  termo expressão' | termo  
expressão'  $\rightarrow$  -termo | +termo | -termo expressão' | +termo expressão'  
termo  $\rightarrow$  fator termo' | fator  
termo'  $\rightarrow$  \*fator | /fator | \*fator termo' | /fator termo'  
fator  $\rightarrow$  (expressão) | ID | NUM

# Análise Sintática LL(1)

- Bibliografia consultada

LOUDEN, K. C. **Compiladores: princípios e práticas**. São Paulo: Pioneira Thompson Learning, 2004 (Cap. 4)

MERINO, M. **Notas de Aulas - Compiladores**, UNIMEP, 2006.