

# Conceitos de Análise Sintática

Pro.f. Marcus Ramos

UNIVASF

Atualizado em 28 de outubro de 2016

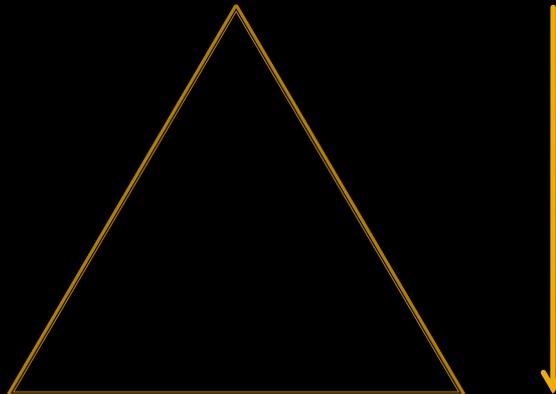
# Referências

The Theory of Parsing, Translation and Compiling  
Volume I: Parsing  
Alfred V. Aho  
Jeffrey D. Ullman  
Prentice Hall 1972

Programming Language Processors in Java  
Compilers and Interpreters  
D. A. Watt  
D. F. Brown  
Pearson Education 2000

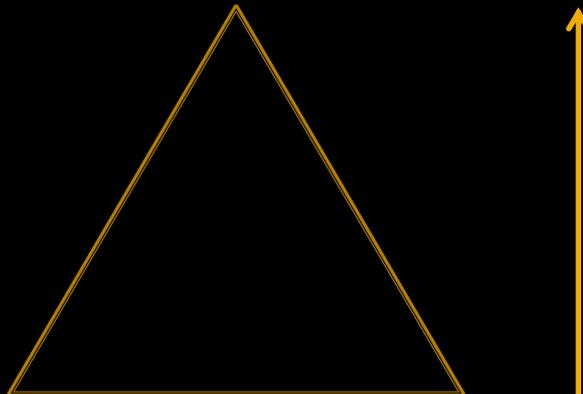
# Análise descendente

- Os movimentos do reconhecedor correspondem ao uso de derivações mais à esquerda;
- A árvore de sintaxe é montada de cima para baixo (da raiz em direção às folhas);
- Caracteriza a classe das gramáticas (e linguagens) LL(k);
- Leitura do arquivo-fonte da esquerda para a direita;
- Emprego de derivações mais à esquerda (na ordem direta);
- Uso de no máximo k símbolos de lookahead.



# Análise ascendente

- Os movimentos do reconhecedor correspondem ao uso de derivações mais à direita;
- A árvore de sintaxe é montada de baixo para cima (das folhas em direção à raiz);
- Caracteriza a classe das gramáticas (e linguagens) LR(k);
- Leitura do arquivo-fonte da esquerda para a direita;
- Emprego de derivações mais à direita (na ordem inversa), ou seja, reduções mais à esquerda (na ordem direta);
- Uso de no máximo k símbolos de lookahead.



# Definição de $\text{first}_k(\alpha)$

Seja  $G=(V, \Sigma, P, S)$  uma gramática livre de contexto,  $\alpha \in V^*$  e  $k$  inteiro.

$\text{first}_k(\alpha) =$

$\{w \in \Sigma^* \mid (\alpha \Rightarrow^* w \text{ e } |w| < k) \text{ ou } (\alpha \Rightarrow^* wx \text{ e } |w|=k \text{ para algum } x)\}$

$\text{first}_k(\alpha)$  é um conjunto de cadeias de símbolos terminais. Ele é formado por:

- prefixos de comprimento  $k$  de todas as cadeias que podem ser geradas a partir de  $\alpha$  pela aplicação das regras de  $G$ ;
- todas as cadeias de comprimento menor que  $k$  que são geradas por  $\alpha$  pela aplicação das regras de  $G$ .

# Gramática LL(k)

Seja  $G=(V, \Sigma, P, S)$  uma gramática livre de contexto.  $G$  é dita LL(k), para algum inteiro  $k$ , se, para quaisquer duas seqüências de derivações mais à esquerda:

1.  $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$
2.  $S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$

**tais que  $\text{first}_k(x) = \text{first}_k(y)$ , isso implicar  $\beta=\gamma$ .**

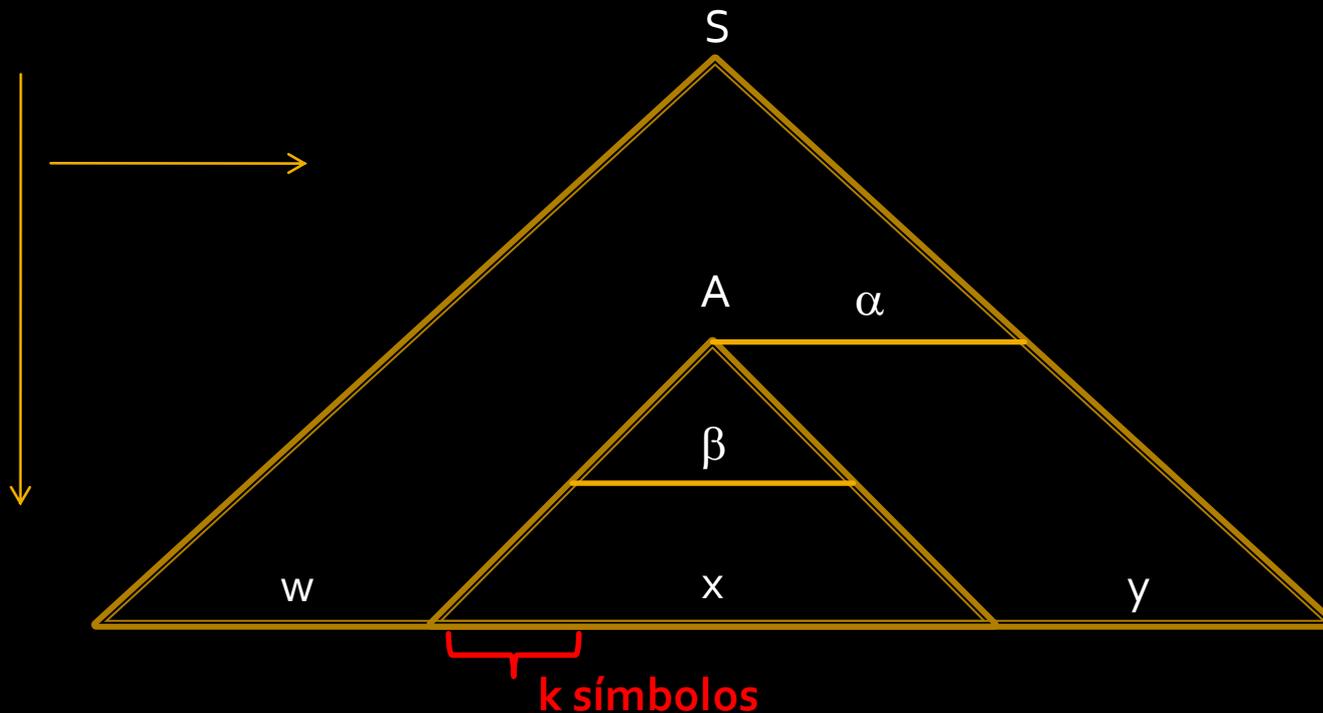
A escolha da derivação para o símbolo não-terminal  $A$  na forma sentencial  $wA\alpha$  é feita de maneira unívoca a partir da análise dos  $k$  primeiros símbolos terminais gerados por  $A\alpha$ .

Em outras palavras: sempre que houver uma única regra em  $G$  que permita gerar os  $k$  primeiros símbolos terminais de  $x$  a partir de  $A$  na forma sentencial  $wA\alpha$ .

# Gramática LL(k)

Considere a seqüência de derivações mais à esquerda e o uso da regra  $A \rightarrow \beta$ :

$S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx\alpha \Rightarrow^* wxy$   
( $w, x, y \in \Sigma^*$ ,  $S, A \in N$  e  $\alpha \in V^*$ )



# micro English

Sentence ::= Subject Verb Object .

Subject ::= I | a Noun | the Noun

Noun ::= rat | car | dog

Verb ::= is | see | sees | like

Object ::= me | a Noun | the Noun

# Análise descendente

the cat sees a dog.  
↑

Sentence  $\Rightarrow$  Subject Verb Object .

- O lookahead "the" é usado para selecionar a regra:  
Sentence  $\Rightarrow$  Subject Verb Object .

# Análise descendente

the cat sees a dog.



# Análise descendente

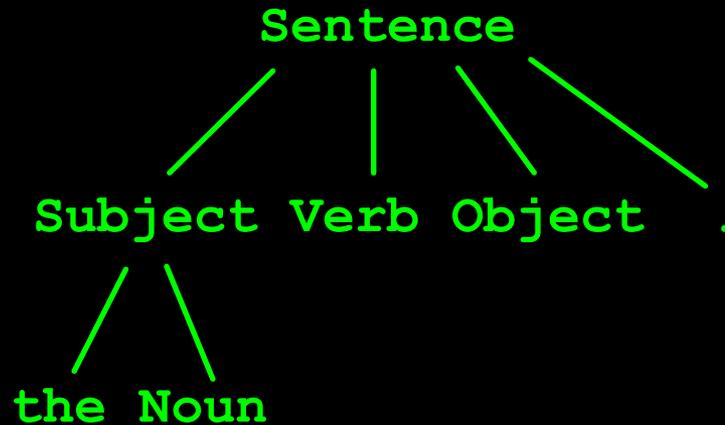
the cat sees a dog.  
↑

Sentence  $\Rightarrow$  Subject Verb Object  $\Rightarrow$  the Noun Verb  
Object .

- O lookahead "the" é usado para selecionar a regra:  
Subject  $\Rightarrow$  the Noun

# Análise descendente

the cat sees a dog.



# Análise descendente

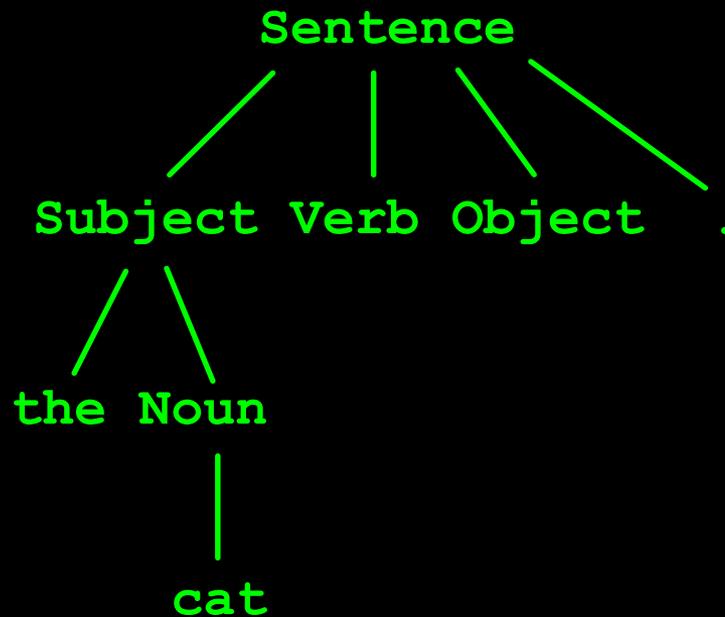
the cat sees a dog.  
↑

Sentence  $\Rightarrow$  Subject Verb Object  $\Rightarrow$  the Noun Verb  
Object .  $\Rightarrow$  the cat Verb Object .

- O lookahead "cat" é usado para selecionar a regra:  
Noun  $\Rightarrow$  cat

# Análise descendente

the cat sees a dog.  
↑



# Análise descendente

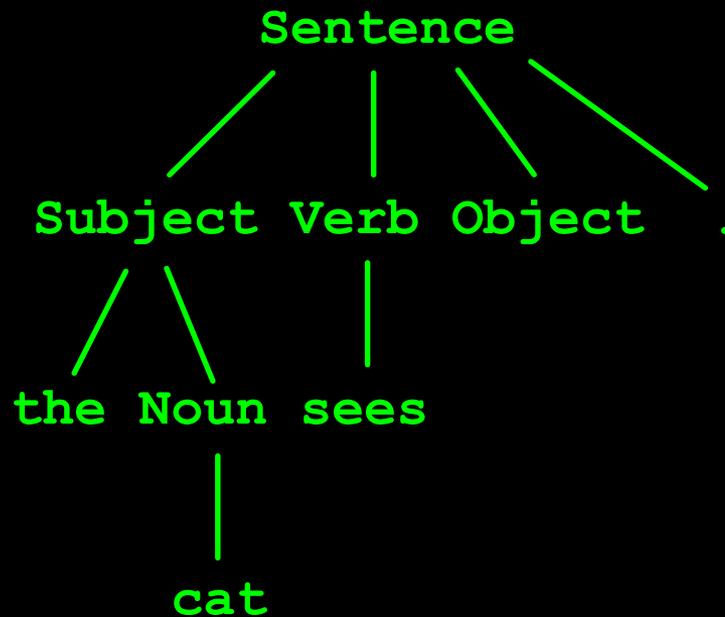
the cat sees a dog.  
          ↑

Sentence  $\Rightarrow$  Subject Verb Object  $\Rightarrow$  the Noun Verb  
Object .  $\Rightarrow$  the cat Verb Object .  $\Rightarrow$  the cat sees  
Object .

- O lookahead "sees" é usado para selecionar a regra:  
Verb  $\Rightarrow$  sees

# Análise descendente

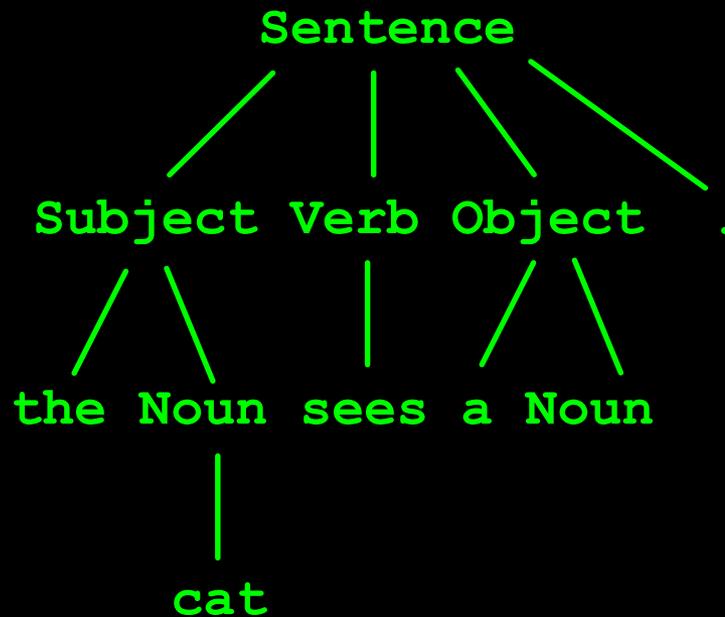
the cat sees a dog.





# Análise descendente

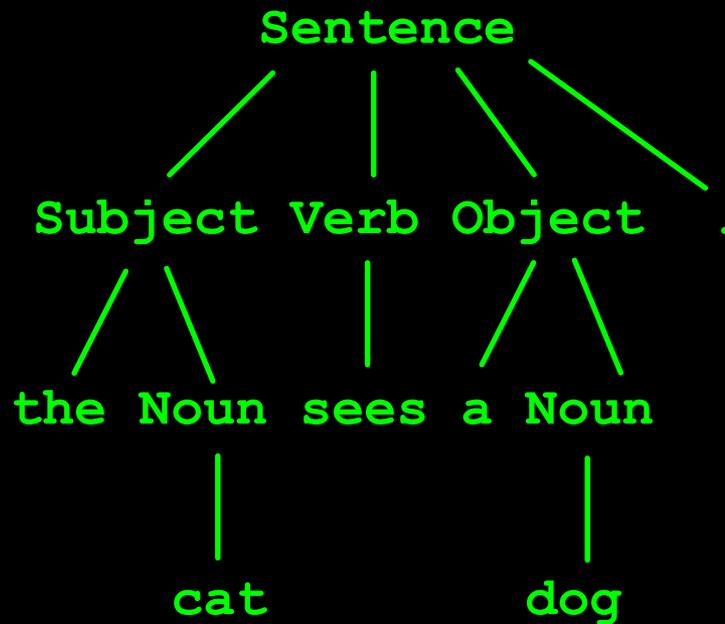
the cat sees a dog.  
↑





# Análise descendente

the cat sees a dog.  
↑



# Gramática LR(k)

Seja  $G=(V, \Sigma, P, S)$  uma gramática livre de contexto.  $G$  é dita LR(k), para algum inteiro  $k$ , se, para quaisquer duas seqüências de derivações mais à direita:

1.  $S \Rightarrow^* \alpha Aw \Rightarrow \alpha \beta w$
2.  $S \Rightarrow^* \gamma Bx \Rightarrow \alpha \beta y$

**tais que  $\text{first}_k(w) = \text{first}_k(y)$ , isso implicar  $\alpha=\gamma$ ,  $A=B$  e  $x=y$ .**

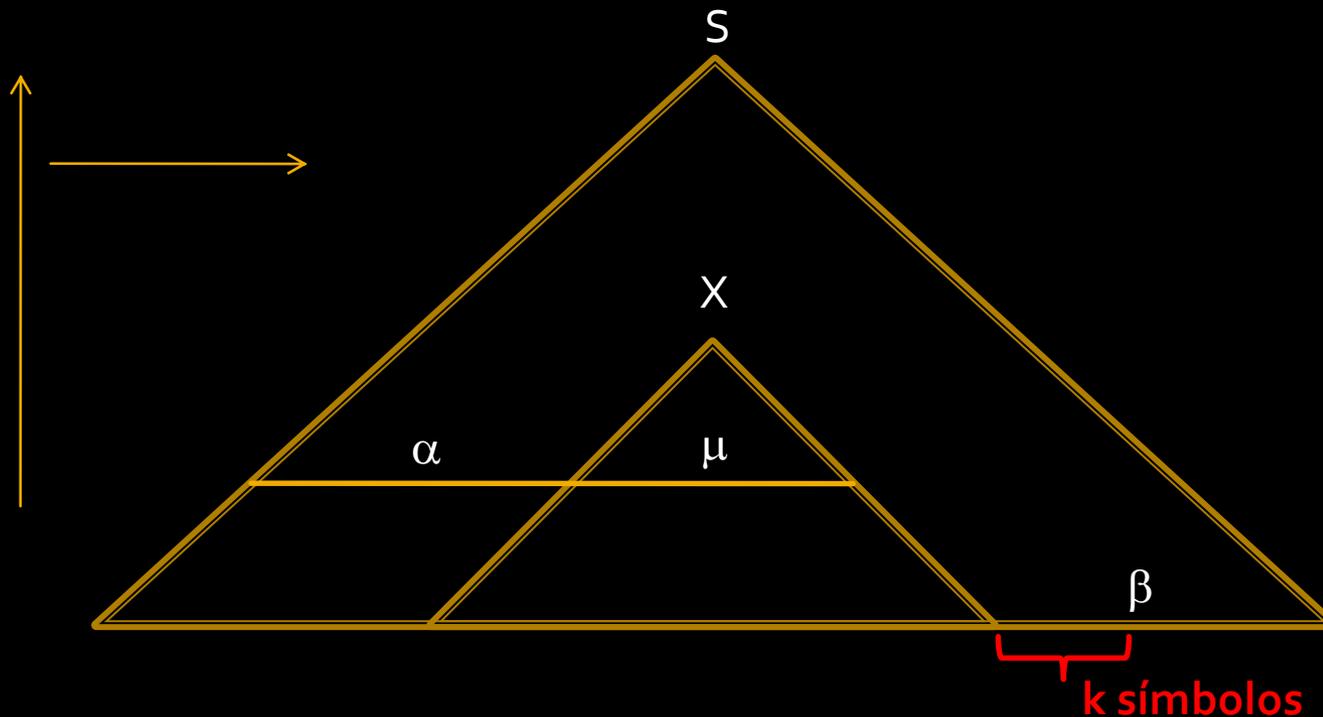
Uma vez identificada, a escolha da redução para a cadeia  $\beta$  é feita de maneira unívoca a partir da análise dos  $k$  primeiros símbolos terminais de  $w$ .

Em outras palavras: sempre que houver uma única regra em  $G$  que permite gerar os  $k$  primeiros símbolos terminais de  $w$  a partir de  $A$  na forma sentencial  $wA\alpha$ .

# Gramática LR(k)

Considere a seqüência de reduções mais à esquerda:

$$w \Rightarrow^* \alpha\mu\beta \Rightarrow \alpha X \beta \Rightarrow^* S \quad (\beta, \gamma \in \Sigma^*, S, X \in N \text{ e } \alpha, \mu \in V^*)$$



# Análise ascendente

the cat sees a dog.  
↑

- Nenhuma redução é possível com o símbolo "the" apenas;
- O cursor avançar para o próximo símbolo.

# Análise ascendente

the cat sees a dog.  
↑

- "cat" é reduzido para "Noun";
- O cursor avança para o próximo símbolo.

the cat sees a dog .  $\Rightarrow$  the Noun sees a dog .

# Análise ascendente

the cat sees a dog.  
↑

- "the Noun" pode ser reduzido para "Subject" ou para "Object";
- O lookahead "sees", no entanto, ocorre apenas depois de "Subject";
- Assim, a redução é feita para "Subject";
- O cursor avança para o próximo símbolo.

the cat sees a dog .  $\Rightarrow$  the Noun sees a dog .  $\Rightarrow$   
Subject sees a dog .

# Análise ascendente

the cat sees a dog.  
↑

- "sees" é reduzido para "Verb" ou para "Object";
- O cursor avança para o próximo símbolo.

the cat sees a dog . ⇒ the Noun sees a dog . ⇒  
Subject sees a dog . ⇒ Subject Verb a dog .

# Análise ascendente

the cat sees a dog.  
                  ↑

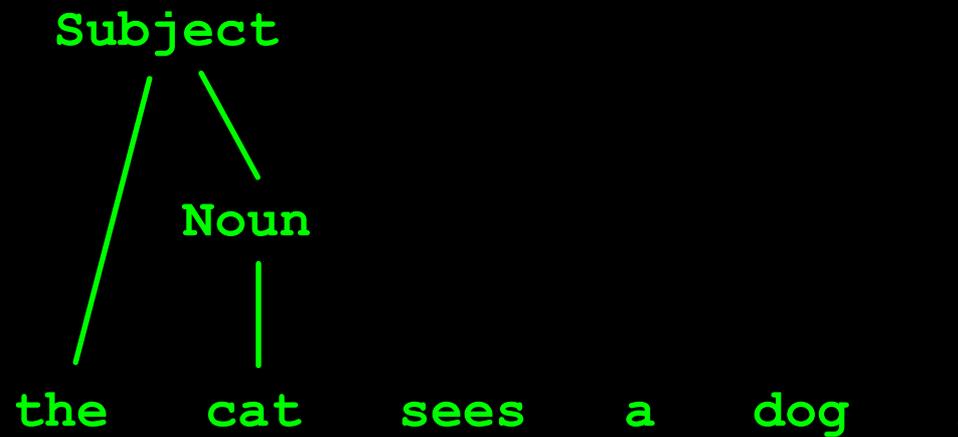
- "sees" é reduzido para "Verb" ou para "Object";
- O cursor avança para o próximo símbolo.

the cat sees a dog . ⇒ the Noun sees a dog . ⇒  
Subject sees a dog . ⇒ Subject Verb a dog .

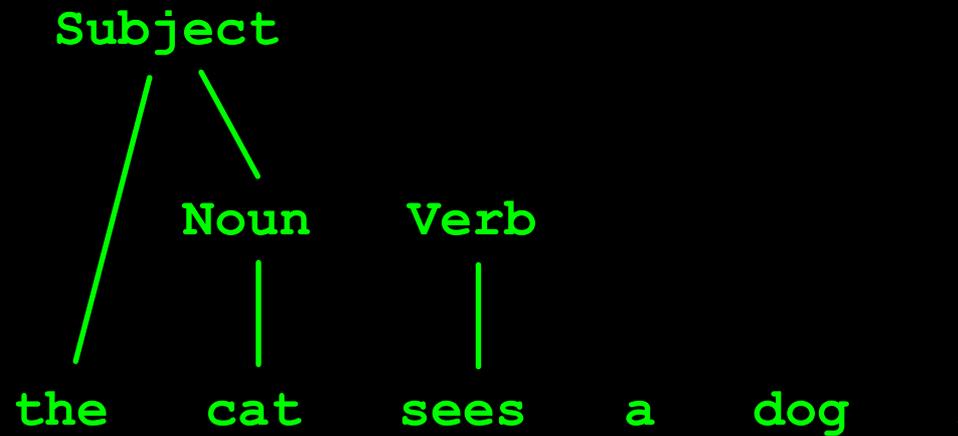
# Análise ascendente

Noun  
|  
the cat sees a dog .

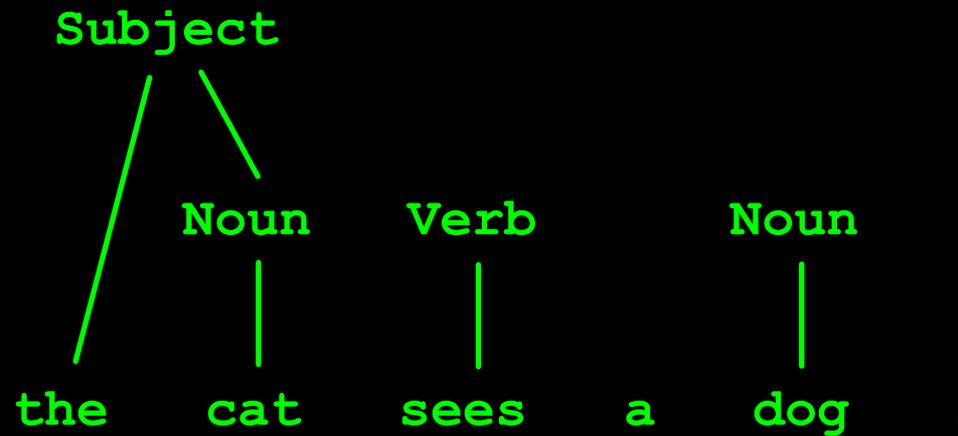
# Análise ascendente



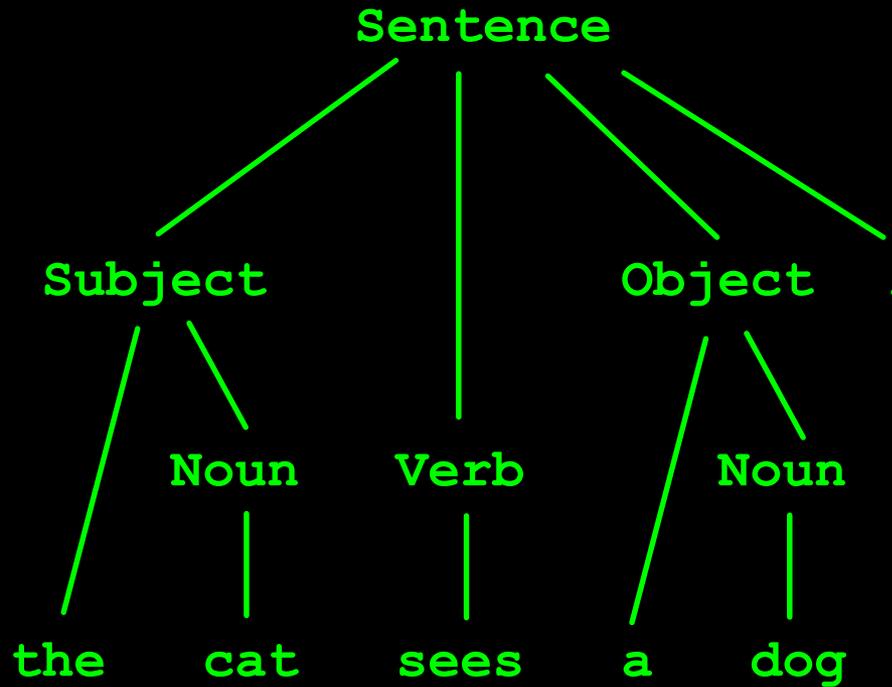
# Análise ascendente



# Análise ascendente



# Análise ascendente



# Resumo

## LL(k):

- Leitura da entrada da esquerda para a direita;
- Ordem direta das derivações mais à esquerda;

## LR(k):

- Leitura da entrada da esquerda para a direita;
- Ordem inversa das derivações mais à direita (ou ordem direta das reduções mais à esquerda);

## RR(k):

- Leitura da entrada da direita para a esquerda;
- Ordem direta das derivações mais à direita;

## RL(k):

- Leitura da entrada da direita para a esquerda;
- Ordem inversa das derivações mais à esquerda (ou ordem direta das reduções mais à direita).

# Ascendente x Descendente

LLC

LR(k)

LL(k)

# Ascendente x Descendente

- Gramática LL(k) é aquela que gera uma linguagem cujas sentenças podem ser analisadas de forma descendente, ou “top-down”;
- Gramática LR(k) é aquela que gera uma linguagem cujas sentenças podem ser analisadas de forma ascendente, ou “bottom-up”;
- Uma linguagem é dita LL(k) (ou LR(k)) se existir pelo menos uma gramática LL(k) (ou LR(k)) que a gere.

# Ascendente x Descendente

- Nem toda LLC pode ser analisada de forma determinística;
- O maior subconjunto das LLCs que podem ser analisadas de forma determinística corresponde ao conjunto das linguagens LR(k);
- Toda linguagem que é analisada de forma descendente pode também ser analisada de forma ascendente;
- Nem toda linguagem que é analisada de forma ascendente por ser analisada de forma descendente;
- As linguagens regulares formam um subconjunto próprio das linguagens LL(k);
- Gramáticas LL(k) e LR(k) são não-ambíguas.

# Implicações

Uma CFG  $G=(V, \Sigma, P, S)$  é dita LL(k) se e somente se:

Para cada forma sentencial  $wA\alpha$  tal que  $S \Rightarrow^* wA\alpha$  por meio do uso exclusivo de derivações mais à esquerda, se  $A \rightarrow \beta, A \rightarrow \gamma \in P$ , então  $\text{first}_k(\beta\alpha) = \text{first}_k(\gamma\alpha)$  implicar  $\beta=\alpha$ .

Em outras palavras, a escolha da substituição a ser aplicada ao símbolo não-terminal  $A$  pode ser feita de forma determinística levando-se em conta a configuração corrente do reconhecedor ( $wA\alpha$ ) e os  $k$  primeiros símbolos terminais gerados pela cadeia  $A\alpha$ .

# Estratégias para verificar a condição LL(k)

Com base na definição anterior:

- Suponha  $A \rightarrow \beta_1 \mid \dots \mid \beta_n$
- Determinar todas as formas sentenciais  $wA\alpha$  em que o não-terminal  $A$  possa comparecer;
- Verificar, para cada uma delas, se  $\text{first}_k(\beta_i\alpha) \cap \text{first}_k(\beta_j\alpha) = \emptyset$  para  $1 \leq i, j \leq n, i \neq j$ .

# Estratégias para verificar a condição $LL(k)$

Dificuldades decorrentes:

- Determinar todas as formas sentenciais  $wA\alpha$  em que  $A$  possa comparecer;
- Determinar  $\text{first}_k(\beta_i\alpha)$ , especialmente quando  $\beta_i$  não gera cadeias de terminais de comprimento  $k$ .

# Estratégias para verificar a condição LL(k)

Exemplo:

- $S \rightarrow aX \mid bX$
- $X \rightarrow cX \mid d$

Considere a derivação do não-terminal X.

Em quais formas sentenciais ele comparece?

Resposta:

$aX, bX, acX, bcX, accX, bccX, acccX, bccccX$  etc.

# Estratégias para verificar a condição LL(k)

- $aX$ :  
Se  $X \rightarrow cX$ , então  $aX \Rightarrow acX$  e  $\text{first}_1(cX) = \{c\}$   
Se  $X \rightarrow d$ , então  $aX \Rightarrow ad$  e  $\text{first}_1(d) = \{d\}$   
Como  $\{c\} \cap \{d\} = \emptyset$ , então a condição LL(1) é válida para a forma sentencial  $aX$ ;
- $bX$ :  
Se  $X \rightarrow cX$ , então  $bX \Rightarrow bcX$  e  $\text{first}_1(cX) = \{c\}$   
Se  $X \rightarrow d$ , então  $bX \Rightarrow bd$  e  $\text{first}_1(d) = \{d\}$   
Como  $\{c\} \cap \{d\} = \emptyset$ , então a condição LL(1) é válida para a forma sentencial  $bX$ ;

# Estratégias para verificar a condição LL(k)

- Situações idênticas acontecem com todas as demais formas sentenciais ( $acX$ ,  $bcX$ ,  $accX$ ,  $bccX$ ,  $acccX$ ,  $bccccX$  etc);
- Portanto:
  - Forma sentencial  $aX$ , não-terminal  $X$ , símbolo corrente  $c$ : deve-se escolher a regra  $X \rightarrow cX$
  - Forma sentencial  $aX$ , não-terminal  $X$ , símbolo corrente  $d$ : deve-se escolher a regra  $X \rightarrow d$
  - Forma sentencial  $bX$ , não-terminal  $X$ , símbolo corrente  $c$ : deve-se escolher a regra  $X \rightarrow cX$
  - Forma sentencial  $bX$ , não-terminal  $X$ , símbolo corrente  $d$ : deve-se escolher a regra  $X \rightarrow d$
  - etc.

# Estratégias para verificar a condição LL(k)

- Por outro lado, cabe observar que a escolha da primeira regra ( $X \rightarrow cX$ ) gera cadeias que começam sempre pelo símbolo "c", independentemente da forma sentencial considerada; da mesma forma, a escolha da segunda regra ( $X \rightarrow d$ ) gera cadeias que iniciam com "d";
- De fato, na lista anterior pode-se perceber que a forma sentencial corrente é irrelevante para se tomar a decisão correta sobre a regra que deve ser aplicada ao não-terminal A;
- Isso sugere para esse caso, portanto, uma simplificação do processo, desconsiderando a forma sentencial corrente e levando em conta apenas o símbolo não-terminal que está sendo derivado e o lookahead.

# Estratégias para verificar a condição LL(k)

Ou seja (válido para esse caso apenas):

A escolha da regra  $X \rightarrow cX$  poderá ser feita sempre que o símbolo corrente for "c", assim como a escolha será pela regra  $X \rightarrow d$  quando o símbolo corrente for "d", sem precisar levar em conta a forma sentencial em que A está sendo derivado.

# Estratégias para verificar a condição LL(k)

Exemplo:

- $S \rightarrow aAaB \mid bAbB$
- $A \rightarrow a \mid ab$
- $B \rightarrow aB \mid a$

Considere a derivação do não-terminal A.

Em quais formas sentenciais ele comparece?

Resposta:

aAaB e bAbB.

# Estratégias para verificar a condição LL(k)

- $aAaB$ :  
Se  $A \rightarrow a$ , então  $aAaB \Rightarrow aaaB$  e  $\text{first}_1(aaB) = \{a\}$   
Se  $A \rightarrow ab$ , então  $aAaB \Rightarrow aabaB$  e  $\text{first}_1(abaB) = \{a\}$   
Como  $\{a\} \cap \{a\} \neq \emptyset$ , então a condição LL(1) não é válida para a forma sentencial  $aAaB$ ;
- $bAbB$  :  
Se  $A \rightarrow a$ , então  $bAbB \Rightarrow babB$  e  $\text{first}_1(abB) = \{a\}$   
Se  $A \rightarrow ab$ , então  $bAbB \Rightarrow babbB$  e  $\text{first}_1(abbB) = \{a\}$   
Como  $\{a\} \cap \{a\} \neq \emptyset$ , então a condição LL(1) não é válida para a forma sentencial  $bAbB$ ;

# Estratégias para verificar a condição LL(k)

- $aAaB$ :  
Se  $A \rightarrow a$ , então  $aAaB \Rightarrow aaaB$  e  $\text{first}_2(aaB) = \{aa\}$   
Se  $A \rightarrow ab$ , então  $aAaB \Rightarrow aabaB$  e  $\text{first}_2(abaB) = \{ab\}$   
Como  $\{aa\} \cap \{ab\} = \emptyset$ , então a condição LL(2) é válida para a forma sentencial  $aAaB$ ;
- $bAbB$  :  
Se  $A \rightarrow a$ , então  $bAbB \Rightarrow babB$  e  $\text{first}_2(abB) = \{ab\}$   
Se  $A \rightarrow ab$ , então  $bAbB \Rightarrow babbB$  e  $\text{first}_1(abbB) = \{ab\}$   
Como  $\{ab\} \cap \{ab\} \neq \emptyset$ , então a condição LL(2) não é válida para a forma sentencial  $bAbB$ ;

# Estratégias para verificar a condição LL(k)

- $bAbB$  :  
Se  $A \rightarrow a$ , então  $bAbB \Rightarrow babB$  e  $\text{first}_3(abB) = \{aba\}$   
Se  $A \rightarrow ab$ , então  $bAbB \Rightarrow babbB$  e  $\text{first}_1(abbB) = \{abb\}$   
Como  $\{aba\} \cap \{abb\} = \emptyset$ , então a condição LL(3) é válida para a forma sentencial  $bAbB$ ;
- A derivação do não-terminal  $A$  requer o lookahead de, no máximo, 3 símbolos.

# Estratégias para verificar a condição LL(k)

Em resumo:

- $aAaB$  com  $A \rightarrow a$   
 $\{aa\}$  (para  $k=2$ ) ou  $\{aaa\}$  (para  $k=3$ )
- $aAaB$  com  $A \rightarrow ab$   
 $\{ab\}$  (para  $k=2$ ) ou  $\{aba\}$  (para  $k=3$ )
- $bAbB$  com  $A \rightarrow a$   
 $\{aba\}$
- $bAbB$  com  $A \rightarrow ab$   
 $\{abb\}$

# Estratégias para verificar a condição LL(k)

Seria possível desconsiderar a forma sentencial corrente nesse caso?

- aAaB com  $A \rightarrow a$   
{aaa}
  - aAaB com  $A \rightarrow ab$   
{aba}
  - bAbB com  $A \rightarrow a$   
{aba}
  - bAbB com  $A \rightarrow ab$   
{abb}
- 
- The diagram illustrates the mapping from grammar rules to language sets. Red arrows indicate the following connections:
- From the first rule (aAaB,  $A \rightarrow a$ ) to the set {aaa, aba}.
  - From the second rule (aAaB,  $A \rightarrow ab$ ) to the set {aba, abb}.
  - From the third rule (bAbB,  $A \rightarrow a$ ) to the set {aba}.
  - From the fourth rule (bAbB,  $A \rightarrow ab$ ) to the set {abb}.

# Estratégias para verificar a condição LL(k)

- Como  $\{aaa, aba\} \cap \{aba, abb\} \neq \emptyset$ , nesse caso não seria possível fazer uma escolha determinística da regra a ser aplicada, sem levar em consideração a forma sentencial corrente.
- Uma alternativa seria verificar se a condição LL(k) seria verificada para valores de  $k \geq 4$ , sem levar em conta a informação de forma sentencial corrente.

# Estratégias para verificar a condição LL(k)

k=4

- aAaB com A  $\rightarrow$  a  
{aaa $\downarrow$ }
  - aAaB com A  $\rightarrow$  ab  
{abaa}
  - bAbB com A  $\rightarrow$  a  
{abaa, aba $\downarrow$ }
  - bAbB com A  $\rightarrow$  ab  
{abba}
- 
- Diagram illustrating the mapping of LL(k) items to their look-ahead strings for k=4:
- Item: aAaB com A  $\rightarrow$  a, {aaa $\downarrow$ } maps to look-ahead string: {aaa $\downarrow$ , **abaa**, aba $\downarrow$ }
  - Item: aAaB com A  $\rightarrow$  ab, {abaa} maps to look-ahead string: {**abaa**, abba}
  - Item: bAbB com A  $\rightarrow$  a, {abaa, aba $\downarrow$ } maps to look-ahead string: {aaa $\downarrow$ , **abaa**, aba $\downarrow$ }
  - Item: bAbB com A  $\rightarrow$  ab, {abba} maps to look-ahead string: {**abaa**, abba}

# Estratégias para verificar a condição LL(k)

k=5

- aAaB com A  $\rightarrow$  a   
{aaa $\downarrow\downarrow$ , aaaa $\downarrow$ , aaaaa,  
{aaa $\downarrow\downarrow$ , aaaa $\downarrow$ , aaaaa}}
  - aAaB com A  $\rightarrow$  ab   
{abaa $\downarrow$ , abaaa}
  - bAbB com A  $\rightarrow$  a   
{aba $\downarrow\downarrow$ , abaa $\downarrow$ , abaaa}
  - bAbB com A  $\rightarrow$  ab   
{abba $\downarrow$ , abbaa}
- {aaa $\downarrow\downarrow$ , aaaa $\downarrow$ , aaaaa,  
aba $\downarrow\downarrow$ , **abaa $\downarrow$** , **abaaa**}
- {**abaa $\downarrow$** , **abaaa**,  
abba $\downarrow$ , abbaa}

# Estratégias para verificar a condição LL(k)

Obtendo uma gramática equivalente LL(4):

$A \rightarrow a \mid ab$

$B \rightarrow aa^*$

$S \rightarrow a(a \mid ab)aaa^* \mid b(a \mid ab)baa^*$

$S \rightarrow (aa \mid aab)aaa^* \mid (ba \mid bab)baa^*$

$S \rightarrow aaaaa^* \mid aabaaa^* \mid babaa^* \mid babbaa^*$

$\text{first}_4(aaaaa^*) = \{aaaa\}$

$\text{first}_4(aabaaa^*) = \{aaba\}$

$\text{first}_4(babaa^*) = \{baba\}$

$\text{first}_4(babbaa^*) = \{babbb\}$

# Estratégias para verificar a condição LL(k)

Obtendo uma gramática equivalente LL(1):

$$S \rightarrow aaaaa^* \mid aabaaa^* \mid babaa^* \mid babbaa^*$$
$$S \rightarrow a (aaaa^* \mid abaaa^*) \mid b (abaa^* \mid abbaa^*)$$
$$S \rightarrow a (a (aaa^* \mid baaa^*)) \mid b (a (baa^* \mid bbaa^*))$$
$$S \rightarrow a (a (aaa^* \mid baaa^*)) \mid b (a (b (aa^* \mid baa^*)))$$

# Estratégias para verificar a condição LL(k)

Quando então usar ou não usar a informação da forma sentencial corrente para fazer a escolha da regra a ser aplicada?

- Considerar o conjunto de formas sentenciais em que A comparece:  $\alpha_i A \gamma_j$ ,  $1 \leq i \leq m$ ,  $1 \leq j \leq n$
- Considerar  $A \rightarrow \beta_1 \mid \dots \mid \beta_k$
- Seja  $X_1 = \text{first}_1(\beta_1 \gamma_1) \cup \dots \cup \text{first}_1(\beta_1 \gamma_m)$
- ...
- Seja  $X_k = \text{first}_1(\beta_k \gamma_1) \cup \dots \cup \text{first}_1(\beta_k \gamma_m)$
- Se  $X_i \cap X_j = \emptyset$ ,  $1 \leq i, j \leq k$ ,  $i \neq j$ , então a forma sentencial é irrelevante para a tomada de decisão.

# Definição de $\text{follow}_k(\beta)$

Seja  $G=(V, \Sigma, P, S)$  uma gramática livre de contexto,  $\beta \in V^*$  e  $k$  inteiro.

$$\text{follow}_k(\beta) = \{w \mid S \Rightarrow^* \alpha\beta\gamma \text{ e } w \in \text{first}_k(\gamma)\}$$

$\text{follow}_k(\beta)$  é o conjunto das cadeias de símbolos terminais que aparecem imediatamente à direita da cadeia  $\gamma$ , consideradas todas as formas sentenciais geradas por  $G$  em que  $\gamma$  faça parte.

# Estratégias

Com base na definição anterior:

- $A \rightarrow \beta_1 \mid \dots \mid \beta_n$
- Verificar se  $\text{first}_k(\beta_i \cdot \text{follow}_k(A)) \cap \text{first}_k(\beta_j \cdot \text{follow}_k(A)) = \emptyset, 1 \leq i, j \leq n, i \neq j.$

# Casos particulares

## Gramáticas LL(1) simples:

- Não existem regras vazias;
- Todas as regras começam com um símbolo terminal;
- As regras de um mesmo não-terminal iniciam com símbolos terminais distintos.

$A \rightarrow \sigma_1 \alpha_1 \mid \sigma_2 \alpha_2 \mid \dots \mid \sigma_n \alpha_n$   
com  $\sigma_i \neq \sigma_j$  para  $i \neq j$  e  $\sigma_i \in \Sigma$ ,  $1 \leq i \leq n$ .

# Exemplo

Gramáticas LL(1) simples:

- $S \rightarrow aS$
- $S \rightarrow bA$
- $A \rightarrow d$
- $A \rightarrow ccA$
  
- S:  
 $\text{first}_1(aS) \cap \text{first}_1(bA) = \{a\} \cap \{b\} = \emptyset$
- A:  
 $\text{first}_1(d) \cap \text{first}_1(ccA) = \{d\} \cap \{c\} = \emptyset$

# Casos particulares

Gramáticas LL(1) sem regras vazias:

- $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$   
 $\text{first}_1(\alpha_i) \cap \text{first}_1(\alpha_j) = \emptyset, i \neq j.$

# Exemplo

Gramáticas LL(1) sem regras vazias:

- $S' \rightarrow S\#$
- $S \rightarrow ABe$
- $A \rightarrow dB \mid aS \mid c$
- $B \rightarrow AS \mid b$
  
- $S'$  :  
 $first_1(S\#) = first_1(ABe\#) = \{a, c, d\}$
- $S$  :  
 $first_1(ABe) = \{a, c, d\}$
- $A$  :  
 $first_1(dB) = \{d\}, first_1(aS) = \{a\}, first_1(c) = \{c\}$
- $B$  :  
 $first_1(AS) = first_1(dBS) \cup first_1(aSS) \cup first_1(cS) = \{d, a, c\}$   
 $first_1(b) = \{b\}$

# Caso geral

Gramáticas LL(1) com regras vazias:

- $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$   
 $\text{first}_1(\alpha_i.\text{follow}(A)) \cap \text{first}_1(\alpha_j.\text{follow}(A)) = \emptyset, i \neq j.$

ou ainda:

- $\text{first}_1(\alpha_i) \cap \text{first}_1(\alpha_j) = \emptyset, i \neq j$ , e, se  $\alpha_i \Rightarrow^* \epsilon$ , então  
 $\text{first}_1(\alpha_j) \cap \text{follow}_1(A) = \emptyset, i \neq j.$

Se o símbolo corrente pertence ao  $\text{follow}_1(A)$ , a regra  $A \rightarrow \epsilon$  deve ser a escolhida.

# Exemplo

Gramáticas LL(1) com regras vazias:

- $S' \rightarrow A\#$
  - $A \rightarrow iB \leftarrow e$
  - $B \rightarrow SB \mid \varepsilon$
  - $S \rightarrow [eC] \mid .i$
  - $C \rightarrow eC \mid \varepsilon$
- 
- $S'$  :  
 $\text{first}_1(A\#) = \{i\}$
  - $A$  :  
 $\text{first}_1(iB \leftarrow e) = \{i\}$
  - $B$  :  
 $\text{first}_1(SB) \cap \text{follow}_1(B) = \{[, .\} \cap \{\leftarrow\} = \emptyset$
  - $S$  :  
 $\text{first}_1([eC]) \cap \text{first}_1(.i) = \{[\} \cap \{.\} = \emptyset$
  - $C$  :  
 $\text{first}_1(eC) \cap \text{follow}_1(C) = \{e\} \cap \{[]\} = \emptyset$

# Recursão à esquerda

Suponha que  $\sigma_1$  comparece à direita de  $X$  nas formas sentenciais geradas por  $G$ , sem uso da recursão à esquerda, e que  $X$  seja recursivo à esquerda:

$$S \rightarrow X \sigma_1$$
$$X \rightarrow X \sigma_2 \mid \sigma_3$$

As formas sentenciais em que  $X$  pode comparecer são:

$$X \sigma_1$$
$$X \sigma_2 \sigma_1$$
$$X \sigma_2 \sigma_2 \sigma_1$$
$$X \sigma_2 \sigma_2 \sigma_2 \sigma_1$$
$$X \sigma_2 \sigma_2 \sigma_2 \sigma_2 \sigma_1$$

...

Qualquer que seja o caso (genericamente,  $X \sigma_2^{k-1} \sigma_1$ ), e qualquer que seja o valor de  $k$ , não é possível escolher de forma unívoca uma única substituição para  $X$  ( $X \sigma_2$  ou  $\sigma_3$ ), pois as formas sentenciais resultantes  $X \sigma_2 \sigma_2^{k-1} \sigma_1$  e  $\sigma_3 \sigma_2^{k-1} \sigma_1$  possuem sempre a cadeia  $\sigma_3 \sigma_2^{k-1}$  como elemento comum dos conjuntos  $\text{first}_k()$  das mesmas.

# Recursão à esquerda

Senão vejamos:

- Suponha que  $k=1$ . Então, se a forma sentencial corrente for  $X \sigma_1$  e o lookahead for  $\sigma_3$ , não será possível determinar a regra a ser aplicada. As novas formas sentenciais nestes casos serão, respectivamente,  $X \sigma_2 \sigma_1$  e  $\sigma_3 \sigma_1$ , e  $\sigma_3$  é o elemento comum aos dois conjuntos  $\text{first}_1$ ;
- Suponha que  $k=2$ . Então, se a forma sentencial corrente for  $X \sigma_2 \sigma_1$  e o lookahead for  $\sigma_3 \sigma_2$ , não será possível determinar a regra a ser aplicada. As novas formas sentenciais nestes casos serão, respectivamente,  $X \sigma_2 \sigma_2 \sigma_1$  e  $\sigma_3 \sigma_2 \sigma_1$ , e  $\sigma_3 \sigma_2$  é o elemento comum aos dois conjuntos  $\text{first}_2$ ;
- Suponha que  $k=3$ . Então, se a forma sentencial corrente for  $X \sigma_2 \sigma_2 \sigma_1$  e o lookahead for  $\sigma_3 \sigma_2 \sigma_2$ , não será possível determinar a regra a ser aplicada. As novas formas sentenciais nestes casos serão, respectivamente,  $X \sigma_2 \sigma_2 \sigma_2 \sigma_1$  e  $\sigma_3 \sigma_2 \sigma_2 \sigma_1$ , e  $\sigma_3 \sigma_2 \sigma_2$  é o elemento comum aos dois conjuntos  $\text{first}_3$ ;
- E assim por diante, para qualquer valor de  $k$  que se considere.

# Recursão à esquerda

$S \rightarrow Xc$

$X \rightarrow Xb \mid a$

Suponha que  $X$  é o não-terminal mais à esquerda que deve ser derivado em  $\alpha X \beta$ :

Lookahead (k)	Primeiros (k) símbolos possíveis de $X\beta$	Regras possíveis para o caso <b>vermelho</b>
1	<b>a</b>	$X \rightarrow Xb$ ou $X \rightarrow a$
2	ac, <b>ab</b>	$X \rightarrow Xb$ ou $X \rightarrow a$
3	ac, abc, <b>abb</b>	$X \rightarrow Xb$ ou $X \rightarrow a$
4	ac, abc, abbc, <b>abbb</b>	$X \rightarrow Xb$ ou $X \rightarrow a$
...	...	...
n	..., <b><math>ab^{n-1}</math></b> , ...	$X \rightarrow Xb$ ou $X \rightarrow a$

# Recursão à esquerda

$S \rightarrow Xc$

$X \rightarrow Xb \mid a$

Suponha que  $X$  é o não-terminal mais à esquerda que deve ser derivado:

- $k=1$  (**a**)
- $k=2$  (ac ou **ab**)
- $k=3$  (ac, abc ou **abb**)
- $k=4$  (ac, abc, abbc ou **abbbb**)
- $k=5$  (ac, abc, abbc, abbbc ou **abbbbb**)
- $k=6$  (ac, abc, abbc, abbbc, abbbbc ou **abbbbbb**)
- $k=7$  (ac, abc, abbc, abbbc, abbbbc, abbbbbc ou **abbbbbbb**)

Sempre que os  $k$  próximos símbolos forem da forma “ $ab^{k-1}$ ”, não será possível escolher de forma unívoca a aplicação da regra  $X \rightarrow Xb$  ou da regra  $X \rightarrow a$ , pois não se sabe se o próximo símbolo é um “c” ou um “b”.

# Recursão à esquerda

$S \rightarrow Xc$

$X \rightarrow Xb \mid a$

Em outras palavras:

- Qualquer que seja o valor de  $k$ , será sempre possível se deparar com um lookahead  $ab^{k-1}$ , de modo que não será possível determinar de forma unívoca a regra a ser aplicada ( $X \rightarrow Xb$  ou  $X \rightarrow a$ ) ao não-terminal  $X$  em questão;
- Sempre existe um lookahead cujo comprimento é maior do que o valor de  $k$  considerado, o qual impede que a escolha seja feita de forma determinística;
- Não é possível fazer uma análise determinística em todos os casos e a gramática não é LL( $k$ ) para nenhum valor de  $k$ .

# Recursão à esquerda

Gramáticas com recursão à esquerda não são LL(k).

A eliminação das recursões à esquerda pode permitir a obtenção de uma gramática LL(k), mas o resultado não é garantido:

$$S \rightarrow Xc$$

$$X \rightarrow Xb \mid a$$

$$S \rightarrow Xc$$

$$X \rightarrow aY$$

$$Y \rightarrow bY \mid \varepsilon$$

ou simplesmente:

$$S \rightarrow ab^*c$$

# Gramática LL(1)

Considere a GLC  $G$ :

$$X_1 \rightarrow \gamma_{11} \mid \gamma_{12} \mid \dots \mid \gamma_{1m}$$

$$X_2 \rightarrow \gamma_{21} \mid \gamma_{22} \mid \dots \mid \gamma_{2n}$$

...

$$X_p \rightarrow \gamma_{p1} \mid \gamma_{p2} \mid \dots \mid \gamma_{pq}$$

$$\text{first}_1(\gamma_{1i}) \cap \text{first}_1(\gamma_{1j}) = \emptyset, 1 \leq i, j \leq m, i \neq j.$$

$$\text{first}_1(\gamma_{2i}) \cap \text{first}_1(\gamma_{2j}) = \emptyset, 1 \leq i, j \leq n, i \neq j.$$

...

$$\text{first}_1(\gamma_{pi}) \cap \text{first}_1(\gamma_{pj}) = \emptyset, 1 \leq i, j \leq q, i \neq j.$$

e, além disso, se  $\gamma_{ij} \Rightarrow^* \varepsilon$ , então:

$$\text{first}_1(\gamma_{ik}) \cap \text{follow}_1(X_i) = \emptyset, \forall k \neq j.$$

# Outros exemplos (I)

- $S \rightarrow bS$
- $S \rightarrow bA$
- $A \rightarrow d$
- $A \rightarrow ccA$

Não é LL(1) mas é LL(2):

- S:  
 $first_2(bS) \cap first_2(bA) = \{bb\} \cap \{bd, bc\} = \emptyset$
- A:  
 $first_1(d) \cap first_1(ccA) = \{d\} \cap \{c\} = \emptyset$

# Outros exemplos (I)

Pode ser convertida na gramática LL(1) equivalente usando fatoração à esquerda:

- $S \rightarrow b(S|A)$
- $A \rightarrow d$
- $A \rightarrow ccA$
  
- S:  
 $first_1(b(S|A)) = \{b\}$
- ( ):  
 $first_1(S) \cap first_1(A) = \{b\} \cap \{c, d\} = \emptyset$
- A:  
 $first_1(d) \cap first_1(ccA) = \{d\} \cap \{c\} = \emptyset$

O parêntesis representa um não-terminal implícito (X):

- $S \rightarrow bX$
- $X \rightarrow S|A$
- $A \rightarrow d|ccA$

# Outros exemplos (II)

- $S \rightarrow aX$
- $S \rightarrow aY$
- $X \rightarrow bX \mid c$
- $Y \rightarrow dY \mid e$

Não é LL(1) mas é LL(2):

- S:  
 $\text{first}_2(aX) \cap \text{first}_2(aY) = \{ab, ac\} \cap \{ad, ae\} = \emptyset$
- X:  
 $\text{first}_1(bX) \cap \text{first}_1(c) = \{b\} \cap \{c\} = \emptyset$
- Y:  
 $\text{first}_1(dY) \cap \text{first}_1(e) = \{d\} \cap \{e\} = \emptyset$

# Outros exemplos (II)

Fatorando à esquerda e agrupando:

- $S \rightarrow a(X|Y)$
- $X \rightarrow bX | c$
- $A \rightarrow dY | e$

Torna-se LL(1), pois:

- $() :$   
 $first_1(X) \cap first_1(Y) = \{b, c\} \cap \{d, e\} = \emptyset$

# Outros exemplos (III)

- $S \rightarrow aXe$
- $X \rightarrow bXY \mid c \mid \varepsilon$
- $Y \rightarrow dY \mid c$

Não é LL(1), pois:

- X:  
 $\text{first}_1(bXY) = \{b\}$   
 $\text{first}_1(c) = \{c\}$   
 $\text{follow}_1(X) = \{c, d, e\}$

# Outros exemplos (III)

Pode ser convertida para LL(1) através da manipulação:

- $S \rightarrow a (bXd^*ce \mid ce \mid e)$

Pois:

- $() :$   
 $first_1(bXd^*ce) = \{b\}$   
 $first_1(ce) = \{c\}$   
 $first_1(e) = \{e\}$

# Outros exemplos (IV)

- $S \rightarrow aXe$
- $X \rightarrow bXY \mid c \mid \varepsilon$
- $Y \rightarrow dY \mid f$

É LL(1), pois:

- X:  
 $\text{first}_1(bXY) = \{b\}$   
 $\text{first}_1(c) = \{c\}$   
 $\text{follow}_1(X) = \{f, d, e\}$

# Objetivo

- Dada uma gramática qualquer, verificar se a mesma é LL(1);
- Em caso negativo, tentar obter uma gramática LL(1) equivalente;
- Uma vez obtida a gramática LL(1) equivalente, usar um método para construção sistemática do analisador sintático;
- Em caso de insucesso, verificar se a mesma é LR(k) e aplicar os métodos correspondentes.



# Conversão

Conversão:

- Fatorações à esquerda;
- Substituições;
- Eliminação de recursões à esquerda.

# Exercício

Provar que não é LL(1):

- $S' \rightarrow S\#$
- $S \rightarrow aAa \mid \varepsilon$
- $A \rightarrow abS \mid c$

# Exercício

Provar que é LL(1):

- $S \rightarrow A\#$
- $A \rightarrow Bb \mid Cd$
- $B \rightarrow aB \mid \varepsilon$
- $C \rightarrow cC \mid \varepsilon$

# Exercício

Provar que é LL(1):

- $S' \rightarrow S\#$
- $S \rightarrow aABC$
- $A \rightarrow a \mid bbD$
- $B \rightarrow a \mid \varepsilon$
- $C \rightarrow b \mid \varepsilon$
- $D \rightarrow c \mid \varepsilon$

# Exercício

Provar que é LL(1):

- $S' \rightarrow S\#$
- $S \rightarrow AB$
- $A \rightarrow a \mid \varepsilon$
- $B \rightarrow b \mid \varepsilon$