Conceitos de Análise Sintática

Pro.f. Marcus Ramos

UNIVASF

Atualizado em 21 de dezembro de 2017 às 14:18

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

Derivações e Reduções

- Regras gramaticais possuem o formato geral $\alpha \rightarrow \beta$
- **DERIVAÇÃO**: subsituição do lado esquerdo de uma regra gramatical (α) pelo lado direito correspondente (β) na forma sentencial corrente;
- **REDUÇÃO**: subsituição do lado direito de uma regra gramatical (β) pelo lado esquerdo (α) correspondente na forma sentencial corrente;

$$\alpha \rightarrow \beta$$

Exemplo

- Considere a gramática:
  - $S \rightarrow XY$
  - $X \rightarrow aX \mid b$
  - $Y \rightarrow cY \mid d$
- Considere a forma sentencial $aXcY$
- Exemplo de derivação: $a\boldsymbol{X}cY \Rightarrow a\boldsymbol{aX}cY$
- Exemplo de redução: $\boldsymbol{aX}cY \Rightarrow \boldsymbol{X}cY$
- Em ambos os casos, foi usada a regra $X \rightarrow aX$.

Derivações mais à esquerda

- Uma seqüência de derivações é dita "mais à esquerda" quando todas as derivações são feitas sempre sobre os símbolos não-terminais situados mais à esquerda na forma sentencial corrente;
- Exemplo (usando a gramática anterior):

$$\boldsymbol{S} \Rightarrow \boldsymbol{X}Y \Rightarrow a\boldsymbol{X}Y \Rightarrow ab\boldsymbol{Y} \Rightarrow abc\boldsymbol{Y} \Rightarrow abcd$$

Derivações mais à direita

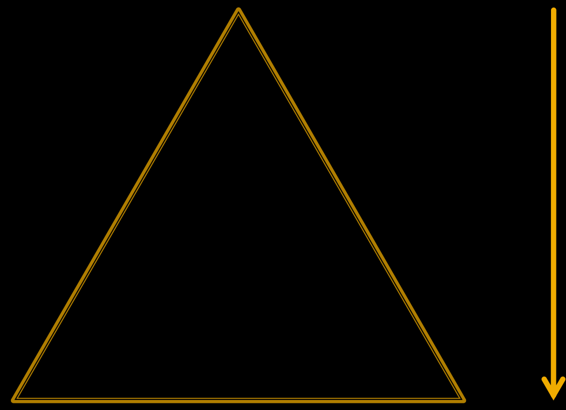
- Uma seqüência de derivações é dita "mais à direita" quando todas as derivações são feitas sempre sobre os símbolos não-terminais situados mais à direita na forma sentencial corrente;
- Exemplo (usando a gramática anterior):
  $\boldsymbol{S} \Rightarrow X\boldsymbol{Y} \Rightarrow Xc\boldsymbol{Y} \Rightarrow \boldsymbol{X}cd \Rightarrow a\boldsymbol{X}cd \Rightarrow abcd$

Reduções mais à esquerda

- Uma seqüência de reduções é dita "mais à esquerda" quando todas as reduções são feitas sempre sobre subcadeias que correspondem ao lado direito de regras e estão situadas mais à esquerda na forma sentencial corrente;
- Exemplo (usando a gramática anterior):
  $a\boldsymbol{b}cd \Rightarrow \boldsymbol{aX}cd \Rightarrow Xc\boldsymbol{d} \Rightarrow X\boldsymbol{cY} \Rightarrow \boldsymbol{XY} \Rightarrow S$
- Observar que uma seqüência de reduções mais à esquerda corresponde sempre à ordem inversa de uma seqüência de derivações mas à direita;
- De fato, no exemplo acima:
  $\boldsymbol{S} \Rightarrow X\boldsymbol{Y} \Rightarrow Xc\boldsymbol{Y} \Rightarrow \boldsymbol{X}cd \Rightarrow a\boldsymbol{X}cd \Rightarrow abcd$
  É uma seqüência de derivações mais à direita (as mesmas formas sentenciais aparecem em ordem invertida);
- Da mesma forma, é possível definir uma seqüência de reduções mais à direita (correspondendo à ordem inversa de uma seqüência de derivações mais à esquerda).

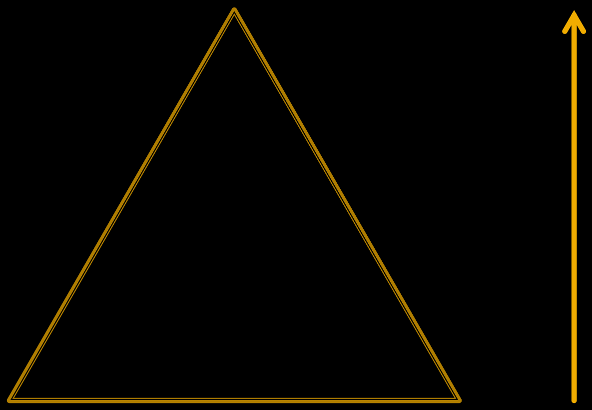
Análise descendente

- 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 <u>derivações mais à esquerda</u> (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 $first_k(\alpha)$

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

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

$first_k(\alpha)$ 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 α pela aplicação das regras de G;
- todas as cadeias de comprimento menor que k que são geradas por α pela aplicação das regras de G.

Exemplos

- Considere a gramática:
  - $S \rightarrow XY$
  - $X \rightarrow aX \mid b$
  - $Y \rightarrow cY \mid d$
- $first_1\,(aX) = \{a\}$
- $first_1\,(b) = \{b\}$
- $first_1\,(XY) = \{a, b\}$
- $first_1\,(S) = \{a, b\}$
- $first_2\,(aX) = \{b, aa, ab\}$
- $first_2\,(b) = \{b\}$
- $first_2\,(XY) = \{aa, ab, bc, bd\}$
- etc.

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 $first_k\ (x) = first_k\ (y)$, isso implicar $\beta=\gamma$.

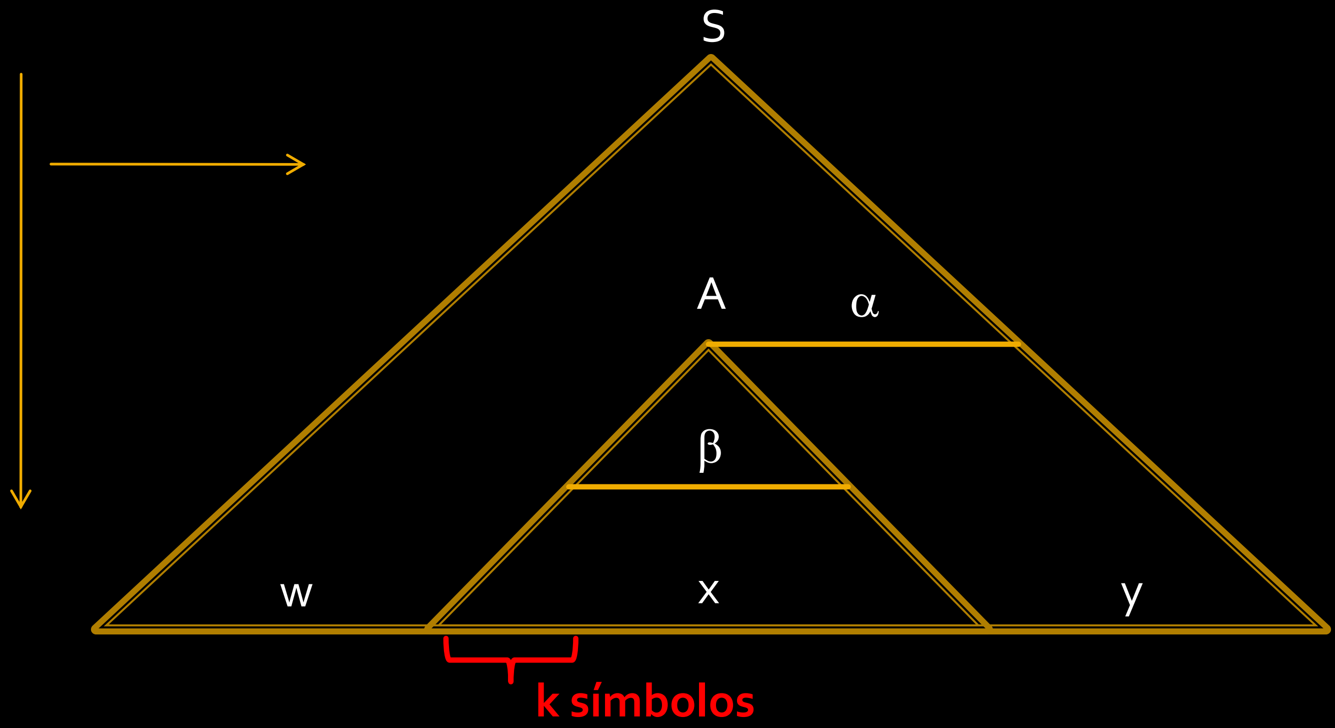
A escolha da derivação para o símbolo não-terminal A na forma sentencial $wA\alpha$ é feita de maneira unívoca a partir da análise dos k primeiros símbolos terminais gerados por $A\alpha$.

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 $A \rightarrow \beta$:

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



micro English

```
Sentence::=  Subject Verb Object .

Subject ::= I | a Noun | the Noun

Noun ::= rat | cat | dog

Verb ::= is | see | sees | like

Object ::= me | a Noun | the Noun
```

Análise descendente

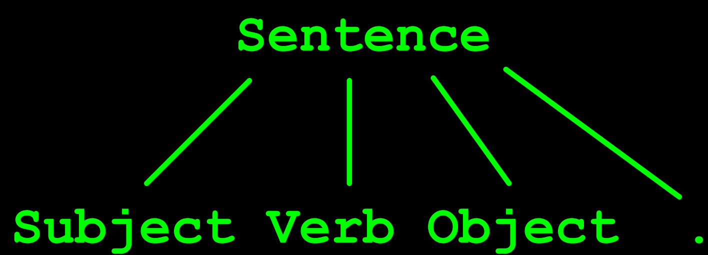
```
the cat sees a dog.
 ↑
```

Sentence ⇒ Subject Verb Object .

- O lookahead “the” é usado para selecionar a regra:
  Sentence ⇒ Subject Verb Object .

Análise descendente

```
the cat sees a dog.
 ↑
```



Análise descendente

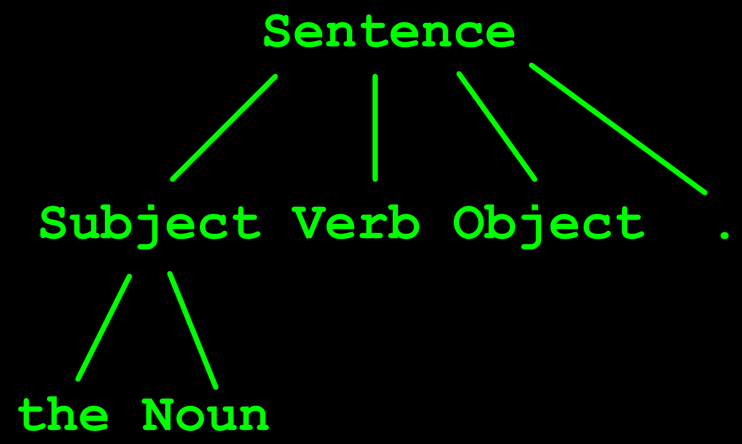
```
the cat sees a dog.
 ↑
```

`Sentence ⇒ Subject Verb Object ⇒ the Noun Verb Object .`

- `O lookahead "the" é usado para selecionar a regra:`
  `Subject ⇒ the Noun`

Análise descendente

```
the cat sees a dog.
```



Análise descendente

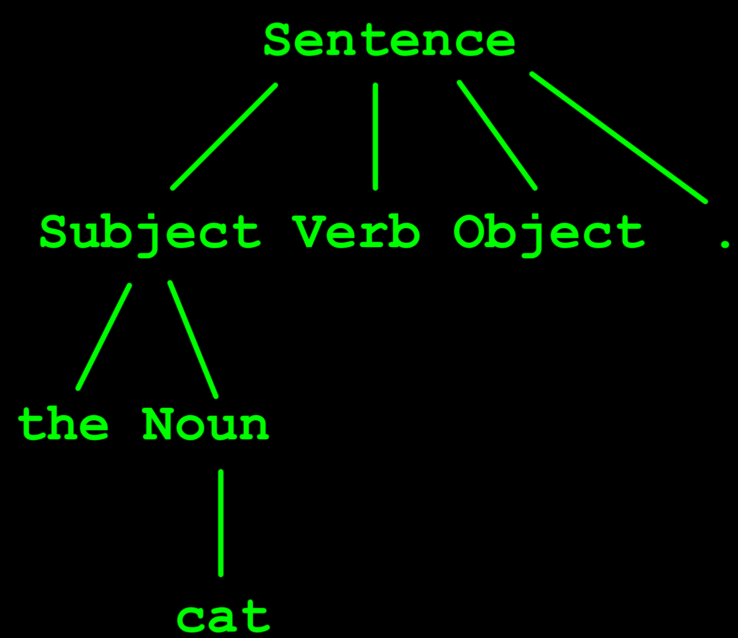
```
the cat sees a dog.
     ↑
```

Sentence ⇒ Subject Verb Object ⇒ the Noun Verb Object . ⇒ the cat Verb Object .

- O lookahead "cat" é usado para selecionar a regra:
  Noun ⇒ cat

Análise descendente

```
the cat sees a dog.
     ↑
```



Análise descendente

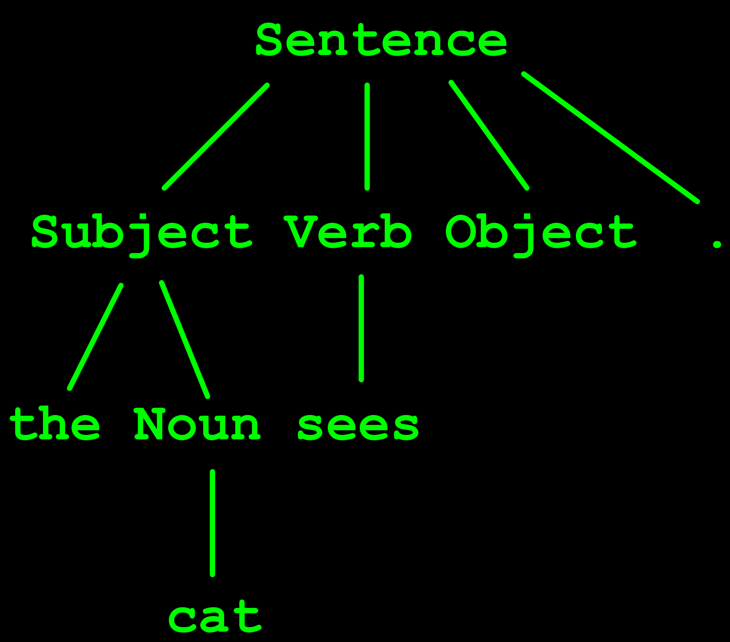
```
the cat sees a dog.
         ↑
```

Sentence ⇒ Subject Verb Object ⇒ the Noun Verb Object . ⇒ the cat Verb Object . ⇒ the cat sees Object .

- O lookahead “sees” é usado para selecionar a regra:
  Verb ⇒ sees

Análise descendente

```
the cat sees a dog.
```



Análise descendente

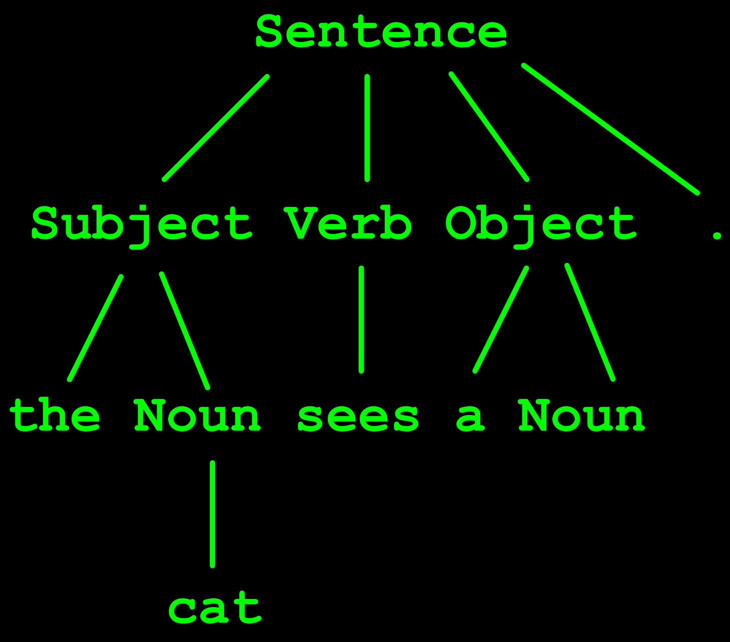
```
the cat sees a dog.
             ↑
```

Sentence ⇒ Subject Verb Object ⇒ the Noun Verb Object . ⇒ the cat Verb Object . ⇒ the cat sees Object . ⇒ the cat sees a Noun .

- O lookahead "a" é usado para selecionar a regra:
  Object ⇒ a Noun

Análise descendente

```
the cat sees a dog.
             ↑
```



Análise descendente

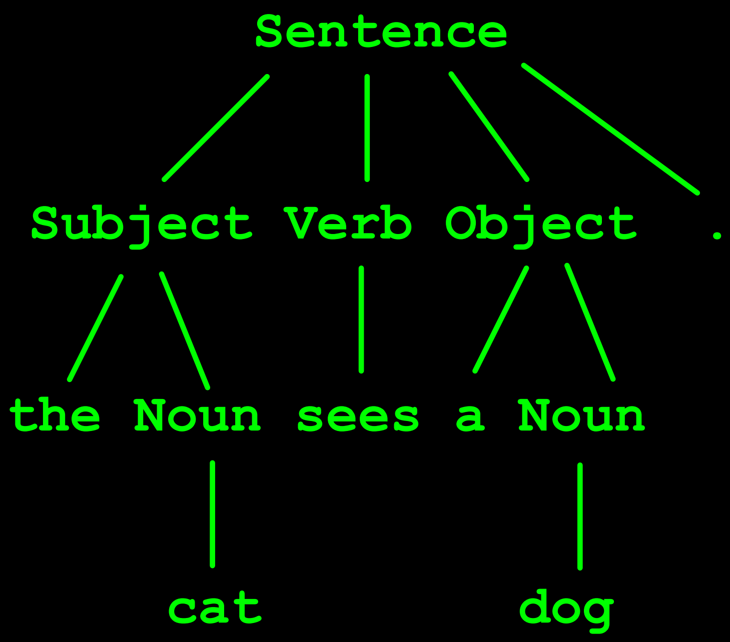
```
the cat sees a dog.
                ↑
```

`Sentence ⇒ Subject Verb Object ⇒ the Noun Verb Object . ⇒ the cat Verb Object . ⇒ the cat sees Object . ⇒ the cat sees a Noun . the cat sees a dog .`

- `O lookahead "dog" é usado para selecionar a regra:`
  `Noun ⇒ 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. $S \Rightarrow^* \alpha A w \Rightarrow \alpha\beta w$
2. $S \Rightarrow^* \gamma B x \Rightarrow \alpha\beta y$

tais que $first_k$ (w) = $first_k$ (y), isso implicar $\alpha=\gamma$, A=B e x=y.

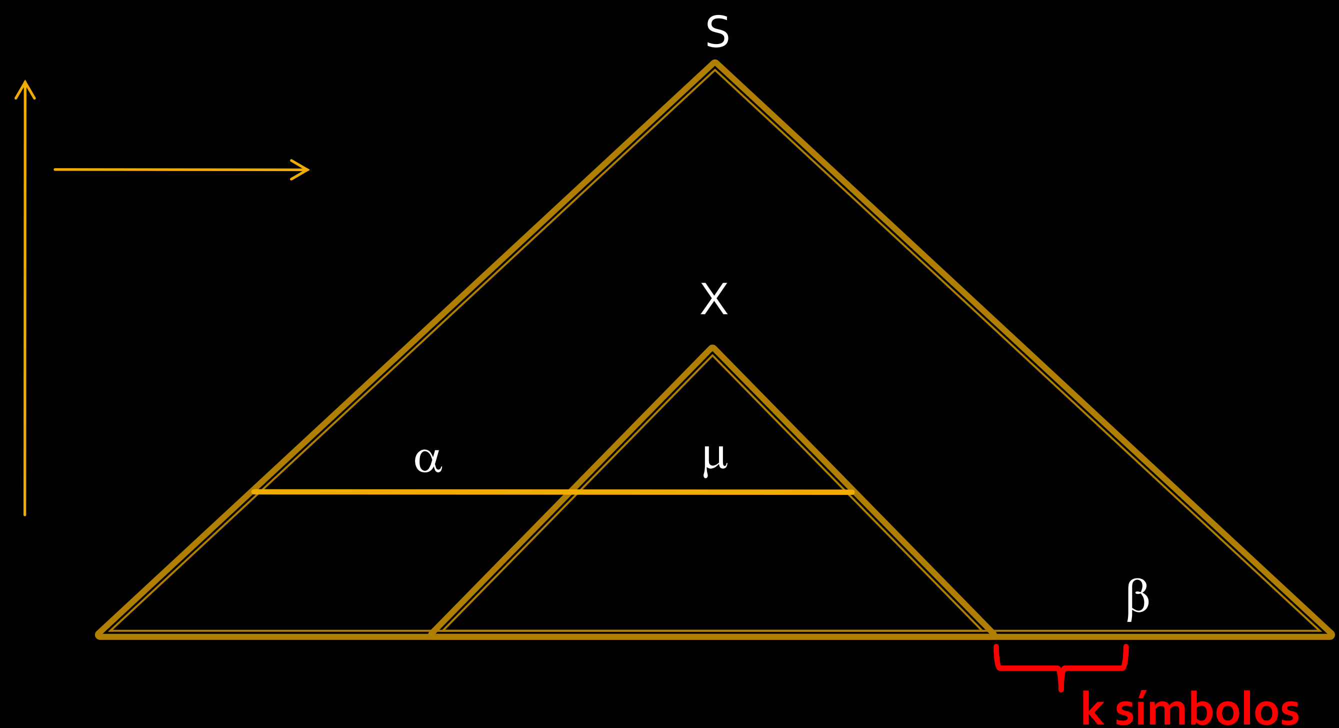
Uma vez identificada, a escolha da redução para a cadeia β é feita de maneira unívoca a partir da análise dos k primeiros símbolos terminais de w.

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:

$w \Rightarrow^* \alpha\mu\beta \Rightarrow \alpha X\beta \Rightarrow^* S$ ($\beta,\gamma \in \Sigma^*$, $S, X \in N$ 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 . ⇒ the Noun sees a dog . ⇒ 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.
               ↑
```

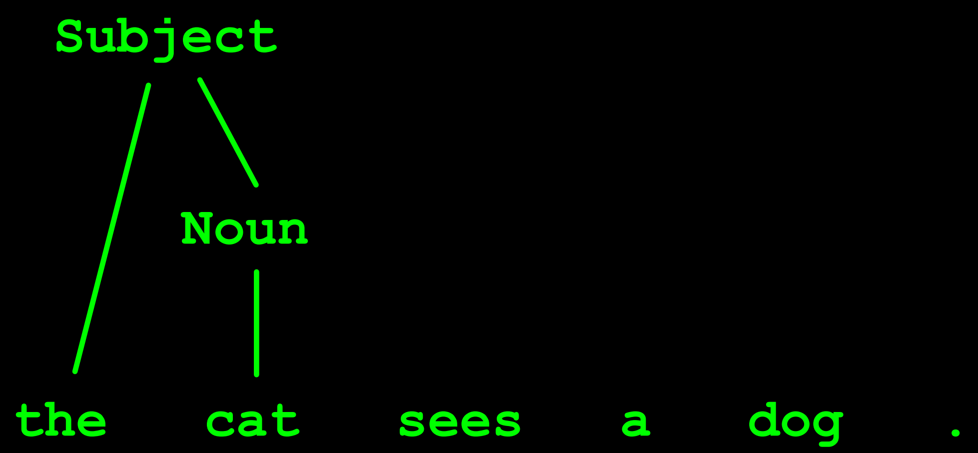
- "sees" é reduzido para "Verb" ou para "Object";
- O cursor avança para o próximo símbolo.

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

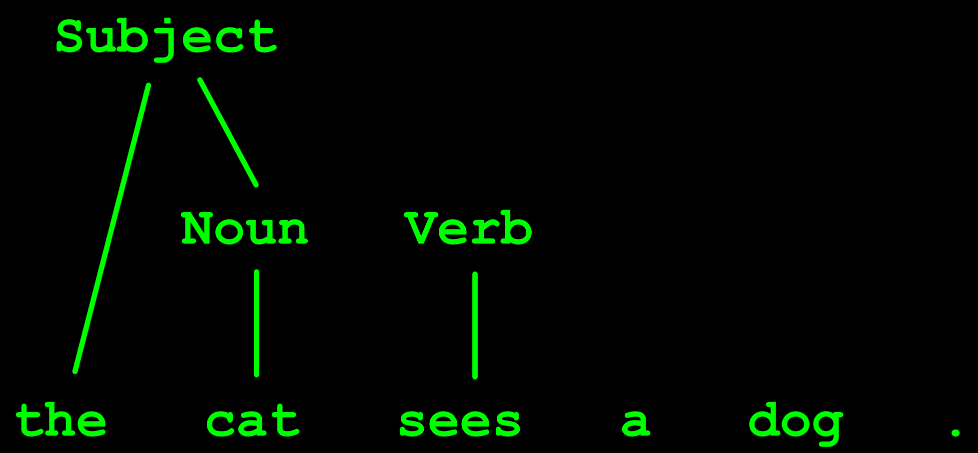
Análise ascendente

```
      Noun
       |
the   cat   sees   a   dog   .
```

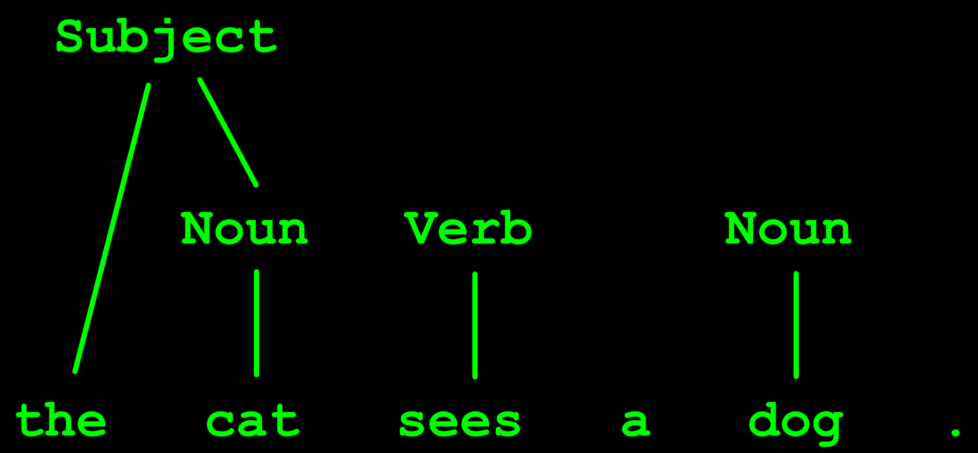
Análise ascendente



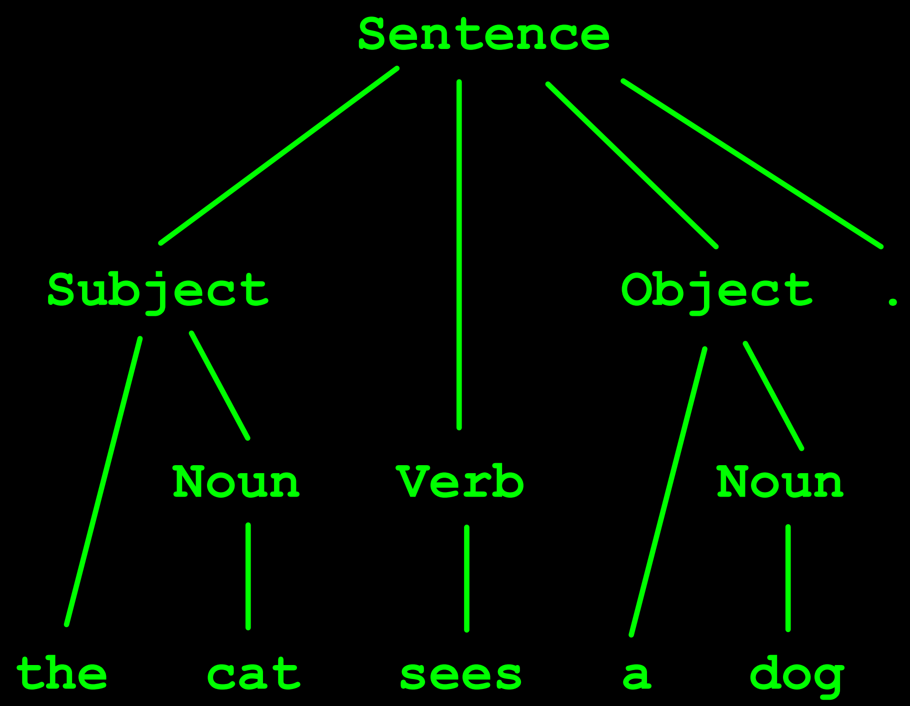
Análise ascendente



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);

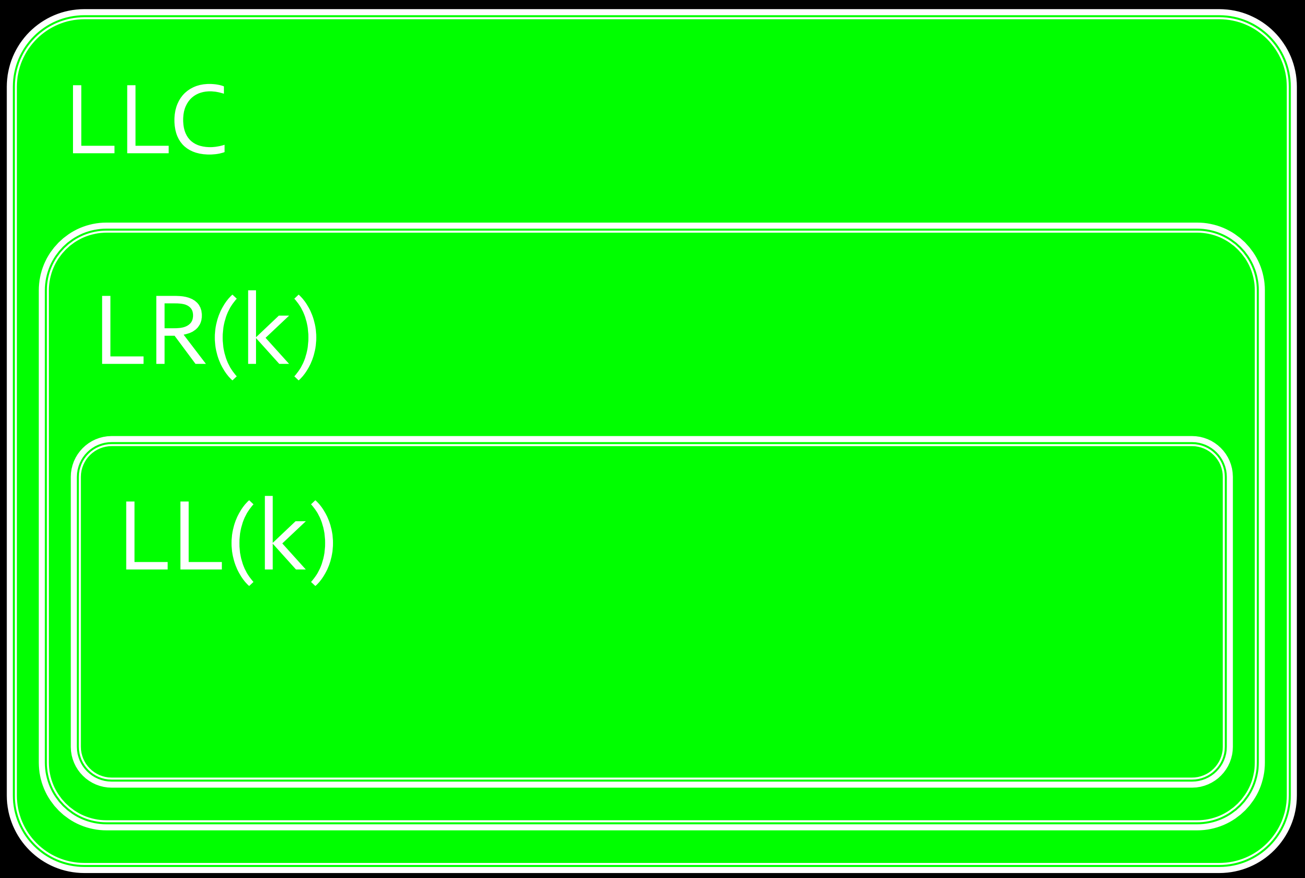
RR(k):

- 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



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.

Ascendente x Descendente

- Problema decidível:
  - G é LL(k) para um dado valor de k?
  - G é LR(k) para um dado valor de k?

- Problemas indecidíveis:
  - Existe algum valor de k tal que G seja LL(k)?
  - Se G não é LL(1), existe alguma gramática G' tal que G' seja LL(1) e L(G)=L(G')?

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 $A \to \beta$, $A \to \gamma \in P$, então $first_k(\beta\alpha) = first_k(\gamma\alpha)$ implicar $\beta=\gamma$.
- 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 $A\alpha$.

Implicações

- Na prática, usa-se apenas A e os k símbolos para fazer a escolha determinística da regra a ser utilizada.
- Eventualmente, usa-se também w se isso trouxer algum benefício (como por exemplo usar um valor menor de k);
- De qualquer forma, w e α são apenas informação de contexto empregada para fazer um reconhecimento de uma linguagem livre de contexto. É diferente de usar informação de contexto para determinar quais sentenças podem e não podem ser geradas.

Estratégias para verificar a condição LL(k)

Com base na definição anterior:

- Suponha $A \to \beta_1 \mid \ldots \mid \beta_n$
- Determinar todas as formas sentenciais $wA\alpha$ em que o não-terminal A possa comparecer;
- Verificar, <u>para cada uma delas</u>, se $first_k(\beta_i\alpha) \cap first_k(\beta_j\alpha) = \emptyset$ 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 $first_k$ $(\beta_i\alpha)$, especialmente quando β_i não gera cadeias de terminais de comprimento k.

Estratégias para verificar a condição LL(k)

Alternativa usada na prática:

- Considerar, para efeito de escolha da regra a ser aplicada, apenas o não-terminal A em questão e os k símbolos do lookahead;
- Os conjuntos $first_k$ ($\beta_i\alpha$), para cada β_i, neste caso, são calculados levando em conta todas as possíveis cadeias α que possam comparecer do lado direito de A;
- Nem sempre a gramática será LL(k) desta forma, mas quando for isto representará uma simplificação importante em relação ao método anterior.

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 $first_1(cX)=\{c\}$
  Se $X \rightarrow d$, então $aX \Rightarrow ad$ e $first_1(d)=\{d\}$
  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 $first_1(cX)=\{c\}$
  Se $X \rightarrow d$, então $bX \Rightarrow bd$ e $first_1(d)=\{d\}$
  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 ($X \to cX$) gera cadeias que começam sempre pelo símbolo "c", independentemente da forma sentencial considerada; da mesma forma, a escolha da segunda regra ($X \to d$) 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 $first_1\ (aaB)=\{a\}$
  Se $A \rightarrow ab$, então $aAaB \Rightarrow aabaB$ e $first_1\ (abaB)=\{a\}$
  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 $first_1\ (abB)=\{a\}$
  Se $A \rightarrow ab$, então $bAbB \Rightarrow babbB$ e $first_1\ (abbB)=\{a\}$
  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 $first_2\ (aaB)=\{aa\}$
  Se $A \rightarrow ab$, então $aAaB \Rightarrow aabaB$ e $first_2\ (abaB)=\{ab\}$
  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 $first_2\ (abB)=\{ab\}$
  Se $A \rightarrow ab$, então $bAbB \Rightarrow babbB$ e $first_1\ (abbB)=\{ab\}$
  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 $first_3$ (abB)={aba}
  Se $A \rightarrow ab$, então $bAbB \Rightarrow babbB$ e $first_1$ (abbB)={abb}
  Como $\{aba\} \cap \{abb\} = \varnothing$, 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}

{aaa,aba}

{aba,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↵}
- aAaB com A $\rightarrow$ ab
  {abaa}
- bAbB com A $\rightarrow$ a
  {abaa, aba↵}
- bAbB com A $\rightarrow$ ab
  {abba}

{aaa↵,**abaa**,aba↵}

{**abaa**,abba}

Estratégias para verificar a condição LL(k)

k=5

- aAaB com A → a
  {aaa↲↲, aaaa↲ ,aaaaa}
- aAaB com A → ab
  {abaa↲, abaaa}
- bAbB com A → a
  {aba↲↲,abaa↲,abaaa}
- bAbB com A → ab
  {abba↲,abbaa}

{aaa↲↲, aaaa↲ ,aaaaa, aba↲↲,**abaa↲**,**abaaa**}

{**abaa↲**, **abaaa**, abba↲,abbaa}

Estratégias para verificar a condição LL(k)

Obtendo uma gramática equivalente LL(4):

A → a | ab
B → aa*
S → a (a|ab) aaa* | b (a|ab) baa*
S → (aa | aab) aaa* | (ba | bab) baa*
S → aaaaa* | aabaaa* | babaa* | babbaa*

$first_4$ (aaaaa*) = {aaaa}
$first_4$ (aabaaa*) = {aaba}
$first_4$ (babaa*) = {baba}
$first_4$ (babbaa*) = {babb}

Estratégias para verificar a condição LL(k)

Obtendo uma gramática equivalente LL(1):

S → aaaaa* | aabaaa* | babaa* | babbaa*
S → a (aaaa* | abaaa*) | b (abaa* | abbaa*)
S → a (a (aaa* | baaa*)) | b (a (baa* | bbaa*))
S → a (a (aaa* | baaa*)) | b (a (b (aa* | 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 $A \rightarrow \beta_1 \mid \ldots \mid \beta_k$
- Seja $X_1 = first_1 (\beta_1\gamma_1) \cup \ldots \cup first_1 (\beta_1\gamma_n)$
- ...
- Seja $X_k = first_1 (\beta_k\gamma_1) \cup \ldots \cup first_1 (\beta_k\gamma_n)$
- 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 comparecem imediatamente à direita da cadeia β, consideradas todas as formas sentenciais geradas por G em que β faça parte.

Exemplo de $\text{follow}_k(\beta)$

$$
\begin{aligned}
S &\longrightarrow aXbY\# \\
X &\longrightarrow aX \\
X &\longrightarrow a \\
Y &\longrightarrow bY \\
Y &\longrightarrow b \\
follow_1(X) &= \{b\} \\
follow_1(Y) &= \{\#\}
\end{aligned}
$$

$$
\begin{aligned}
S &\longrightarrow aSXc\# \\
X &\longrightarrow bX \\
X &\longrightarrow \varepsilon \\
follow_1(S) &= \{b, c\} \\
follow_1(X) &= \{c\}
\end{aligned}
$$

Estratégias

Com base na definição anterior:

- $A \rightarrow \beta_1 \mid \dots \mid \beta_n$
- Verificar se $first_k\ (\beta_i.follow_k\ (A)) \cap first_k\ (\beta_j.follow_k\ (A)) = \emptyset$, $1 \leq i\ ,\ j \leq n$, $i \neq j$;
- Se todos os β_i geram cadeias não-vazias e, além disso, os β_i geram sentenças com k símbolos terminais, então a condição acima é equivalente à:
- Verificar se $first_k\ (\beta_i) \cap first_k\ (\beta_j) = \emptyset$, $1 \leq i\ ,\ j \leq n$, $i \neq j$.

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.

$A \rightarrow \sigma_1 \alpha_1 \mid \sigma_2 \alpha_2 \mid \ldots \mid \sigma_n \alpha_n$
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

<u>*Gramáticas LL(1) sem regras vazias*</u>:

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

Exemplo

Gramáticas LL(1) sem regras vazias:

- $S' \to S\#$
- $S \to ABe$
- $A \to dB \mid aS \mid c$
- $B \to AS \mid b$

- S':
  $\mathrm{first}_1(S\#)=\mathrm{first}_1(ABe\#)=\{a,c,d\}$
- S:
  $\mathrm{first}_1(ABe)=\{a,c,d\}$
- A:
  $\mathrm{first}_1(dB)=\{d\}$, $\mathrm{first}_1(aS)= \{a\}$, $\mathrm{first}_1(c)=\{c\}$
- B:
  $\mathrm{first}_1(AS)=\mathrm{first}_1(dBS)\cup\mathrm{first}_1(aSS)\cup\mathrm{first}_1(cS)=\{d,a,c\}$
  $\mathrm{first}_1(b)=\{b\}$

Caso geral

Gramáticas LL(1) com regras vazias:

- $A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$
  $first_1\ (\alpha_i.follow\ (A)) \cap first_1\ (\alpha_j.follow\ (A)) = \varnothing,\ i \neq j.$

ou ainda:

- **$first_1\ (\alpha_i) \cap first_1\ (\alpha_j) = \varnothing$, $i \neq j$, e, se $\alpha_i \Rightarrow^* \varepsilon$, então**
  $first_1\ (\alpha_j) \cap follow_1\ (A) = \varnothing$, $j \neq i$.

Se o símbolo corrente pertence ao $follow_1$ (A), a regra $A \rightarrow \varepsilon$ deve ser a escolhida.

Exemplo

Gramáticas LL(1) com regras vazias:

- $S' \to A\#$
- $A \to iB \leftarrow e$
- $B \to SB \mid \varepsilon$
- $S \to [eC] \mid .i$
- $C \to eC \mid \varepsilon$

- S':
  $first_1(A\#)=\{i\}$
- A:
  $first_1(iB{\leftarrow}e)=\{i\}$
- B:
  $first_1(SB) \cap follow_1(B)=\{[,.\} \cap \{\leftarrow\}=\varnothing$
- S:
  $first_1([eC]) \cap first_1(.i)=\{[\} \cap \{.\}=\varnothing$
- C:
  $first_1(eC) \cap follow_1(C)=\{e\} \cap \{]\}=\varnothing$

Recursão à esquerda

Suponha que σ_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 **σ_3**), pois as formas sentenciais resultantes **$X\ \sigma_2$** $\sigma_2^{\ k-1}\ \sigma_1$ e **σ_3** $\sigma_2^{\ k-1}\ \sigma_1$ possuem sempre a cadeia **$\sigma_3\ \sigma_2^{\ k-1}$** como elemento comum dos conjuntos $first_k$ () 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 σ_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 σ_3 é o elemento comum aos dois conjuntos $first_1$;

- 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 $first_2$;

- 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 $first_3$;

- 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 **vermelho**
1	**a**	X → Xb ou X → a
2	ac, **ab**	X → Xb ou X → a
3	ac, abc, **abb**	X → Xb ou X → a
4	ac, abc, abbc, **abbb**	X → Xb ou X → a
...	...	...
n	..., **ab^{n-1}**, ...	X → Xb ou X → 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 **abbb**)
- k=5 (ac, abc, abbc , abbbc ou **abbbb**)
- k=6 (ac, abc, abbc , abbbc, abbbbc ou **abbbbb**)
- k=7 (ac, abc, abbc , abbbc, abbbbc, abbbbbc ou **abbbbbb**)

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 ambas permitem chegar no mesmo resultado.

Recursão à esquerda

Suponha que num certo ponto da análise a forma sentencial obtida pelo uso exclusivo de derivações mais à esquerda seja wXb^{k-1}. Suponhamos ainda que a entrada neste ponto seja ab^{k-1}. Então, não é possível determinar de forma unívoca a regra a ser usada na substituição de X.

Senão, vejamos:

1. Se usarmos a regra $X \rightarrow Xb$ inicialmente, e depois a regra $X \rightarrow a$, obtemos a seqüência:
   $wXb^{k-1} \Rightarrow wXbb^{k-1} \Rightarrow wabb^{k-1}$
2. Se usarmos a regra $X \rightarrow a$ inicialmente, obtemos a seqüência:
   $wXb^{k-1} \Rightarrow wab^{k-1} \Rightarrow wab^{k-1}$

Em ambos os casos, os k símbolos do lookahead formam a cadeia ab^{k-1}. Logo, não é possível escolher deterministicamente a regra do X, qualquer que seja o valor de k considerado.

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 nãoterminal 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:

$X_1 \rightarrow \gamma_{11} \mid \gamma_{12} \mid \ldots \mid \gamma_{1m}$
$X_2 \rightarrow \gamma_{21} \mid \gamma_{22} \mid \ldots \mid \gamma_{2n}$
...
$X_p \rightarrow \gamma_{p1} \mid \gamma_{p2} \mid \ldots \mid \gamma_{pq}$

$first_1(\gamma_{1i}) \cap first_1(\gamma_{1j}) = \varnothing$, $1 \leq i, j \leq m$, $i \neq j$.
$first_1(\gamma_{2i}) \cap first_1(\gamma_{2j}) = \varnothing$, $1 \leq i, j \leq n$, $i \neq j$.
...
$first_1(\gamma_{pi}) \cap first_1(\gamma_{pj}) = \varnothing$, $1 \leq i, j \leq q$, $i \neq j$.

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

$first_1(\gamma_{ik}) \cap follow_1(X_i) = \varnothing$, $\forall\, k \neq j$.

Gramática LL(1) - Alternativa

Considere a GLC G:

$X_1 \to \gamma_{11} \mid \gamma_{12} \mid \ldots \mid \gamma_{1m}$
$X_2 \to \gamma_{21} \mid \gamma_{22} \mid \ldots \mid \gamma_{2n}$
...
$X_p \to \gamma_{p1} \mid \gamma_{p2} \mid \ldots \mid \gamma_{pq}$

$\text{first}_1\ (\gamma_{1i}).\text{follow}_1(X_1) \cap \text{first}_1\ (\gamma_{1j}).\text{follow}_1(X_1) = \varnothing$, $1 \leq i, j \leq m$, $i \neq j$.
$\text{first}_1\ (\gamma_{2i}).\text{follow}_1(X_2) \cap \text{first}_1\ (\gamma_{2j}).\text{follow}_1(X_2) = \varnothing$, $1 \leq i, j \leq n$, $i \neq j$.
...
$\text{first}_1\ (\gamma_{pi}).\text{follow}_1(X_p) \cap \text{first}_1\ (\gamma_{pj}).\text{follow}_1(X_p) = \varnothing$, $1 \leq i, j \leq q$, $i \neq j$.

Outros exemplos (I)

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

Não é LL(1) mas é LL(2):

- S: $\text{first}_2(bS) \cap \text{first}_2(bA) = \{bb\} \cap \{bd, bc\} = \emptyset$
- A: $\text{first}_1(d) \cap \text{first}_1(ccA) = \{d\} \cap \{c\} = \emptyset$

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:
  $\mathtt{first}_1(b(S|A))=\{b\}$
- ():
  $\mathtt{first}_1(S) \cap \mathtt{first}_1(A)=\{b\}\cap\{c,d\}=\varnothing$
- A:
  $\mathtt{first}_1(d)\cap\mathtt{first}_1(ccA)=\{d\}\cap\{c\}=\varnothing$

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

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

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

Outros exemplos (II)

Fatorando à esquerda e agrupando:

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

Torna-se LL(1), pois:

- `()`:
  $\texttt{first}_1(X) \cap \texttt{first}_1(Y) = \{b, c\} \cap \{d, e\} = \emptyset$

Outros exemplos (III)

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

Não é LL(1), pois:

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

Outros exemplos (III)

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

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

Pois:

- ( ):
  $\mathrm{first}_1(bXd^*ce)=\{b\}$
  $\mathrm{first}_1(ce)=\{c\}$
  $\mathrm{first}_1(e)=\{e\}$

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.

G não LL(1) → G′ LL(1) → Analisador sintático determinístico

Conversão

Conversão:

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

Fatoração à esquerda

$X \longrightarrow \alpha\beta \mid \alpha\gamma$

A condição não é verificada para o não-terminal X, pois ambas as regras começam com α.
Para resolver, deve-se colocar em evidência o termo comum mais à esquerda:

$X \longrightarrow \alpha\ (\beta \mid \gamma)$

A condição LL(k) é verificada para a regra, desde que seja verificada dentro dos parênteses também.

Substituições

$X \longrightarrow \alpha Y \beta$
$Y \longrightarrow \gamma \mid \omega$

O não-terminal Y é substituído pela sua definição:

$X \longrightarrow \alpha(\gamma|\omega)\beta$

O não-terminal Y e suas regras podem ser eliminados da gramática. A gramática resultante é equivalente à original.

Recursões à esquerda

$X \longrightarrow X\alpha \mid X\beta \mid \gamma \mid \omega$

O não-terminal X é substituído por:

$X \longrightarrow (\gamma|\omega)(\alpha|\beta)^*$

A condição LL(k) deve ser verificada para a regra acima.

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):

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