

Conceitos de Análise Sintática

Pro.f. Marcus Ramos

UNIVASF

Atualizado em 19 de abril de 2026 às 08:40

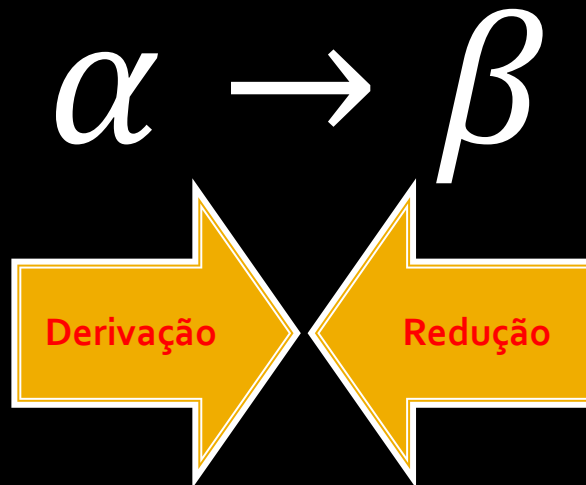
Referências

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**: substituição do lado esquerdo de uma regra gramatical (α) pelo lado direito correspondente (β) na forma sentencial corrente;
- **REDUÇÃO**: substituição do lado direito de uma regra gramatical (β) pelo lado esquerdo (α) correspondente na forma sentencial corrente;



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: $aXcY \Rightarrow aaXcY$
- Exemplo de redução: $aXcY \Rightarrow XcY$
- 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):
 $S \Rightarrow XY \Rightarrow aXY \Rightarrow abY \Rightarrow abcY \Rightarrow abcd$

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):
 $S \Rightarrow XY \Rightarrow XcY \Rightarrow Xcd \Rightarrow aXcd \Rightarrow abcd$

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):
 $abcd \Rightarrow aXcd \Rightarrow Xcd \Rightarrow XcY \Rightarrow XY \Rightarrow S$
- 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:
 $S \Rightarrow XY \Rightarrow XcY \Rightarrow Xcd \Rightarrow aXcd \Rightarrow abcd$
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).

Determinismo x Não-determinismo

O principal objetivo da análise sintática é:

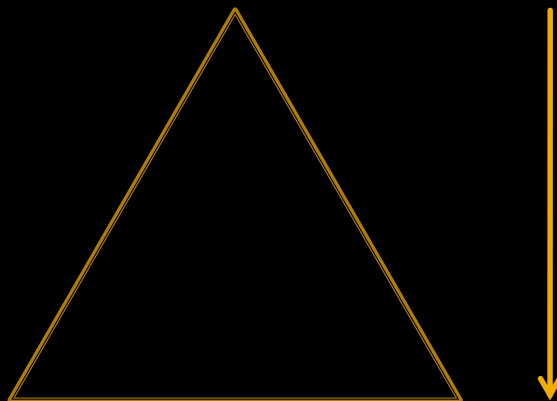
- Descobrir as regras que foram usadas na geração da cadeia dada (se existir tal conjunto de regras);
- Para isso, é usada a gramática livre de contexto que gera a linguagem sendo analisada;
- Mesmo fixando uma ordem de derivação (por exemplo mais à esquerda), ainda é possível que um certo não-terminal possua várias alternativas de substituição;
- Por exemplo, $A \rightarrow \beta_1 \mid \dots \mid \beta_n$

Determinismo x Não-determinismo

- Então, como escolher de forma determinística a regra em questão para cada não-terminal?
- Mesmo que haja uma única opção, como garantir que ela pode ser usada?
- A escolha não-determinística sempre funciona, mas onera muito o tempo da análise sintática;
- O tempo é o maior possível justamente quando há erros no programa-fonte;
- Então, como escolher de forma determinística a regra em questão?
- Resposta: análise descendente ou ascendente.

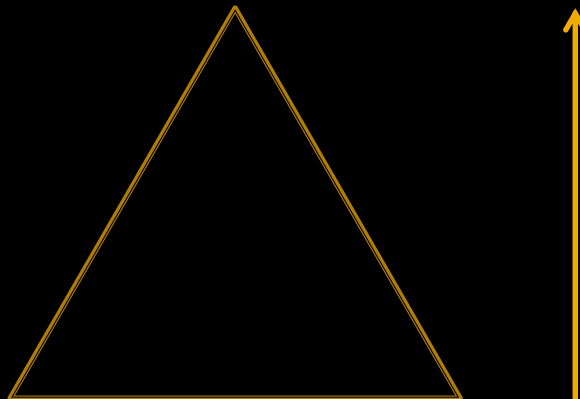
Análise descendente

- 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 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:
 - $S \rightarrow XY$
 - $X \rightarrow aX \mid b$
 - $Y \rightarrow cY \mid d$
- $first_1(aX) = \{a\}$
- $first_1(b) = \{b\}$
- $first_1(XY) = \{a, b\}$
- $first_1(S) = \{a, b\}$
- $first_2(aX) = \{aa, ab\}$
- $first_2(b) = \{b\}$
- $first_2(XY) = \{aa, ab, bc, bd\}$
- 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 $\text{first}_k(x) = \text{first}_k(y)$, isso implicar $\beta=\gamma$.

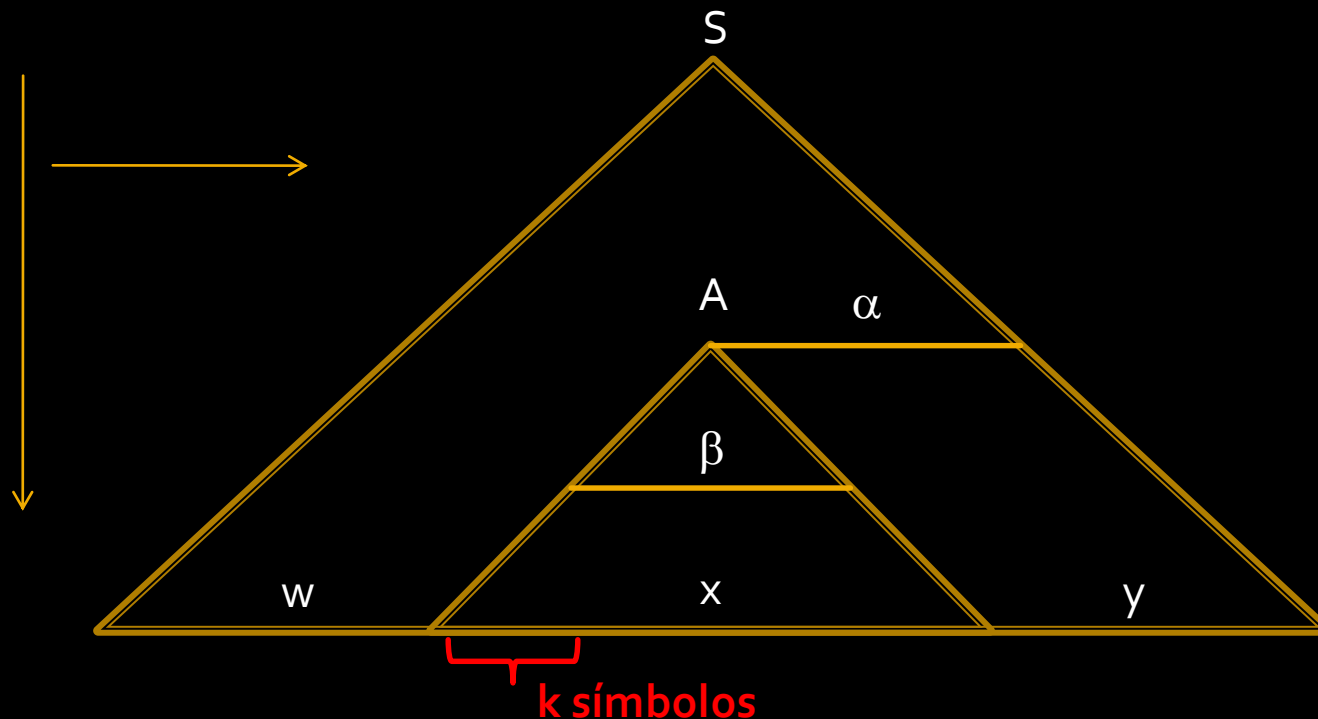
A escolha da derivação para o símbolo não-terminal A na forma sentencial $wA\alpha$ é feita de maneira unívoca a partir da análise dos k primeiros símbolos terminais gerados por $A\alpha$.

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 $\alpha \in V^*$)



Gramática LL(k)

Em outras palavras, a substituição do símbolo não-terminal (mais à esquerda) na forma sentencial corrente em um reconhecedor descendente (aquele que usa derivações mais à esquerda) é feita levando-se em conta:

$$S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx\alpha \Rightarrow^* wxy$$

- O símbolo não-terminal corrente – A.
- O lookahead corrente (k símbolos) – $\text{first}_k(\beta\alpha)$.
- A forma sentencial corrente – w e α .

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

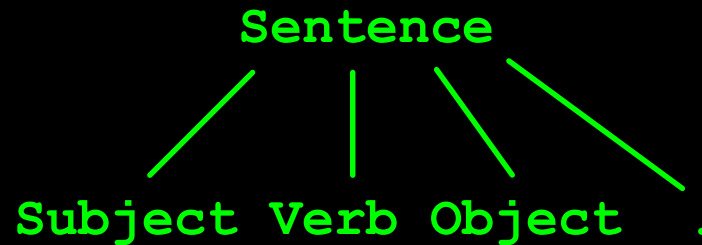
the cat sees a dog.
↑

Sentence $\Rightarrow$
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 $\Rightarrow$

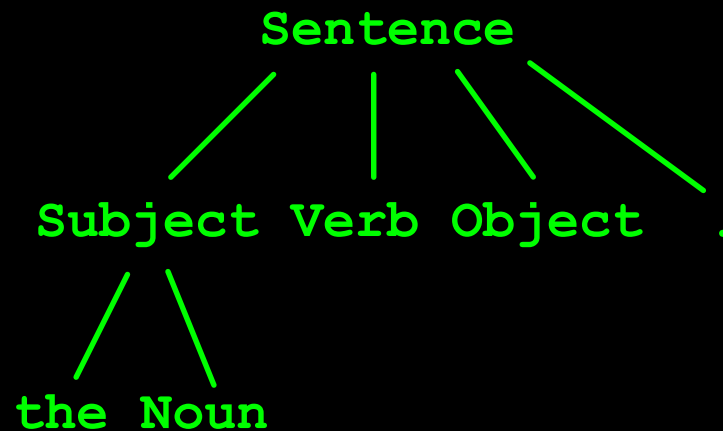
Subject Verb Object . $\Rightarrow$

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 . $\Rightarrow$

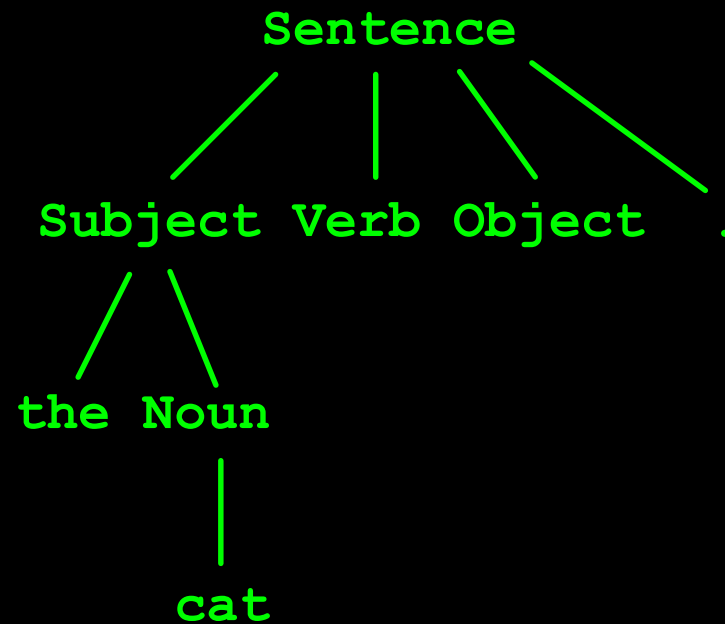
the Noun Verb Object . $\Rightarrow$

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 . $\Rightarrow$

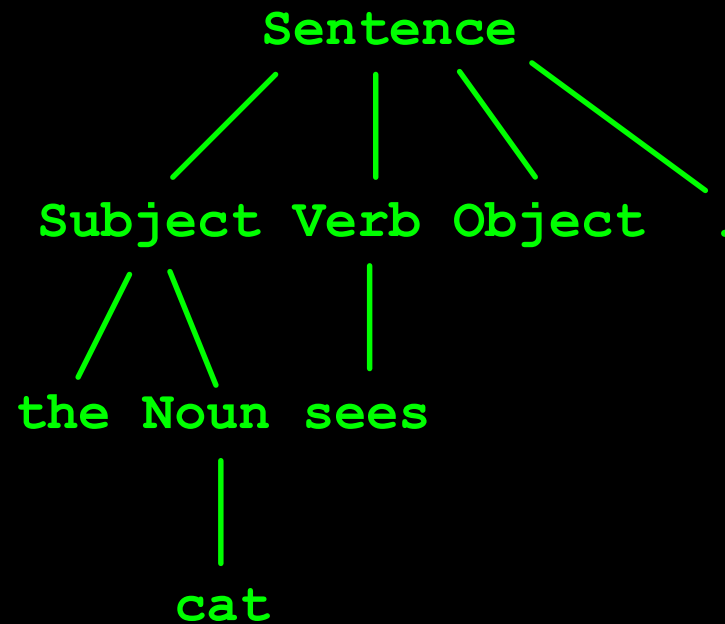
the cat Verb Object . $\Rightarrow$

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.
 ↑

Sentence $\Rightarrow$

Subject Verb Object . $\Rightarrow$

the Noun Verb Object . $\Rightarrow$

the cat Verb Object . $\Rightarrow$

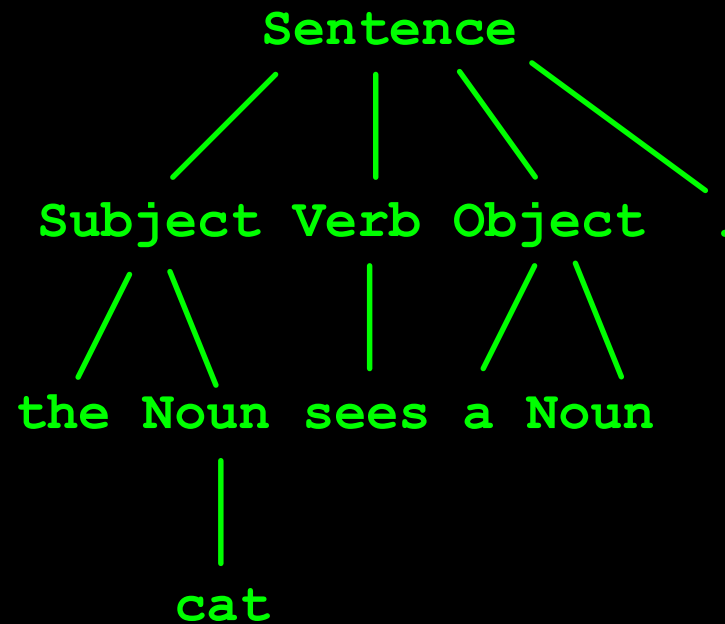
the cat sees Object . $\Rightarrow$

the cat sees a Noun .

- O lookahead "a" é usado para selecionar a regra:
Object $\rightarrow$ a 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 . $\Rightarrow$

the cat sees Object . $\Rightarrow$

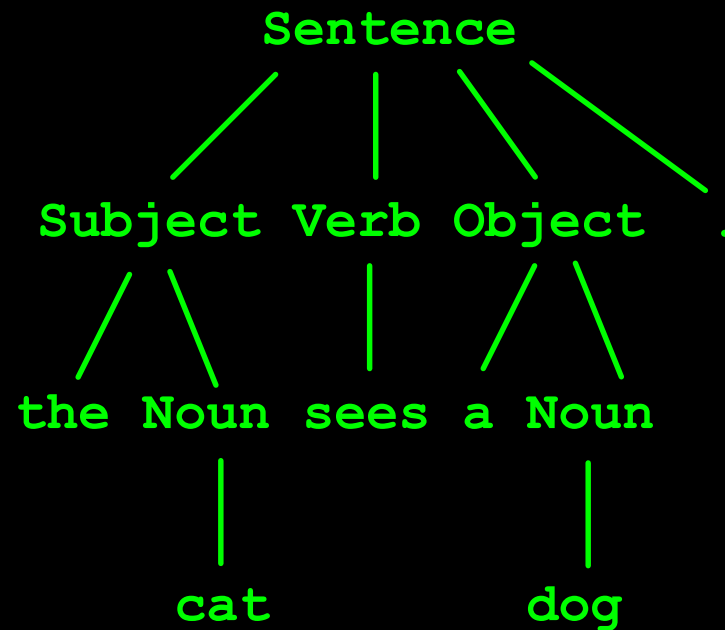
the cat sees a Noun . $\Rightarrow$

the cat sees a dog .

- O lookahead "dog" é usado para selecionar a regra:
Noun $\rightarrow$ 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. $\alpha\beta w \Rightarrow \alpha Aw \Rightarrow^* S$
2. $\alpha\beta y \Rightarrow \gamma Bx \Rightarrow^* S$

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

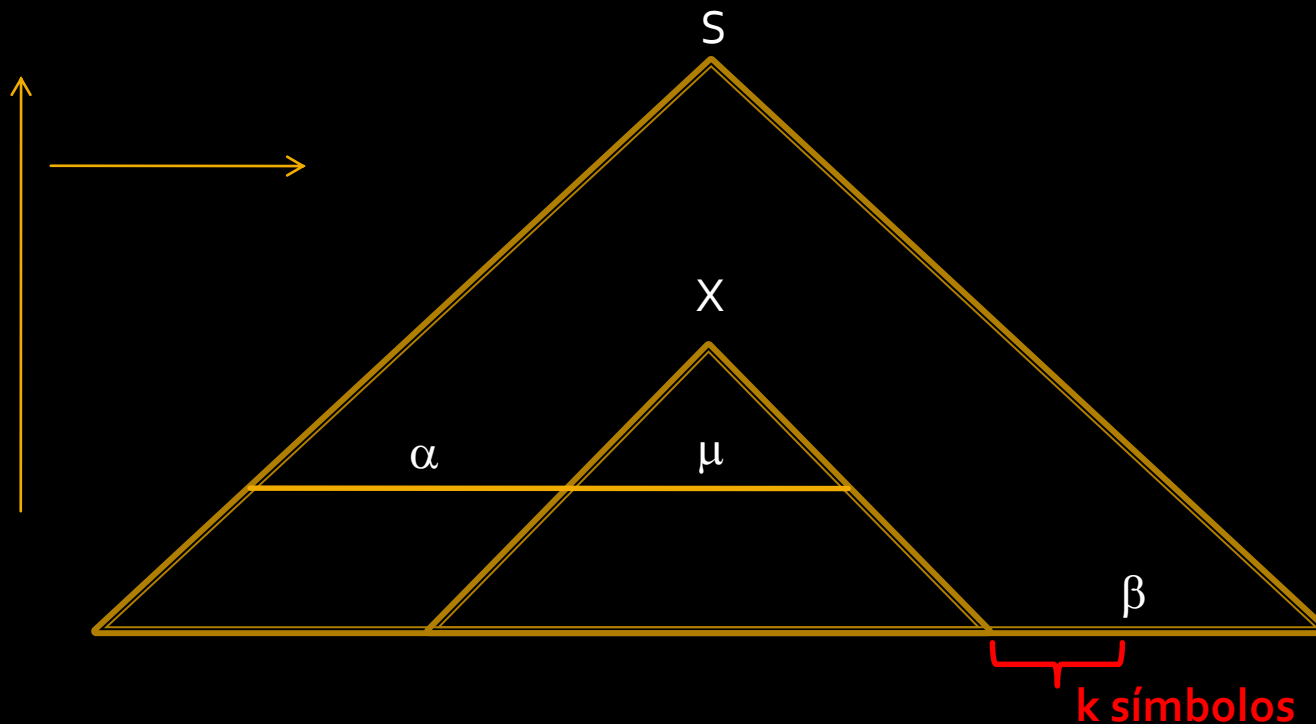
Uma vez identificada, a escolha da redução para a cadeia β é feita de maneira unívoca a partir da análise dos k primeiros símbolos terminais de w .

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 αAw .

Gramática LR(k)

Considere a seqüência de reduções mais à esquerda:

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



Gramática LR(k)

Em outras palavras, a operação de um reconhecedor ascendente (aquele que usa reduções mais à esquerda) por meio da regra $X \rightarrow \mu$ consiste das seguintes etapas:

$$\alpha\mu\beta \Rightarrow \alpha X\beta \Rightarrow^* S$$

- Reconhecer o lado direito de alguma regra que esteja situado mais à esquerda na forma sentencial corrente – μ .
- O lookahead corrente (k símbolos) – $\text{first}_k(\beta)$.
- A forma sentencial corrente – α e β .

Análise ascendente

the cat sees a dog .
↑

- Nenhuma redução é possível com o símbolo "the" apenas;
- O cursor avança 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 permanece no símbolo corrente.

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";
- 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 .
 ↑

- Nenhuma redução é possível com o símbolo "a" apenas ou qualquer prefixo da forma sentencial corrente;
- O cursor avança para o próximo símbolo.

Análise ascendente

the cat sees a dog .
 ↑

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

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

Análise ascendente

the cat sees a dog .
 ↑

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

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

Análise ascendente

the cat sees a dog .
↑

- Nenhuma redução é possível com o símbolo "." apenas ou qualquer prefixo da forma sentencial corrente;
- O cursor avança para o próximo símbolo.

Análise ascendente

the cat sees a dog .
 ↑

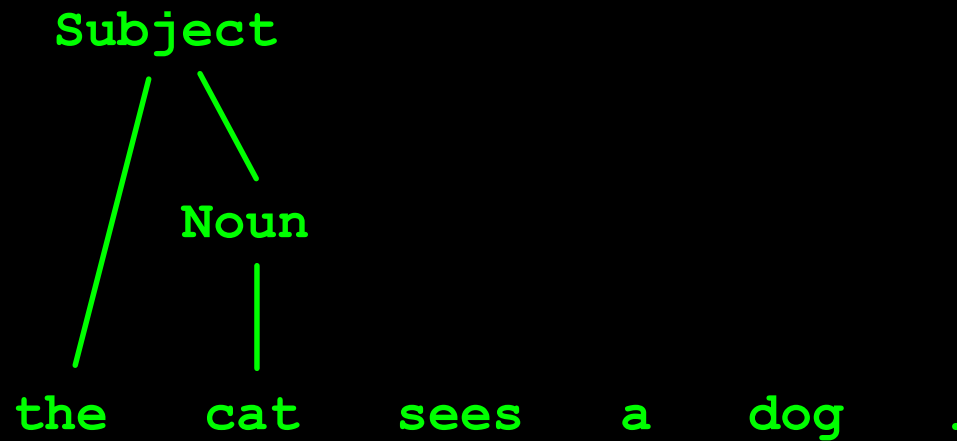
- "Subject Verb Object ." é reduzido para "Sentence";
- A análise chega na raiz da gramática e a cadeia de entrada está esgotada;
- A análise ascendente termina.

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

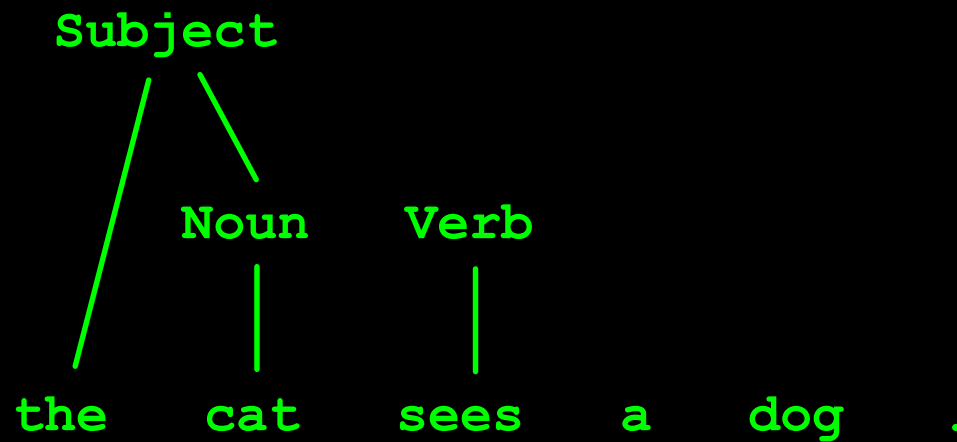
Análise ascendente

Noun
|
the cat sees a dog .

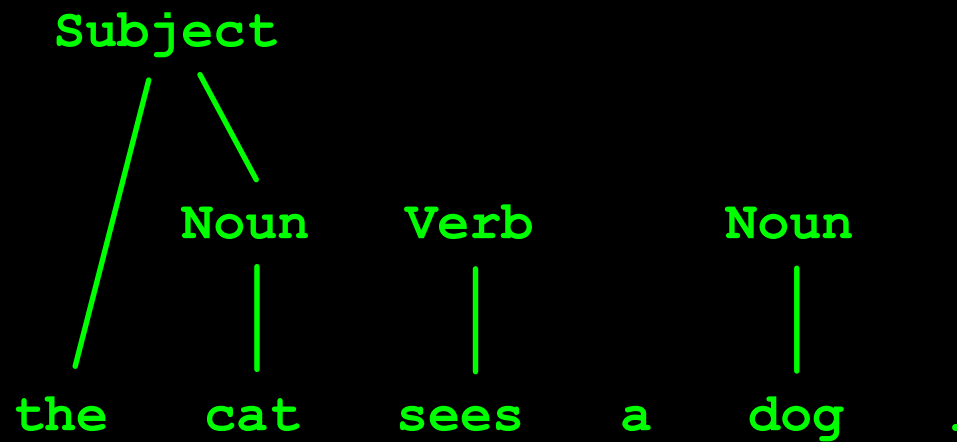
Análise ascendente



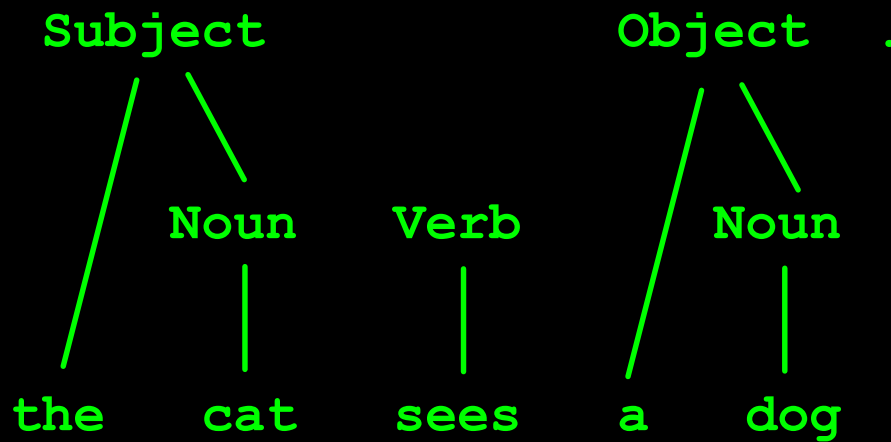
Análise ascendente



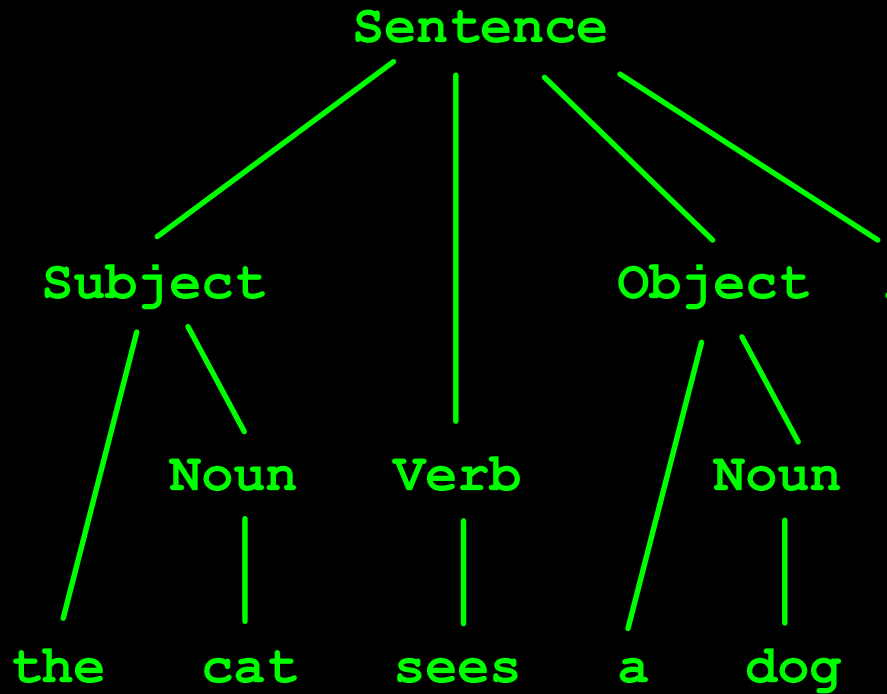
Análise ascendente



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 $A \rightarrow \beta, A \rightarrow \gamma \in P$, então $\text{first}_k(\beta\alpha) = \text{first}_k(\gamma\alpha)$ 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.
- Ou seja, espera-se que a escolha determinística da substituição de A possa ser feita levando-se em conta apenas o lookahead, independentemente da forma sentencial onde A ocorre mais à esquerda;
- 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 (desconsiderando a configuração corrente):

- 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 o conjunto delas, se $\text{first}_k(\beta_i\alpha) \cap \text{first}_k(\beta_j\alpha) = \emptyset$ para $1 \leq i, j \leq n, i \neq j$;
- Ou seja, se independentemente da forma sentencial a escolha da substituição de A pode ser feita levando-se em conta apenas o lookahead.

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 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 $\text{first}_k(\beta_i\alpha)$, para cada β_i , neste caso, são calculados levando em conta todas as possíveis cadeias α que possam comparecer do lado direito de A , para todas as formas sentenciais em que A aparece;
- 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 $LL(k)$

Alternativa usada na prática:

- Desconsiderar a forma sentencial corrente;
- Verificar, para cada não-terminal da gramática, se as suas regras podem ser escolhidas de forma determinística e qual o menor valor de k em cada caso;
- Cada não-terminal terá, portanto, um valor de k ;
- A gramática será $LL(k)$ se todos os não-terminais o forem e, neste caso, o valor do k da gramática será o maior de todos os k dos não-terminais analisados.

Exemplos e estratégias

No primeiro exemplo mostrado a seguir:

- A forma sentencial corrente não é considerada;
- A condição $LL(1)$ é verificada para os não-terminais S e X ;
- Logo, a gramática é $LL(1)$ e não necessita da forma sentencial corrente.

Exemplos e estratégias

No segundo exemplo apresentado a seguir:

- Sem considerar a forma sentencial corrente, o não-terminal S verifica a condição $LL(1)$;
- Sem considerar a forma sentencial corrente, o não-terminal A não verifica a condição $LL(k)$ para $k=1, 2, 3, 4, 5$ etc
- Levando-se em conta a forma sentencial corrente, o não-terminal A verifica a condição $LL(3)$;
- Portanto, levando-se em conta apenas S e A , a gramática é $LL(3)$ e necessita levar em conta a forma sentencial corrente (informação de contexto);
- O não-terminal B não é analisado.

Estratégias para verificar a condição LL(k)

Exemplo 1:

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

Considere a derivação do não-terminal X.

Em quais formas sentenciais ele comparece?

Resposta:

aX, bX, acX, bcX, accX, bccX, acccX, bcccccX 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 , $accX$, $bccX$, $acccX$, $bcccccX$ 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 2:

- No exemplo a seguir, analisamos o caso do não-terminal A ;
- Levando-se em conta apenas esse símbolo não-terminal, a gramática é $LL(3)$ se for considerada a configuração (forma sentencial) corrente no momento de se decidir a regra que deve ser escolhida;
- Se forma sentencial corrente não for considerada, veremos que a gramática não é $LL(3)$, nem $LL(4)$ nem $LL(5)$ (e paramos a análise por aí);
- Entretanto, podemos obter uma gramática equivalente que seja $LL(4)$ e também outra $LL(1)$ – ambas sem levar em conta a forma sentencial corrente.

Estratégias para verificar a condição LL(k)

Exemplo 2:

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

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.

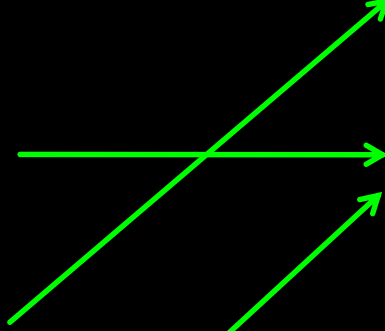
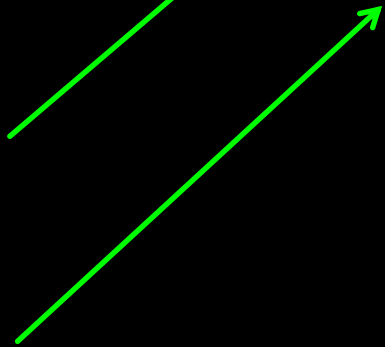
Estratégias para verificar a condição LL(k)

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?

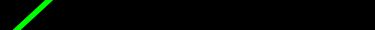
- $aAaB$ com $A \rightarrow a$
 $\{aaa\}$ 
- $aAaB$ com $A \rightarrow ab$
 $\{aba\}$ 
- $bAbB$ com $A \rightarrow a$
 $\{aba\}$ 
- $bAbB$ com $A \rightarrow ab$
 $\{abb\}$ 

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.

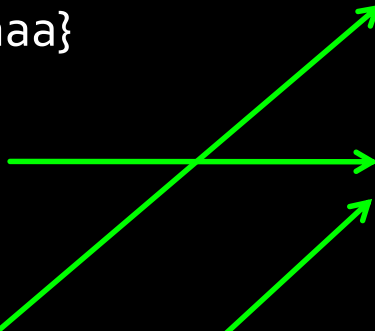
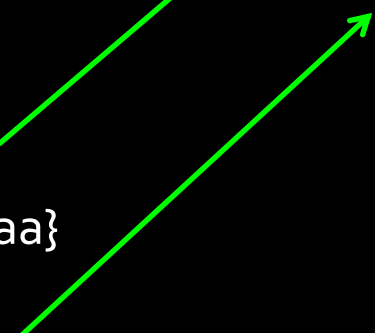
Estratégias para verificar a condição LL(k)

k=4

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

Estratégias para verificar a condição LL(k)

k=5

- aAaB com $A \rightarrow a$  {aaa↓↓, aaaa↓, aaaaa, aba↓↓, **abaa↓**, **abaaa**}
- aAaB com $A \rightarrow ab$  {**abaa↓**, **abaaa**, abba↓, abbaa}
- bAbB com $A \rightarrow a$  {aba↓↓, abaa↓, abaaa}
- bAbB com $A \rightarrow ab$  {abba↓, 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 se pode dispensar o uso da forma sentencial corrente para fazer a escolha da regra a ser aplicada?

- Considerar o conjunto de todas as 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 $X_1 = \text{first}_1(\beta_1 \gamma_1) \cup \dots \cup \text{first}_1(\beta_1 \gamma_n)$
- ...
- Seja $X_k = \text{first}_1(\beta_k \gamma_1) \cup \dots \cup \text{first}_1(\beta_k \gamma_n)$
- 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.

Estratégias para verificar a condição LL(k)

- Para todos os efeitos, a partir de agora, vamos desconsiderar a forma sentencial corrente como informação a ser usada na escolha da regra de derivação;
- Uma gramática será dita LL(k) se a escolha da regra puder sempre ser feita de forma determinística e levando-se em conta apenas o não-terminal corrente e os k símbolos do look-ahead;
- Ainda assim, precisamos lidar com gramáticas com regras que produzem a cadeia vazia, direta ou indiretamente. Quando aplicar a regra vazia?

Estratégias para verificar a condição LL(k)

- A regra vazia deve ser usada quando o símbolo corrente for o símbolo que vem à direita do não-terminal em análise.
- Assim, para que a escolha seja sempre determinística, deve-se garantir que o(s) símbolo(s) que comparecem à direita sejam diferentes dos símbolos que iniciam as regras não-vazias.
- É necessário definir $\text{follow}_k(\beta)$.

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

$$S \rightarrow aXbY\#$$

$$X \rightarrow aX$$

$$X \rightarrow a$$

$$Y \rightarrow bY$$

$$Y \rightarrow b$$

$$\text{follow}_1(X) = \{b\}$$

$$\text{follow}_1(Y) = \{\#\}$$

$$S \rightarrow aSXc\#$$

$$X \rightarrow bX$$

$$X \rightarrow \varepsilon$$

$$\text{follow}_1(S) = \{b, c\}$$

$$\text{follow}_1(X) = \{c\}$$

Estratégias

Com base na definição anterior:

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

Casos particulares

- Em seguida serão apresentados alguns casos particulares de gramáticas LL(1);
- Eles tem apenas finalidade didática e estão em ordem crescente de complexidade;
- Cada gramática deve ser analisada conforme as suas especificidades;
- Em todos os casos, utilizamos o critério LL(1) sem considerar a forma sentencial corrente. Ou seja, trata-se de gramáticas que geram linguagens que podem ser analisadas de forma descendente levando-se em conta apenas o não-terminal e o símbolo de entrada corrente.

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

Se o símbolo corrente pertence ao $\text{follow}_1(A)$, a regra $A \rightarrow \varepsilon$ 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' :
 $first_1(A\#) = \{i\}$
 - A :
 $first_1(iB \leftarrow e) = \{i\}$
 - B :
 $first_1(SB) \cap follow_1(B) = \{[, .\} \cap \{\leftarrow\} = \emptyset$
 - S :
 $first_1([eC]) \cap first_1(.i) = \{[\} \cap \{.\} = \emptyset$
 - C :
 $first_1(eC) \cap follow_1(C) = \{e\} \cap \{]\} = \emptyset$

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:

$$\begin{aligned} S &\rightarrow X \sigma_1 \\ X &\rightarrow X \sigma_2 \mid \sigma_3 \end{aligned}$$

As formas sentenciais em que X pode comparecer são:

$$\begin{aligned} &X \sigma_1 \\ &X \sigma_2 \sigma_1 \\ &X \sigma_2 \sigma_2 \sigma_1 \\ &X \sigma_2 \sigma_2 \sigma_2 \sigma_1 \\ &X \sigma_2 \sigma_2 \sigma_2 \sigma_2 \sigma_1 \\ &\dots \end{aligned}$$

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 $\sigma_3 \sigma_2^{k-1} \sigma_1$ 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 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

De outra forma:

- Suponha a gramática anterior e suponha que a regra $X \rightarrow X \sigma_2$ foi usada n vezes na derivação de uma cadeia de entrada, produzindo a sentença $\sigma_3 \sigma_2^n \sigma_1$; a forma sentencial corrente é, genericamente, $X \sigma_2^m \sigma_1$, $m \leq n$. **Como então escolher uma regra para X em $X \sigma_2^m \sigma_1$ de tal forma que a análise seja determinística?**
- Observe que o look-ahead σ_3 pertence simultaneamente aos conjuntos $\text{first}_1(X \sigma_2)$ e $\text{first}_1(\sigma_3)$; da mesma forma, a cadeia $\sigma_3 \sigma_2$ pertence simultaneamente aos conjuntos $\text{first}_2(X \sigma_2)$ e $\text{first}_1(\sigma_3)$ e assim por diante até $\sigma_3 \sigma_2^n$; ou seja, a cadeia $\sigma_3 \sigma_2^n$ pertence simultaneamente aos conjuntos $\text{first}_{n+1}(X \sigma_2)$ e $\text{first}_{n+1}(\sigma_3)$; logo, cadeias de comprimento $n+1$ não permitem fazer a escolha determinística da regra a ser usada;
- Apenas a look-ahead $\sigma_3 \sigma_2^n \sigma_1$ (de comprimento $n+2$) permite fazer a escolha da regra correta a ser aplicada; se $n=0$, deve-se usar a regra $X \rightarrow \sigma_3$; se $n \geq 1$, deve-se usar a regra $X \rightarrow X \sigma_2$;
- Mas como não sabemos quantas vezes no máximo a regra $X \rightarrow X \sigma_2$ foi usada na forma sentencial corrente, então **não é possível estimar um look-ahead de comprimento máximo para a gramática.**

Recursão à esquerda

Exemplo:

- Suponha a sentença $\sigma_3 \sigma_1$; neste caso ($n=0$), basta um look-ahead de comprimento 2 para decidir entre a regra $X \rightarrow \sigma_3 (\sigma_3 \sigma_1)$ e a regra $X \rightarrow X \sigma_2 (\sigma_3 \sigma_2)$ na forma sentencial $X \sigma_1$;
- Suponha a sentença $\sigma_3 \sigma_2 \sigma_1$; neste caso ($n=1$), basta um look-ahead de comprimento 3 para decidir entre a regra $X \rightarrow \sigma_3 ()$ e a regra $X \rightarrow X \sigma_2 ()$ na forma sentencial $X \sigma_1$;

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	$X \rightarrow Xb$ ou $X \rightarrow a$
2	$ac, $ ab	$X \rightarrow Xb$ ou $X \rightarrow a$
3	$ac, abc, $ abb	$X \rightarrow Xb$ ou $X \rightarrow a$
4	$ac, abc, abbc, $ abbb	$X \rightarrow Xb$ ou $X \rightarrow a$
...	...	...
n	$\dots, $ ab^{n-1} $, \dots$	$X \rightarrow Xb$ 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 **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 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:

$$S \rightarrow ab^*c$$

Recursão à esquerda

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

Suponha que G contenha as seguintes regras:

$$X \rightarrow Xb \mid a$$

Suponha $k=1$:

$\text{first}_1(Xb)$ inclui "a" e $\text{first}_1(a)$ inclui "a"

Suponha $k=2$:

$\text{first}_1(Xb)$ inclui "ab" e $\text{first}_1(a)$ inclui "ab"

Suponha $k=3$:

$\text{first}_1(Xb)$ inclui "abb" e $\text{first}_1(a)$ inclui "abb"

...

Recursão à esquerda

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

Suponha que G contenha as seguintes regras: $X \rightarrow Xb \mid a$

Suponha que a cadeia de entrada seja:

... abbbb ... bbb ...

k símbolos

e o símbolo corrente da entrada seja "a".

- Por mais que se olhe k símbolos de lookahead, sempre existe a possibilidade de o próximo símbolo (depois dos k símbolos) ser "b" e, assim, não ser possível fazer a escolha determinística da regra a ser usada.

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:

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

Note que a forma sentencial corrente não é considerada.

Gramática LL(1) - alternativa

Considere a GLC G:

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

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

...

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

$$\text{first}_1(\gamma_{1i}).\text{follow}_1(X_1) \cap \text{first}_1(\gamma_{1j}).\text{follow}_1(X_1) = \emptyset, 1 \leq i, j \leq m, i \neq j.$$

$$\text{first}_1(\gamma_{2i}).\text{follow}_1(X_2) \cap \text{first}_1(\gamma_{2j}).\text{follow}_1(X_2) = \emptyset, 1 \leq i, j \leq n, i \neq j.$$

...

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

Note que a forma sentencial corrente não é considerada.

Uso de parênteses

Numa regra $X \rightarrow \dots (\alpha \mid \beta) \dots$

O termo $(\alpha \mid \beta)$ se comporta como se fossem regras de um “símbolo não-terminal anônimo (ou implícito)”, representado pelos parênteses;

Ou seja, é como se existisse um certo símbolo não-terminal Y com as regras:

$$Y \rightarrow \alpha \mid \beta$$

Naturalmente, deve-se considerar $X \rightarrow \dots (\alpha \mid \beta) \dots$ como $X \rightarrow \dots Y \dots$

- Independentemente de o não-terminal ser explícito (Y) ou implícito (com regras agrupadas por meio de parênteses), a condição LL(1) deve sempre ser verificada. No caso, deve-se garantir a escolha determinística da regra α ou da regra β por meio de look-aheads apropriados.

Outros exemplos (I)

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

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

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

Torna-se LL(1), pois:

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

Outros exemplos (III)

- $S \rightarrow aXe$
 - $X \rightarrow bXY \mid c \mid \varepsilon$
 - $Y \rightarrow dY \mid c$
- O não-terminal S satisfaz à condição $LL(1)$ pois há uma única regra para o mesmo;
- O não-terminal Y satisfaz também pois os conjuntos first são disjuntos;
- A gramática, no entanto, não é $LL(1)$ pois X não satisfaz à condição $LL(1)$:
- X :
 $first_1(bXY) = \{b\}$
 $first_1(c) = \{c\}$
 $follow_1(X) = \{c, d, e\}$

Outros exemplos (III)

Tentativa de manipulação gramatical:

- Eliminação da recursão à direita de Y ;
- Substituição e eliminação de Y ;

Produz a seguinte gramática equivalente:

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

Outros exemplos (III)

O não-terminal S satisfaz à condição LL(1) pois:

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

No entanto, o não-terminal X continua não satisfazendo à condição LL(1), pois:

- $X :$
 $\text{first}_1(bXd^*c) = \{b\}$
 $\text{first}_1(c) = \{c\}$
 $\text{follow}_1(X) = \{c, d\}$

Outros exemplos (IV)

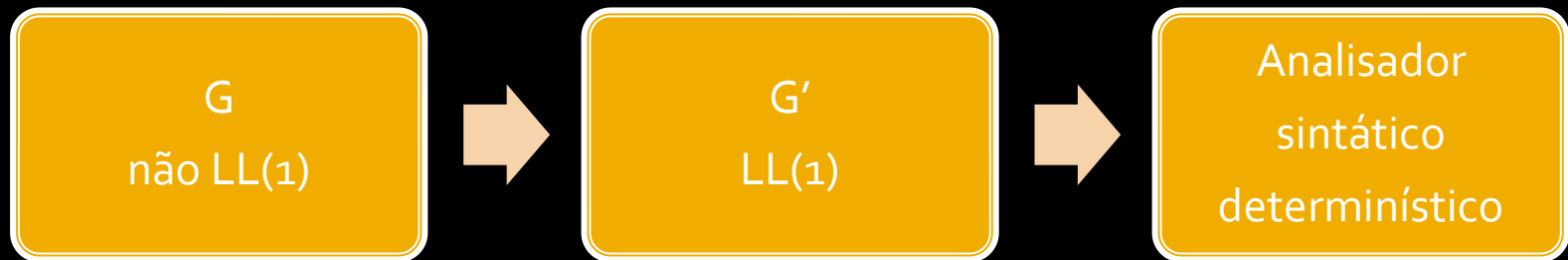
- $S \rightarrow aXe$
- $X \rightarrow bXY \mid c \mid \varepsilon$
- $Y \rightarrow dY \mid f$

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.

Fatoração à esquerda

$$X \rightarrow \alpha\beta \mid \alpha\gamma$$

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:

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

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

Substituições

$$X \rightarrow \alpha Y \beta$$

$$Y \rightarrow \gamma \mid \omega$$

O não-terminal Y é substituído pela sua definição:

$$X \rightarrow \alpha(\gamma \mid \omega)\beta$$

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

Recursões à esquerda

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

O não-terminal X é substituído por:

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

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):

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