LINGUAGENS FORMAIS: Teoria e Conceitos

Material adicional, versão do dia 12 de maio de 2024 às 15:33

1. Página 51:
   Acrescentar no penúltimo parágrafo, logo depois de "do conjunto original.":
   Em outras palavras, suponha que A e B sejam conjuntos finitos. Então, se $|A| = |B|$, A e B necessariamente possuem a mesma quantidade de elementos. Se A possuir mais elementos do que B, então $|A| > |B|$. Suponha agora que A e B sejam conjuntos infinitos. Então, se $|A| = |B|$, A e B podem ou não possuir a mesma quantidade de elementos. Ou seja, o fato de um conjunto infinito possuir mais elementos do que outro conjunto infinito não garante que as suas respectivas cardinalidades sejam diferentes (exemplo: $\mathbb{Z}$ possui mais elementos do que $\mathbb{N}$ mas ambos possuem a mesma cardinalidade; $\mathbb{R}$ possui mais elementos do que $\mathbb{N}$ e a cardinalidade do primeiro é maior do que a do segundo). Se A é finito e B é infinito, então necessariamente $|A| < |B|$.

2. Página 69:
   Acrescentar, logo depois do Exemplo 2.13:
   Note que a operação de concatenação é distributiva em relação à operação de união, tanto à esquerda quanto à direita:
   $$X(Y \cup Z) = XY \cup XZ$$
   $$(X \cup Y)Z = XZ \cup YZ$$

3. Página 80:
   Acrescentar, antes do Exemplo 2.31:
   Uma gramática $G = (V, \Sigma, P, S)$ é dita "bem formada" se e somente se:

   - $\Sigma \subset V$.
   - Σ é finito e não-vazio.
   - $N = V - \Sigma$ é finito e não-vazio.
   - $S \in N$.
   - Se $\alpha \to \beta \in P$, então $\alpha \in V^* N V^*$ e $\beta \in V^*$.

   Caso contrário, diz-se que G está "malformada".

4. Página 82:
   Acrescentar, antes do parágrafo "A completa identificação...":
   Para que uma gramática G defina uma linguagem L, as duas condições seguintes devem ser observadas simultaneamente:

   - Toda sentença de L pode ser gerada por G.
   - Toda cadeia não pertencente à L não pode ser gerada por G (ou seja, não existe nenhuma cadeia não pertencente à L que possa ser derivada em G).

   Exemplo 2.34 Considere a linguagem L formada por todas as cadeias sobre o alfabeto $\{a, b\}$ que começam com a e terminam com b. A gramática G_i com as regras $S \to aXb$, $X \to aX$, $X \to bX$, $X \to a$, $X \to b$ não gera L pois falha em gerar a sentença ab, pertencente à ela (viola a primeira condição acima). Ou seja, $L(G_i) = L - \{ab\}$. Por outro lado, a gramática G_{ii} com as regras $S \to aXb$, $S \to a$, $X \to aX$, $X \to bX$, $X \to \varepsilon$ não gera L pois permite a derivação da cadeia a, que não pertence à L (viola a segunda condição acima). Ou seja, $L(G_{ii}) = L \cup \{a\}$. A gramática G_{iii} com as regras $S \to aXb$, $X \to aX$, $X \to bX$, $X \to \varepsilon$ gera exatamente L, pois ambas as condições acima são verificadas.

   Note que a escolha do não-terminal inicial (raiz da gramática) é determinante para a formação da linguagem gerada pela respectiva gramática.

Exemplo 2.35 Considere novamente o Exemplo 2.33. Se a raiz de G_1 fosse A e não S_1, então $L(G_1) = \{\varepsilon, 12\}$. Ou seja, a linguagem seria finita e composta por duas sentenças, uma de comprimento zero (ε) e outra de comprimento dois (12). O símbolo não-terminal S_1, neste caso, não teria nenhuma utilidade na gramática e na linguagem gerada por ela.

5. Página 82:
   Parágrafo "A completa identificação...":
   onde se lë:
   " Observe-se, a título de ilustração, o caso da gramática G_2 apresentada no Exemplo 2.34."
   leia-se:
   " Observe-se, a título de ilustração, os casos das gramáticas G_2 e G_3 apresentadas, respectivamente, nos Exemplos 2.36 e 2.37."

6. Página 82:
   Acrescentar, antes do Exemplo 2.34:
   Exemplo 2.36 A gramática $G_2 = (\{a,b,S,X\},\{a,b\},\{S \to aSb, S \to X, X \to ab\},S)$ é tal que $L(G_2) = \{a^n b^n$ com $n \geq 1\}$, portanto infinita. Trata-se de uma gramática simples cuja respectiva linguagem gerada não é difícil de ser deduzida. De fato, $S \Rightarrow^i a^i S b^i \Rightarrow a^i X b^i \Rightarrow a^i abb^i = a^{i+1} b^{i+1}$.

   Uma pequena mudança na linguagem pode introduzir uma grande complexidade na correspondente gramática. Veja o Exemplo 2.37, que apresenta uma gramática que gera a linguagem $a^n b^n c^n$ com $n \geq 1$.

7. Página 82:
   Exemplo 2.34:
   Substituir *Exemplo* 2.34 por *Exemplo* 2.37
   substituir G_2 por G_3, V_2 por V_3, Σ_2 por Σ_3 e P_2 por P_3

8. Página 83:
   Exemplo 2.35:
   Substituir *Exemplo* 2.35 por *Exemplo* 2.38
   substituir G_3 por G_4 e G_4 por G_5

9. Página 94:
   Acrescentar, antes do parágrafo "É possível...":
   Diz-se que um reconhecedor R define uma linguagem L quando:

   - Todas as sentenças de L são aceitas por R, e
   - Todas as cadeias não pertencentes à L são rejeitadas por R (ou seja, não existe nenhuma cadeia não pertencente à L que seja aceita por R).

10. Página 110:
    Acrescentar, antes do Exemplo 3.1:
    ou seja, $\alpha \in N$ e $\beta \in \{\varepsilon\} \cup \Sigma \cup N \cup \Sigma N$. Logo, $|\alpha| = 1$ e $|\beta| \leq 2$.

11. Página 110:
    Acrescentar, antes do Exemplo 3.2:
    ou seja, $\alpha \in N$ e $\beta \in \{\varepsilon\} \cup \Sigma \cup N \cup N\Sigma$. Logo, $|\alpha| = 1$ e $|\beta| \leq 2$.

12. Página 114:
    Substituir o parágrafo "A constatação..." por:
    As regras da GLD G_1 empregadas na derivação anterior são as seguintes, por ordem de uso:

$$
\begin{array}{c}
S \to \mu_n A_1 \\
A_1 \to \mu_{n-1} A_2 \\
A_2 \to \mu_{n-2} A_3 \\
A_3 \to \mu_{n-3} A_4 \\
\ldots \\
A_{n-3} \to \mu_3 A_{n-2} \\
A_{n-2} \to \mu_2 A_{n-1} \\
A_{n-1} \to \mu_1
\end{array}
$$

Por outro lado, a Tabela 3.1 mostra que, para cada uma das regras acima da GLD G_1, existe uma regra correspondente na GLE G_2, a saber:

Regra da GLD G_1	Regra da GLE G_2
$S \to \mu_n A_1$	$A_1 \to \mu_n$
$A_1 \to \mu_{n-1} A_2$	$A_2 \to A_1 \mu_{n-1}$
$A_2 \to \mu_{n-2} A_3$	$A_3 \to A_2 \mu_{n-2}$
$A_3 \to \mu_{n-3} A_4$	$A_4 \to A_3 \mu_{n-3}$
...	...
$A_{n-3} \to \mu_3 A_{n-2}$	$A_{n-2} \to A_{n-3} \mu_3$
$A_{n-2} \to \mu_2 A_{n-1}$	$A_{n-1} \to A_{n-2} \mu_2$
$A_{n-1} \to \mu_1$	$S \to A_{n-1} \mu_1$

Finalmente, a aplicação destas regras da GLD G_2, porém na ordem inversa, garante que a sentença derivada na GLD G_1 possa ser derivada na GLE G_2. Em outras palavras, temos que $L(G_1) \subseteq L(G_2)$.

13. Página 115:
Substituir "A Tabela 3.1" por:
De forma similar ao caso anterior, a Tabela 3.1

14. Página 174:
Inserir, antes do prágrafo "Em resumo:"
Em outras palavras: antes da execução do passo 2 do Algoritmo 3.14, as equações das variáveis X_i (no lado esquerdo) podem possuir X_j, $1 \leq j \leq m$ (ou seja, qualquer variável) do lado direito. Quando o passo 2 é executado sobre a equação da variável X_i, esta passa a conter, do lado direito, apenas referências à X_j, $j \geq i$. Ao término do passo 2, todas as equações de X_i contém referências apenas para X_j, $j \geq i$. Portanto, a última equação (de X_m, que contém apenas referências ao próprio X_m, no pior caso) é resolvida (por meio do Teorema 3.13) e tem início o passo 4 do algoritmo. Quando a solução de um certo X_i (obtida pelo Teorema 3.13) é substituida em todas as equações anteriores (das variáveis X_j, $j < i$), estas equações passam a conter, do lado direito, apenas a própria variável que está sendo definida. Uma equação com, por exemplo X_i do lado esquerdo, conterá apenas X_i do lado direito. Para resolver esta equação, basta usar novamente o Teorema 3.13.

15. Página 181:
Substituir, na penúltima linha, "autômato finito M" por:
"autômato finito M (eventualmente não determinístico e com transições em vazio)

16. Página 184:
Inserir, debaixo da linha "$\Sigma = \{a,b,c\}$" e antes de "Considere-se a cadeia *abbbc*":

$$
\begin{array}{rcl}
P & = & \{q_0 \to aq_1, \\
 & & q_1 \to bq_1, \\
 & & q_1 \to cq_2, \\
 & & q_1 \to q_2, \\
 & & q_2 \to cq_2, \\
 & & q_2 \to \varepsilon\}
\end{array}
$$

17. Página 221:
    Onde se lê:
    "uma sentença w de comprimento suficientemente longo pertencente à linguagem,"
    Leia-se:
    "uma sentença w de comprimento suficientemente longo pertencente à linguagem (sem, no entanto, especificar qual seria esta sentença),"

18. Página 222:
    Substituir "Logo, L não é regular." por:
    Portanto, $m*(1+\mid y\mid)$ é divisível por um número que não é 1 nem $m*(1+\mid y\mid)$. Logo. $m*(1+\mid y\mid)$ não é um número primo, a hipótese é falsa e L não é regular.

19. Página 235:
    Inserir, antes do Teorema 3.34 e depois do Exemplo 3.85:
    "Uma maneira alternativa de verificar se $L=\emptyset$ consiste em determinar o conjunto de estados acessíveis do autômato (Teorema 3.7) e verificar se ele contém algum estado final. Ou, ainda, determinar o conjunto de estados úteis do autômato (Teorema 3.8) e verificar se ele contém o estado inicial. Este método pode ser usado em qualquer tipo de autômato finito."

20. Página 341:
    Inserir uma nova Seção 4.11, conforme abaixo:

4.11 Pertencimento

Dada uma gramática livre de contexto $G=(V,\Sigma,P,S)$ e uma cadeia $w\in\Sigma^*$, como determinar se $w\in$ Ł(G)? A reposta para esta pergunta, fundamental na Teoria de Linguagens, é essencial para a construção de analisadores sintáticos, que por sua vez são necessários à construção de processadores de linguagens (compiladores e interpretadores).

Existem diversas maneiras de responder à esta pergunta, algumas simples e ineficientes, outras mais complexas porém eficientes. Iniciaremos com algoritmos de tempo exponencial, tendo por base as Formas Normais de Chomsky e de Greibach. Em seguida, apresentaremos o algoritmo CYK, que gera a resposta em tempo polinomial. Nas Seções 4.12 e 4.13 serão apresentadas técnicas de análise sintática determinística que permitem que a resposta seja gerada em tempo linear com o comprimento da cadeia de entrada (w).

Forma Normal de Chomsky

(a) Obter G' na Forma Normal de Chomsky ($A\rightarrow BC\mid a$), tal que $L(G)=L(G')$.

(b) Considerar $n=|w|$.

(c) Se $n>0$, então fazer todas as derivações com $2*n-1$ passos.

(d) Se $n=0$, então fazer todas as derivações com 1 passo.

(e) Se alguma dessas derivações gera w, então $w\in L(G)$.

(f) Caso contrário, $w\notin L(G)$..

A geração de uma cadeia de comprimento n numa gramática na Forma Normal de Chomsky demanda, necessariamente, a aplicação de $n-1$ regras do tipo $A\rightarrow BC$ (para gerar uma forma sentencial com n símbolos não-terminais) e também a aplicação de n regras do tipo $A\rightarrow a$ para transformar a forma sentencial numa sentença. Portanto, a geração de uma cadeia de n símbolos requer a aplicaçao de $2*n-1$ regras ou passos de derivação. Existe um conjunto finito de seqüências de derivação com qualquer quantidade de passos. Basta obter todas elas e verificar se alguma produz a cadeia informada na entrada. Em caso afirmativo, a cadeia pertence à linguagem. Em caso negativo, ela não pertence.

Suponha que o maior número de regras de um mesmo símbolo não terminal em G seja k. Então, uma árvore de derivação com $2n-1$ passos gera no máximo:

$$k^{2n-1}$$

formas sentenciais após os $2n-1$ passos. Em seguida, estas formas sentenciais devem ser inspecionadas para verificar se alguma delas corresponde à w e assim determinar a resposta do algoritmo. Logo, o tempo deste algoritmo é exponencial em n, onde $n = |w|$. Note que este tempo não leva em conta o tempo necessário para a geração de todas as formas sentenciais da árvore de derivação. O número total de formas sentenciais da árvore é:

$$1 + k + k^2 + ... + k^{2n-1} = \sum_{i=0}^{2n-1} k^i = \frac{k^{2n}-1}{k-1}$$

portanto o tempo total é exponencial também.

Exemplo 4.56 Considere G com as seguintes regras:

$$\begin{aligned} G &= (\{S,A,B,a,b,c\},\{a,b,c\},P,S) \\ P &= \{S \to AB \mid c, \\ &\quad A \to AA \mid a, \\ &\quad B \to BB \mid b\} \end{aligned}$$

Portanto, $k = 2$. Considere-se a cadeia abb:

- $n = |abb| = 3$.
- Deve-se pesquisar todas as seqüências de derivação com até $2 * 3 - 1 = 5$ passos.
- Para simplificar, serão consideradas apenas derivações mais à esquerda.

Na Figura 1 são apresentados os vários caminhos para a geração da cadeia aab em G.

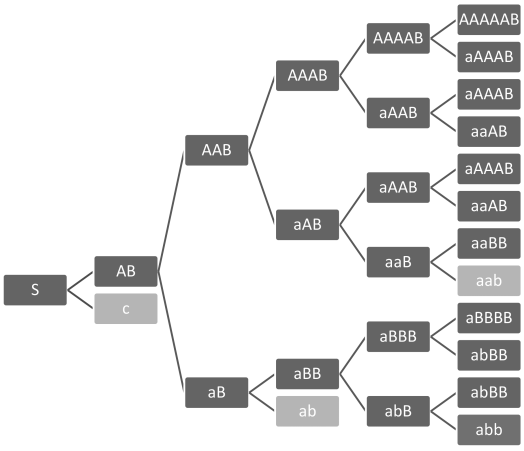

Figura 1: Sequências de derivações para a cadeia aab

No máximo, 32 formas sentenciais precisariam ser examinadas ($2^{2*3-1} = 2^5 = 32$). No entanto, para esta gramática em particular, apenas 12 formas sentenciais distintas são geradas.

Por último, cumpre notar que simular G diretamente pode não funcionar, pois pode haver seqüências infinitas de derivações em G. Por exemplo, se G possuir as regras unitárias $X \to Y$ e $Y \to X$, é possível que a seqüência de derivações torne-se infinita:

$$... \Rightarrow \alpha X \beta \Rightarrow \alpha Y \beta \Rightarrow \alpha X \beta \Rightarrow \alpha Y \beta \Rightarrow ...$$

Forma Normal de Greibach

Como todas as regras iniciam com um símbolo terminal, é suficiente gerar todas as formas sentenciais que produzem cadeias do comprimento da cadeia de entrada, e depois verificar se alguma delas corresponde à mesma.

(a) Obter G' na Forma Normal de Greibach, tal que $L(G') = L(G)$.

(b) Considerar $n = |w|$.

(c) Considerar k como o maior número de regras definido para um símbolo não terminal, entre todos os não terminais de G'.

(d) Obter todas as sequências de derivações mais à esquerda que geram formas sentenciais cujo prefixo seja uma cadeia de terminais de comprimento máximo n (existem no máximo k^n sequências distintas).

(e) Verificar se alguma dessas sequências de derivação corresponde à geração da cadeia w; em caso afirmativo, $w \in L(G)$; caso contrário, $w \notin L(G)$.

O número de formas sentenciais geradas por este método, com prefixo formado por n símbolos terminais, é:

$$k^n$$

Logo, o tempo é exponencial com o comprimento da cadeia de entrada n. Note que este tempo não leva em conta o tempo necessário para a geração de todas as formas sentenciais da árvore de derivação. O número total de formas sentenciais da árvore é:

$$1 + k + k^2 + ... + k^n = \sum_{i=0}^{n} k^i = \frac{k^{2n+1} - 1}{k - 1}$$

portanto o tempo total é exponencial também.

Exemplo 4.57 Considere-se a gramática a seguir, já apresentada na Forma Normal de Greibach:

$$\begin{aligned} G &= (\{S,B,C,a,b,c\},\{a,b,c\},P,S) \\ P &= \{S \to aBC \mid bBC, \\ &\quad B \to bB \mid b, \\ &\quad C \to c\} \end{aligned}$$

Considere-se a cadeia $abbc \in L(G)$. Então, $n = 4$ e $k = 2$ (pois existem duas produções para S, duas para B e apenas uma para C). Existem no máximo $2^4 = 16$ sequências distintas de derivações mais à esquerda que geram como prefixo uma cadeia de terminais de comprimento máximo 4, não havendo necessidade de se inspecionar outras sequências. A Tabela 1 relaciona todas as oito sequências (para o caso particular desta gramática) que geram cadeias de terminais de comprimento máximo 4. Observe que, no caso particular desta gramática, o número de formas sentenciais que deve ser inspecionado é apenas 6.

Tabela 1: Derivações mais à esquerda que geram prefixos de comprimento máximo 4

Sequência			ΣV^*		$\Sigma\Sigma V^*$		$\Sigma\Sigma\Sigma V^*$		$\Sigma\Sigma\Sigma\Sigma V^*$
1	S	$\Rightarrow$	aBC	$\Rightarrow$	$abBC$	$\Rightarrow$	$abbBC$	$\Rightarrow$	$abbbBC$
2								$\Rightarrow$	$abbbC$
3						$\Rightarrow$	$abbC$	$\Rightarrow$	$abbc$
4				$\Rightarrow$	abC	$\Rightarrow$	abc		
5		$\Rightarrow$	bBC	$\Rightarrow$	$bbBC$	$\Rightarrow$	$bbbBC$	$\Rightarrow$	$bbbbBC$
6								$\Rightarrow$	$bbbbC$
7						$\Rightarrow$	$bbbC$	$\Rightarrow$	$bbbc$
8				$\Rightarrow$	bbC	$\Rightarrow$	bbc		

Como se pode verificar, a terceira sequência de derivações produz a cadeia $abbc$. Por outro lado, a cadeia $bcbc$, também de comprimento 4, não pertence à linguagem, uma vez que nenhuma das oito sequências da Tabela 1 gera o prefixo $bcbc$ e, portanto, nenhuma sequência de derivações é capaz de gerar a cadeia $bcbc$. O tempo de execução deste algoritmo é proporcional a k^n, onde n é o comprimento da cadeia de entrada. Logo, esse tempo varia exponencialmente com o tamanho da cadeia, o que é um resultado considerado ineficiente e, portanto, indesejável do

ponto de vista prático.

Algoritmo CYK

O Algoritmo CYK[1], diferentemente dos algoritmos anteriores, de tempo exponencial, possui tempo polinomial, variando com o cubo do comprimento da cadeia de entrada ($O(n^3)$). Ao contrário daqueles, no entanto, o Algoritmo CYK apenas verifica se uma certa cadeia w pertence à linguagem gerada por uma gramática livre de contexto, não informando as regras que foram usadas na geração da mesma. Todos os algoritmos apresentados nesta seção funcionam para qualquer linguagem livre de contexto. O exemplo apresentado a seguir trata do pertencimento à uma linguagem livre de contexto não determinística. Os métodos discutidos nas Seções 4.12 e 4.13 possuem tempos lineares (portanto melhores do que o CYK), porém funcionam apenas para linguagens livres de contexto determinísticas.

O algoritmo funciona, em essência, identificando os símbolos não terminais de G que geram subcadeias w de comprimentos sucessivamente maiores, até que se chegue ao comprimento n. O Algoritmo CYK pressupõe que a gramática em questão, digamos $G = (V, \Sigma, P, S)$, está na Forma Normal de Chomsky. Ele considera a tabela apresentada a seguir:

n	X_{1n}						
$n-1$	$X_{1(n-1)}$	X_{2n}					
$n-2$	$X_{1(n-2)}$	$X_{2(n-1)}$	X_{3n}				
...							
3	X_{13}	X_{24}	X_{35}	...	$X_{(n-2)n}$		
2	X_{12}	X_{23}	X_{34}	...	...	$X_{(n-1)n}$	
1	X_{11}	X_{22}	X_{33}	...	...	...	X_{nn}
	σ_1	σ_2	σ_3	...	...	...	σ_n

Nesta tabela os símbolos $\sigma_1\sigma_2...\sigma_n$ dispostos na linha horizontal inferior representam a cadeia de entrada $w \in \Sigma^*$ com comprimento n. As linhas são numeradas de baixo para cima, de 1 até n, e referem-se aos comprimentos das diferentes subcadeias que podem ser encontradas em w. A tabela é então preenchida com X_{ij}, onde cada X_{ij} representa um conjunto de símbolos não terminais capazes de derivar a cadeia $\sigma_i\sigma_{i+1}...\sigma_j$. Por exemplo, na linha 2 encontramos X_{12}, X_{23}, X_{34}, ..., $X_{(n-1)n}$, cada um dos quais contém os símbolos não terminais capazes de derivar, respectivamente, $\sigma_1\sigma_2$, $\sigma_2\sigma_3$, $\sigma_3\sigma_4$, ..., $\sigma_{n-1}\sigma_n$.

Trata-se, portanto, de preencher a tabela com os valores de X_{ij}. Caso $S \in X_{1n}$, então $w \in L(G)$. Caso contrário, $w \notin L(G)$. O método para calcular X_{ij} é apresentado a seguir, e as linhas são preenchidas de baixo para cima (ou seja, da linha ! até a linha n).

No início, os valores de X_{ij}, com $i = j$, são determinados diretamente. Como se trata de cadeias de comprimento 1, e como G está na Forma Normal de Chomsky, a única possibilidade de geração destas cadeias unitárias é através de regras to tipo $A \to a$. Em outras palavras, X_{11} contém os símbolos não terminais capazes de derivar diretamente σ_1, X_{22} contém os símbolos não terminais capazes de derivar diretamente σ_2 e assim por diante. Portanto, a linha 1 da tabela é preenchida diretamente.

As demais linhas da tabela (2 até n, nesta ordem) são preenchidas como segue. Antes disso cumpre observar que, ao preecher uma linha, todas as linhas inferiores já estarão preenchidas. Desta maneira, todas as possibilidades de geração de cadeias de comprimento menor que o comprimento corrente (o que inclui prefixos e sufixos próprios), assim como os não terminais que as geram, já terão sido identificados.

Note que os índices i e j de X_{ij} não se referem, respectivamente, à linha e à coluna do mesmo, como seria usual. A linha onde X_{ij} ocorre é dada por $j - i + 1$ e a coluna corresponde ao valor de i.

Considere então X_{ij}, com $i \neq j$ (o caso $i = j$ foi tratado anteriormente e corresponde à linha 1 da tabela). O conjunto de símbolos não terminais que definem o valor de X_{ij} são os A tais que $A \Rightarrow \sigma_i...\sigma_j$. Como a geração não é direta,

[1] O nome refere-se às letras iniciais de Cocke, Younger e Kasami, respectivamente John Cocke, Daniel Younger e Tadao Kasami que, juntamente com Jacob T. Schwartz, descobriram o algoritmo de forma independente. O algoritmo foi publicado pela primeira vez por Itiroo Sakai em 1961 (citar sakai1961).

e G está na Forma Normal de Chomsky, segue que uma regra do tipo $A \rightarrow BC$ deve ser usada. Existem então $j-i$ possibilidades para se gerar esta cadeia:

- $B \Rightarrow \sigma_i$ e $C \Rightarrow \sigma_{i+1}...\sigma_j$.
- $B \Rightarrow \sigma_i\sigma_{i+1}$ e $C \Rightarrow \sigma_{i+2}...\sigma_j$.
- $B \Rightarrow \sigma_i\sigma_{i+1}\sigma_{i+2}$ e $C \Rightarrow \sigma_{i+3}...\sigma_j$.
- ...
- $B \Rightarrow \sigma_i\sigma_{i+1}\sigma_{i+2}...\sigma_{j-1}$ e $C \Rightarrow \sigma_j$.

ou seja, $B \Rightarrow \sigma_i...\sigma_k$ e $C \Rightarrow \sigma_{k+1}\sigma_j$ para $i \leq k < j$. Todas estas possibilidades devem então ser consideradas para determinar o valor de X_{ij}. Em outras palavras, para determinar se $A \in X_{ij}$ deve-se considerar todos os casos válidos em que:

- $i \leq k < j$.
- $B \in X_{ik}$.
- $C \in X_{(k+1)j}$.
- $A \rightarrow BC$ é uma regra de G.

. Note que, para cada X_{ij}, com $1 \leq i, j \leq n$, $i \neq j$, os seguintes pares são comparados:

- Começando com X_{ii} com $X_{(i+1)j}$.
- Depois $X_{i(i+1)}$ com $X_{(i+2)j}$.
- ...
- Até $X_{i(j-1)}$ com X_{jj}.

Portanto, o conjunto X_{ij} é obtido a partir de $X_{ii}X_{(i+1)j} \cup X_{i(i+1)}X_{(i+2)j} \cup ... \cup X_{i(j-1)}X_{jj}$, sendo formado pelo lado esquerdo das regras que possuem $X_{ii}X_{(i+1)j}$ ou $X_{i(i+1)}X_{(i+2)j}$ ou ... ou $X_{i(j-1)}X_{jj}$ do lado direito.

A Figura 2 ilustra esta situação. Observe que X_{ij} está na coluna i e por ele passa uma diagonal e uma vertical. As comparações são feitas entre os pares que estão na coluna (de baixo para cima, se aproximando de X_{ij}) com os pares que estão na diagonal (da esquerda para a direita, se afastando de X_{ij}).

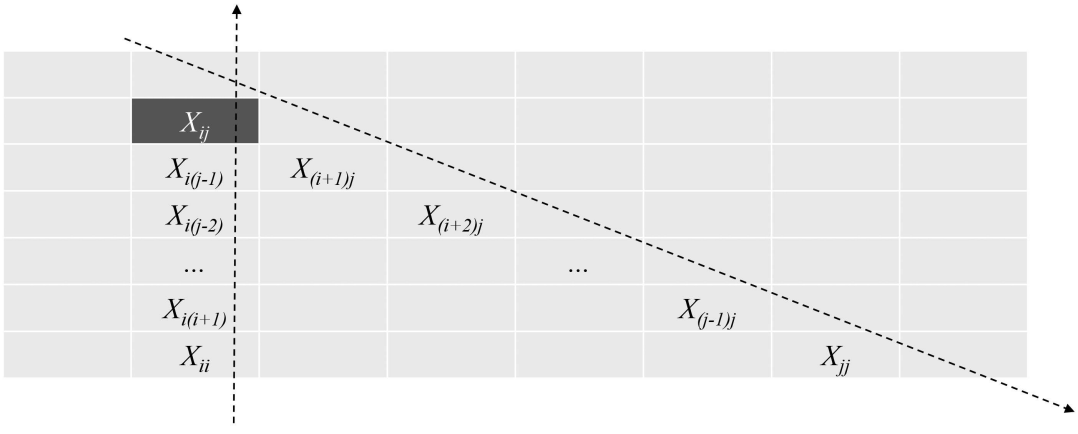

Figura 2: Cálculo de X_{ij}

Por exemplo:

- O cálculo de X_{12} envolve os pares X_{11} e X_{22} (para $k = 1$).

- O cálculo de X_{13} envolve os pares X_{11} e X_{23} (para $k = 1$) e depois os pares X_{12} e X_{33} (para $k = 2$).
- O cálculo de X_{14} envolve os pares X_{11} e X_{24} (para $k = 1$), depois os pares X_{12} e X_{34} (para $k = 2$) e finalmente os pares X_{13} e X_{44} (para $k = 3$).

Como a construção de cada X_{ij} demanda a comparação com até $n-1$ pares, segue que o tempo para calcular cada X_{ij} é $O(n)$, onde n é o comprimento da cadeia de entrada. Como a tabela possui:

$$\sum_{i=1}^{n} i = \frac{n*(n+1)}{2} = O(n^2)$$

elementos, então o tempo total do Algoritmo CYK é $O(n)*O(n^2) = O(n^3)$.

Exemplo 4.58 Considere a linguagem $\{ww^R \mid w \in \{a,b\}^+\}$ gerada pela gramática G abaixo:

$$\begin{aligned} G &= (\{S,a,b\},\{a,b\},P,S) \\ P &= \{S \to aSa \mid bSb \mid aa \mid bb\} \end{aligned}$$

Colocada na Forma Normal de Chomsky, G torna-se G' apresentada abaixo:

$$\begin{aligned} G' &= (\{S,A,B,X,Y,a,b\},\{a,b\},P,S) \\ P &= \{S \to AX \mid BY \mid AA \mid BB, \\ &\quad X \to SA, \\ &\quad Y \to SB, \\ &\quad A \to a, \\ &\quad B \to b\} \end{aligned}$$

Suponha agora que se deseja determinar se a cadeia $aa \in L(G)$.

2	$\{S\}$	
1	$\{A\}$	$\{A\}$
	a	a

Com $X_{11} = X_{22} = \{A\}$ e $X_{12} = \{S\}$ (pois $k = 1$, com X_{11} e X_{22}). Portanto, $aa \in L(G)$. Suponha agora que se deseja determinar se a cadeia $ab \in L(G)$.

2	$\{\}$	
1	$\{A\}$	$\{B\}$
	a	a

Com $X_{11} = \{A\}$, $X_{22} = \{B\}$ e $X_{12} = \{\}$ (pois para $k = 1$, não existe lado direito de regra para X_{11} e X_{22}). Portanto, $ab \notin L(G)$. Suponha agora que se deseja determinar se a cadeia $aabaabaa \in L(G)$.

8	$\{S\}$							
7	–	$\{X\}$						
6	–	$\{S\}$	–					
5	–	–	$\{X\}$	–				
4	–	–	$\{S\}$	–	–			
3	$\{Y\}$	–	–	$\{Y\}$	–	–		
2	$\{S\}$	–	–	$\{S\}$	–	–	$\{S\}$	
1	$\{A\}$	$\{A\}$	$\{B\}$	$\{A\}$	$\{A\}$	$\{B\}$	$\{A\}$	$\{A\}$
	a	a	b	a	a	b	a	a

onde:

- $X_{11} = X_{22} = X_{44} = X_{55} = X_{77} = X_{88} = \{A\}$ pois $A \to a$.

- $X_{33} = X_{66} = \{B\}$ pois $B \to b$.
- $X_{12} = \{S\}$ pois $k = 1$ e $S \to AA$.
- $X_{45} = \{S\}$ pois $k = 4$ e $S \to AA$.
- $X_{78} = \{S\}$ pois $k = 7$ e $S \to AA$.
- $X_{13} = \{Y\}$ pois $k = 2$ e $Y \to SB$.
- $X_{46} = \{Y\}$ pois $k = 5$ e $Y \to SB$.
- $X_{36} = \{S\}$ pois $k = 3$ e $S \to BY$.
- $X_{37} = \{X\}$ pois $k = 6$ e $X \to SA$.
- $X_{27} = \{S\}$ pois $k = 2$ e $S \to AX$.
- $X_{28} = \{X\}$ pois $k = 7$ e $X \to SA$.
- $X_{18} = \{S\}$ pois $k = 1$ e $S \to AX$.

Como $S \in X_{18}$, segue que $aabaabaa \in L(G)$. Note, este exemplo, que cada X_{ij} contém apenas um símbolo não terminal. No caso geral, X_{ij} pode conter vários símbolos não terminais, conforme eles tenham o mesmo lado direito satisfazendo as condições anteriores.

Análise determinística

Os algoritmos apresentados nesta seção determinam o pertencimento de uma cadeia a uma linguagem livre de contexto em tempo exponencial (usando a Forma Normal de Chomsky ou a Forma Normal de Greibach) ou cúbico com o comprimento da cadeia de entrada (CYK). Soluções ainda mais eficientes (de tempo linear), para classes restritas de gramáticas e linguagens livres de contexto, são discutidas nas Seções 4.12 e 4.13.

21. `Página 392:`
    `Remover todo o conteúdo a partir de` "Conforme o Algoritmo 4.15" `até` "presente publicação." `(página 394).`
    `Adicionar no lugar` "Conforme a Seção 4.11"

22. `Página 394:`
    `Substituir` "Por outro lado, conforme o Algoritmo 4.11" `por` "Conforme o Algoritmo 4.11"

23. `Página 395:`
    `Inserir, ao final do parágrafo` "Uma maneira alternativa..."
    "Em outras palavras, basta verificar se a raiz da gramática é um símbolo útil (Teorema 4.5). Este método pode ser usado em qualquer tipo de gramática livre de contexto."

24. `Página 455:`
    `Inserir, depois do parágrafo` "*Linguagens sensíveis ao contexto...*"
    `o parágrafo:`
    "*Uma Máquina de Turing com fita limitada é, por definição, não determinística, mas também é possível definir uma versão determinística dela (modificando a função de transição de maneira correspondente). Entretanto, não se sabe ainda se as Máquinas de Turing com fita limitada não determinísticas reconhecem a mesma classe de linguagens que as Máquinas de Turing com fita limitada determinísticas (citar linz2011, citar wiki-lba).*"

25. `Página 552:`
    `Teorema 7.7:`
    `Inserir, depois de` "L_U, L_P e L_K":
    ", respectivamente Teoremas 6.5, 6.6 e 6.7
    `Inserir, depois de` "que as aceitam":
    " (respectivamente Teoremas 7.8, 7.9 e 7.10)"