CAPÍTULO 6

SIMPLIFICAÇÃO DE GRAMÁTICAS LIVRES DE CONTEXTO E FORMAS NORMAIS

6.1. Introdução

No Cap. 5 concluiu-se que certas gramáticas são melhores do que outras para efeitos de *parsing.* Sabemos também que para uma mesma linguagem é possível definirem-se muitas gramáticas.

Poderemos por isso realçar a conveniência de, dada uma gramática livre de contexto numa forma qualquer, encontrar uma gramática equivalente que tenha a forma mais adequada para o *parsing*. Esta questão é o objecto do presente Capítulo.

Veremos como eliminar as produções unitárias e as produções vazias de uma gramática qualquer.

Há gramáticas de formas especiais, apelidadas de normais no sentido que elas definem uma norma à qual qualquer gramática livre de contexto se pode reduzir. Estudaremos as duas mais conhecidas que têm os nomes dos seus autores: a de Chomsky e a de Greibach. Chomsky, Professor de Linguística no Massachusetts Institute of Technology (MIT), um reputado linguista e cidadão, estudou as linguagens humanas e a partir daí elevou-se a um dos teóricos principais em gramáticas formais (isto é, gramáticas das linguagens computacionais). Sheila Greibach, Professora de Ciências da Computação da Universidade da Califórnia em Los Angeles (UCLA), é também originária da linguística e da matemática aplicada.

O grande objectivo das formas canónicas é a obtenção de gramáticas adequadas para o parsing e para a construção de autómatos de pilha para linguagens livres de contexto não regulares (assunto do Capítulo 7).

6.2. Métodos para transformar gramáticas

Para se chegar a uma forma canónica é necessário previamente "limpar" as gramáticas de algumas limitações estruturais que lhes conferem características desadequadas. O carácter vazio é de evitar; as produções que não permitam um *parsing* eficiente, como as produções unitárias (que não aumentam o tamanho das formas sentenciais), ou as produções lambda (que reduzem o tamanho das formas sentenciais), ou as produções inúteis (que são isso mesmo), devem ser eliminadas.

6.2.1. Eliminação do carácter vazio λ

A eliminação das produções-λ, ou produções vazias (ou esvaziadoras, na medida em que vão esvaziando a forma sentencial, …) é uma necessidade para que se torne possível um método de parsing exequível. Mas será que a eliminação das produções-λ não amputa a gramática (e a sua linguagem) de propriedades essenciais ? De facto uma gramática sem produções-λ não pode produzir a cadeia vazia. Qual a diferença entre uma linguagem que contenha λ e uma que o não contenha? A resposta é dada pela propriedade seguinte:

Seja L uma linguagem livre de contexto que contenha λ.

Seja $G = (V, T, S, P)$ uma gramática livre de contexto para $L-\{\lambda\}$, isto é, L sem λ.

Adicione-se uma nova variável S_0 ao conjunto de variáveis V, fazendo S_0 a variável inicial (em vez de S).

Adicione-se também a P uma nova produção

$$S_0 \rightarrow S \mid \lambda$$

Obtém-se assim uma nova gramática. Esta nova gramática gera a linguagem L. Por isso as conclusões que se possam tirar para a linguagem $L-\{\lambda\}$ também se aplicam a L.

Por outro lado, dada uma qualquer gramática livre de contexto G, há um método de obtenção de uma gramática G' tal que $L(G') = L(G) - \{\lambda\}$.

Do ponto de vista prático não há diferenças entre linguagens livres de contexto que incluam λ e as que não incluam. Dada uma linguagem sem λ, a partir da sua gramática obtém-se uma linguagem que difere desta apenas por conter λ, introduzindo aquela pequena modificação na sua gramática.

Por isso daqui para a frente neste capítulo suporemos apenas linguagens sem λ, a menos que se diga o contrário. Se queremos passar a incluir λ, basta introduzir a nova produção acima indicada.

6.2.2. Uma regra geral de substituição de produções

Teorema 6.2.2.1.

Seja $G = (V,T,S,P)$ uma gramática livre de contexto. Suponha-se que P contém uma produção da forma

$$A \rightarrow x_1 B x_2.$$

sendo A e B variáveis distintas.

Seja

$$B \rightarrow y_1 \mid y_2 \mid ... \mid y_n$$

o conjunto de todas as produções em P que têm B como lado esquerdo. Se A produz uma forma sentencial contendo B e se B produz diversas formas então pode-se ir directamente de A para estas formas, eliminando B, que deixa por isso de ser necessária.

A nova gramática será $G' = \{V, T, S, P'\}$, em que P' se constrói de P

- eliminando a produção $A \rightarrow x_1 B x_2$
- adicionando a produção $A \rightarrow x_1 y_1 x_2 \mid x_1 y_2 x_2 \mid ... \mid x_1 y_n x_2 \mid$

Ter-se-á a igualdade

$$L(G') = L(G)$$

Este teorema está demonstrado em Linz, p. 151. No entanto ele é bastante intuitivo. Se de A se vai para B e de seguida de B se pode ir para vária destinos, quer dizer que de A se pode ir também para vários destinos, indirectamente através de B. Mas se assim é poderemos colocar esses destinos directamente em A, eliminando B que tem um papel de mero intermediário… Tem que se ter cuidado em não perder nenhum dos destinos originariamente possíveis.

Exemplo:

$$S \rightarrow aAb$$
$$A \rightarrow ab|a|b|\lambda$$

Substitui-se por

$$S \rightarrow aabb|aab|abb|ab$$

Que produz as mesmas 4 cadeias.

6.2.3. Eliminação das produções inúteis

Se tivermos uma produção, numa dada gramática, que nunca pode ser utilizada, ela é inútil, e para não criar eventuais problemas importa removê-la, ficando a gramática mais *limpa*.

Seja a gramática

$$S \rightarrow aSb \mid \lambda \mid A,$$

$$A \rightarrow aA$$

Por exemplo derivando

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

obtém-se uma cadeia terminal.

Mas se em vez disso utilizarmos a produção $S \rightarrow A$,

$$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow ....$$

nunca mais é possível passar de uma forma sentencial para uma sentença (cadeia só com símbolos terminais). Cai-se numa "armadilha". Por isso aquela produção nunca se pode utilizar, sendo inútil. Neste caso ela além de inútil é prejudicial dado que produz um ciclo infinito. Vejamos mais formalmente as situações em que uma variável é inútil. Serão todas aquelas em que a variável não é útil.

Definição 6.2.3.1.

Seja $G = (V, T, S, P)$ uma gramática livre de contexto. A variável $A \in V$ diz-se **útil** se e só se existir pelo menos um $w \in L(G)$ tal que

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$$

com x, y em $(V \cup T)^*$.

Uma variável é **útil** se ocorrer pelo menos numa derivação, isto é, se contribuir pelo menos para a derivação de uma sentença. Caso contrário é **inútil**. Uma produção é inútil se envolve alguma variável inútil.

Uma variável pode ser inútil por duas razões:

i) nunca pode ser alcançada a partir da variável de início, como por exemplo o B em

$$S \to A$$
$$A \to aA \mid \lambda$$
$$B \to bA$$

ii) não pode derivar uma cadeia terminal (sentença) porque cria um ciclo infinito, com por exemplo a variável A em

$$S \to aSb \mid \lambda \mid A,$$
$$A \to aA$$

Felizmente é possível eliminar todas as variáveis e todas as produções inúteis de uma gramática, conforme é certificado pelo teorema seguinte, que afirma que para qualquer linguagem livre de contexto existe uma gramática sem variáveis ou produções inúteis.

Teorema 6.2.3.1

Seja $G = (V, T, S, P)$ uma gramática livre de contexto. Então existe uma gramática equivalente $G' = (V', T', S, P')$ que não contém qualquer variável ou produção inútil.

Exemplo 6.2.3.1

Seja a CFG com as produções seguintes (e só essas):

$$S \to ABC \mid b$$
$$A \to a$$
$$B \to b$$

A variável *C* só entra na primeira produção. Se a usarmos, nunca mais nos libertaremos de *C* que por isso é uma variável inútil. Eliminando-a, bem como à produção que a contém, fica

$$S \rightarrow b$$
$$A \rightarrow a$$

Mas a segunda produção nunca pode ser usada, porque de *S* não se chega a *A* ; por isso elimina-se , reduzindo-se a gramática final simplesmente a

$$S \rightarrow b$$

Exemplo 6.2.3.2

Seja a gramática $G = (V, T, S, P)$, em $\Sigma = \{a, b\}$, sendo $V = \{S, A, B, C\}$ e

$$P: \quad S \rightarrow aS \mid A \mid C$$
$$A \rightarrow a$$
$$B \rightarrow aa$$
$$C \rightarrow aCb$$

reduz-se simplesmente a

$$S \rightarrow aS \mid A$$
$$A \rightarrow a$$

De facto, por inspecção visual, verifica-se que a variável *C* é inútil (cria um ciclo infinito) e deve ser eliminada, bem como todas as produções em que entra. A variável *B* nunca é alcançável a partir de *S*, e portanto também é inútil. Ficam apenas as produções que contêm *S* ou *A*.

Estes exemplos são muito simples e resolvem-se por mera inspecção visual. Em gramáticas maiores e mais complicadas é necessário um procedimento algorítmico para a remoção das produções inúteis, que tem as seguintes etapas.

Etapa A - Construção de uma gramática intermédia cujas variáveis são todas úteis. Seja V_1 conjunto das variáveis úteis. Inicialmente está vazio, $\varnothing$, e vai-se enchendo em voltas sucessivas.

1. $V_1 = \varnothing$, inicialização

2. Se $A \rightarrow x_1 x_2 x_3 ... x_m$ com $x_i \in V_1 \cup T$, colocar A em V_1.

   Na 1ª volta só entram em V_1 variáveis que tenham do lado direito apenas símbolos terminais. De facto neste instante V_1 está vazia e portanto $V_1 \cup T$=T.

   Na 2ª volta e seguintes entram as que tenham do lado direito símbolos terminais e/ou variáveis já incluídas em V_1 nas voltas anteriores.

   Até que não se possam meter mais variáveis em V_1.

   Depois de completado este processo, todas as variáveis que possam gerar cadeias terminais estão em V_1. Por outro V_1 só contém variáveis úteis. Ver uma demonstração mais formal em Linz, p. 155.

3. As produções P_1 da gramática intermédia são todas as produções da gramática original cujos símbolos estão todos em $V_1 \cup T$ (i.e., todas as suas variáveis estão em V_1). Se houver uma variável que não esteja em V_1, ela não produz qualquer cadeia final, e será portanto inútil e prejudicial. Portanto todas as produções que têm estas variáveis inúteis são eliminadas na etapa A.

Etapa B. A partir da gramática intermédia obtém-se a gramática final.

Além das produções já eliminadas na etapa A, pode haver outras que sejam inúteis, não por serem prejudiciais, mas por serem inalcançáveis.

Encontrem-se por isso todas as variáveis que não podem ser alcançadas a partir de *S* e eliminam-se, bem como as suas produções. Para isso usa-se o grafo de dependências como ferramenta auxiliar.

O grafo das dependências tem um nó para cada variável ainda existente. Entre duas variáveis existe uma aresta se as duas estão em lados opostos da mesma produção, ou seja, se uma produz a outra. Depois de desenhado o grafo, procuram-se os caminhos desde a variável inicial *S* até cada um das restantes variáveis. Se entre *S* e uma variável A não existe nenhum caminho, então A não é alcançável e deve ser eliminada, bem como todas as produções em que entre.

Exemplo 6.2.3.3

Seja a gramática com as produções

$$S \rightarrow aS \mid A|C$$
$$A \rightarrow a$$
$$B \rightarrow aaD$$
$$C \rightarrow aCD$$
$$D \rightarrow bD|\lambda$$

Construção da gramática intermédia cujas variáveis são todas úteis:

1.- $V_1 = \varnothing$,

2. 1ª volta

- colocar A em V_1 porque $A \rightarrow a$
- colocar D em V_1 porque $D \rightarrow \lambda$ $V_1 = \{A, D\}$

2ª volta

- como $S \rightarrow A$, S vai para V_1
- como $B \rightarrow aaD$, B vai para V_1 $V_1 = \{S, A, B, D\}$

3ªvolta

-não há mais variáveis para introduzir em B

3. Gramática intermédia G_1 : produções de P_1 são as produções originais cujos símbolos estão todos em $V_1 \cup T$ (ou seja, todas as suas variáveis estão em V_1)

$$S \rightarrow aS \mid A$$
$$A \rightarrow a$$
$$B \rightarrow aaD$$
$$D \rightarrow bD|\lambda$$

Na segunda parte obtém-se a gramática G final a partir da gramática intermédia G_1. Para isso basta encontrar todas as variáveis que não são alcançáveis a partir de S. Depois eliminam-se bem como todas as suas produções. Poderemos recorrer ao grafo de dependências.

Figura 6.2.3.1 Grafo de dependência do exemplo 6.2.3.3

Do grafo conclui-se que só ficam as variáveis *S* e *A*, dado que *B* e *D* não são alcançáveis a partir de *S*.

Assim a gramática final terá apenas as produções

$$S \to aS \mid A$$
$$A \to a$$

E aplicando a regra de substituição ficará

$$S \to aS \mid a$$

que gera a linguagem $L(G)=\{a^n, n\geq 1\}$.

6.2.4. Remoção das produções λ

Já sabemos o que são produções-λ: qualquer produção de uma gramática livre de contexto da forma

$$A \to \lambda$$

Definição 6.2.4.1. **Variável anulável**

Qualquer variável *A* para a qual é possível a derivação

$$A \overset{*}{\Rightarrow} \lambda$$

chama-se **anulável (*nullable*).** De facto ela pode-se anular (eliminar) nas formas sentenciais em que aparecer.

Se uma gramática gera uma linguagem que não contém λ, isso não obriga a que a gramática seja desprovida de produções-λ. Por exemplo

$$S \to aA$$

$$A \rightarrow aA|\lambda$$

gera a linguagem das cadeias de a's, a^n, $n \geq 1$. Não contém λ, embora a sua gramática contenha $A \rightarrow \lambda$. Quando tal acontece é possível eliminar as produções-λ da gramática.

Neste caso, fazendo

$$S \rightarrow aA|a$$
$$A \rightarrow aA|a$$

obtém-se a mesma linguagem (confirme o leitor que assim é).

Também para eliminar as produções-λ é necessário um procedimento algorítmico para os casos mais complicados. Ele é fundamentado no teorema seguinte:

Teorema 6.2.4.1.

Seja G uma gramática livre de contexto tal que λ não pertence a $L(G)$. Então existe uma gramática equivalente G' sem produções-λ que gera a mesma linguagem.

Demonstração:

A demonstração faz-se desenvolvendo o algoritmo que permite encontrar G':

1ª etapa- Encontrar o conjunto V_N das variáveis anuláveis, que está inicialmente vazio.

1. Para todo o A tal que existe $A \rightarrow \lambda$, incluir A em V_N.

   Estas são as variáveis anuláveis directamente, em um passo.

2. Para todo o B tal que existe a produção

   $$B \rightarrow A_1A_2 \ldots A_n$$

   com $A_1, A_2, \ldots, A_n$ pertencentes todas a V_N., incluir B em V_N.

   Estas são as variáveis anuláveis indirectamente, através de outras, em mais de um passo.

3. Repetir o passo 2 até que não seja possível adicionar mais variáveis a V_N

Depois da primeira etapa sabemos quais são as variáveis anuláveis. Se duas delas aparecerem simultaneamente na mesma produção (lado direito), podem-se anular simultaneamente ou uma de cada vez, de modo que temos três situações distintas que têm que se considerar para que a gramática não se altere (tudo o que se podia produzir antes tem que se continuar a poder produzir depois). Daí a segunda etapa de reconstituição das possíveis produções.

2ª etapa - Substituição de produções

1. Procurar as produções em P da forma

   $A \rightarrow x_1x_2...x_m,\ m \geq 1, \quad x_i \in V \cup T$

   tendo do lado direito caracteres terminais e/ou várias variáveis, podendo algumas ser anuláveis.

2. Colocar em P'

   - esta produção $A \rightarrow x_1x_2...x_m,\ m \geq 1, \quad x_i \in V \cup T$

   - as que resultam desta pela substituição das suas variáveis anuláveis por λ em todas as combinações possíveis : se x_i e x_j são anuláveis, cria-se
     (i) uma produção em que x_i é substituída por λ,
     (ii)outra em que x_j é substituída por λ, e
     (iii) ainda outra em que x_j e x_j são ambas substituídas por λ.

   - se todas as x_i são anuláveis, a produção $A \rightarrow \lambda$ não se introduz em P'.

Ver também Hopcroft, 259.

Exemplo 6.2.4.1 (Hopcroft, 261)

Sejam as produções

$S \rightarrow AB$
$A \rightarrow aAA \mid \lambda$
$B \rightarrow bBB \mid \lambda$

1ª etapa:

1ª volta A e B são directamente anuláveis.

2ªvolta Depois S também o é, indirectamente através de A e B.

Portanto todas as variáveis são anuláveis.

2ª etapa. Consideramos uma produção de cada vez

$S \to AB$ desmultiplica-se em

$S \to AB$, a produção original

$S \to A$, fazendo $B=\lambda$

$S \to B$, fazendo $A=\lambda$

$S \to \lambda$, fazendo $A=\lambda$ e $B=\lambda$, mas este caso não se retém.

$A \to aAA$ multiplica-se em

$A \to aAA$

$A \to aA$

$A \to a$

$B \to bBB$ multiplica-se em

$B \to bBB$

$B \to bB$

$B \to b$

De modo que teremos, concluída a segunda etapa, a gramática

$S \to AB \mid A \mid B$

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

sem variáveis anuláveis.

Estas produções não permitem derivar a cadeia vazia. E se tivermos o caso de uma CFL com λ, como fazer ?

Se $\lambda \in L(G)$, consideramos em primeiro lugar a mesma linguagem mas sem λ a que chamamos $L-\{\lambda\}$. Poderemos afirmar agora que existe uma gramática CFG G_1 sem produções-λ tal que $L(G_1) = L(G) - \{\lambda\}$. Depois, e como vimos anteriormente, introduz-se uma nova produção

$$S_0 \rightarrow S|\lambda$$

Naturalmente que para incluir esta possibilidade se introduziu uma variável anulável S_0, o novo axioma.

Exemplo 6.2.4.2

Seja a gramática com as produções

$$S \rightarrow aAbBc$$
$$A \rightarrow BC$$
$$B \rightarrow bB|b|\lambda$$
$$C \rightarrow D|\lambda$$
$$D \rightarrow c$$

Obtenção do conjunto das variáveis anuláveis:

1º $B \rightarrow \lambda$

$C \rightarrow \lambda$ donde $V_N=\{B, C\}$

2º $A \rightarrow BC$ donde $V_N=\{A, B, C\}$

Agora $V_N=\{A, B, C\}$

3º Não há mais

$S \rightarrow aAbBc$ resulta em $S \rightarrow aAbBc$

$S \rightarrow abBc$ (anula A)

$S \rightarrow aAbc$ (anula B)

$S \rightarrow abc$ (anula A e B)

$A \rightarrow BC$ resulta em $A \rightarrow BC$

$A \rightarrow C$ (anula B)

$A \rightarrow B$ (anula A)

$B \to bB$ resulta em	$B \to bB$	
	$B \to b$	(anula B)
$C \to D$ resulta em	$C \to D$	
$D \to c$ resulta em	$D \to c$	

Chegamos assim a 11 produções para a gramática final.

6.2.5. Remoção das produções unidade

As produções unidade não aumentam o comprimento das formas sentenciais, o que dificulta o *parsing.* Por isso devem ser eliminadas, sem que a gramática se altere.

Definição 6.2.5.1. Produção unidade

Qualquer produção de uma gramática livre de contexto da forma

$$A \to B$$

em que $A, B \in V$ são variáveis, chama-se **produção unidade.** O seu nome vem do facto de não alterar o comprimento da forma sentencial, ou seja, de multiplicar este comprimento por 1, a unidade.

Qualquer para (A, B) para o qual se possa encontrar uma sequência de derivações

$$A \overset{*}{\Rightarrow} B$$

usando apenas produções unidade é um **par unidade.**

Pares unidade em um passo encontram-se por inspecção visual. Mas numa gramática com muitas produções e um grande número de variáveis, já não é tão evidente.

Neste caso para descobrir todos os pares unidade, entre as variáveis de uma gramática, poderemos usar o seguinte algoritmo de indução (Hopcroft, p. 263):

Base :

(A, A) é um par unidade para todo o A em V: $A \overset{*}{\Rightarrow} A$ em zero passos.

Indução:

Se (A, B) é um par unidade e
$B \to C$ é uma produção, sendo C uma variável,
então (A, C) é um par unidade.

Os grafos de dependência também podem ajudar a identificar os pares unidade.

Identificados os pares unidade, vamos ver como os eliminaremos sem alterar a gramática. Tal é sempre possível, conforme afirma o teorema seguinte.

Teorema 6.2.5.1.

Seja $G = (V, T, S, P)$ qualquer gramática livre de contexto sem produções-λ. Então existe uma outra gramática livre de contexto $G' = (V', T', S', P')$ equivalente a G que não contém qualquer produção unidade.

Demonstração:

A demonstração faz-se também aqui desenvolvendo o procedimento de eliminação dos pares unidade (demonstração construtiva).

1º Qualquer produção do tipo $A \to A$ pode ser removida sem qualquer efeito (é evidente)

2º Colocar em P' todas as produções não-unidade de P

3º Procurar todos os pares unidade

4º Para todos os pares unidade (A, B) em P tais que

$$B \to y_1 \mid y_2 \mid ... \mid y_n \text{ em } P'$$

adicionar a P'

$$A \to y_1 \mid y_2 \mid ... \mid y_n$$

o que corresponde à regra geral de substituição de produções que vimos no parágrafo 6.2.2.

Exemplo 6.2.5.1 (Linz, 160)

Remover todas as produções unitárias da gramática cujas produções são

$$S \to Aa \mid B$$
$$B \to A \mid bb$$
$$A \to a \mid bc \mid B$$

1º e 2º. Procurar os pares unidade

$S \to B$, logo (S, B) é par unidade
$B \to A$, logo (B, A) é par unidade
$A \to B$, logo (A, B) é par unidade

Se (S, B) e (B, A) são pares unidade (S, A)-também o é.

O grafo de dependência das variáveis confirma esta dedução. Os seus vértices são as variáveis que entram em pares unidade. Existe uma aresta entre dois vértices se as respectivas variáveis fazem um par unidade. A aresta é orientada segundo a produção do par unidade. No caso dá Fig. 6.2.5.1.

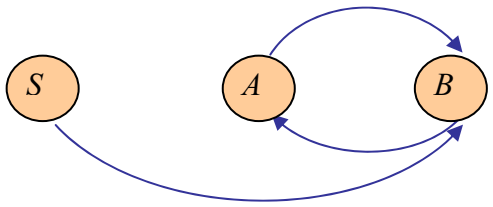


Figura 6.2.5.1 Grafo de dependências das variáveis de pares unidade do exemplo 6.2.5.1.

Vai-se de S a A através de B.

3º Adicionar a P' todas as produções não unitárias de P

$$S \to Aa$$
$$B \to bb$$

$$A \to a \mid bc$$

4º Introduzir as derivações que substituem os pares unidade de P com produções de P', como na tabela

Par unidade em P	Produção em P'	A criar em P'
$S \overset{*}{\Rightarrow} B$	$B \to bb$	$S \to bb$
$A \overset{*}{\Rightarrow} B$	$B \to bb$	$A \to bb$
$B \overset{*}{\Rightarrow} A$	$A \to a \mid bc$	$B \to a \mid bc$
$S \overset{*}{\Rightarrow} A$	$A \to a \mid bc$	$S \to a \mid bc$

Tabela 6.2.5.1.

Tem-se finalmente a nova gramática simplificada

$$S \to Aa \mid bb \mid a \mid bc \mid$$
$$B \to bb \mid a \mid bc$$
$$A \to a \mid bc \mid bb$$

Exemplo 6.2.5.2. (Hopcroft, 263)

Seja a gramática do exemplo 5.3.7 de criação de expressões numa linguagem de programação em que os identificadores são definidos do modo seguinte: uma letra seguida por letras ou números; as letras são a e b e os números 0 e 1, ou seja, o alfabeto é $\Sigma=\{0,1,a,b\}$. As operações aritméticas usadas são a multiplicação * e a soma +. As suas produções são

$$E \to T \mid E+T$$
$$T \to F \mid T*F$$
$$F \to I \mid (E)$$
$$I \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$$

Várias são produções unidade. Os pares unidade são:

-directos (E,T), (T,F), (F,I)

- indirectos (E,F), (E,I), (F,I)

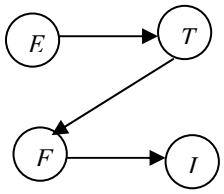


Fig. 6.2.5.1

Note-se que para além destes pares unidade teremos ainda os da base do algoritmo de Hopcroft. No grafo de dependências representar-se-iam como arestas reflexivas em cada vértice (do vértice para ele mesmo).

Atendendo ao grafo de dependências, é mais conveniente eliminar ao pares unidade começando pelo fim do grafo, neste caso pelo par (F,I), depois sucessivamente (T,F), (E,T):

(F,I) : $F \to I$ resulta em $F \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$

(T,F): $T \to F$ resulta em $T \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \mid (E)$

(E,T): $E \to T$ resulta em $E \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \mid (E) \mid T*F$

Juntando agora as produções não-unidade restantes, teremos as produções

$E \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \mid (E) \mid T*F \mid E+T$

$T \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \mid (E) \mid T*F$

$F \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \mid (E)$

$I \to a \mid b \mid c \mid 0 \mid 1 \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$

equivalentes às originais, pois geram as mesmas expressões.

Juntando as três operações de simplificação que estudámos até aqui, poderemos enunciar o teorema seguinte.

Teorema 6.2.5.2

Seja L uma linguagem livre de contexto que não contém qualquer λ. Então existe uma gramática livre de contexto que gera L e que não contém

- produções inúteis
- produções-λ

- produções-unidade

Podem-se remover todas as produções indesejáveis através da seguinte sequência de passos:

1º - Remover as produções-λ

2º - Remover as produções-unidade

3º - Remover as produções inúteis.

Depois destas simplificações poderemos prosseguir no sentido de encontrar gramáticas que facilitem o *parsing*. As formas normais, que têm esse objectivo, requerem estas simplificações prévias.

6.3. Formas normais (ou canónicas) de Chomsky e de Greibach

Na família das gramáticas livres de contexto existem muitas formas normais (i.e., que obedecem a certas normas, são normalizadas e permitem normalizar em relação a elas qualquer gramática CFG). Algumas revestem-se de importância particular e por isso as estudaremos de imediato.

6.3.1. Forma normal de Chomsky

Na forma normal de Chomsky a parte direita de qualquer produção não pode ter mais de dois símbolos.

Definição 6.3.1.1.

Uma gramática livre de contexto está na forma normalizada de Chomsky se todas as produções são da forma

$$A \to BC$$

ou

$$A \to a$$

em que A, B, C pertencem a V e $a \in T$.

Esta gramática ou aumenta em uma unidade em cada produção o tamanho das formas sentenciais, ou substitui uma variável de cada vez por um símbolo terminal. Para derivar uma cadeia com n símbolos finais são necessárias no máximo $2n$-1 produções. Olhando para as produções vê-se que esta gramática trabalha em duas fases: primeiro cria a forma sentencial apenas com n variáveis. Para duas variáveis basta uma produção, três variáveis duas produções, etc., n variáveis n-1 produções. De seguida substitui cada variável por um símbolo terminal, precisando de n produções para substituir as n variáveis. Logo n-1+n dá $2n$-1.

Exemplo 6.3.1.1.

Vejamos as produções na FC Chomsky

$$S \to SB|a$$
$$B \to b$$

Fazendo a derivação

$$S \Rightarrow SB \Rightarrow SBB \Rightarrow SBBB \Rightarrow SBBBB \Rightarrow SBBBb \Rightarrow SBBbb \Rightarrow SBbbb \Rightarrow Sbbbb \Rightarrow abbbb$$

obtém-se uma cadeia de 5 caracteres em 9 derivações.

Teorema 6.3.1

Qualquer gramática livre de contexto G=(V, T, S, P} com $\lambda \notin L(G)$ tem uma gramática equivalente G' = (V', T', S, P') na forma normal de Chomsky.

Pressupõe-se que G não tem produções λ nem produções inúteis nem produções unidade nem símbolos inúteis.

Demonstração.

Este teorema demonstra-se construindo o algoritmo para transformação na forma de Chomsky.

1º passo

Para todo o símbolo terminal a que aparece numa produção de corpo (lado direito) com comprimento igual ou superior a 2, cria-se uma nova variável B_a. Esta variável terá uma só produção $B_a \to a$. Usa-se B_a em vez de a nas produções de corpo maior ou igual a 2.

Cada produção fica com um corpo que é ou um terminal simples, ou pelo menos duas variáveis e nenhum terminal.

Vejamos um exemplo para acompanhar o desenvolvimento,

Exemplo 6.3.1.2. Seja

$$S \rightarrow ABa$$
$$A \rightarrow aab$$
$$B \rightarrow Ac$$

Introduzem-se B_a, B_b, B_c para a, b, c , ficando

$$S \rightarrow ABB_a$$
$$A \rightarrow B_aB_aB_b$$
$$B \rightarrow AB_c$$
$$B_a \rightarrow a$$
$$B_b \rightarrow b$$
$$B_c \rightarrow c$$

2º passo

Partem-se as produções $A \rightarrow C_1C_2....C_n, n \geq 3$, num conjunto de produções com duas variáveis no lado direito. Para isso introduzem-se novas variáveis $D_1, D_2,..., D_{n-2}$. A produção original é substituída pelas n-1 produções

$$A \rightarrow C_1D_1$$
$$D_1 \rightarrow C_2D_2$$
$$\ldots$$
$$D_{n-2} \rightarrow C_{n-1}D_n$$

Ainda no mesmo exemplo, introduzem-se as variáveis D_1 (para a primeira produção) e D_2 (para a segunda produção). A forma normal de Chomsky será finalmente,

$$S \rightarrow AD_1$$
$$D_1 \rightarrow BB_a$$
$$A \rightarrow B_aD_2$$

$D_2 \rightarrow B_aB_b$

$B \rightarrow AB_c$

$B_a \rightarrow a$

$B_b \rightarrow b$

$B_c \rightarrow c$

Passou-se de 3 para 8 produções. Normalmente é este o custo da passagem à FN de Chomsky (FNC): um grande aumento do número das produções.

Exemplo 6.3.1.3.

Considere-se a gramática não-ambígua do Exemplo 6.2.5.2.

$E \rightarrow T \mid E+T$

$T \rightarrow F \mid T*F$

$F \rightarrow I \mid (E)$

$I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$

Reduzir esta gramática à FN de Chomsky.

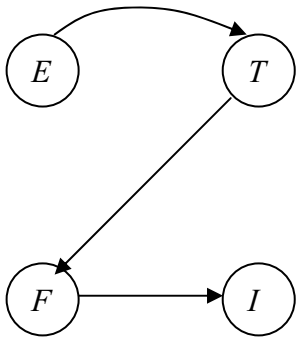
Olhando com atenção para as produções, nota-se que existem várias produções unidade. A sua remoção é a primeira etapa para a resolução do problema.

Remoção das produções unidade (Hopcroft, 263)

Temos que identificar todos os pares unidade. Para isso aplica-se o algoritmo que vimos:

Base: pares unidade (*E*, *E*), (*T*, *T*), (*F*, *F)*, (*I*, *I*).

Passo indutivo:



1.(*E*, *E*) e a produção $E \rightarrow T$ dá o par unidade (*E*,*T*)

2.(*E*, *T*) e a produção $T \rightarrow F$ dá o par unidade (*E*,*F*)

3.(*E*, *F*) e a produção $F \rightarrow I$ dá o par unidade (*E*,*I*)

4. (*T*, *T*) e a produção $T \rightarrow F$ dá o par unidade (*T*,*F*)

5. (T, F) e a produção $F \rightarrow I$ dá o par unidade (T,I)

6. (F, F) e a produção $F \rightarrow I$ dá o par unidade (F,I)

Encontramos assim 10 pares unidade, incluindo os da base. O grafo de dependências também evidencia esses pares unidade.

Agora aplicamos a regra de substituição para os pares unidade na tabela seguinte.

Par	Produções
(E, E)	$E \rightarrow E+T$
(E, T)	$E \rightarrow T*F$
(E, F)	$E \rightarrow (E)$
(E, I)	$E \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$
(T, T)	$T \rightarrow T*F$
(T, F)	$T \rightarrow (E)$
(T, I)	$T \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$
(F, F)	$F \rightarrow (E)$
(F, I)	$F \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$
(I, I)	$I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$

Teremos assim a gramática sem produções unidade, composta pelas 10 produções da segunda coluna da tabela.

Note-se que a gramática não tem nem produções-λ nem produções vazias. Está por isso pronta para ser passada à FN de Chomsky.

Os símbolos terminais desta gramática são nove, $\{a, b, c, 0, 1, +, *, (,)\}$. Qualquer um deles aparece no corpo de uma produção com mais de um símbolo. Por isso é conveniente

introduzir nove variáveis correspondentes aos símbolos terminais, e nove produções adicionais nas quais essas variáveis são substituídas pelo seu carácter terminal. Assim

$A \to a$	$Z \to 0$	$M \to *$
$B \to b$	$U \to 1$	$L \to ($
$C \to c$	$P \to +$	$R \to)$

Introduzindo as variáveis substituindo os caracteres finais nos corpos das produções com mais de um carácter, teremos as produções seguintes

Produções actuais	Novas produções
$E \to E+T$	$E \to EPT$
$E \to T*F$	$E \to TMF$
$E \to (E)$	$E \to LER$
$E \to a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$	$E \to a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$
$T \to T*F$	$T \to TMF$
$T \to (E)$	$T \to LER$
$T \to a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$	$T \to a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$
$F \to (E)$	$F \to LER$
$F \to a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$	$F \to a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$
$I \to a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1$	$I \to a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$

$A \to a$

$B \to b$

$C \to c$

$Z \to 0$

$U \to 1$

$$P \rightarrow +$$

$$M \rightarrow *$$

$$L \rightarrow (\quad R \rightarrow)$$

Agora todos os corpos das produções ou têm mais de duas variáveis ou têm um símbolo terminal.

Para concluirmos a passagem á forma normal de Chomsky resta resolver o problema da existência de mais de duas variáveis em alguns corpos. Um conjunto de três variáveis , com por exemplo *EPT* transforma-se num conjunto duas e mais um produção adicional:

$$E \rightarrow EPT \qquad E \rightarrow EX_1$$

$$X_1 \rightarrow PT$$

obtendo-se assim duas produções na FN de Chomsky.

As novas produções são as da tabela seguinte.

Produções actuais	Novas produções: FN Chomsky
$E \rightarrow EPT$	$E \rightarrow EX_1$ $X_1 \rightarrow PT$
$E \rightarrow TMF$	$E \rightarrow TX_2$ $X_2 \rightarrow MF$
$E \rightarrow LER$	$E \rightarrow LX_3$ $X_3 \rightarrow ER$
$E \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$	$E \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$
$T \rightarrow TMF$	$T \rightarrow TX_2$
$T \rightarrow LER$	$T \rightarrow LX_3$

$T \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$	$T \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$
$F \rightarrow LER$	$F \rightarrow LX_3$
$F \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$	$F \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$
$I \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$	$I \rightarrow a \mid b \mid c \mid IA \mid IB \mid IC \mid IZ \mid IU$
$A \rightarrow a$	$A \rightarrow a$
$B \rightarrow b$	$B \rightarrow b$
$C \rightarrow c$	$C \rightarrow c$
$Z \rightarrow 0$	$Z \rightarrow 0$
$U \rightarrow 1$	$U \rightarrow 1$
$P \rightarrow +$	$P \rightarrow +$
$M \rightarrow *$	$M \rightarrow *$
$L \rightarrow ($	$L \rightarrow ($
$R \rightarrow)$	$R \rightarrow)$

Temos um total de 50 produções na FN de Chomsky, para 16 produções à partida.

6.3.2. Forma normal de Greibach

Se colocarmos restrições não no número de símbolos da parte direita das produções, mas nas posições relativas admissíveis das variáveis e dos símbolos terminais, teremos a forma normalizada de Greibach.

Definição 6.3.2.1

Uma gramática livre de contexto está na **forma normalizada de Greibach** se todas as produções têm a forma

$$A \rightarrow ax,$$

em que $a \in T$ e $x \in V^*$ (por exemplo *aB, aBC, aBCD, ...*).

As gramáticas de (Sheila) Greibach têm propriedades interessantes:

- uma frase de tamanho *n* é derivada em *n* passos, dado que em cada produção se introduz exactamente um carácter terminal numa forma sentencial.

Exemplo 6.3.2.1

Seja o exemplo 6.3.1.1

$$S \to SB|a$$
$$B \to b$$

- se aplicarmos um PDA para uma forma normal de Greibach, obtém-se um PDA sem transições-λ, o que prova que é sempre possível eliminar tais transições num PDA (estudaremos os PDA no Cap. 7, mas fica aqui a nota).

Teorema 6.3.2.1. Existência da Forma Normal de Greibach

Para toda a gramática livre de contexto G tal que $\lambda \notin L(G)$, existe uma gramática equivalente G' na forma normal de Greibach.

A demonstração deste teorema não é tão simples como o da FN de Chomsky. De facto não existe um algoritmo simples para transformar qualquer gramática numa FN Greibach.

Podem usar-se as regras de substituição de produções vistas anteriormente, como nos exemplos de Linz 6.9, 6.10.

Exemplo 6.3.2.1

$S \to 0S1 \mid 1S0 \mid 1 \mid 0|$ na FN Greibach.

Introduzindo duas novas variáveis *A* para *0* e *B* para *0,* obtém-se directamente a FCG.

$S \to 0SA \mid 1SA \mid 0 \mid 1|$
$A \to 0$
$B \to 1$

Este exemplo é simples porque todas a produções iniciais de iniciam com um carácter final (0 ou 1). No entanto, quando assim não acontece, o processo torna-se bem mais complicado. Existe um algoritmo

Hopcroft sugere (p. 271) expandir a primeira variável de cada produção até se obter um terminal. Vejamos um exemplo.

Exemplo 6.3.2.2 (Ex. 6.2.13 Linz)

Converter na forma normal de Greibach

$$S \to ABb \mid a$$
$$A \to aaA \mid B$$
$$B \to bAb$$

Para obtermos um terminal na primeira produção, substitui-se o A

$$S \to ABb|a \text{ e } A \to aaA \mid B \text{ vai dar } S \to aaBb|BBb|a$$

Mas agora é necessário substituir B na segunda produção

$$S \to BBb \text{ e } B \to bAb \text{ vai dar } S \to bAbBb$$

Agora o primeiro carácter em qualquer produção é um símbolo terminal a ou b.

Introduzindo de seguida variáveis adicionais para os caracteres terminais misturados com variáveis,

$$Z_a \to a \qquad\qquad Z_b \to \text{b}$$

substituindo nas produções anteriores resulta na solução:

$$S \to aZ_aABZ_b \mid bAZ_aBZ_b \mid a$$
$$A \to aZ_aA \mid bAZ_b$$
$$B \to bAZ_b$$
$$Z_a \to a$$
$$Z_b \to b$$

Se se reduzir previamente a gramática, obtém-se simplesmente $S \to a$, que é uma FN de Greibach.

De facto A e B, na gramática original, são variáveis que não permitem obter um cadeia final, sendo por isso variáveis inúteis. Fica assim apenas $S \to a$.

No entanto este procedimento, de procurar chegar a um carácter terminal no início de cada produção, não funciona quando existem ciclos recursivos à esquerda que nunca atingem um terminal, como por exemplo em $A \rightarrow AB$. É necessário neste caso curto-circuitar o processo, criando uma produção que (i)introduz um terminal como primeiro símbolo do corpo e (ii) tem variáveis a seguir a ele para gerar todas as sequências de variáveis que poderiam ter sido derivadas no caminho de geração daquele terminal. É possível, para qualquer CFG, eliminar os ciclos recursivos.

Existe um algoritmo para passar à FN Greibach a partir da FN de Chomski . Este algoritmo pode ver-se por exemplo em http://www.cs.uky.edu/~lewis/texts/theory/languages/cf-lang.pdf. Nesta passagem reside uma das principais utilidades da FN Chomsky: é fácil chegar a ela, a partir de qualquer gramática e depois usa-se o algoritmo para passas à de Greibach. Esta usa-se por exemplo para o projecto de autómatos de pilha, matéria a estudar no próximo capítulo.

Bibliografia.

An Introduction to Formal Languages and Automata, Peter Linz, 3rd Ed., Jones and Bartelett Computer Science, 2001.

Introduction to Automata Theory, Languages and Computation, 2nd Ed., John Hopcroft, Rajeev Motwani, Jeffrey Ullman, Addison Wesley, 2001.

Elements for the Theory of Computation, Harry Lewis and Christos Papadimitriou, 2nd Ed., Prentice Hall, 1998.

Introduction to the Theory of Computation, Michael Sipser, PWS Publishing Co, 1997.