CAPÍTULO 3

EXPRESSÕES REGULARES, LINGUAGENS REGULARES E GRAMÁTICAS REGULARES

3.1 Introdução 117

3.2 Expressões Regulares 117

3.3 Regras algébricas para expressões regulares 126

3.4 Relação entre expressões regulares e linguagens regulares 130

3.5 Gramáticas regulares 140

3.6 Síntese das equivalências 151

3.1. Introdução

Vimos que uma linguagem é regular se for aceite por um DFA ou um NFA. Poderemos assim representar uma linguagem pelo seu autómato. Se o autómato tiver um elevado número de estados, não é possível, por simples inspecção visual, ver qual a sua linguagem. Interessa por isso uma forma mais simples de representar uma linguagem regular, que seja de utilização expedita, de modo análogo ao de uma fórmula química de uma substância: olhando para ela extrai-se logo uma grande quantidade de informação sobre a sua natureza.

É esse o objectivo das expressões regulares: são formas simples, expeditas, com muita informação, de representar linguagens regulares.

Mas se assim é, e se a uma expressão regular corresponde sempre uma linguagem regular, para uma expressão regular há-de haver um autómato finito que represente a mesma linguagem, dado que toda a linguagem regular tem o seu DFA ou NFA.

Há portanto uma estreita relação entre expressões regulares e linguagens regulares. Estudá-la-emos neste capítulo.

Já vimos que uma linguagem pode ser definida por uma gramática. Dada uma linguagem regular, pode interessar conhecer uma gramática que a gere, e neste caso diz-se gramática regular. E é sempre possível encontrá-la, com recurso aos autómatos finitos. Autómatos finitos, gramáticas regulares, expressões regulares, são três formas de representar o mesmo conjunto: uma certa linguagem regular. Estudaremos neste capítulo a forma como essas três representações se relacionam.

3.2. Expressões regulares

Uma expressão regular é uma forma de descrever linguagens regulares. São expressões com símbolos de um dado alfabeto e caracteres operadores sobre as cadeias, isto é, são expressões algébricas. Dado um alfabeto Σ, uma expressão regular é desenvolvida a partir de

- os símbolos do alfabeto
- os operadores de

- união : $\cup$ ou +
- concatenação: $\cdot$
- fecho-estrela : *

- de parênteses

Exemplo 3.2.1.

Consideremos o alfabeto Σ=$\{a, b, c, ..., z\}$.

Nele a linguagem finita $L = \{a\}$ é definida pela expressão regular : a

A linguagem $L = \{a, b, c\}$, é definida pela expressão regular: $a \cup b \cup c$ ou $a+b+c$. O operador união em conjuntos corresponde ao operador adição nas expressões regulares.

Muitas linguagens são infinitas e não se podem por isso escrever por enumeração explícita. A linguagem definida pela expressão regular $(a+b\cdot c)^*$ é uma linguagem infinita devido ao fecho-estrela. Por definição de *, teremos

$$L= \{(a+b\cdot c)^0 \cup (a+b\cdot c)^1 \cup (a+b\cdot c)^2 \cup (a+b\cdot c)^3 \ \ldots \ \}$$

$$= \{\lambda, a+b.c, (a+bc)(a+b.c), (a+bc)(a+b.c)(a+b.c), \ldots\}$$

$$= \{\lambda, a, bc, aa, abc, bca, bcbc, aaa, aabc, abcbc, abca, bcbcbc, \ldots \}$$

No alfabeto $\Sigma = \{a, b, c\}$ a linguagem $(a + b)^*$ é composta pelas cadeias cadeias λ, a, b, aa, bb, ab, ba, aaa, $abbaa$, $bbbaabab$, ...Qualquer cadeia que contenha só a's e b's pode ser obtida desta expressão regular. Como obter $ababaa$? Fica o desafio ao leitor.

No mesmo alfabeto a linguagem a^*b^* é composta pelas cadeias $a^n b^m$, $n,m \geq 0$ ou seja λ, a,b, aab, $aabb$, abb, bbb, $aabbbb$,...

Finalmente, ainda no mesmo alfabeto, a linguagem $(c+\varnothing)$ contém apenas a cadeia c.

As expressões regulares são formas sintéticas de exprimirmos as operações sobre cadeias que estudámos no Cap. 1. São como que expressões algébricas num dado alfabeto. No entanto poderemos defini-las formalmente.

3.2.1. Definição formal, recursiva, de expressão regular

Uma definição por recursividade usa como sabemos um conjunto de primitivas e uma recursão. Assim é também para as expressões regulares:

Definição 3.2.1. Seja um Σ um dado alfabeto. Então:

1. As **expressões regulares primitivas** são compostas por zero ou um carácter:

   $\varnothing$, λ e $a \in \Sigma$ (para $\forall a \in \Sigma$) são todas expressões regulares primitivas

2. **Recursão**: indica como se obtêm expressões regulares a partir umas das outras:

   Se r_1 e r_2 são expressões regulares, também o serão

   2.1 r_1+r_2, (soma de duas expressões regulares)

   2.2 $r_1 \cdot r_2$, (concatenação de duas expressões regulares)

   2.3 r_1^* , r_2^* (fecho-estrela de uma expressão regular)

   2.4 (r_1), (r_2) (parêntese de uma expressão)

3. Uma cadeia é uma expressão regular se e só se ela puder ser derivada a partir das expressões regulares primitivas e pela aplicação de um número finito de vezes das regras de 2 (princípio da recursividade)

Exemplo 3.2.2

Seja o alfabeto $\Sigma=\{0,1\}$

A expressão regular seguinte define a linguagem indicada

$01^* = \{0, 01, 011, 0111,\ldots\}$, 0 concatenado com um número arbitrário de 1's.

Se concatenarmos toda e qualquer cadeia desta linguagem com 01 obtém-se

$(01^*)(01) = \{001, 0101, 01101, 011101, \ldots..\}$

Por outro lado

(0+1)* = {0, 1, 00, 01, 10, 11, …..}, i.e., toda e qualquer as cadeias com "0" e "1".

Analisemos agora a expressão regular

(0+1)*00(0+1)*

À esquerda temos (0+1)*, qualquer cadeia de 0's e 1's. À direita temos a mesma coisa. Mas no centro temos o par 00. Quer dizer que esta expressão regular representa qualquer cadeia que contenha o par 00 precedido por qualquer número de 0's e 1's e seguido de qualquer número de 0's e 1's, ou seja,

{**00**, 1**00**1, **00**110, 101**00**, 01010**00**0111000…..},

i.e., todas as cadeias de 0 e 1 contendo "**00**" em **qualquer** posição. O par 00 tem que aparecer uma vez, mas pode aparecer além disso qualquer número de vezes.

Exemplo 3.2.3

No alfabeto $\Sigma = \{a,b,c\}$, será $(a+b\cdot c)^*\cdot(c+\varnothing)$ uma expressão regular ?

Para responder temos que ver se ela se pode obter a partir das expressões regulares primitivas por uma recursão:

Os símbolos $a,b,c,\varnothing$ são expressões regulares primitivas.

$b.c$ é uma expressão regular (*er*) , do item 2.2. da definição,

$a+b.$c é uma *er*, do item 2.1 da definição,

$(a+b.c)$ é uma *er*, do item 2.4 da definição,

$(a+b.c)^*$ é uma *er*, do item 2.3 da definição,

$(c+\varnothing)$ é uma *er*, dos itens 2.1 e 2.4 da definição,

$(a+b\cdot c)^*\cdot(c+\varnothing)$ é uma *er* do item 2.2 da definição. *q.e.d*

E $(a + c +)$ é expressão regular ?

a , c são *er* primitivas

a+*c* é uma *er*

a+*c*+ não é *er*, logo (a+c+) não é expressão regular.

Exemplo 3.2.4.

Seja a expressão regular (1+10)*. Que linguagem define ?

Pela definição de fecho-estrela teremos sucessivamente:

$(1+10)^* = (1+10)^0 \cup (1+10) \cup (1+10)^2 \cup (1+10)^3 \cup \ldots$

$= \lambda \cup (1+10) \cup (1+10)(1+10) \cup (1+10)(1+10)(1+10) \cup \ldots$

$= \{\lambda, 1+10, (1+10)(1+10), (1+10)(1+10)(1+10), \ldots\}$

$= \{\lambda, 1+10, 11, 110, 101, 1010, (11+110+101+1010)(1+10), \ldots\}$

$= \{\lambda, 1+10, 11, 110, 101, 1010, 111, 1110, 1101, 11010, 1011, 10110, 10101, 101010, \ldots\}$

Qual a característica comum a todas estas cadeias?

Elas começam todas por 1 e podem ter um número arbitrário de 1's seguidos. Os 0's são introduzidos pela parcela 10 quando esta é potenciada. Mas sendo potenciada, introduz 10, 1010, 101010, …, ou seja, quando introduz um zero mete sempre 1 antes, e por isso não é possível introduzir dois zeros seguidos. O facto de ambas as parcelas da soma se iniciarem por 1 faz com que todas as suas concatenações possíveis se iniciem por 1. Poderemos concluir que a linguagem é composta por todas as cadeias em {0,1} que se iniciam por 1 e não têm qualquer par de zeros.

Exemplo 3.2.5

E no caso (0+1)*011 ?

A primeira parte (0+1)* produz todas as cadeias. A segunda parte, 011, concatena o sufixo 011 a todas as cadeias da primeira parte. Portanto temos a linguagem de todas as cadeias em {0,1} que terminam em 011.

Exemplo 3.2.6. Para 0*1*:

Teremos um número arbitrário de 0's (de 0*) seguido de um número arbitrário de 1's (de 1*). Podemos resumir dizendo que é a linguagem de todas as cadeias que não têm um "0" depois de "1".

Exemplo 3.2.7. Seja agora 00*11*.

Esta obteve-se da anterior colocando um 0 obrigatório no início, e um 1 obrigatório no meio. Portanto tem pelo menos um 0 e um 1.Além disso não aparece qualquer 0 depois de 1.A linguagem é portanto composta por todas as cadeias com pelo menos um "0" e um "1", e nenhum "0" depois de "1".

3.2.2. Linguagem associada a uma expressão regular

Definição 3.2.2. Se r é uma expressão regular, $L(r)$ denota a linguagem associada com r.

Esta linguagem é definida pelas regras seguintes:

1. $r=\varnothing$ é uma expressão regular , o conjunto vazio
2. $r=\lambda$ é uma expressão regular, o conjunto $\{\lambda\}$ com um elemento
3. Para todo o $a \in \Sigma$, $r=a$ é uma expressão regular denotando $\{a\}$

   Se r_1 e r_2 são expressões regulares, então

4. $L(r_1+r_2) = L(r_1) \cup L(r_2)$, união de duas linguagens
5. $L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$, concatenação de $L(r_1)$ com $L(r_2)$
6. $L((r_1)) = L(r_1)$
7. $L(r_1^*) = (L(r_1))^*$

As regras 4 a 7 usam-se para reduzir recursivamente uma linguagem L a expressões mais simples.

As regras 1 a 3 são as condições terminais para esta recursão.

Para se verificar qual a linguagem que corresponde a uma dada expressão regular, aplicam-se aquelas regras tantas vezes quantas as necessárias.

A **precedência dos operadores** é a seguinte

1º- fecho- estrela (*),

2º- concatenação (·),

3º- (+) união

Exemplos 3.2.8:

i) Seja o alfabeto $\Sigma=\{x\}$.

A linguagem de xx^* será $L(xx^*) = \{x, xx, xxx,...\} = L(x^+)$.

ii) E $L(x(xx)^*) = \{x, xxx, xxxxx,..\} = L(x^{impar})$.

Em $\Sigma = \{ a, b, c \}$

iii) A linguagem de $(a+c)b^*$ será

$L((a+c)b^*) = L(a+c)L(b^*)$ (regra5)= $(L(a) \cup L(c))\ (L(b))^*$ (regra 4 e regra 7) =

$= \{ a, c \}\ \{ \lambda, b, bb, bbb, \ldots \}$ (regra 3) =

$=\{ a, c, ab, cb, abb, cbb, abbb,..\}$

iv) $L(c+\varnothing) = \{ c \}$ (regra 4 e regra 3)

Em $\Sigma =$ $\{a, b\}$

iv) $L((a+b).(a+b).(a+b)) = L(a+b)\ L(a+b)\ L(a+b)$ (regra 4)

$= \{a,b\}\ \{a,b\}\ \{a,b\}$ (regra 3)

$= \{aaa, aba, abb, baa, bba,..\}$

v) Em $\Sigma = \{0,1\}$ escrever a expressão regular para a linguagem das cadeias que contêm um número ímpar de zeros, ou seja,

$L = \{w : w$ contém um número ímpar de zeros $\}$

Nota:

$(a+b)^* = (a+b)^* + (a+b)^*$ (a união de um conjunto consigo próprio dá o próprio conjunto)

$(a+b)^* = (a+b)^*(a+b)^*$ (a concatenação de qualquer cadeia de *a*'s e *b*'s com qualquer outra cadeia de *a*'s e *b*'s dá sempre uma cadeia de *a*'s e *b*'s; por outro lado $\lambda\lambda=\lambda$).

$(a+b)^* = a(a+b)^* + b(a+b)^* + \lambda$ (a união de todas as cadeias começadas por *a* com todos as cadeias começadas por *b* e com a cadeia vazia dá o conjunto de todas as cadeias com *a*'s e *b*'s mais λ.

Exercícios:

1. Qual é a linguagem representada pela expressão regular:

i) $(1+01+001)^*(\lambda+0+00)$

ii) $((0+1)(0+1))^* + ((0+1)(0+1)(0+1))^*$

2. Exemplos de linguagens complementares

Seja $\Sigma = \{0,1\}$.Encontrar uma expressão regular para as linguagens:

i) $L(r) = \{w \in \Sigma^* : w$ tem pelo menos um par de zeros consecutivos $\}$

ii) $L(r) = \{w \in \Sigma^* : w$ não tem qualquer par de zeros consecutivos $\}$

Resolução:

i) Uma cadeia da linguagem tem que conter "00" em algum lado. Antes ou depois do par de zeros pode conter qualquer cadeia arbitrária. Como sabemos uma cadeia arbitrária de zeros e uns é dada por $(0+1)^*$.

A solução tem por isso três partes; o par 00 precedido e sucedido por $(0+1)^*$ ou seja,

$$r_1 = (1+0)*00(1+0)*$$

Note-se que o enunciado não impede a existência de mais pares de zeros para além do obrigatório. Nunca confundir "ter um par de zeros" com "ter um só par de zeros".

ii) Esta linguagem é a complementar da anterior. No caso dos autómatos sabemos como desenhar o autómato do complemento a partir do autómato original: basta trocarmos a missão dos estados, passando os aceitadores a não aceitadores e os não aceitadores a aceitadores.

Com expressões regulares não há nenhuma técnica assim expedita de encontrar o complemento. Tem que se pensar na questão desde a base.

Se a linguagem não tem qualquer par de zeros consecutivos, quer dizer que sempre que aprece um "0" ele deve ser seguido imediatamente por um "1", i.e., o 0 só pode aparecer na cadeia "01"

Antes ou depois de "01" pode aparecer um número arbitrário de 1's, produzindo a cadeia 1....1011.....1. Portanto a linguagem contém (1...1011...1)*, ou, equivalentemente, (1*011*)*. Note-se que ao fecharmos (fecho-estrela) a expressão estamos a replicá-la um número arbitrário de vezes, permitindo assim gerar cadeias com um número arbitrário de zeros. Mas cada zero aparece acompanhado à esquerda e à direita por um número qualquer de 1's. Note-se que obrigamos todas estas cadeias a terminar em 1, o que é limitativo.

Não estão ainda incluídas as cadeias que terminam em "0". Poderemos obtê-las da expressão anterior concatenando-lhe um zero no fim, obtendo-se (1*011*)*0 .

Faltam também as cadeias que só têm 1's; para elas basta adicionar 1*.

Finalmente juntam-se as cadeias que só têm 1's excepto o último que é "0", 1*0.

Teremos a união destes subconjuntos todos, pelo operador + em expressões regulares:

$$(1*011*)* + (1*011*)*0 + 1* + 1*0$$

Este resultado responde à pergunta. Por uma questão de elegância podemos procurar simplificar a expressão.

Pondo em evidência os factores comuns às parcelas obtém-se a

$$r_2 = (1*011*)* (\lambda+0) + 1* (\lambda+0)$$

Pensando de outro modo, talvez mais simples, as cadeias sem zeros consecutivos são a repetição arbitrária das subcadeias "1" e "01", isto é, (1+01)*, às quais se pode ou não adicionar um zero no fim , obtendo-se r_3

$$r_3 = (1+01)*(0+\lambda)$$

equivalente à anterior, mas ainda mais elegante.

Para uma dada linguagem existe um número ilimitado de expressões regulares equivalentes. Importa no entanto obter uma que seja simples.

Não existe nada na forma de r_1 e r_2 ou r_1 e r_3 que sugira que definem linguagens complementares. Este facto indica uma das limitações das expressões regulares.

3.3.Regras algébricas para expressões regulares

As expressões regulares são expressões algébricas sobre as quais se podem fazer algumas operações (união, concatenação, fecho estrela). Essas operações devem obedecer a um certo número de regras. Podemos agrupá-las em regras comutativas, regras associativas, regras distributivas, identidades e anuladores, regras de fecho.

Duas expressões regulares são equivalentes se elas denotam a mesma linguagem. Quando se simplifica uma expressão regular, obtêm-se sucessivamente expressões regulares equivalentes.

3.3.1 Regras comutativas e associativas

Sejam L, M e N expressões regulares.

A união de duas expressões regulares é comutativa

$$L + M = M + L$$

A demonstração desta propriedade faz-se atendendo à natureza da operação em causa.

De facto a união de duas linguagens é o conjunto das cadeias de ambas, postas no mesmo "saco", por qualquer ordem. Podemos colocar primeiro as de *L* e depois as de *M*, ou ao contrário, que o conjunto resultante da união não se altera. Portanto a união é comutativa.

A união de linguagens é associativa

$$(L + M) + N = L + (M + N)$$

Se queremos fazer a união de três conjuntos, poderemos unir os primeiros dois e depois unir o resultante com o terceiro; ou podemos unir os dois últimos, unindo depois o primeiro com o conjunto resultante.

A concatenação de expressões regulares é associativa

$$(LM)N = L(MN)$$

A concatenação de três cadeias pode fazer-se indiferentemente de dois modos: concatenar as duas primeiras e depois a terceira à direita; ou concatenar as duas últimas e depois a primeira à esquerda.

Mas não é comutativa

$$LM \neq ML$$

A prova é imediata.

3.3.2 Regras distributivas

A concatenação tem alguma analogia com o produto algébrico e a união com a adição. A concatenação é distributiva à esquerda em relação à união:

$$L\,(M + N) = LM + LN$$

Por exemplo *aba*(*bba*+*abb)*=*ababba*+*abaabb.* Atendendo ao significado de +, união, temos que concatenar o prefixo *aba* com cada uma das cadeias dentro do parêntese. O resultado é a união dessas concatenações.

E também distributiva à direita

$$(M + N)L = ML + NL$$

Prova semelhante, agora com um sufixo em vez de prefixo.

A união não é distributiva em relação à concatenação nem à direita nem à esquerda

$$(MN) + L \neq (M + L)(N + L)$$

$$L + (MN) \neq (L + M)(L + N)$$

Para provar que assim é, faz-se prova por contradição: supõe-se que é igual e procura-se um caso em que não o seja. As propriedades têm que ser gerais, válidas para todas as linguagens e todas as cadeias em cada linguagem.

$$(ab)+c= (a+c)(b+c)$$

Ora $(a+c)(b+c)=ab+ac+cb+cc$ por definição de união e de concatenação. Mas isto é diferente de $(ab)+c$, o que é suficiente para a prova por contradição.

3.3.3 Identidades e anuladores (zeros)

Numa certa álgebra, a identidade é o elemento que operado com qualquer outro elemento dá esse elemento, tal como 1 na multiplicação numérica.

O conjunto vazio, $\varnothing$, é a identidade para a união porque

$$\varnothing + L = L + \varnothing = L$$

Já para a concatenação, a identidade é a cadeia vazia porque

$$\lambda L = L \lambda = L$$

O anulador é o elemento que operado com outro elemento dá sempre o conjunto vazio.

No caso da concatenação é o conjunto vazio

$$\varnothing L = L\varnothing = \varnothing$$

Para a união não existe anulador. E para a intersecção?

3.3.4. Regras do fecho-estrela

O fecho estrela é uma operação muito importante em expressões regulares quando se trata de linguagens infinitas.

Ele goza de algumas propriedades cuja demonstração fica ao cuidado do leitor.

(i) $(L^*)^* = L^*$

(ii) $L+ = LL^* = L^*L$

(iii) $L^* = L^+ + \lambda$

(iv) $\varnothing^* = \lambda$

(v) $\lambda^* = \lambda$

(vi) $(L+M)^* = (L^* M^*)^*$

Esta última merece alguns comentários, já que as restantes são mais ou menos evidentes.

Vejamos um exemplo de duas cadeias:

$$(a+b)^* = (a^*b^*)^*$$

À esquerda temos qualquer cadeia com qualquer número de *a*'s e *b*'s e por qualquer ordem. Qualquer destas cadeias também pode ser obtida pela parte direita. Se começa por *b*'s, faz-se primeiro a^0b^*, depois a^*b^0 para os *a*'s, novamente a^0b^* para *b*'s e assim sucessivamente. Repare-se que o fecho externo permite fazê-lo tantas vezes quantas as necessárias.

Poderemos ver que qualquer cadeia gerada pela expressão da direita pode ser gerada pela expressão da esquerda e vice-versa. Para uma prova mais formal ver Hopcroft p. 118.

3.3.5. Outras regras algébricas

Para provar uma regra algébrica qualquer, por exemplo

$$L + ML = (L + M)L$$

tem que se provar que qualquer cadeia gerada pela *er* da esquerda é também gerada pela RE da direita.

Para provar que é falsa, basta dar um contra exemplo.

Para auxiliar a prova podem-se substituir os símbolos das *er* por caracteres de um alfabeto, no caso por exemplo

$$(a+ba) = (a + b)a$$

que facilmente se vê ser falso (e portanto aquela propriedade é falsa).

3.4. Relação entre expressões regulares e linguagens regulares

As expressões regulares são como vimos uma forma de especificar linguagens regulares. Assim sendo para toda a linguagem regular existe uma expressão regular e, vice-versa, para toda a expressão regular existe uma linguagem regular.

Façamos a prova em duas etapas.

3.4.1 De uma expressão regular a um NFA: expressões regulares definem linguagens regulares

Uma linguagem é regular, como vimos no Cap.2, de for aceite por um DFA ou por um NFA (dado que estes são equivalentes a DFA).

Se pudermos construir um NFA a partir de uma expressão regular *r* qualquer, então essa expressão regular denotará uma linguagem regular $L(r)$. De facto assim é. Para o provar, recorre-se à definição recursiva de $L(r)$ e segue-se a definição 3.1 de expressão regular.

1º Definem-se NFA para as expressões regulares primitivas) os três primeiros elementos da definição 3.1)

i) NFA aceitador de $L_1=\varnothing$:

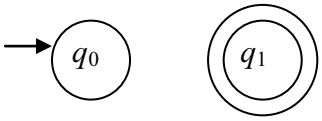


Figura 3.4.1. Aceitador do conjunto vazio

$r=\varnothing$

Não havendo possibilidade de alcançar o estado aceitador, e linguagem é o vazio.

ii) NFA aceitador de $L_2=\{\lambda\}$

$r=\lambda$

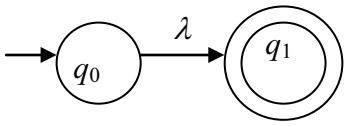


Figura 3.4.2. Aceitador de λ

iii) NFA aceitador de $L_3=\{a\}$

$r=a$

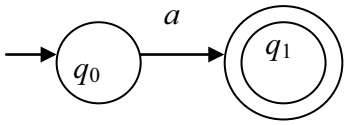


Figura 3.4.3. Aceitador de *a*

2º Admitamos que temos uma expressão regular *r* e que o NFA que aceira $L(r)$ é representado pela Figura 3.4.4 genérica,

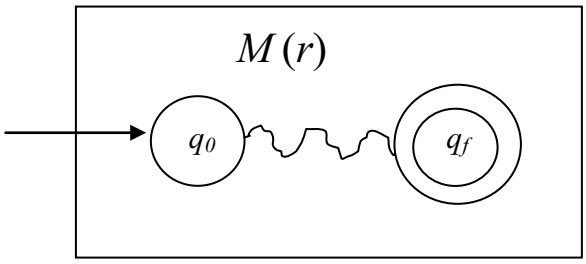


Figura 3.4.4 Um antómato *M* genérico

com o estado inicial q_0 e o estado final q_F. Note-se que qualquer NFA pode ser representado com um só estado final.

3º Suponhamos agora que temos dois NFAs $M(r_1)$ e $M(r_2)$ que aceitam as linguagens $L(r_1)$ e $L(r_2)$ definidas pelas expressões regulares r_1 e r_2. Construam-se os NFA para a segunda parte da Definição 3.2.2 do modo seguinte:

iv) $L(r_1+r_2)$

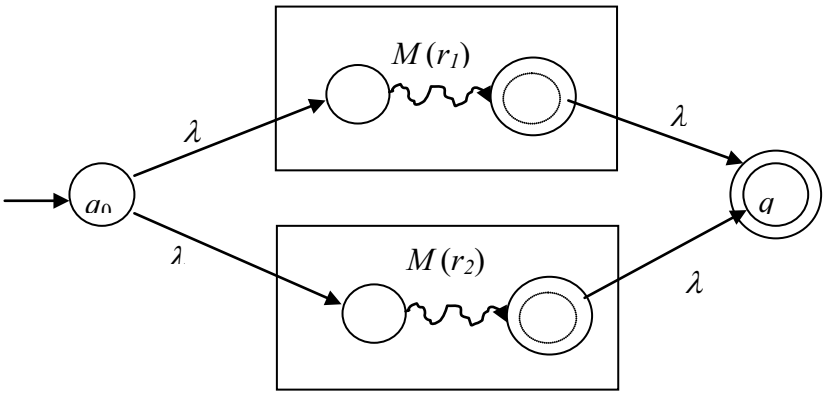


Figura 3.4.5. Autómato da soma de expressõe regulares

O NFA aceita a união das linguagens, L_1 pela parte de cima e L_2 pela de baixo.

v) $L(r_1r_2)$

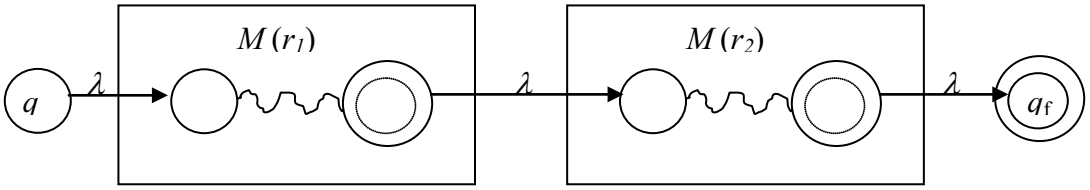


Figura 3.4.6. Autómato da concatenação de duas expressões regulares

O NFA aceita uma cadeia que resulte da concatenação de uma cadeia de L_1 (que leva o autómato ao estado aceitador da de $M(r_1)$) com uma cadeia de L_2 que lava depois a q_f.

vi) $L(r_1^*)$

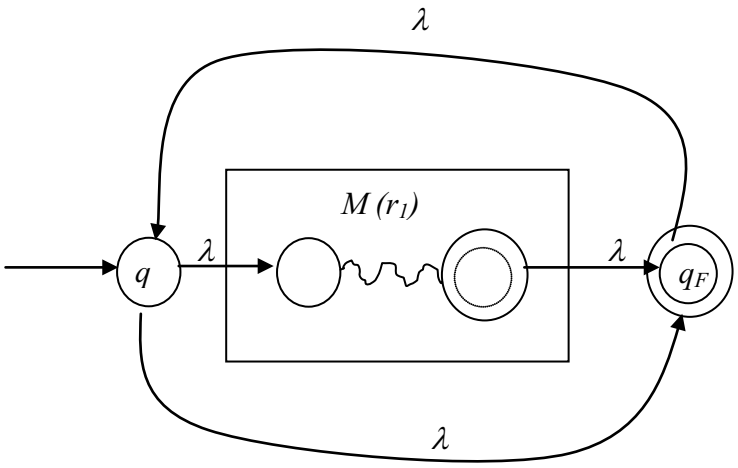


Figura 3.4.7. Autómato do fecho-estrela de uma expressão regular.

Este NFA já não é tão evidente. Repare-se que

$$r_1^* = \lambda + r_1 + r_1r_1 + r_1r_1r_1 + \ldots$$

O λ vai pela parte de baixo do NFA directamente para o aceitador, portanto está incluído na linguagem do NFA.

O r_1 vai através da máquina desde q_0 até q_f.

Já r_1r_1 parte de q_0 , vai até q_f e volta para q_0 pela parte de cima, com λ.

As outras potências de r_1 dão tantas voltas quantas as necessárias até ao estado final.

Usando estas máquinas é possível construir um NFA para qualquer expressão regular. Pode-se por isso enunciar o teorema:

Teorema 3.4.1. $L(r)$ é uma linguagem regular

Se r é uma expressão regular, existe algum autómato finito não-determinístico que aceita $L(r)$. Logo $L(r)$ é uma linguagem regular.

Exemplo 3.4.1.1

$r=1^*$

Sendo 1 uma expressão regular primitiva, desenha-se o seu autómato. Depois aplica-se-lhe a regra do fecho estrela.

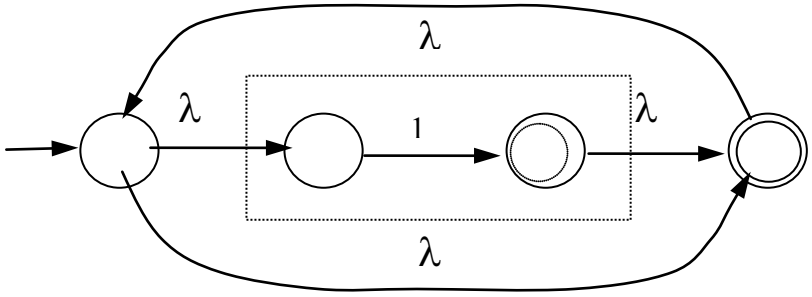


Figura 3.4.8 Autómato NFA de 1*

Agora pode-se demonstrar que este NFA é equivalente a

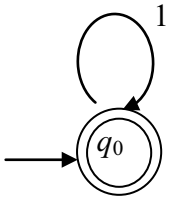


Figura 3.4.10. Autómato equivalente de 1*

Calculando o DFA equivalente obtém-se, em $\Sigma=\{0,1\}$

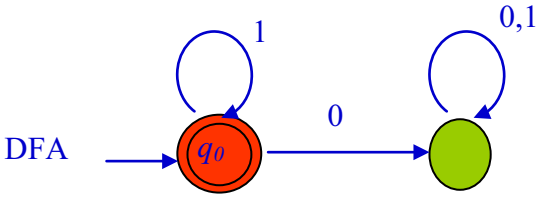


Figura 3.4.11. DFA de 1*

Exemplo 3.4.1.2.

$r=a+a^*b$

Desenham-se os NFA de *a* e de *b*, expressões regulares primitivas. A partir do de *a* desenha-se o de a^* que se concatena com o de *b.* Finalmente coloca-se um outro de *a* em paralelo com o de a^*b.

$$r=ab+(b+ab)^*$$

Desenha-se o de *a* que se concatena com o de *b*. Desenha-se novamente o de *b* que se paraleliza com o de *ab*. Agora aplica-se o fecho estrela ao conjunto obtido. Desenha-se depois um outro de *a e* um outro de *b* que se concatenam, colocando-se o conjunto em paralelo com o anterior.

3.4.2. De um NFA a uma expressão regular: expressões regulares para linguagens regulares

A toda a linguagem regular se pode associar um NFA e portanto um grafo de transições. Partindo do estado q_0, procuram-se todos os caminhos possíveis até ao estado final e as suas etiquetas.

É possível depois encontrar uma expressão regular que gere todas essas etiquetas. Para facilitar esta operação, usam-se os grafos de transição generalizados.

Grafos de transição generalizados são grafos de transição em que as arestas podem ser etiquetadas por expressões regulares (e não só por caracteres). No Cap. 2 chamámos-lhe δ^*. A etiqueta de um caminho desde o estado inicial até ao estado final é a concatenação das etiquetas das arestas do caminho, i.e., a concatenação de expressões regulares e portanto é uma expressão regular. As cadeias expressas por essa expressão regular são um subconjunto da linguagem aceite pelo grafo de transição generalizado; a linguagem total será a união de todos os subconjuntos gerados deste modo.

Temos então o seguinte caminho pata encontrar uma expressão regular para uma dada linguagem

NFA → grafos de transição generalizados → expressões regulares

O grafo de um NFA pode considerar-se um grafo generalizado desde que se interpretem apropriadamente as suas etiquetas.

Uma aresta etiquetada com um único carácter *a* interpreta-se como etiquetada pela expressão regular *a*.

Uma aresta etiquetada por vários caracteres *a, b, ...,* interpreta-se como etiquetada pela expressão regular *a* + *b* + ...

Pode-se assim afirmar que para toda a linguagem regular existe um grafo de transição generalizado que a aceita. Por outro lado toda a linguagem aceite por um grafo generalizado é uma linguagem regular.

A equivalência entre grafos generalizados define-se em termos da linguagem aceite.

Consideremos o exemplo da figura 3.4.12 de um grafo generalizado com estados Q= {*q, q*$_i$*, q*$_j$*, ...* }, em que *q* não é nem um estado inicial nem um estado final.

Pode-se obter um grafo generalizado equivalente, com menos um estado, eliminando o estado *q*.

No proceddimento tem que se assegurar que que não se altera a linguagem aceite pelo NFA e denotada pelo conjunto das etiquetas que se podem gerar desde o estado inicial até ao estado final.

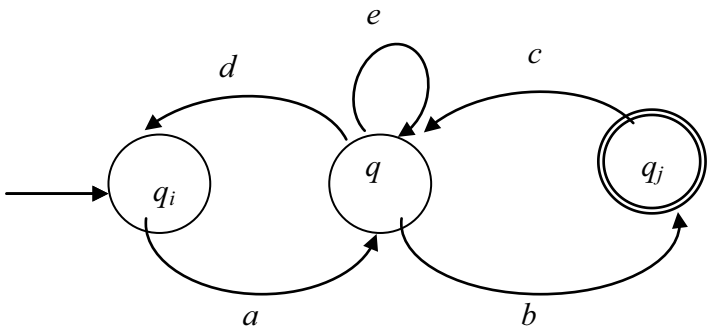


Figura 3.4.12. Grafo genérico de três estados. O do meio pode ser eliminado.

Para que isso aconteça temos que analisar com atenção as etiquetas (expressões regulares) das arestas das transição possíveis entre os estados que hão-de restar. Por exemplo para ir de *q*$_i$ até *q*$_i$, um ciclo fechado, vai-se com *a* até *q,* com um número qualquer de *e*'s, ou

seja e^*, mantém-se em q e depois com d regressa a q; concatenando dá ae^*d. Para ir de q_i até q_j passa-se em q e aí pode ler-se um número arbitrário de e's, seguindo depois para q_j com b. E de modo análogo para as outras transições. Resumindo,

$$q_i \to q_i : \; ae^*d$$

$$q_i \to q_j \; : \; ae^*b$$

$$q_j \to q_j \; : \; ce^*d$$

$$q_j \to q_i \; : \; ce^*b$$

Obtém-se assim o grafo de transição generalizado sem o estado q, Fig. 3.4.13.

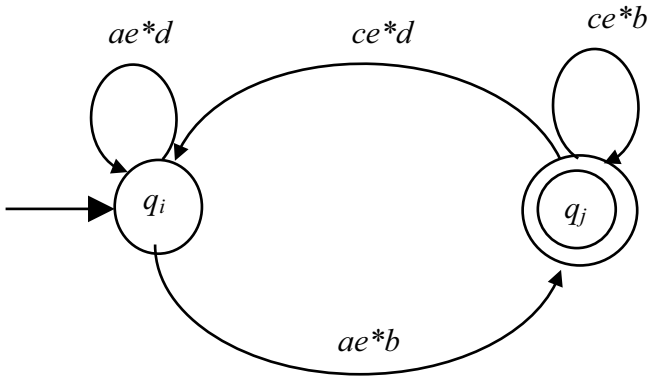


Figura 3.4.13. Grafo resultante da eliminação do estado intermédio da Fig. 3.4.12.

Note-se que no estado q entram e saem arestas para ambos os estados adjacentes e para si próprio. No caso de alguma delas não existir, omite-se o correspondente no grafo simplificado.

Este procedimento assegura que a linguagem aceite não é alterada. Num grafo com mais estados, mais complicado, o processo completo exige que este procedimento seja feito sucessivamente para todos os pares (q_i, q_j) em $Q - \{q\}$ antes de se remover q (i.e., todos os pares que estejam ligados a q). Por outro lado elimina-se um estado de cada vez, até restarem apenas dois. As transições generalizadas permitem-nos reduzir qualquer autómato finito a um outro com apenas dois estados.

Pode provar-se que a construção seguida produz um grafo generalizado equivalente ao inicial.

Pode agora, e em consequência, enunciar-se o teorema seguinte.

Teorema 3.4.2 (3.2). Seja L uma linguagem regular. Então existe uma expressão regular r tal que $L=L(r)$.

Demonstração:

Existe um NFA que aceita L, com um só estado final e tal que o estado inicial q_0 não é estado final. Pode-se interpretar como um grafo de transição generalizado

Aplica-se-lhe o procedimento anterior de eliminar sucessivamente vértices q, até que se fique apenas com o estado inicial e o estado final, obtendo-se o grafo da Figura 3.4.14 seguinte, em que r_1, r_2, r_3, r_4 são expressões regulares.

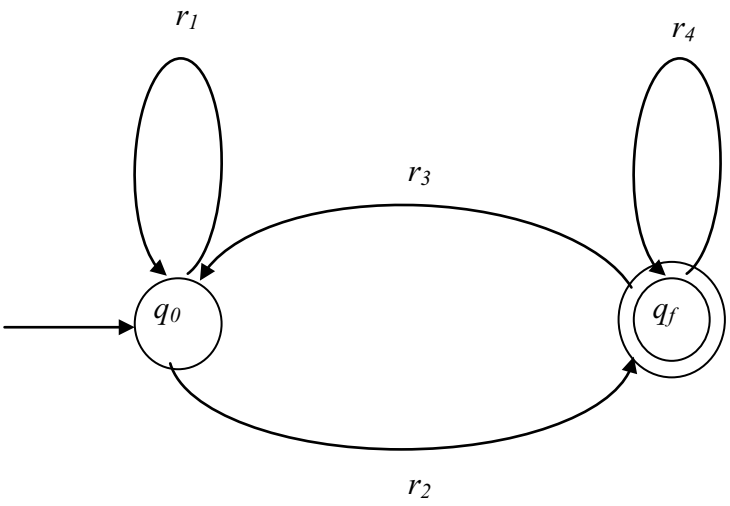


Figura 3.4.14, Qualquer NFA pode ser reduzido a esta forma.

A expressão regular que denota a linguagem aceite pelo grafo é

$$r=r_1{}^*r_2(r_4 + r_3r_1{}^*r_2)^*$$

De facto, para se ir de q_0 a q_f , pode-se ir por r_1 (um número arbitrário de vezes), seguido de r_2 ,

- seguido de r_4 ou, voltando para trás, de r_3 seguido de r_1* seguido de r_2 , e esta volta um número arbitrário de vezes.

No caso de se obter um grafo em que o estado inicial também é estado final,

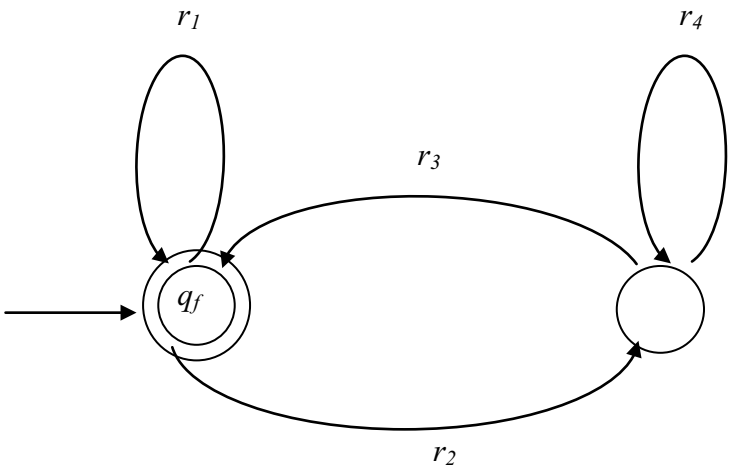


Figura 3.4 15. Caso em que o estado inicial é aceitador.

a expressão regular é

$$r=(r_1^*+(r_2r_4^*r_3))^*$$

Exemplo 3.4.2.1

Calcular a expressão regular do autómato da Fig. 3.4.16.

Antes de aplicar a construção subjacente ao teorema 3.2, é necessário introduzir uma pequena alteração no autómato para que o estado inicial não seja o aceitador. Basta para isso introduzir um estado adicional q que passa a ser o inicial e do qual se transita para A através de λ. Resulta um NFA equivalente.

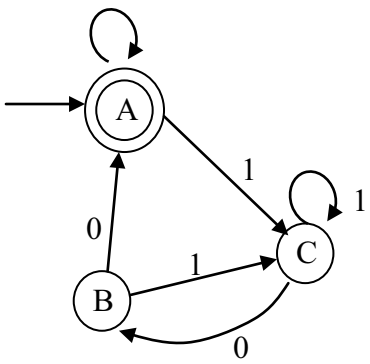


Figura 3.4.16. Exemplo 3.4.2.1

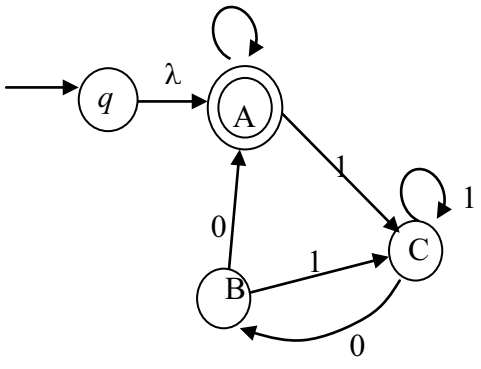


Figura 3.4.17. Introdução de um estado inicial na Fig. 3.4.16

Elimine-se o estado C. O caminho de A para B, passando por C é 11*0. De B para B passando por C é 11*0. De B para A passando por C não há. Teremos por isso

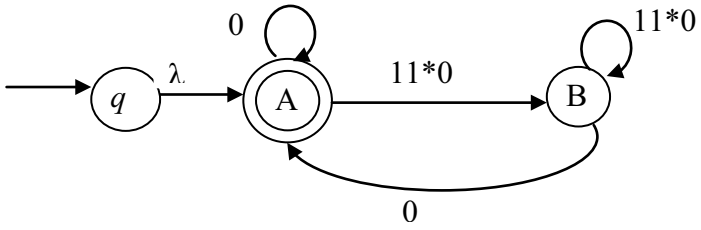
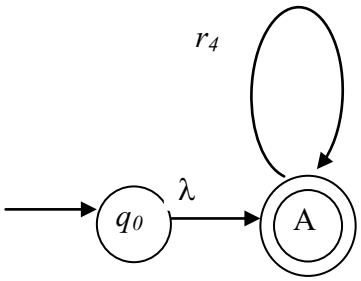


Figura 3.4.17. Eliminação do estado C.

Elimine-se agora B. Basta calcular o caminho de A para A passando por B, que é 11*0(11*0)*0. Ainda de A para A temos 0 que se soma ao anterior. Logo os caminhos totais de A para A são 0+11*0(11*0)*0 um número arbitrário de vezes, ou seja (0+11*0(11*0)*0)*. Logo teremos o autómato



$$r_4 = (\ 0 + 11*0(11*0)*0\ )*$$

Figura 3.4.18. Eliminação do estado B

Alicando agora o resultado geral obtém-se a expressão regular do autómato inicial,

$$r = \lambda(r_4 + \varnothing)^* = r_4^* = r_4$$

3.5. Gramáticas Regulares

Já vimos duas maneiras de especificar linguagens regulares: autómatos finitos (DFA ou NFA), e expressões regulares. Ambos têm as suas vantagens e inconvenientes. De uma podemos obter a outra. As gramáticas são uma terceira forma de especificação de linguagens,

como vimos no Cap. 1. No caso das linguagens regulares teremos gramáticas regulares, uma classe de gramáticas lineares.

3.5.1. Gramáticas lineares à esquerda e à direita

Definição 3.5.1. Gramática regular

Uma gramática $G = (V,T,S,P)$ diz-se **linear à direita** se todas as suas produções são da forma

$$A \rightarrow xB,$$

$$A \rightarrow x,$$

em que

$A, B \in V$, variáveis do conjunto das variáveis

$x \in T,$ símbolo do conjunto dos símbolos terminais

O seu nome deriva do facto de a variável aparecer à direita do símbolo terminal, e por isso as cadeias vão sendo produzidas da esquerda para a direita. É linear por ter apenas uma variável, tal como uma equação A=xB, é linear em B se x for uma constante.

Uma gramática diz-se **linear à esquerda** se todas as suas produções têm a forma

$$A \rightarrow Bx,$$

$$A \rightarrow x,$$

Uma gramática diz-se **regular** se ela é ou linear à esquerda ou linear à direita. Na parte direita das produções aparece no máximo uma variável e ela situa-se sempre na posição mais à direita ou sempre na posição mais à esquerda do lado direito das produções.

Exemplos de gramáticas regulares:

Exemplo 3.5.1.1

$G_1 = (\{S\}, \{a,b\}, S, P_1)$, linear à direita

$$P_1 : S \rightarrow abA$$

$$P_2 : S \rightarrow a$$

Também se pode escrever em forma mais compacta por

$$P : S \rightarrow abA|a$$

Como derivar *ababa* de G_1 ?

$$S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa$$

Aplicando P_1 sucessivamente duas vezes e depois P_2 para terminar.

De cada vez que se aplica P_1 introduz-se *ab*. Aplicando duas vezes, fica $(ab)^2S$ depois para eliminar *S* tem que se aplicar P_2 resultando em $(ab)^2a$. Aplicando três vezes P_1 fica $(ab)^3S$ e depois P_2 termina a derivação em $(ab)^3a$.

Generalizando conclui-se que a gramática G_1 produz a linguagem (regular) definida pela expressão regular

$$r=(ab)^*a$$

Exemplo 3.5.1.2

$G_2 = (\{S,A,B\}, \{a,b\}, S, P_2\}$, linear à esquerda

$$P_1 \qquad S \rightarrow Aab$$

$$P_2 \qquad A \rightarrow Aab \mid B$$

$$P_3 \qquad B \rightarrow a$$

Como derivar *aababab* de G_2 ?

$$S \Rightarrow Aab \Rightarrow Aabab \Rightarrow Aababab \Rightarrow Bababab \Rightarrow aababab$$

Aplicando P_2 sucessivamente conclui-se que esta gramática gera a linguagem regular definida pela expressão regular $r=aab(ab)^*$.

Exemplos de gramáticas não-regulares

Exemplo 3.5.1.3.

$G_3 = (\{S,A,B\}, \{a,b\}, S, P\}$

$$P_1 \quad S \Rightarrow A$$

$$P_2 \quad A \Rightarrow aB$$

$$P_3 \quad A \Rightarrow \lambda$$

$$P_4 \quad B \Rightarrow Ab$$

Esta tem produções lineares à direita (P_2) e outras lineares à esquerda (P_4) (P_1 e P_3 tanto são lineares à esquerda como à direita) por isso é **linear** mas **não regular**. Uma gramática é linear se do lado direito de cada produção aparece no máximo uma variável, seja à direita seja à esquerda. Se é regular é linear, mas nem todas as lineares são regulares.

Exemplo 3.5.1.4.

$G_4 = (\{S\}, \{a,b\}, S, P\}$

$$P: \quad S \Rightarrow aSSa \mid \lambda$$

é não linear porque do lado direito de uma produção aparecem duas variáveis (no caso duas vezes S).

As gramáticas regulares estão associadas com linguagens regulares. Para toda a linguagem regular existe uma gramática regular.

3.5.2. Gramáticas lineares à direita geram linguagens regulares

Uma linguagem gerada por uma gramática linear à direita é regular. Para o provar, constrói-se um NFA que imite as derivações da gramática linear à direita. Se se conseguir, é suficiente para provar que a linguagem é regular.

Numa gramática linear à direita, qualquer forma sentencial tem uma e uma só variável que é o símbolo mais à direita. Uma passo numa derivação resultante de uma produção resulta em

$$ab...c\underline{D} \Rightarrow ab...c\underline{dE}$$

Um NFA pode imitar esta produção se tiver um estado D e um estado E e entre os dois uma aresta etiquetada por d :

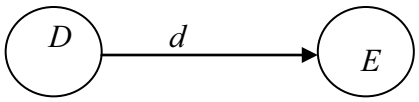


Figura 3.5.1 Produção $D \Rightarrow dE$

Estando no estado D e aparecendo d transita para E. É como se o autómato estivesse a ler as cadeias geradas pela gramática.

Os estados do autómato correspondem às variáveis das formas sentenciais. A parte da cadeia já processada foi obtida por construções semelhantes anteriores. Daí o teorema 3.5.1

Teorema 3.5.1.(3.3) Uma gramática linear à direita produz uma linguagem regular

Seja $G = (V,T,S,P)$ uma gramática linear à direita. Então $L(G)$ é uma linguagem regular.

Para o demonstrar constrói-se um NFA que imite as produções da gramática como visto acima.

Sejam

$V = \{V_0, V_1, ..., V_n\}$ o conjunto das variáveis da gramática

$S = V_0$ a variável inicial (axioma)

P : $V_0 \Rightarrow v_1 V_i$

$V_i \Rightarrow v_2 V_j$

...

$V_n \Rightarrow v_l$

em que os v's são sub-cadeias de um ou mais símbolos terminais.

Se w é uma cadeia em $L(G)$, então, necessariamente, a sua produção será

$V_0 \Rightarrow v_1 V_i$

$\Rightarrow v_1 v_2 V_j$

....

$\overset{*}{\Rightarrow} v_1v_2...v_kV_n$

$\Rightarrow v_1v_2...v_kv_l = w$

O NFA a ser construído imita cada derivação "consumindo" um *v* de cada vez. V_0, V_i, V_j, ..., V_n são estados do autómato, mas existem outros entre eles quando os *v*'s são cadeias com mais de um carácter. Além disso é preciso acrescentar o estado final.

Por exemplo se tivermos a produção:

$$V_i \rightarrow a_1a_2a_3...a_mV_j$$

então o NFA terá um caminho ligando V_i a V_j, ou seja $\delta(V_i, V_j)$ existe e será definida por

$$\delta^*(V_i, a_1a_2a_3...a_m) = V_j$$

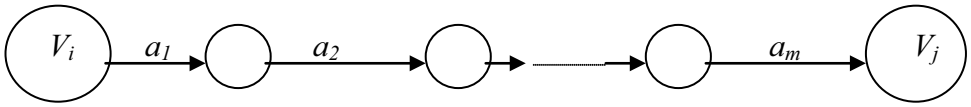


Figura 3.5.2. Produção $V_i \rightarrow a_1a_2a_3...a_mV_j$

Para uma produção

$$V_i \rightarrow a_1a_2a_3...a_m$$

a função de transição generalizada será o estado final V_f.

$$\delta^*(V_i, a_1a_2a_3...a_m) = V_f$$

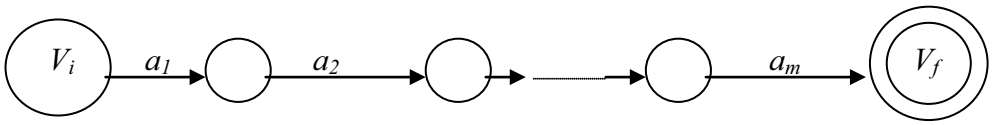


Figura 3.5.3. Produção $V_i \rightarrow a_1a_2a_3...a_m$

Se uma cadeia $w \in L(G)$ então ela foi gerada por um conjunto de produções definidas acima. No NFA existe, por construção, um caminho etiquetado w que vai de V_0 até V_f ou seja, w é aceite pelo NFA.

Se uma cadeia w é aceite pelo NFA, então, pelo modo como este foi construído, para aceitar w o autómato tem que passar por uma sequência de estados $V_0, V_i, \ldots$ até V_f usando caminhos etiquetados $v_1, v_2, \ldots,$ ou seja, w tem necessariamente a forma

$$w = v_1 v_2 \ldots v_k v_l$$

e por isso a derivação

$$V_0 \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \overset{*}{\Rightarrow} v_1 v_2 \ldots v_k V_n \Rightarrow v_1 v_2 \ldots v_k v_l = w$$

é possível e em consequência w pertence a $L(G)$.

q.e.d.

Exemplo 3.5.2.1

Seja $G = (\{S, A, B\}, \{a,b\}, S, P)$

com P:

$$S \rightarrow aA \mid aB ,$$

$$A \rightarrow bB \mid b ,$$

$$B \rightarrow aA \mid bB .$$

Derivação da cadeia *ababab* :

$$S \Rightarrow aA \Rightarrow abB \Rightarrow abaA \Rightarrow ababB \Rightarrow ababaA \Rightarrow ababab$$

Para se construir o NFA corresondente, cria-se um estado por cada variável mais um estado final, resultando em 4 estados *S,A,B,F*. Depois as transições entre cada dois estados extraem-se das produções: a etiqueta é o símbolo terminal da produção repectiva. As produções que permitem completar a derivação, as que têm no lado direito apenas caracteres

do alfabeto ou λ, levam necessariamente a uma estado final; no caso é apenas a produção $A \rightarrow b$. Resulta o autómato seguinte da Fig.3.5.4.

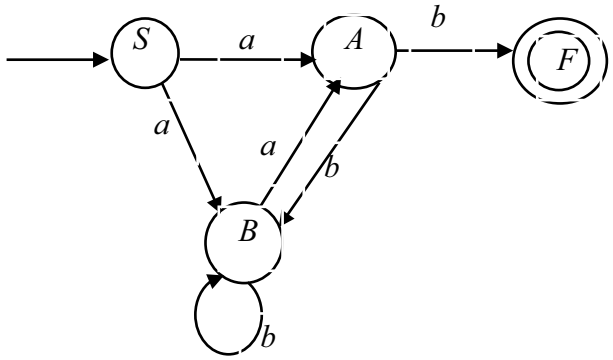


Figura 3.5.4. Autómato da gramática do exemplo 3.5.2.1.

3.5.3. Gramáticas lineares à direita a partir de autómatos finitos

Para provar que toda a linguagem regular pode ser gerada por uma gramática linear à direita

(i) constrói-se o DFA para a linguagem e

(ii) inverte-se a construção apresentada no teorema anterior 3.3. Os estados do DFA transformam-se nas variáveis da gramática, e os símbolos produtores das transições são os terminais do lado direito das produções. Teremos assim o

Teorema 3.5.2.(3.4) Se L é uma linguagem regular no alfabeto Σ, então existe uma gramática linear à direita $G=(V,\Sigma, S,P)$ tal que $L=L(G)$.

Demonstração:

Seja $M=(Q,\Sigma,\delta, q_0,F)$ o DFA que aceita L, tal que

$$Q= \{q_0 ,q_1,...q_n \}$$

$$\Sigma =\{a_1, a_2, ..., a_m\}$$

Vamos fazer a prova do teorema construindo uma gramática linear à direita a partir do DFA. Para isso

1º Construa-se a gramática linear à direita $G = (V, \Sigma, S, P)$, tal que

$V = \{q_0, q_1, \ldots q_n\}$ as variáveis da gramática são os estados do DFA

$S = q_0$

2º Para cada transição $\delta(q_i, a_j) = q_k$ no DFA M, introduz-se em P a produção

$q_i \rightarrow a_j q_k$ cria-se uma produção por cada transição entre dois estados

3º Se q_k faz parte de F, acrescenta-se a P a produção

$q_k \rightarrow \lambda$ cria-se uma produção que permite terminar as derivações.

A gramática G gerada deste modo pode produzir toda e qualquer cadeia de L.

Por outro lado, se uma cadeia w pertence a L, então ela pode ser derivada pela gramática G. A demonstração mais detalhada ser vista por exemplo em Linz p. 94.

Se partirmos de um NFA, em vez de um DFA, o teorema é igualmente válido. A construção é análoga, com as pequenas diferenças; agora teremos

2º Para cada transição $\delta(q_i, a_j) = q_k$ no DFA M, introduz-se em P a produção

$q_i \rightarrow a_j q_k$ cria-se uma produção por cada transição entre dois estados

Se existir também $\delta(q_i, a_j) = q_l$, introduz-se em P a produção ou seja, neste caso,

$q_i \rightarrow a_i q_l$, ou seja $q_i \rightarrow a_j q_k \mid a_j q_l$

Para cada transição $\delta(q_i, \lambda) = q_k$ no NFA M, introduz-se em P a produção $q_i \rightarrow q_k$ (mais além chamaremos unitária a esta produção)

A única diferença advém da escolha possível a partir de um estado, que se reflecte em duas produções possíveis.

Exemplo 3.5.3.1

Construir a gramática linear à direita pata a linguagem $L(aab^*a)$.

Começando por desenhar o NFA respectivo obtém-se

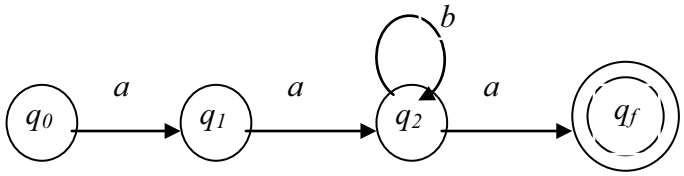


Figura 3.5.5.NFA para *aab*a*

A tabela de transições no NFA e das produções da gramática é a seguinte:

Tabela 3.5.3.1.Exemplo 3.5.3.1

Transições em M	Produções em G
$\delta(q_0, a)=\{q_1\}$	$q_0 \rightarrow aq_1$
$\delta(q_1, a)=\{q_2\}$	$q_1 \rightarrow aq_2$
$\delta(q_2,b)=\{q_2\}$	$q_2 \rightarrow bq_2$
$\delta(q_2, a)=\{q_f\}$	$q_2 \rightarrow aq_f$
$q_f \in F$	$q_f \rightarrow \lambda$

Para gerar por exemplo a cadeia *aaba* usam-se as formas sentenciais

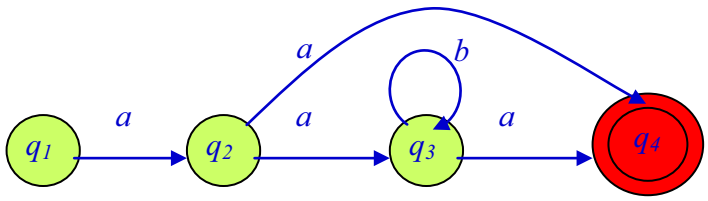
$$q_0 \Rightarrow aq_1 \Rightarrow aaq_2 \Rightarrow aabq_2 \Rightarrow aabaq_f \Rightarrow aaba$$

Exemplo 3.5.3.2.

Construir a gramática linear à direita para a linguagem $L(aab^*a+aa+ab^*a)$

A linguagem é a união de três:

*aab*a+aa*



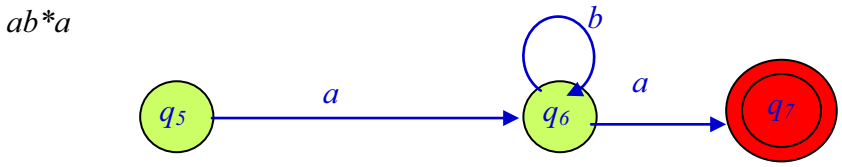


Figura 3.5.6. Resolução por partes do exemplo 3.5.3.2

Reduz-se a um NFA com um só estado inicial e um só estado final criando um estado inicial global e um estado final global com as necessárias transições λ-

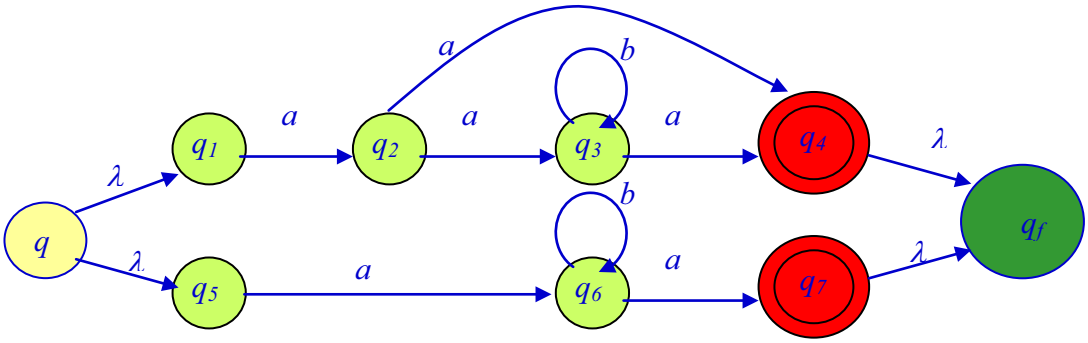


Figura 3.5.7. NFA resultante para o exemplo 3.5.3.2

Poderemos construir a tabela 3.5.3.2. das transições e das produções da gramática equivalente:

Tabela 3.5.3.2.

Transições em M	Produções em G
$\delta(q_0, \lambda)=\{q_1, q_5\}$	$q_0 \rightarrow q_1 \mid q_5$
$\delta(q_1, a)=\{q_2\}$	$q_1 \rightarrow aq_2$
$\delta(q_2,a)=\{q_3, q_4\}$	$q_2 \rightarrow aq_3 \mid aq_4$
$\delta(q_3, a)=\{q_4\}$	$q_3 \rightarrow aq_4$
$\delta(q_3, b)=\{q_3\}$	$q_3 \rightarrow bq_3$
$\delta(q_4, \lambda)=\{q_f\}$	$q_4 \rightarrow q_f$
$\delta(q_5, a)=\{q_6\}$	$q_5 \rightarrow aq_6$
$\delta(q_6, b)=\{q_6\}$	$q_6 \rightarrow bq_6$
$\delta(q_6,a)=\{q_7\}$	$q_6 \rightarrow aq_7$
$\delta(q_7, \lambda)=\{q_f\}$	$q_7 \rightarrow q_f$
$q_f \in F$	$q_f \rightarrow \lambda$

3.5.4. Equivalência entre linguagens regulares e gramáticas regulares

Nos parágrafos anteriores vimos a equivalência entre linguagens regulares, autómatos finitos e gramáticas lineares à direita. Que se passa com as gramáticas lineares à esquerda ?

Tal como para as gramáticas lineares à direita, também para as gramáticas lineares à esquerda se pode enunciar o teorema de equivalência:

Teorema 3.5.3 (3.5.)A linguagem L é regular se e só se existir uma gramática linear à esquerda G tal que $L=L(G)$.

Para demonstrar este teorema recorre-se à construção de uma gramática linear à esquerda que gera a linguagem reversa de L. Assim se prova que a reversa de L é regular. Mas se a reversa de L é regular, também o é, porque a reversão preserva a propriedade de regular. Para mais detalhes ver p. ex. em Linz p. 98.

Podemos finalmente enunciar o teorema de equivalência entre linguagens regulares e gramáticas regulares:

Teorema 3.5.4.(3.6) Uma linguagem L é regular se e só se existir uma gramática regular G tal que $L=L(G)$.

Este teorema é a conjugação dos dois anteriores.

3.6. Síntese das equivalências

Vimos neste capítulo que dada uma expressão regular, poderemos construir um autómato finito NFA ou DFA, e vice-versa. Vimos também que dado um DFA ou um NFA podemos a partir deles extrair uma gramática regular; e dada uma gramática regular poderemos construir um autómato finito equivalente. Temos assim diversas equivalências possíveis (note-se que qualquer relação de equivalência é uma relação transitiva). Estas equivalências estão esquematizadas na figura seguinte.

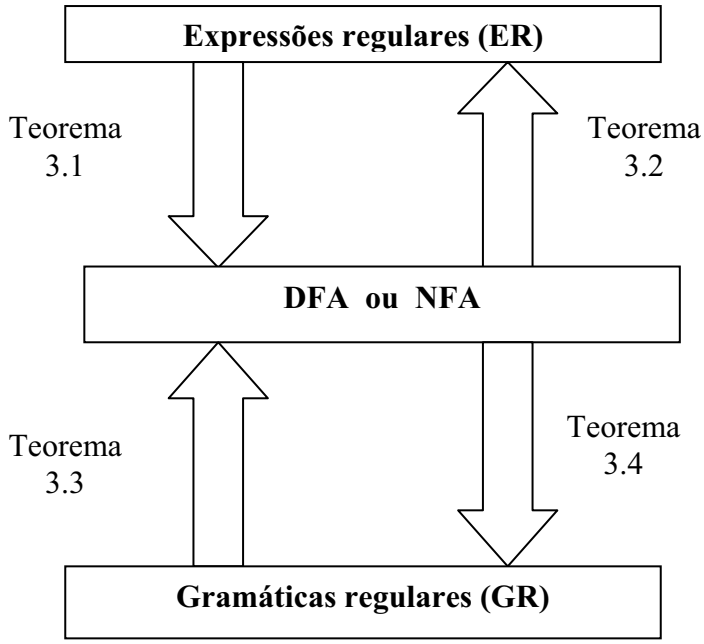


Figura 3.6.1. Várias formas de descrever as linguagens regulares (LR)

Teorema 3.1.(3.4.1) ER $\rightarrow$ NFA

Teorema 3.2 LR $\rightarrow$ ER (LR $\rightarrow$ DFA $\rightarrow$ GTG $\rightarrow$ ER)

Teorema 3.3 Gramáticas lineares à direita $\rightarrow$ LR

(Gramáticas lineares à direita $\rightarrow$ NFA $\rightarrow$ LR)

Teorema 3.4 LR $\rightarrow$ Gramáticas Lineares à direita

(LR $\rightarrow$ DFA $\rightarrow$ Gramáticas lineares à direita)

Bibliografia:

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

Models of Computation and Formal Languages, R. Gregory Taylor, Oxford University Press, 1998.

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.