Linguagens - 1 e 2

19 de maio de 2024 15:42

Introdução

Alfabeto: conjunto finito de símbolos cujas combinações formam os elementos da linguagem

Sintaxe: Diz quais, de todas as combinações de símbolos do alfabeto, pertencem à linguagem

Semântica: descreve o significado dos elementos da linguagem

Ao conjunto de todas as strings sobre um alfabeto T chama-se T^*
Sejam L_1 e L_2 subconjuntos de T^*. Define-se a concatenação de duas linguagens

$$L = L_1 \ . \ L_2 = \{\mu \in T^* | \mu = \mu_1 . \mu_2 \wedge \mu_1 \in L_1 \wedge \mu_2 \in L_2\}$$

Ist é, é o conjunto das strings que se obtêm concatenando strings de L_1 com strings de L_2.

• Expressões Regulares

1. ϕ carateriza a linguagem {} (linguagem vazia - a que não pertence qualquer string).
2. ε carateriza a linguagem $\{\varepsilon\}$ (linguagem cuja única string é a string vazia. É de notar a diferença com a anterior!).
3. **a** carateriza a linguagem {a}
4. **p + q** carateriza a linguagem $L_p \cup L_q$
5. **p q** carateriza a linguagem $L_p . L_q$
6. **(p)** carateriza a linguagem L_p
7. $\mathbf{p}^*$ carateriza a linguagem L_p^*
8. $\mathbf{p}^+$ carateriza a linguagem L_p^+

a) $e = (a)^+$

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

c) $e = ('+' + '-' + \varepsilon)('0' + \dots + '9')^+$

d)

e = sinal . mantissa . expoente
sinal = $('+' + '-' + \varepsilon)$
mantissa = digitos . (digito . '.' + '.' . digito + ε) . digitos
expoente = ε + ('E' . sinal . digito . $\underbrace{\text{digitos}}_{\text{pode ser } \varepsilon}$)
digitos = digito = ('0' + ... '9')

$- 321.2E-123$

- Álgebra das expressões regulares

1. $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
2. $\alpha + \beta = \beta + \alpha$
3. $\alpha + \phi = \phi + \alpha = \alpha$
4. $\alpha + \alpha = \alpha$
5. $(\alpha \,.\, \beta) \,.\, \gamma = \alpha \,.\, (\beta \,.\, \gamma)$
6. $\alpha \,.\, \varepsilon = \varepsilon \,.\, \alpha = \alpha$
7. $\alpha \,.\, (\beta + \gamma) = \alpha \,.\, \beta + \alpha \,.\, \gamma$
8. $(\beta + \gamma) \,.\, \alpha = \beta \,.\, \alpha + \gamma \,.\, \alpha$

Normal

9. $\alpha^+ = \alpha \,.\, \alpha^* = \alpha^* \,.\, \alpha$
10. $\alpha^* = \varepsilon + \alpha^+$
11. $(\alpha + \varepsilon)^+ = (\alpha + \varepsilon)^* = \alpha^*$
12. **se** $\boxed{X = \beta + \alpha \,.\, X}$ **então** $X = \alpha^* \,.\, \beta$
13. **se** $X = \beta + X \,.\, \alpha$ **então** $X = \beta \,.\, \alpha^*$

Mais interessante

$$X_1 = \beta + \alpha . X$$
$$X_2 = \beta + \alpha . (\beta + \alpha . X)$$
$$= \beta + \alpha . \beta + \alpha . \alpha . X$$
$$X_3 = \beta + \alpha . \beta + \alpha . \alpha . \beta + \alpha . \alpha . \alpha . X$$
$$\Rightarrow X = (\varepsilon + \alpha + \alpha . \alpha + \alpha . \alpha . \alpha \cdots) . \beta$$
$$= \alpha^* . \beta$$

$$X_1 = \beta + X . \alpha$$
$$X_2 = \beta + (\beta + X . \alpha)\alpha$$
$$= \beta + \beta . \alpha + X . \alpha . \alpha$$

$$X_3 = \beta + \beta . \alpha + (\beta + X . \alpha) . \alpha \alpha$$
$$= \beta + \beta . \alpha + \beta . \alpha . \alpha + X . \alpha \alpha$$
$$= \beta . (\varepsilon + \alpha + \alpha . \alpha + \alpha . \alpha . \alpha + \cdots)$$

$$\Rightarrow X = \beta . \alpha^*$$

Exercício 1.2)

- *a execução da tarefa* $\boldsymbol{A}$ *seguida da execução da tarefa* $\boldsymbol{B}$ *se representa pela expressão regular* $\boldsymbol{A.B}$
- *a execução da tarefa* **se c então A senão B** *se representa pela expressão regular* $\boldsymbol{c.A} + \neg \boldsymbol{c.B}$
- *a execução da tarefa* **skip** *se representa pela expressão regular* ε.
- *a execução da tarefa* **abort** *se representa pela expressão regular* ϕ.

a)
i) enquanto c fazer A
ii) repetir A até c

i) A primeira ação a executar é o cálculo da condição c; se esta for falsa não se executa mais nada; senão executa-se a ação A e depois voltamos ao mesmo ponto. Isto é,

$$\neg c.\varepsilon + c.A.(\neg c.\varepsilon + c.A.(\cdots))$$
$$= \neg c.\varepsilon + c.A.\neg c + c.A.c.A.\neg c + \cdots$$
$$= (c.A)^*.\neg c$$

↳ faz sentido

ii) $A.(c.\varepsilon + \neg c.(A.(c.\varepsilon + \neg c.A.\cdots)))$
$$= A.c + A.\neg c.A.c + A.\neg c.A.\neg c.A.\neg c + \cdots$$
$$= (A.\neg c)^*.A.c$$

(b) *Usando as propriedades de expressões regulares (e o facto de* $\neg\neg c = c$*) mostre que:*

(i) *(enquanto c fazer A) = (Se c então (repetir A até ¬ c) senão skip)*

(ii) *(repetir A até c) = (A; enquanto ¬c fazer A)*

$$(c.A)^*.\neg c = c.(A.c)^+.A.\neg c + \neg c.\varepsilon$$
$$= (c.A)^+.\neg c + \neg c.\varepsilon$$
$$= ((c.A)^+ + \varepsilon).\neg c$$
$$= (c.A)^*.\neg c$$

ii) $(A.\neg c)^*.A.c$
$$= A.((c\neg.A)^*.c)$$
$$= (A.\neg c)^*.A.c$$

ou

$$A.(c.\varepsilon + \neg c.(A.(c.\varepsilon + \neg c.A\cdots)))$$
$$= A.c + A.\neg c.A.c + A.\neg c.A.\neg c.A.c + \cdots$$

$$= (A.\neg C)^{*}.A.C$$

(c) *Considere o procedimento P definido como*

$$P = \textbf{se } c \textbf{ então } A \textbf{ senão } (B;P)$$

(i) *Determine uma expressão regular que caraterize a sua execução.*

(ii) *Usando as propriedades das expressões regulares, determine uma versão iterativa (usando **enquanto**) do procedimento **P**.*

$P =$ se c então A senão $(B;P)$

$$c.A + \neg c.B.P$$

$$c.A + \neg c.B.(c.A + \neg c.B.P)$$

$$= c.A + \neg c.B.c.A + \neg c.B.\neg c.B + P + \ldots$$

$$= (\neg c.B)^{*}.c.A$$

ii)

$$\underbrace{(c.A)^{*}.\neg c}$$

$$\big(\overbrace{(\neg c.B)^{*}.c}\big).A$$

$=$ (enquanto $\neg c$ fazer B); A

1.2 GRAMÁTICAS

T : conjunto finito de símbolos terminais (alfabeto)
N : conjunto de símbolos não terminais
S : símbolo não terminal especial chamado símbolo inicial
P : é um conjunto de produções, sendo que:

> Uma produção é un par $(\alpha\ ,\ \beta)$, e escrevemos $\alpha \to \beta$, em que α e β são sequências finitas de símbolos terminais e não terminais. Dada uma produção $\alpha \to \beta$ dizemos que α é o lado direito e que β é o lado esquerdo da produção.

$$\alpha \in (T \cup N)^{*} \text{ e } \beta \in (T \cup N)^{*}$$

Dizemos que α deriva imediatamente em β e notamos α ⇒ β
se e só se

$$\exists_{\Phi, \Psi \in (N \cup T)^{*}, (\delta_1 \to \delta_2) \in P} \ . \ \alpha$$

$$= \Phi \delta_1 \Psi \wedge \beta = \Phi \delta_2 \Psi$$

Isto é, exsite uma produção em que
- o lado esquero é uma substring de α
- β pode ser obtida substituindo δ1 por δ2 em α.

$\alpha \to$ lado direito

- β pode ser obtida substituindo δ1 por δ2 em α.

α → lado direito
β → lado esquerdo

$G = (T, N, S, P)$

$L = \{ \mu \in T^* . S \Rightarrow \mu \}$

(conjunto de strings que se podem derivar do símbolo inicial S)

Exercício 1.3

Exercício 1.3 *Escreva gramáticas para especificar a sintaxe das seguintes linguagens:*

(a) strings constituidas por um ou mais ***a****'s*

$S \rightarrow a$
$S \rightarrow aS$

ou $S \rightarrow a \mid aS$

(b) strings constituidas por um ou mais ***a****'s seguidas de zero ou mais* ***b****'s*

$S \rightarrow AB$
$A \rightarrow a \mid aA$
$B \rightarrow \varepsilon \mid bB$

(c) strings que representam os números inteiros

$S \rightarrow AB$
$A \rightarrow \varepsilon \mid \text{'+'} \mid \text{'-'}$
$B \rightarrow D \mid DB$
$D \rightarrow \text{'0'} \mid \text{'1'} \mid \cdots \mid \text{'9'}$

(d) strings que representam os números reais (em Pascal, i.e., com parte inteira, parte decima e expoente opcionais mas tem de existir parte inteira ou parte decimal)

$S \rightarrow ABE \mid ABDBE$
$A \rightarrow \varepsilon \mid \text{'+'} \mid \text{'-'}$
$B \rightarrow C \mid CB$
$C \rightarrow \text{'0'} \mid \cdots \mid \text{'9'} \mid$
$D \rightarrow \text{'.'}$
$E \rightarrow \text{'E'}\ A\ B \mid \varepsilon$

$S \rightarrow$ Sinal Mantissa Expoente
Sinal $\rightarrow \text{'+'} \mid \text{'-'} \mid \varepsilon$

S → Sinal Mantissa Expoente
Sinal → '+' | '-' | ε
Mantissa → Digitos Meio Digitos
Digitos → ε | Digito Digitos
Meio → Digito '.' | '.' Digito | ε
Digito → '0' | ... | '9'
Expoente → ε | 'E' Sinal Digito Digitos

Pelo menos 1 digito

Uma das ideias chaves na escrita de gramáticas é identificar os blocos constituintes das strings a caraterizar e associar-lhes um símbolo não terminal, como no exemplo acima.

Exercício 1.4 *Escreva gramáticas para especificar a sintaxe das seguintes linguagens:*

(a) *strings constituidas por zero ou mais **a**'s seguidas de zero ou mais **b**'s sendo o número de **a**'s igual ao número de **b**'s.*

S → a S b | ~~ab~~ | ε

↳ não é preciso

S → a S B | ε

(b) *strings constituidas por zero ou mais **a**'s seguidas de zero ou mais **b**'s seguidas de zero ou mais **c**'s sendo o número de **a**'s mais o número de **b**'s igual ao número de **c**'s.*

$a^* b^* c^*$

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

S → a S c | b S c | ε

(c) *strings constituidas por **a**'s e **b**'s sendo o número de **a**'s igual ao número de **b**'s.*

~~S → Aux | ε~~

~~Aux → ab | a Aux b~~

↳ a's e b's podem estar misturados

S → a B | b A | ε

A → a S | b A A (strings com mais um a do que b's)

$S \rightarrow aB \mid bA \mid \varepsilon$

$A \rightarrow aS \mid bAA$ (strings com mais um a do que b's)

$B \rightarrow bS \mid aBB$ (strings com um b a mais do que a's)

aaababbb

aB talvez

↳ $aBB \rightarrow \boxed{b}S$

↳ $aBB \rightarrow aBB \rightarrow \boxed{b}S$

↳ $\boxed{b}S$ ↳ $\boxed{b}S$

↳ ε ↳ ε

aababbb

a BB

a (aBB) (aBB)

1.5) $T = \{-, M, I\}$

$\forall x \in (-)^*,\ xM-Ix \in L$

$\forall x, y, z \in (-)^*,$

$xMyIz \in L \Rightarrow xMy-Iz- \in L$

$S \rightarrow -S- \mid M\ Aux$

$Aux \rightarrow -Aux- \mid -I-$

$-M-I--$

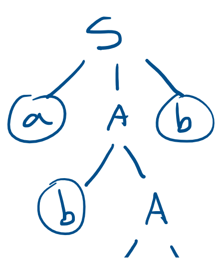
$-M--I--$

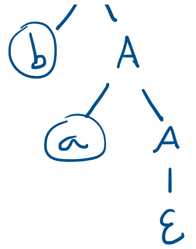
pode haver sempre um maior número de $-$

1.6) $S \rightarrow aAb$

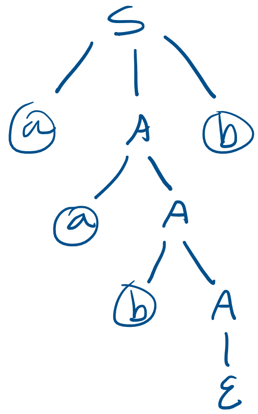
$A \rightarrow aA \mid bA \mid \varepsilon$

a) abab





b) aabb



2 Linguagens Regulares

2.1 Gramáticas Regulares

Uma gramática independente do contexto G=(T,N,S,P) diz-se regular à direita, quando todas as produções são da forma:

A -> µ

ou A -> µ B

em que A,B ∈ N e µ ∈ T*

2.1

(a) strings constituidas por um ou mais a's seguidas de zero ou mais b's

não terminal

$$\left.\begin{array}{l} S \to a\,A \\ A \to B \mid a\,A \\ B \to \varepsilon \mid b\,B \end{array}\right\}$$ à direita

Não terminal

$$\left.\begin{array}{l} S \to A \mid S\,b \\ A \to a \mid A\,a \end{array}\right\}$$ à esquerda

(Início)

mais b's

à direita quando todas as produções são da forma:

$$A \rightarrow \mu$$

$$\text{ou } A \rightarrow \mu\ B$$

e em que A,B $\in$ N e $\mu \in T^*$

Isto é, todas as produções têm do lado direito, no máximo **um** símbolo não terminal (e este é o último símbolo do lado direito).

à esquerda quando todas as produções são da forma:

$A \rightarrow \mu$ terminal

ou $A \rightarrow B\ \mu$ não terminal

e em que A,B $\in$ N e $\mu \in T^*$

Isto é, têm do lado direito, no máximo **um** símbolo não terminal (e este é o primeiro símbolo do lado direito).

$S \rightarrow A \mid S\,b$ (Início)

$A \rightarrow a \mid A\,a$

aaabbb

S

Sb

$a \leftarrow A\,a \leftarrow A\,a \leftarrow S\,b \leftarrow S\,b$

Gramática regular à esquerda

b) strings que representam números interos

à direita:

$S \rightarrow \text{'+'}\,N \mid \text{'-'}\,N \mid N$

$N \rightarrow \text{'0'} \mid \cdots \mid \text{'9'}$

$N \rightarrow \text{'0'}\,N \mid \cdots \mid \text{'9'}\,N$

à esquerda

+12

$S \rightarrow Sinal \mid S\,N$

$Sinal \rightarrow \text{'+'} \mid \text{'-'} \mid \varepsilon$

$N \rightarrow \text{'0'} \mid \ldots \mid \text{'9'}$

ou

- S → A '0' | A '1' | ...| A '8' | A '9'
- S → S '0' | S '1' | ...| S '8' | S '9'
- A → '+' | '-' | ε

Exercício 2.2)

a) Escreva uma gramática regular à direita para caracterizar a sintaxe dessa linguagem

$S \rightarrow I \mid E$

$I \rightarrow A \mid +A \mid -A$

à direita

$I \rightarrow A \mid +A \mid -A$
$A \rightarrow d \mid dA$
$E \rightarrow F \mid +F \mid -F$
$F \rightarrow dF \mid d \cdot A$

à direita

$S \rightarrow A d$
$A \rightarrow \varepsilon \mid '+' \mid '-' \mid Ad \mid F d '.'$
$F \rightarrow \varepsilon \mid '+' \mid '-' \mid Fd$

à esquerda

2.2 Conversão de gramáticas em expressões regulares

Exercício 2.3

- $S \rightarrow Sinal\ RealSemSinal$
- $Sinal \rightarrow \varepsilon | + | -$
- $RealSemSinal \rightarrow Inteiro\ ParteDecimal$
- $Inteiro \rightarrow digito \mid digito\ Inteiro$
- $ParteDecimal \rightarrow \varepsilon \mid \text{'.'}\ Inteiro$

Determine uma expressão regular equivalente.

$S = Sinal\ RealSemSinal$
$Sinal = \varepsilon + '+' + '-'$
...

Solução:

$S = (\varepsilon + '+' + '-')(digito)^+(\varepsilon + '.'\ (digito)^+)$

2.2.1 Gramáticas Simplesmente Regulares

Ao símbolo Z chama-se <u>símbolo final</u>

2.2.2 Conversão de GR em GSR

Seja $G_1 = (T_1, N_1, S_1, P_1)$ uma gramática regular à direita. Então é possível escrever uma gramática $G_2 = (T_2, N_2, S_2, P_2)$ simplesmente regular à direita tal que a linguagem gerada é a mesma.

Um processo standard para atingir este objectivo é o seguinte:

- $S_2 = S_1$
- $T_2 = T_1$ → terminais
- $N_2 = N_1 \cup \{X_1, X_2, ..., X_n\} \cup Z$ → não terminais
- $Z \to \varepsilon \in P_2$ → Conjunto de produções
- **se** $A \to B \in P_1$ **então** $A \to B \in P_2$ } igual
- **se** $A \to a\,B \in P_1$ **então** $A \to a\,B \in P_2$ } igual
- **se** $A \to a \in P_1$ **então** $A \to a\,Z \in P_2$ } igual + Z
- **se** $A \to a_1 a_2 ... a_m B \in P_1$ **então**
  - $A \to a_1 X_1 \in P_2$
  - $X_1 \to a_2\, X_2 \in P_2$
  - ....
  - $X_{m-2} \to a_{m-1}\, X_{m-1} \in P_2$
  - $X_{m-1} \to a_m B \in P_2$

  separação
- **se** $A \to a_1 a_2 ... a_m \in P_1$ **então**
  - $A \to a_1 X_1 \in P_2$
  - $X_1 \to a_2\, X_2 \in P_2$
  - ....
  - $X_{m-2} \to a_{m-1}\, X_{m-1} \in P_2$
  - $X_{m-1} \to a_m Z \in P_2$

2.2.1 Gramáticas Simplesmente Regulares (GSR)

Uma gramática G=(T,N,S,P) diz-se **simplesmente regular**

à direita quando todas as produções são da forma:

$$A \to a\,B$$

em que $A,B \in N$ e $\mathbf{a} \in T$, com excepção para a produção

$$Z \to \varepsilon$$

à esquerda quando todas as produções são da forma:

$$A \to B\,a$$

em que $A,B \in N$ e $a \in T^*$, com excepção para a produção

$$Z \to \varepsilon$$

Ao símbolo Z chama-se o **símbolo final**.

Exercício 2.4

$S \to I \mid E$

$I \to A \mid +A \mid -A$

$A \to d \mid dA$

$E \to F \mid +F \mid -F$

$F \to dF \mid d \cdot A$

GR para GSR

$$\boxed{\begin{array}{l} S \to I \mid E \\ I \to A \mid \text{'+'}A \mid \text{'-'}A \\ A \to dZ \mid dA \\ E \to F \mid \text{'+'}F \mid \text{'-'}F \\ F \to dF \mid d\,X_1 \\ X_1 \to \text{'.'}\,A \\ Z \to \varepsilon \end{array}}$$

$S \to A\,d$

$A \to \varepsilon \mid \text{'+'} \mid \text{'-'} \mid Ad \mid F\,X_1$

$F \to \varepsilon \mid \text{'+'} \mid \text{'-'} \mid Fd$

$X_1 \to d\,\text{'.'}$

$Z \to \varepsilon$

2.3 Autómatos

$A = (T, Q, S, Z, \delta)$

T é o alfabeto
Q é um conjunto de estados
S ∈ Q é um estado especial chamado estado inicial
Z ∈ Q é um conjunto de estados chamados estados finais
δ é uma função de transição de estados

Representamos graficamente um automato da seguinte forma:
- a cada estado fazemos corresponder um círculo com o identificador de estado;
- os estados finais são assinalados fazendo-lhes corresponder um círculo diferente;
- sempre que para um dado estado q, existe um símbolo a do alfabeto tal que δ(q, a) = q', então marcamos uma seta do círculo correspondente ao estado q para o símbolo q' com a etiqueta a.
- o estado inicial é marcado fazendo-lhe chegar uma seta sem origem

Dado um autómato, γ é uma string da linguagem caraterizada por esse aut´omato sse existir um caminho do estado inicial para um estado final tal que a concatenação das etiquetas de todos os seus ramos é igual a γ

2.3.1 Automatos Não Determinísticos (AND)

$$\delta : Q \times (T \cup \{\varepsilon\}) \to P(Q) \cup \{\perp\}$$

A função de transição de estado é uma função que dado um estado e um símbolo do alfabeto dá como resultado um conjunto de estados. Esta função é parcial, i.e., existem pares de valores (q,s) tais que δ(q, s) = ⊥.

2.3.2 Conversão de GSR em AND

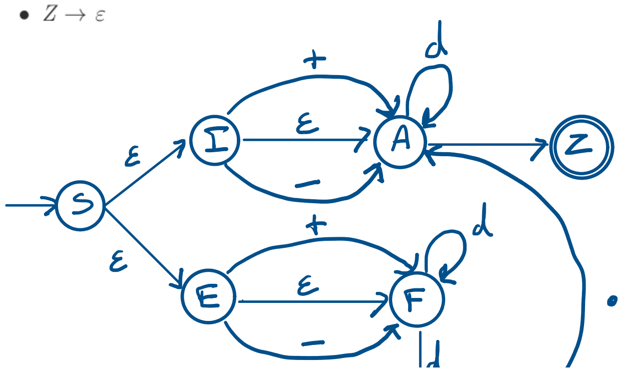
Dada um gramática simplesmente regular à direita o autómato não deterministico correspondente pode ser obtido da seguinte forma:

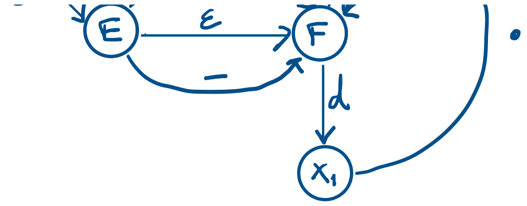
N

- a cada símbolo não terminal corresponde um estado.
- o estado a que corresponde o símbolo S (símbolo inicial da gramática) é o estado inicial do autómato.
- o estado correspondente ao símbolo Z é o estado final do autómato.
- para cada produção da forma (A → B) ∈ P coloca-se uma seta do estado correspondente a A para o estado correspondente a B com etiqueta ε. $A \xrightarrow{\varepsilon} B$
- para cada produção da forma (A → a A) ∈ P coloca-se uma seta do estado correspondente a A para o estado correspondente a B com etiqueta **a**. $A \xrightarrow{a} B$

Exercício 2.5

- $S \to I \mid E$
- $I \to A \mid \text{'+'}\ A \mid \text{'-'}\ A$
- $A \to d\ Z \mid d\ A$
- $E \to \text{'+'}\ F \mid \text{'-'}\ F \mid F$
- $F \to d\ F \mid d\ X_1$
- $X_1 \to \text{'.'}\ A$
- $Z \to \varepsilon$

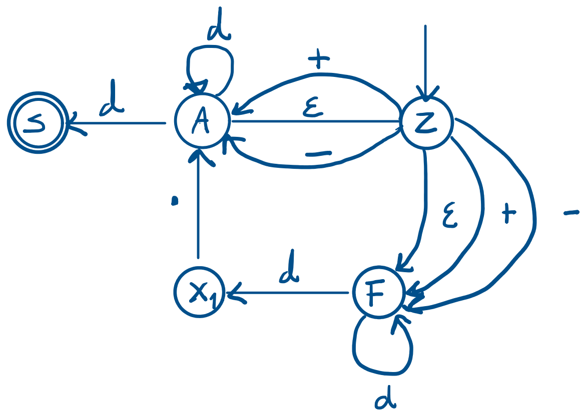




Dada uma gramática simplesmente regular à esquerda, o autómato não deterministico correspondente pode ser obtido da seguinte forma:

- a cada símbolo não terminal corresponde um estado.
- o estado a que corresponde o símbolo S (símbolo inicial da gramática) é o estado final do autómato.
- o estado correspondente ao símbolo Z é o estado inicial do autómato.
- para cada produção da forma (A → B) ∈ P coloca-se uma seta do estado correspondente a B para o estado correspondente a A com etiqueta ε.
- para cada produção da forma (A → a A) ∈ P coloca-se uma seta do estado correspondente a B para o estado correspondente a A com etiqueta **a**.

- $S \rightarrow A\ d$
- $A \rightarrow Z \mid Z\ '+' \mid Z\ '-' \mid A\ d \mid X_1\ '.'$
- $X_1 \rightarrow F\ d$
- $F \rightarrow Z \mid Z\ '+' \mid Z\ '-' \mid F\ d$
- $Z \rightarrow \varepsilon$



2.3.3 Autómatos Determinísticos

$$\delta : Q \times T \rightarrow Q \cup \{\perp\}$$

Um autómato determinístico A é um autómato em que a função de transição de estado é uma função que dado um estado e um símbolo do alfabeto dá como resultado um estado.
Esta função é parcial, i.e., existem pares de valores (q,s) tais que δ(q,s) = ⊥.

2.3.4 Conversão de AND em AD

Seja N = (T,Q,S,Z,δ) um autómato determinístico.

Considere-se a seguinte função:

fecho : P(Q) $\rightarrow$ P(Q)

fecho (X) = X $\cup \bigcup_{x \in X} fecho(\delta(x, \varepsilon))$

isto é, o fecho de um conjunto de estados X: é o conjunto de estados a que se pode chegar dos estados de X através de transições por ε.

Para a partir de N = (T,Q,S,Z,δ) se obter o autómato D = (T',Q',S',Z',δ') correspondente procede-se da seguinte forma:

- T' = T
- S' = fecho ({S})
- δ' (X,t) = fecho ($\cup_{q \in X}, (\delta(q, t))$)
- Q' define-se recursivamente da seguinte forma:
  - (i) fecho({S}) $\in$ Q'
  - (ii) **se** q $\in$ Q' **então** δ' (q,t) $\in$ Q'

Exercício 2.8

S → A | aS | bA

A → ab | abA | baS

a) S → A | aS | bA

A → aX_1 | aX_2 | bX_3

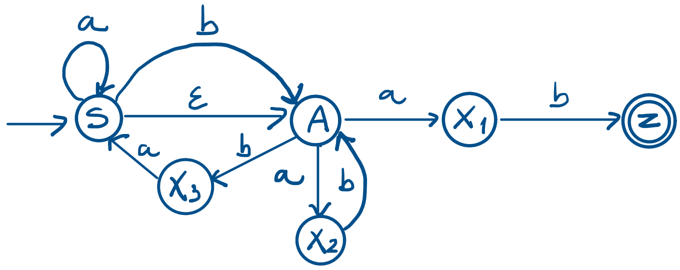
X_1 → bZ

X_2 → bA

X_3 → aS

Z → ε

b)

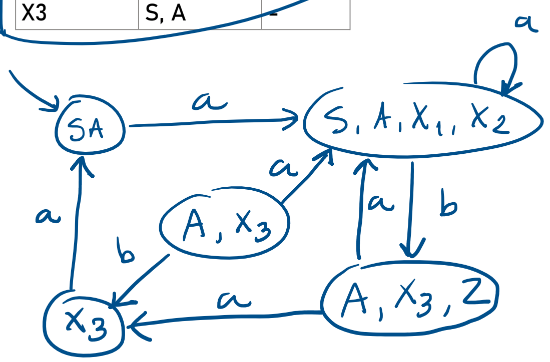


c) fecho S

fecho (X) = X $\cup \bigcup_{x \in X} fecho(\delta(x, \varepsilon))$

Estado	a	b
S, A	S, A, X1, X2	A, X3
S, A, X1, X2	S, A, X1, X2	A, X3, Z
A, X3	X1, X2, S, A	X3
A, X3, Z	X1, X2, S, A	X3
X3	S, A	–

Uma linha para cada novo estado encontrado

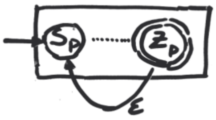


2.3.5 Conversão de ER em AND

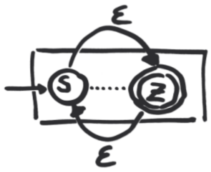
4. $\mathbf{e} = \mathbf{p} + \mathbf{q}$ sendo **p** e **q** expressões regulares

5. $\mathbf{e} = \mathbf{p}\ .\ \mathbf{q}$ sendo **p** e **q** expressões regulares

6. $\mathbf{e} = \mathbf{p}^{+}$ sendo **p** uma expressão regular

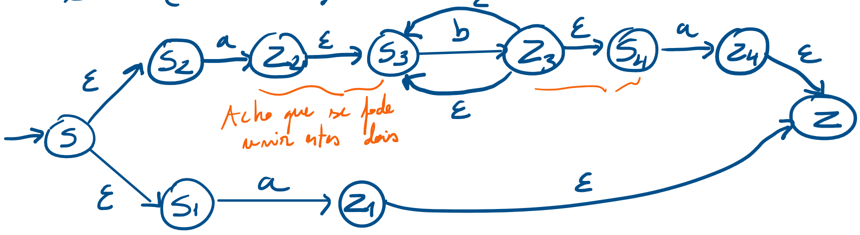


7. $\mathbf{e} = \mathbf{p}^{*}$ sendo **p** uma expressão regular



Exercício 2.9

$e = (a\, b^{*} a) + a$



2.4 Reconhecedores Baseados em AD

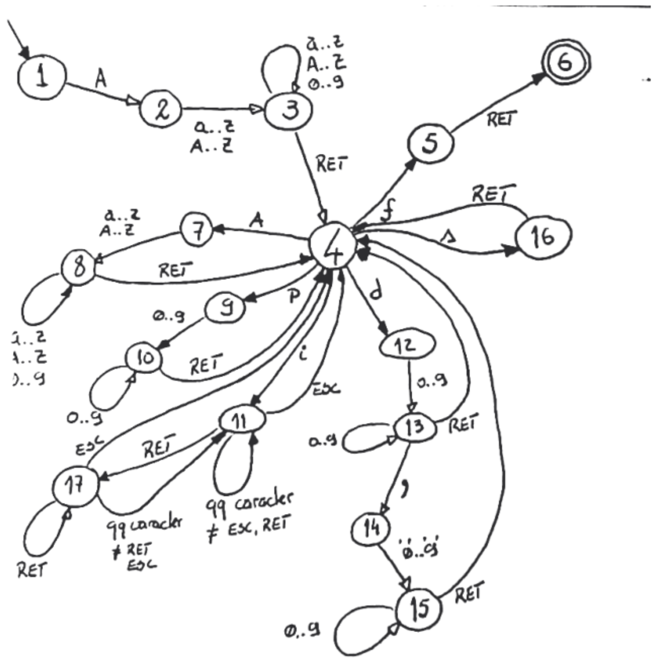
2.5 Análise de um exemplo

Vamos tentar escrever um editor de texto muito simples que permita:

- abrir um ficheiro **A** ***ficheiro***
- posicionar numa determinada linha **P** ***linha***
- inserir texto terminado por ESC **i** ***texto*** **esc**
- apagar a linha atual **d**
- apagar as linhas desde uma posição a outra **d** ***linha***$_1$ ***linha***$_2$
- escrever o ficheiro **s**
- terminar a edição **f**

Todos os comandos (excepto o de inserção de texto) devem ser terminados com RETURN.

S → Comando S | Fim
Comando → ComAbr | ComPos | ComIns | ComApa | ComEsc
ComAbr → A nome RETURN
ComPos → P inteiro RETURN
ComIns → i texto ESC
ComApa → d RETURN | d inteiro ',' inteiro RETURN
ComEsc → s RETURN
Fim → f RETURN
nome → alfabético | nome alfanumérico
inteiro → digito | digito inteiro
texto → ε| linha texto
linha → RETURN | char linha
numérico →'0' | '1' | ... | '9'
alfanumérico → numérico | alfabético
alfabético → 'a' | 'b' | ... | 'z' | 'A' | 'B' | ... | 'Z'
char → qualquer carater diferente de ESC e de RETURN



makes sense

Est\Simb	...	RET	...	ESC	...	A	...	P	...	Z	...	,	...	a	...
1						2									
2						3	3	3	3	3				3	3
3		4				3	3	3	3	3				3	3
4						7		9							
5		6													
6															
7						8	8	8	8	8				8	8
8		4				8	8	8	8	8				8	8
9															
10		4													
11	11	17	11	4	11	11	11	11	11	11	11	11	11	11	11
12															
13		4										14			
14															
15		4													
16		4													
17	11	17	11	4	11	11	11	11	11	11	11	11	11	11	11

Est\Simb	...	d	...	f	...	i	...	s	...	z	...	0	...	9	...
1															
2	3	3	3	3	3	3	3	3	3	3					
3	3	3	3	3	3	3	3	3	3	3		3	3	3	
4		12		5		11		16							
5															
6															
7	8	8	8	8	8	8	8	8	8	8					
8	8	8	8	8	8	8	8	8	8	8		8	8	8	
9												10	10	10	
10												10	10	10	
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
12												13	13	13	
13												13	13	13	
14												15	15	15	
15												15	15	15	
16															
17	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11