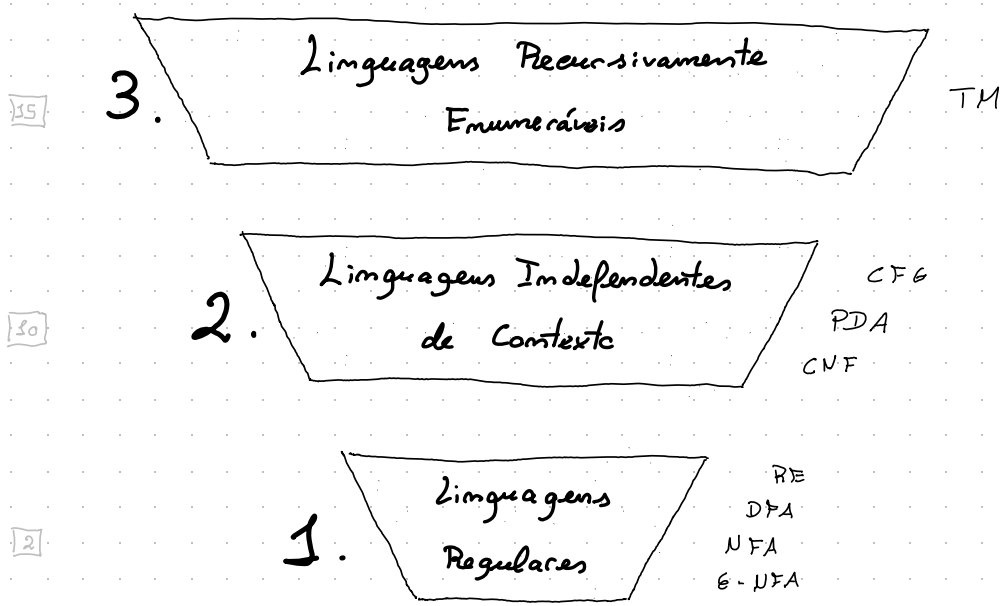
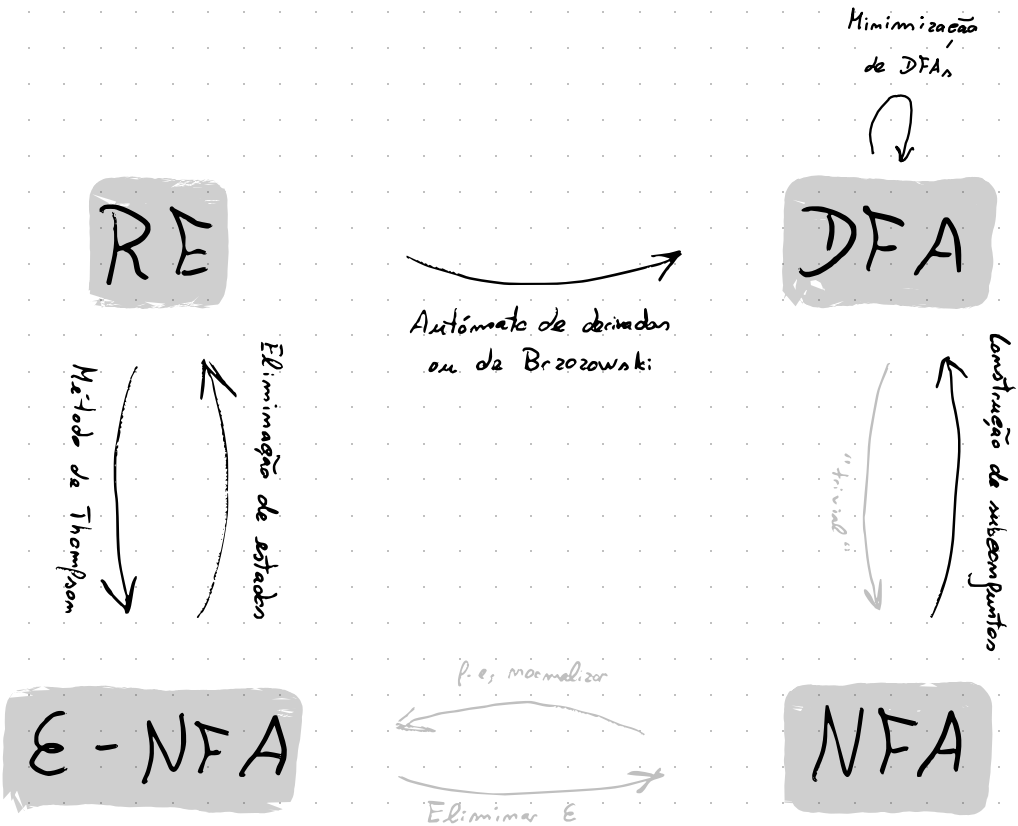
Teoria da Computação

Lista de algoritmos



1. Linguagens Regulares

- Fecho de Kleene
- Autómato produto
- Lema da repetição



Autómatos:

- **RE** | Expressão Regular
- **DFA** | Autómato Finito Determinístico
- **NFA** | Autómato Finito Não-determinístico
- **ε-NFA** | NFA com transições por ε

Fecho de kleene

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots = \bigcup_{m \geq 0} \Sigma^m$$

Ex// $\Sigma = \{a, b\}$

$\Sigma^* = \{ \varepsilon,$
$\Sigma^0 \rightarrow$
$\Sigma^1 \rightarrow a, b$
$\Sigma^2 \rightarrow aa, ab, ba, bb,$
$\Sigma^3 \rightarrow aaa, aab, aba, abb, baa, bab, bba, bbb,$
$\ldots \quad \ldots \}$

Autómato produto

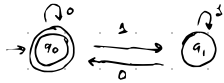
$A_3 = \langle Q_3, \Sigma, \delta_3, i_3, F_3 \rangle$

- $Q_3 = Q_1 \times Q_2$
- $i_3 = (i_1, i_2)$
- δ é aplicado a ambos os estados
- F_3 é definido por uma operação booleana à escolha

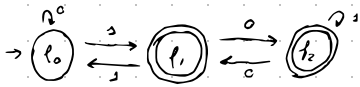
Ex// Ficha 3.1, ex. 2 a)

② $\Sigma = \{a, b\}$

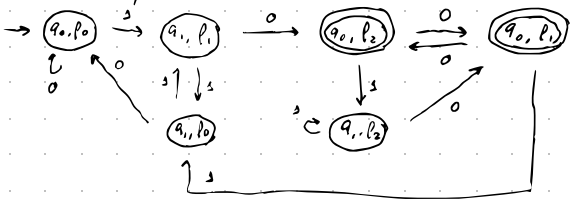
- Autómato: Múltiplos de 2 | $A_1 = \langle \{q_0, q_1\}, \Sigma, \delta_1, q_0, \{q_0\} \rangle$



- Autómato: Não múltiplos de 3 | $A_2 = \langle \{l_0, l_1, l_2\}, \Sigma, \delta_2, l_0, \{l_1, l_2\} \rangle$



- Autómato produto: $A_1 \cap A_2 = A_3$



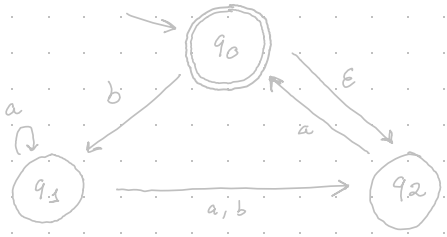
Construção de subconjuntos

NFA
↳ DFA

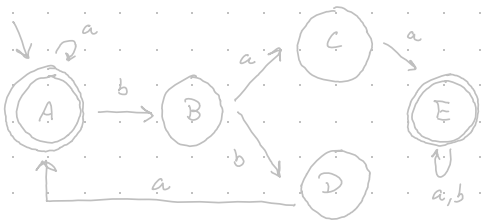
1. Construir uma tabela de transições em que a primeira linha é o conjunto de estados iniciais.
2. Preencher o conjunto de possíveis estados para cada palavra consumida.
3. Se surgir um conjunto de estados novo, criar uma nova linha.
4. Repetir até não surgir mais nenhum novo conjunto.
5. (opcional) Renomear cada um dos conjuntos, já que representam estados do DFA.

Ex: Ficha 4.1 | Ex. 1b)

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



δ_D	a	b
→ ✲ $\{q_0, q_2\}$	$\{q_0, q_2\}$	$\{q_1\}$
$\{q_1\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_2\}$
$\{q_2\}$	$\{q_0, q_2\}$	$\emptyset$
✲ $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$

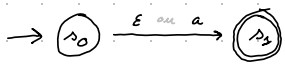


δ_D	a	b
→ ✲ A	A	B
B	C	D
C	E	D
D	A	$\emptyset$
✲ E	E	F

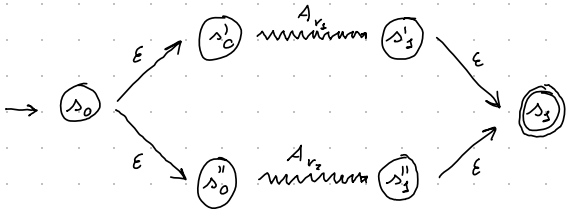
Método de Thompson

RE
↳ ε - NFA

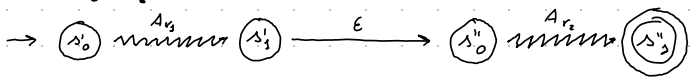
- $r = \varepsilon$ ou $r = a$, $a \in \Sigma$



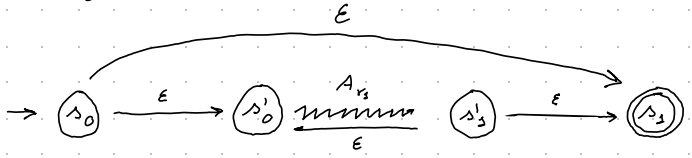
- $r = r_1 + r_2$



- $r = r_1 \cdot r_2$



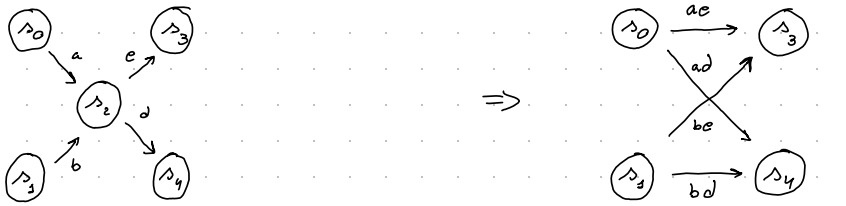
- $r = r_1^*$



Eliminação de estados

ε-NFA
↳ RE

A partir de um autómato normalizado, colapsar transições e estados até apenas existir um estado inicial e final e uma única transição entre eles que corresponde à expressão regular.



...

Autómato de derivadas / Brzozowski

RE
↳ DFA

1. Derivar a RE em relação a cada símbolo do alfabeto
2. Repetir a cada nova RE até não aparecer mais nenhuma expressão nova.
3. Sendo cada RE um estado, construir um autómato em que:
   - ↳ Estado inicial | RE inicial
   - ↳ Transições entre estados | Dita pelo símbolo consumido na derivada
   - ↳ Estado final | Todos os estados que contêm ϵ

Derivada de uma RE por um símbolo

$D_a(a) = \epsilon$

$D_a(\emptyset) = D_a(\epsilon) = D_a(b) = \emptyset$

$D_a(r_1 + r_2) = D_a(r_1) + D_a(r_2)$

$D_a(r_1 \cdot r_2) = D_a(r_1)\, r_2 + \epsilon(r_1)\, D_a(r_2)$

$D_a(r_1^*) = D_a(r_1)\, r_1^*$

Ex. Ficha 5.1 | Ex. 3a)

A $D_a(a^* + ba)$
$= D_a(a^*) + D_a(ba)$
$= D_a(a)\, a^* + D_a(b)\, a + \epsilon(b)\, D_a(a)$
$= \epsilon a^* + \emptyset a + \emptyset\epsilon$

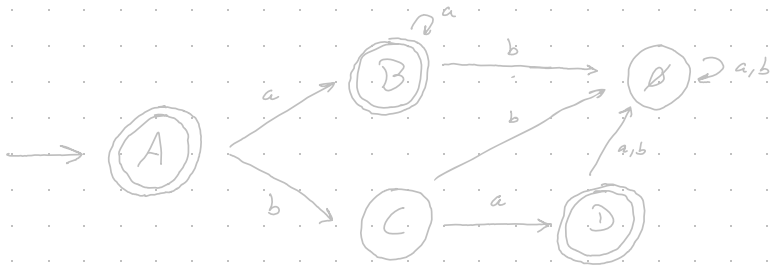
$D_b(a^* + ba)$
$= D_b(a^*) + D_b(ba)$
$= D_b(a)\, a^* + D_b(b)\, a + \epsilon(b)\, D_b(a)$
$= \emptyset a^* + \epsilon a + \emptyset\emptyset$

B $= a^*$

C $= a$

B $D_a(a^*) = D_a(a)\, a^* = \epsilon a^* = a^*$
$\emptyset$ $D_b(a^*) = D_b(a)\, a^* = \emptyset a^* = \emptyset$

D $D_a(a) = \epsilon$
$\emptyset$ $D_b(a) = \emptyset$



Minimização de DFAs

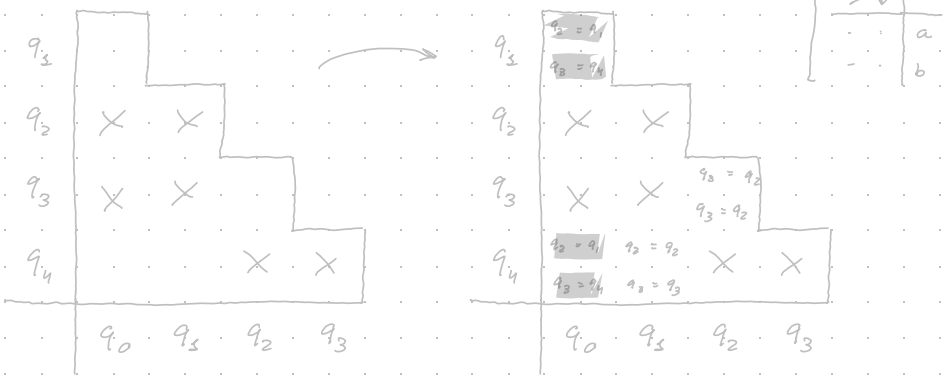
1. Construir uma tabela triangular inferior, em que cada coluna representa um estado exceto o último, e cada linha representa um estado exceto o primeiro.
2. Preencher os pares de estados que são incompatíveis. Num passo inicial é quando um estado é final, mas o outro não é.
3. Continuar a assinalar pares incompatíveis. A partir daqui um par é incompatível quando os estados transitam para um par já assinalado, após um símbolo do alfabeto ter sido consumido.
4. Repetir até que não sejam assinalados mais pares de estados. Os conjuntos não assinalados representam estados compatíveis e cada par pode ser representado num só estado.
5. Construir o novo autómato em que cada par compatível é representado num único estado.

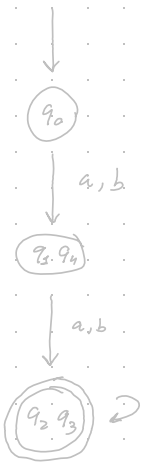
Ex. Ficha 5.2 | Ex. 5b)

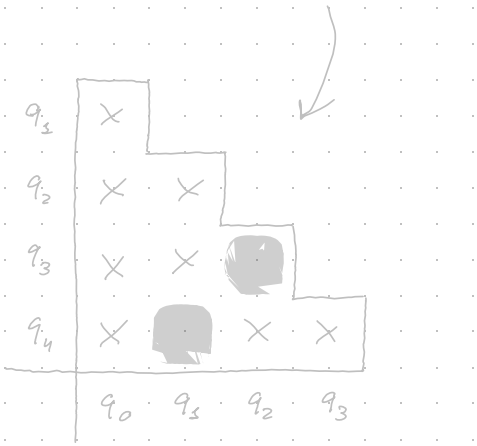
$A = \langle \{q_0, q_1, q_2, q_3, q_4\}, \{a,b\}, \delta, q_0, \{q_2, q_3\} \rangle$ (ver δ na ficha)

(*) Codificação:

	a	b
→		
-		







∴ q_1 é compatível com q_4
q_2 é compatível com q_3

Lema da repetição

Útil para provar que uma linguagem não é regular.

⚠ Não necessariamente prova que a linguagem é regular caso falso.

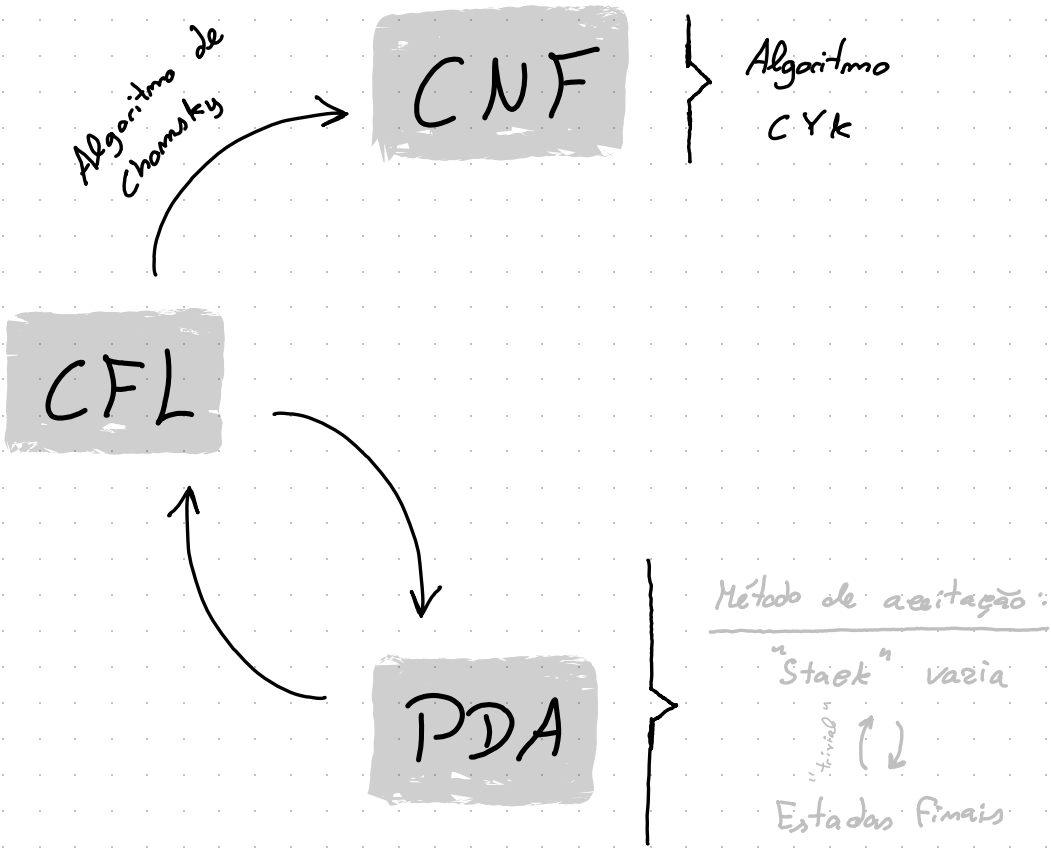
⚠ O valor de n não pode ser concretizado. O lema tem de ser verdadeiro para todo n.

⚠ As subpalavras u, v e w não podem ser concretizadas. O lema tem ele ser verdadeiro para todas elas.

1. Assumir que a linguagem L é regular
2. Escolher uma palavra x
   ↳ $x \in L$ ↳ $|x| \geq n$
3. Dividir x em 3 subpalavras: u, v e w
   ↳ $x = uvw$ ↳ $|uv| \leq n$ ↳ $|v| \geq 1$
4. Escolher um valor de i para que $uv^iw \notin L$.
5. Concluir que L não satisfaz o lema da repetição e portanto que não é regular.

2. Linguagens Independentes de Contexto

- Lema da repetição



Autómatos:

CFL | Linguagem fora de contexto
PDA | Autómato de pilha
CNF | Chomsky Normal Form

CFL → PDA

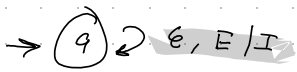
O autómato de pilha só terá um único estado e aceitará por "stack" vazia.
Todas as transições são laços para o mesmo estado.
Cada produção do CFG corresponde a um laço:

- **Terminais**

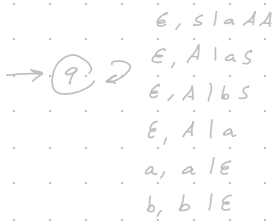
Ex:
Terminal "a"

- **Variáveis**

Ex:
E → I



Ex:
S → a AA
A → a S | b S | a



PDA → CFL

Ex. Ficha 9 : Exercício 4. d)
Pág. 23 e 22 de Exercícios .pdf

1. Criar uma lista de todas as possíveis variáveis.
Cada variável tem a codificação $[pXq]$, em que $p, q \in Q$ e $X \in \Gamma$, e representa a transição de p para q, eliminando X ("pop").

Ex II
$Q = \{p, s\}$
$\Gamma = \{z, 0, 1\}$

$V = \{ [pzp], [pzs], [szs], [szp], [p0p], [p0s], [s0s], [s0p], [p1p], [p1s], [s1s], [s1p] \}$

2. Escrever as produções iniciais : $S \to [q_0 z_0 p]$, em que q_0 é o estado inicial e z_0 é o símbolo inicial da pilha.

Ex II
$Q = \{p, s\}$
$q_0 = p$
$z_0 = z$

$S \to [pzp] \mid [pzs]$

3. Escrever as produções para cada transição:

$\delta(q, a, X) = \{(r, y_1 y_2 \dots y_k), \dots\}$

$[q X r_k] \to a\, [r\, y_1\, r_1]\, [r_1\, y_2\, r_2] \dots [r_{k-1}\, y_k\, r_k]$

- k : Nº de símbolos no topo da "stack"
- r_1, r_2, r_k : Combinação de todos os estados

Ex II $Q = \{p, s\}$

$k = 2$

r_1	$r_2 = r_k$
p	p
s	s
p	s
s	p

- **Se $k = 0$ ou $y_1 \dots y_k = \epsilon$**
  - Se a codificação dos estados corresponde à transição, isto é, $r_k = r$:
    $[q X r] \to a$
  - Caso contrário : $r_k \neq r$
    $[q X r_k] \to \emptyset$

Ex II

$\delta(p, 0, z) = \{(p, 0z)\}$ ($k = 2$)

$[pzp] \to 0\, [p0p]\, [pzp]$
$[pzs] \to 0\, [p0s]\, [szs]$
$[pzp] \to 0\, [p0s]\, [szp]$
$[pzs] \to 0\, [p0p]\, [pzs]$

$Q = \{p, s\}$

r_1	$r_2 = r_k$
p	p
s	s
s	p
p	s

Ex II

$\delta(p, 0, 1) = \{(p, \epsilon)\}$ ($k = 0$)

$[p1p] \to 0$
$[p1s] \to \emptyset$

$r_0 = r_k$
p
s

4. Organizar produções e eliminar produções que tenham variáveis que
- levam a "loops" infinitos
- Não tenham nenhuma produção

Ex II
$[p0s] \to 0\, [p0s]$
$[szp] \to \emptyset$

5. (opcional) Renomear variáveis para melhor legibilidade.

Algoritmo de Chomsky

CFL
↳ CNF

CNF : Chomsky Normal Form

Um CFG sem transições ε, em que todas as produções são da forma:

- $A \to BC$
- $A \to a$

1. Eliminar produções ε

Ex. $S \to AB$, $A \to aAA \mid \varepsilon$ ⟹ $S \to AB \mid B$, $A \to aAA \mid aA \mid a$

2. Eliminar produções unitárias $(A \to B)$

Ex. Eliminação por expansão:

$S \to S1 \mid A$, $A \to 12 \mid B$, $B \to 0$

Casos base	$(S \to S)$	$S \to S1$
	$(A \to A)$	$A \to 12$
	$(B \to B)$	$B \to 0$
Casos indutivos	$(S \to A)$	$S \to 12$
	$(A \to B)$	$A \to 0$
	$(S \to B)$	$S \to 0$

⟹ $S \to S1 \mid 12 \mid 0$, $A \to 12 \mid 0$, $B \to 0$

3. Eliminar produções inúteis

a) Eliminar produções não geradoras **b)** Eliminar produções não alcançáveis

Ex. $S \to AB \mid a$, $A \to b$ ⟹ $S \to a$, $A \to b$ ⟹ $S \to a$

4. Substituir terminais por variáveis

Ex. $S \to Aa \mid a$, $A \to aS$ ⟹ $S \to AX \mid a$, $A \to XS$, $X \to a$

5. Encurtar produções a duas variáveis ("Long Bodies")

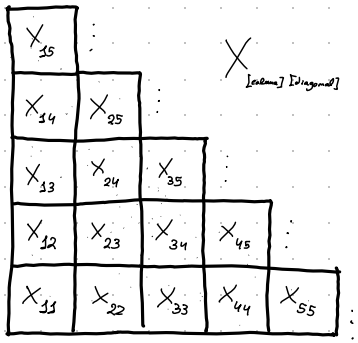
Ex. $S \to zzzU \mid Uzz$ ⟹ $S \to zC_1 \mid UC_2$, $C_1 \to zzU$, $C_2 \to zz$

Algoritmo CYK

1. Criar uma tabela triangular e quadrada do tamanho do "input", em que cada coluna corresponde a um símbolo.

2. Preencher a 1ª linha a partir do input: Para cada elemento, escrever todas as produções que incluem o símbolo correspondente.

3. Preencher as restantes linhas:
   - → 2ª linha | $X_{12} = X_{11} \times X_{22}$
   - → 3ª linha | $X_{13} = X_{11} \times X_{23} \cup X_{12} \times X_{33}$
   - → 4ª linha | $X_{14} = X_{11} \times X_{24} \cup X_{12} \times X_{34} \cup X_{13} \times X_{44}$
   - ...



4. A palavra foi aceite se o último elemento contiver a variável inicial da CNF

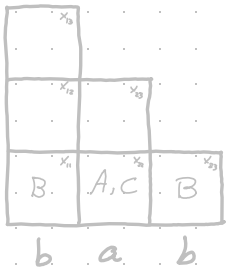
Ex// "Input" / Entrada: b a b

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

1ª linha

b: Produção B: (B → b)

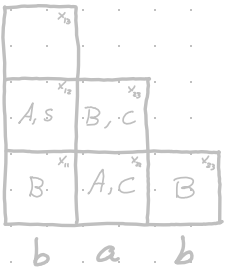
a: Produção A: (A → a); Produção C: (C → a)



2ª linha

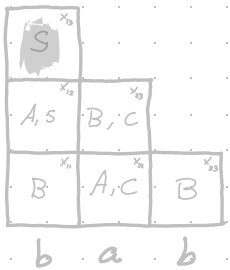
$X_{12} = X_{11} \times X_{23}$
$= \{B\} \times \{A, C\}$
$= \{BA, BC\} \rightarrow \{A, S\}$

$X_{23} = X_{22} \times X_{33}$
$= \{A, C\} \times \{B\}$
$= \{AB, CB\} \rightarrow \{S, C\}$



3ª linha

$X_{13} = X_{11} \times X_{23} \cup X_{12} \times X_{33}$
$= \{B\} \times \{B, C\} \cup \{A, S\} \times \{B\}$
$= \{BB, BC, AB, SB\}$
$\hookrightarrow \{S\}$



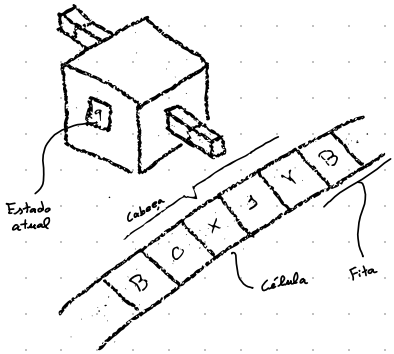
$X_{33} \in \{S\} \rightarrow$ Palavra aceite //

Linguagens

3. Recursivamente Enumeráveis

TM

1 Máquina de Turing



$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

- **Q** — Estados de controlo
- **Σ** — Alfabeto de entrada
- **Γ** — Alfabeto da fita (Inclui todos os símbolos de entrada)
- **δ** — Função de transição: $\delta(Q, X) = (P, Y, D)$
  - **q** — Estado atual
  - **X** — Símbolo atual da fita
  - **P** — Estado a transitar
  - **Y** — Símbolo a substituir
  - **D** — Direção (← ou →)
- **q_0** — Estado inicial
- **B** — Símbolo branco da fita
- **F** — Estados finais