PCS 2215

Fundamentos de Engenharia de Computação II

Linguagens, Gramáticas e Autômatos

Anna Helena Reali Costa

Professora Responsável

versão: 1.6 (outubro 2002)

Conteúdo

1. Circuitos Seqüenciais e Máquinas de Estado Finito.
2. Autômatos Finitos.
3. Linguagens e Gramáticas.
4. Autômatos Finitos não Determinísticos.
5. Relação entre Linguagens e Autômatos.
6. Autômatos de Pilha.
7. Máquinas de Turing.
8. Linguagens Formais e Dispositivos Computacionais.

1. Circuitos Seqüenciais e Máquinas de Estado Finito

- Circuitos seqüenciais:
  - são aqueles nos quais sua saída depende não somente das entradas, mas também de seu estado no instante em que as entradas foram introduzidas (têm memória).
- Máquina de estado finito:
  - é um modelo abstrato de uma máquina que possua uma memória primitiva.

1. Circuitos Sequenciais e Máquinas de Estado Finito

- ***Definição***: Uma máquina de estado finito $M = (I, O, S, f, g, \sigma)$ consiste de:
  - Um conjunto finito I de símbolos de entrada
  - Um conjunto finito O de símbolos de saída
  - Um conjunto finito S de estados
  - Uma função próximo estado $f: S \times I \rightarrow S$
  - Uma função de saída $g: S \times I \rightarrow O$
  - Um estado inicial $\sigma \in S$

1. Circuitos Sequenciais e Máquinas de Estado Finito

- ***Ex 10.1.5*** Seja I={a,b}, O={0,1}, S={σ_0, σ_1} e $\sigma = \sigma_0$. As funções f e g são definidas pelas regras dadas na tabela:

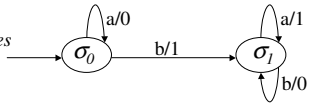
	f		g	
S \ I	a	b	a	b
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_1	1	0

$f(\sigma_0,a)= \sigma_0 \quad g(\sigma_0,a)= 0$
$f(\sigma_0,b)= \sigma_1 \quad g(\sigma_0,b)= 1$
$f(\sigma_1,a)= \sigma_1 \quad g(\sigma_1,a)= 1$
$f(\sigma_1,b)= \sigma_1 \quad g(\sigma_1,b)= 0$

1. Circuitos Sequenciais e Máquinas de Estado Finito

- ***Definição:*** Seja M = (I, O, S, f, g,σ) uma máquina de estado finito. O **diagrama de transições** de M é um digrafo G cujos vértices são membros de S. Uma seta indica o estado inicial σ. Uma aresta orientada (σ_1,σ_2) existe em G se existir uma entrada i com $f(\sigma_1,i)=\sigma_2$. Neste caso, se $g(\sigma_1,i)= 0$, a aresta (σ_1,σ_2) é rotulada com *i/0*.

Diagrama de transições para o ex. 10.1.5:



1. Circuitos Sequenciais e Máquinas de Estado Finito

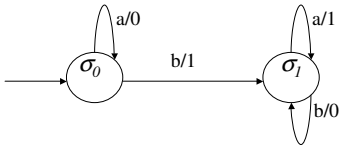
- ***Definição:*** Seja M = (I, O, S, f, g,σ) uma máquina de estado finito. Uma cadeia de entrada para M é uma cadeia sobre I. A cadeia $y_1 \ldots y_n$ é a cadeia de saída de M correspondendo à cadeia de entrada $x_1 \ldots x_n$, caso existam os estados $\sigma_0, \sigma_1 \ldots, \sigma_n \in S$ com:

$\sigma_0 = \sigma$, $\sigma_i = f(\sigma_{i-1}, x_i)$, $y_i = g(\sigma_{i-1}, x_i)$ para i=1,…,n;

Pode-se pensar em M = (I, O, S, f, g,σ) como um computador simples: começa no estado σ, consome uma cadeia de caracteres sobre I e produz uma cadeia de saída.

1. Circuitos Sequenciais e Máquinas de Estado Finito

- ***Exemplo***: Encontre a cadeia de saída correspondente à cadeia de entrada *aababba* para a máquina de estado finito abaixo:



1. Circuitos Sequenciais e Máquinas de Estado Finito

- Exercício:
  - projetar uma máquina de estado finito que forneça 1 como saída caso um número par de 1's seja fornecido numa cadeia de bits de entrada e que forneça 0, no caso contrário.

2. Autômato Finito

- ***Definição*** **:**

Um **autômato finito** Af = (*I, O, S, f, g*,σ) é uma máquina de estado finito onde:
  - o conjunto de símbolos de saída é {0,1} e
  - o estado corrente determina a última saída.

- Aqueles estados para os quais a última saída é 1 são chamados **estados de aceitação.**
- *Diagramas de transições de um AF: os estados de aceitação são representados por círculos duplos e os símbolos de saída são omitidos.*

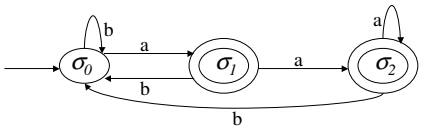
2. Autômato Finito

- ***Exemplo***: Desenhe o diagrama de transições da máquina de estado finito *At* definida abaixo. O estado inicial é σ_0. Mostre que *At* é um autômato finito e determine o conjunto de estados de aceitação.

	f		*g*	
S \ *I*	**a**	**b**	**a**	**b**
σ_0	σ_1	σ_0	1	0
σ_1	σ_2	σ_0	1	0
σ_2	σ_2	σ_0	1	0

2. Autômato Finito

Resposta do exemplo: A máquina de estado finito *At* é um AF uma vez que seu conjunto de símbolos de saída é {0,1} e, para cada estado σ, todas as arestas que chegam em σ têm o mesmo rótulo de saída.



2. Autômato Finito

- *Definição alternativa de um AF:*

Um **autômato finito** Af é definido por:

- Um conjunto finito *I* de símbolos de entrada
- Um conjunto finito *S* de estados
- Uma função próximo estado *f: S* x *I* $\rightarrow S$
- Um subconjunto *A* de *S* de estados de aceitação
- Um estado inicial $\sigma \in S$

Af = (I, S, f, A, σ)

2. Autômato Finito

- ***Definição:*** Seja Af = (I, S, f, A, σ) um autômato finito. Seja $\alpha = x_1 \ldots x_n$ uma cadeia sobre *I*. Caso existam os estados $\sigma_0, \ldots, \sigma_n$ satisfazendo:
  - *(a)* $\sigma_0 = \sigma$
  - *(b)* $\sigma_i = f(\sigma_{i-1}, x_i)$ para i=1,…,n;
  - *(c)* $\sigma_n \in A$

  dizemos que α é ***aceita*** por Af.

A cadeia vazia (ou nula) λ é aceita por Af se e somente se $\sigma \in$ A.

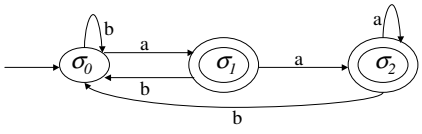
2. Autômato Finito

- Seja Ac(Af) o conjunto de cadeias aceito por Af. Dizemos que Af aceita Ac(Af).
- Seja $\alpha = x_1 \ldots x_n$ uma cadeia sobre *I*. Os estados $\sigma_0, \ldots, \sigma_n$ são definidos pelas condições (a) e (b) da definição anterior. O caminho ($\sigma_0, .., \sigma_n$) é um ***caminho representando*** α **em** ***Af*****.**
  - *Se o caminho P representa a cadeia* α *em Af, então Af aceita* α *se e somente se P terminar em um estado de aceitação.*

2. Autômato Finito

Exercícios:

1. A cadeia *abaa* é aceita pelo AF abaixo?
2. A cadeia *abab* é aceita pelo AF abaixo?



2. Autômato Finito

- ***Respostas***:
- 1. O caminho P = $(\sigma_0, \sigma_1, \sigma_0, \sigma_1, \sigma_2)$ representa a cadeia *abaa*. Como o estado final σ_2 é um estado de aceitação, a cadeia é aceita pelo AF dado.
- 2. O caminho P = $(\sigma_0, \sigma_1, \sigma_0, \sigma_1, \sigma_0)$ representa a cadeia *abab*. Como o estado final σ_0 não é um estado de aceitação, a cadeia não é aceita pelo AF dado.

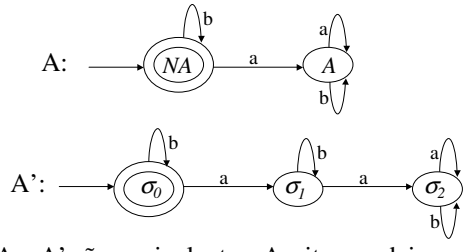
2. Autômato Finito

- ***Definição:***

Os autômatos finitos A e A' são equivalentes se Ac(A) = Ac(A').

- Se definirmos a relação R num conjunto de AF's pela regra ARA', se A e A' forem equivalentes, R é uma relação de equivalência. Cada classe de equivalência consiste de um conjunto de AF's mutuamente equivalentes.

2. Autômato Finito



A e A' são equivalentes. Aceitam cadeias sobre {a,b} que não possuem a's.

3. Linguagens e Gramáticas

- ***Definição:*** Seja **A** um conjunto finito de símbolos. Uma linguagem (formal) **L** sobre **A** é um subconjunto de **A***.
  - *Ex*: Dado A={*a,b*}. O conjunto L de todas as cadeias sobre A que contêm um número ímpar de *a*'s é uma linguagem sobre A.

3. Linguagens e Gramáticas

- Como podemos descrever uma linguagem L, isto é, especificar exatamente quais cadeias pertencem a L?
  - Se L for finita, pode-se simplesmente enumerar seus elementos;
  - Pode-se descrever um algoritmo para determinar (*reconhecer*) se cada cadeia pertence a L;
  - Pode-se descrever um procedimento que permita *gerar (produzir)* apenas elementos de L, através do estabelecimento de uma *gramática*.

3. Linguagens e Gramáticas

- ***Definição:***

  Uma **gramática** $G = (N, T, P, \sigma)$ consiste de:
  - um conjunto finito N de símbolos não terminais;
  - um conjunto finito T de símbolos terminais, onde $N \cap T = \varnothing$;
  - um subconjunto P de $[(N \cup T)^* - T^*] \times (N \cup T)^*$, chamado conjunto de ***produções;***
  - um símbolo inicial $\sigma \in N$.

3. Linguagens e Gramáticas

- Uma produção $(A,B) \in P$ é escrita $A \rightarrow B$, onde $A \in [(N \cup T)^* - T^*]$ e $B \in (N \cup T)^*$.
  - Assim, A deve conter pelo menos um símbolo não terminal e B pode conter qualquer combinação de símbolos terminais e não terminais.

3. Linguagens e Gramáticas

- *Exemplos*:

  Seja a gramática $G = (N, T, P, \sigma)$, com:

  a) $N=\{\sigma\}$, $T=\{a,b\}$, $P=\{\sigma \rightarrow \sigma a,\ \sigma \rightarrow b\}$

  b) $N=\{\sigma,S\}$, $T=\{0,1\}$, $P=\{\sigma \rightarrow \lambda, \sigma \rightarrow S, S \rightarrow 1S, S \rightarrow 0S, S \rightarrow 1, S \rightarrow 0\}$

3. Linguagens e Gramáticas

- ***Definição:***

  Seja a gramática G = (N, T, P, σ). Se $\alpha \rightarrow \beta$ é uma produção e $x\alpha y \in (N \cup T)^*$, dizemos que $x\beta y$ é **diretamente derivável** de $x\alpha y$ e escrevemos: $x\alpha y \Rightarrow x\beta y$.

  Se $\alpha_i \in (N \cup T)^*$ para $i=1,..,n$ e α_{i+1} é diretamente derivável de α_i para $i=1,..,n-1$, dizemos que **α_n é derivável de α_1**: $\alpha_1 \Rightarrow \alpha_n$

3. Linguagens e Gramáticas

- Derivação de α_n (a partir de α_1):

  $\alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow \alpha_n$

  Por convenção, qualquer elemento de $(N \cup T)^*$ é derivável de si mesmo.

- A **linguagem L(G) gerada por G** consiste de todas as cadeias sobre T derivadas de σ.

3. Linguagens e Gramáticas

- *ex.*: Seja a gramática G = (N, T, P, σ), com N={σ, S}, T={a,b},

  P={$\sigma \rightarrow b\sigma$, $\sigma \rightarrow aS$, $S \rightarrow bS$, $S \rightarrow b$}

(a) A cadeia abSbb é diretamente derivável de aSbb, escrita como aSbb ⇒ abSbb, usando a produção $S \rightarrow bS$.

(b) A cadeia bbab é derivável de σ, escrita $\sigma \Rightarrow bbab$. A derivação é:
$\sigma \Rightarrow b\sigma \Rightarrow bb\sigma \Rightarrow bbaS \Rightarrow bbab$

3. Linguagens e Gramáticas

- **BNF** ("Backus normal form" ou "Backus-Naur form"): modo alternativo de descrever uma gramática G.
  - símbolos não terminais: inclusos em <..>
  - produção $S \rightarrow T$, escrita: S::=T
  - produções da forma: $S::=T_1$, $S::=T_2$, …, $S::=T_n$ podem ser combinadas em $S::= T_1 | T_2 | \ldots | T_n$. *(lê-se "ou" para "|")*

3. Linguagens e Gramáticas

- Ex.: uma gramática para inteiros - um inteiro é definido como uma cadeia contendo um sinal opcional (+ ou -), seguido por uma cadeia de dígitos (0 a 9).

Símbolo inicial: <inteiro>

<digito>::= 0 |1| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<inteiro>::= <inteiro com sinal> | <inteiro sem sinal>

<inteiro com sinal>::= + < inteiro sem sinal> |
- <inteiro sem sinal>

<inteiro sem sinal>::= <digito> |
<digito> <inteiro sem sinal>

3. Linguagens e Gramáticas

ex.: derivação do inteiro -901

<inteiro> ⇒ <inteiro com sinal>
⇒ - <inteiro sem sinal>
⇒ - <digito> <inteiro sem sinal>
⇒ - <digito> <digito> <inteiro sem sinal>
⇒ - <digito> <digito> <digito>
⇒ - 9 <digito> <digito>
⇒ - 90 <digito>
⇒ - 901.

3. Linguagens e Gramáticas

- Notação equivalente: G = (N, T, P, σ)

N={<digito>,<inteiro>,<inteiro com sinal>,<inteiro sem sinal>}, T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -}

P = {<digito>→ 0, <digito>→ 1, …., <digito>→ 9,
<inteiro>→<inteiro com sinal>,
<inteiro>→<inteiro sem sinal>,
<inteiro com sinal>→ + <inteiro sem sinal>,
<inteiro com sinal>→ - <inteiro sem sinal>,
<inteiro sem sinal>→ <digito>,
<inteiro sem sinal>→ <digito> <inteiro sem sinal>}

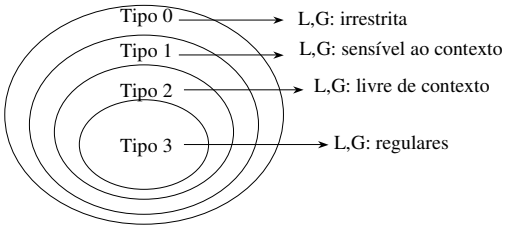
Símbolo inicial σ : <inteiro>

3. Linguagens e Gramáticas

- Conforme as restrições impostas ao formato das produções de uma gramática, varia-se corrrepondentemente a classe de linguagens que tal gramática gera.
- A teoria mostra que há quatro classes de gramáticas, capazes de gerar quatro classes correspondentes de linguagens, de acordo com a denominada ***hierarquia de Chomsky.***

3. Linguagens e Gramáticas

- Hierarquia de Chomsky: estabelece uma relação de inclusão entre as gramáticas.



3. Linguagens e Gramáticas

- **Convenção da produção nula**

Se uma gramática permitir produção da cadeia nula, ela deverá ser da forma $\sigma \rightarrow \lambda$, onde σ é o símbolo inicial e não pertence ao lado direito de qualquer produção e λ é a cadeia nula. Assim, pode-se tratar esta produção como um caso especial.

3. Linguagens e Gramáticas

- ***Definição (inclui itens a,b,c e d):***
  Seja G uma gramática e λ a cadeia nula.

(a) Se toda produção estiver na forma $A \rightarrow a$ ou $A \rightarrow aB$ ou $A \rightarrow \lambda$, com $A,B \in N$, $a \in T$, G é uma **gramática regular** (ou **tipo 3**).

  – Nesta gramática, pode-se substituir um símbolo não terminal por: (i) um símbolo terminal, (ii) um símbolo terminal seguido por um não terminal ou (iii) pela cadeia nula.

3. Linguagens e Gramáticas

- *Exemplo*: a gramática G = (N, T, P, σ), com T={a,b}, N={σ, S}, P={$\sigma \rightarrow b\sigma$, $\sigma \rightarrow aS$, $S \rightarrow bS$, $S \rightarrow b$} é **regular.**
- Derivações possíveis: ab, abb, bbbabb,…
- $L(G)=\{b^n a b^m \mid n \geq 0, m \geq 1\}$ $\Rightarrow$ G é uma gramática regular, portanto, a linguagem L(G) que ela gera é uma linguagem regular!

  *Linguagens regulares permitem concatenação de símbolos, união e * (fechamento transitivo e recursivo) de conjuntos de símbolos.*

3. Linguagens e Gramáticas

(b) Se toda produção estiver na forma $A \rightarrow \delta$, com $A \in N$, $\delta \in (N \cup T)^*$, G é uma **gramática livre de contexto** (ou **tipo 2**).

- Nesta gramática, pode-se substituir A (um não terminal isolado) por δ sempre que se queira, independentemente do contexto em que A esteja inserido.

3. Linguagens e Gramáticas

- *Exemplo*: a gramática G = (N, T, P, σ), com T={a,b}, N={σ}, P={σ→aσb, σ→ab} é **livre de contexto.**
- Derivações possíveis: ab, aabb, aaabbb,…
- $L(G)=\{a^n b^n \mid n=1,2,\ldots\}$ ⇒ L é uma linguagem livre de contexto e não é uma linguagem regular!

Linguagens livres de contexto permitem, além das operações permitidas para a linguagem regular, operações de aninhamento.

3. Linguagens e Gramáticas

(c) Se toda produção estiver na forma $\alpha A \beta \rightarrow \alpha \delta \beta$, com $\alpha, \beta \in (N \cup T)^*$, $A \in N$, $\delta \in (N \cup T)^* - \{\lambda\}$, G é uma **gramática sensível ao contexto** (ou **tipo 1**).

- Nesta gramática, pode-se substituir A por δ se A estiver dentro do contexto α e β.
- Na gramática do tipo 1, $|\alpha A \beta| \leq |\alpha \delta \beta|$, exceto para a produção $\sigma \rightarrow \lambda$, com σ sendo o símbolo inicial e λ, a cadeia nula.

3. Linguagens e Gramáticas

- *Ex.*: a gramática G = (N, T, P, σ), com T={a,b,c}, N={σ,A,B,C,D,E}, P={σ→aAB, σ→aB, A→aAC, A→aC, B→Dc, D→b, CD→CE, CE→DE, DE→DC, Cc→Dcc} é **sensível ao contexto**
  (ex.: CE→DE diz que C pode ser substituído por D caso C seja seguido por E)
- Derivações possíveis: abc, aabbcc, aaabbbccc,…
- $L(G)=\{a^n b^n c^n \mid n=1,2,\ldots\}$ ⇒ não existe uma gramática livre de contexto G com L=L(G); assim, L não é uma linguagem livre de contexto!

Ex: declarações e uso de variáveis em programas fazem parte de linguagens sensíveis ao contexto.

3. Linguagens e Gramáticas

(d) Se toda produção de G estiver na forma $\alpha \rightarrow \beta$, com $\alpha \in [(N \cup T)^* - T^*]$ e $\beta \in (N \cup T)^*$, G é uma **gramática irrestrita** (ou **tipo 0**).

- Nesta gramática, nenhuma limitação é imposta.

3. Linguagens e Gramáticas

- *Exemplo* : a gramática G = (N, T, P, σ), com T={a,b}, N={σ,B,C},
  P={σ→BC, BC→CB, B→b, C→a} é **irrestrita.**
  - Esta produção somente é permitida em gramáticas irrestritas: BC→CB

L(G)={ab, ba}

3. Linguagens e Gramáticas

- Uma gramática regular é uma gramática livre de contexto.
- Uma gramática livre de contexto, sem produções do tipo $A \rightarrow \lambda$, é uma gramática sensível ao contexto.
- Uma gramática sensível ao contexto é uma gramática irrestrita.

3. Linguagens e Gramáticas

- ***Definição:***

  Uma linguagem L é sensível ao contexto (respectivamente livre de contexto, regular) se existe uma gramática sensível ao contexto G (respectivamente livre de contexto, regular) com L=L(G).

3. Linguagens e Gramáticas

- A gramática para inteiros dada (slide 29) é livre de contexto. Se mudarmos as produções para:

<digitos>::= 0<digitos> |1<digitos> |... | 9<digitos> | λ

<inteiro>::= + <inteiro sem sinal> | - <inteiro sem sinal> |
0<digitos> |1<digitos> |... | 9 <digitos>

<inteiro sem sinal>::= 0<digitos> |1<digitos> |... | 9<digitos>

resultará numa gramática G regular. Como a linguagem L=L(G) gerada não foi modificada, concluímos que L é uma linguagem regular.

3. Linguagens e Gramáticas

- ***Definição***:

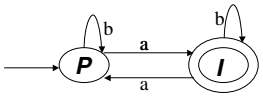
As gramáticas G e G' são equivalentes se L(G) = L(G').

4. Autômatos Finitos Não Determinísticos

- Nesta seção mostraremos que gramáticas regulares e autômatos finitos são essencialmente equivalentes, no sentido em que ambos são especificações de uma linguagem regular (a gramática, como *geradora* da linguagem e o autômato, como *reconhecedor* da linguagem).

4. Autômatos Finitos Não Determinísticos

- Seja o autômato finito AF abaixo, o qual aceita cadeias sobre {a,b} que contêm um número ímpar de a's.



- Determinar a gramática regular equivalente.

4. Autômatos Finitos Não Determinísticos

- **AF ⇒ G:** os símbolos de entrada {a,b} do AF são os símbolos terminais de G. Os estados **P** e **I** são os símbolos não terminais. O estado inicial **P** é o símbolo inicial. Os arcos do AF correspondem às produções de G. Se existir um arco rotulado por *x* de S para S', escreve-se a produção: S→*x*S'.

No exemplo, temos: $\boldsymbol{P} \rightarrow b\boldsymbol{P}$, $\boldsymbol{P} \rightarrow a\boldsymbol{I}$, $\boldsymbol{I} \rightarrow a\boldsymbol{P}$, $\boldsymbol{I} \rightarrow b\boldsymbol{I}$

- Além disso, se S for um estado de aceitação, inclui-se: S→λ. *No exemplo:* $\boldsymbol{I} \rightarrow \lambda$.

4. Autômatos Finitos Não Determinísticos

- Assim, a gramática G=(N,T,P,**P**), com N={**I**,**P**}, T={a,b} e P={$\boldsymbol{P} \rightarrow b\boldsymbol{P}$, $\boldsymbol{P} \rightarrow a\boldsymbol{I}$, $\boldsymbol{I} \rightarrow a\boldsymbol{P}$, $\boldsymbol{I} \rightarrow b\boldsymbol{I}$, $\boldsymbol{I} \rightarrow \lambda$} gera a linguagem L(G), a qual é a mesma que o conjunto de cadeias aceitas pelo autômato finito AF.

4. Autômatos Finitos Não Determinísticos

- **Teorema**:

  Seja A um autômato finito dado por um diagrama de transições. Seja σ o estado inicial. Seja T o conjunto dos símbolos de entrada e N, o conjunto de estados. Definir produções S→*x*S' se existir um arco rotulado *x* de S para S' e S→λ se S for um estado de aceitação. Seja a gramática regular G=(N,T,P,σ). Desta forma, o conjunto de cadeias aceitas por A é igual a L(G).

4. Autômatos Finitos Não Determinísticos

- Seja a gramática regular G=(N,T,P,σ), com N={σ, C}, T={a,b}, P={σ→bσ, σ→aC, C→bC, C→b}. Determinar o AF correspondente.
- **G ⇒ AF:** Os símbolos não terminais serão os estados. Para cada produção da forma S→*x*S', desenhar uma aresta de S a S', com rótulo *x* (produções σ→bσ, σ→aC, C→bC). A produção C→b equivale a: C→bF, F→λ, sendo F um símbolo não terminal adicional. A produção F→λ indica que F é um estado de aceitação.

4. Autômatos Finitos Não Determinísticos

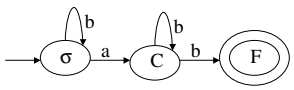


Diagrama de transições do autômato resultante.

- Este autômato é um **autômato finito não determinístico** (estando no estado C, caso receba b como entrada, o próximo estado não está determinado: pode ser C ou F)

4. Autômatos Finitos Não Determinísticos

- ***Definição :***

Um autômato finito não determinístico At=(I,S,f,A,σ) consite de:

(a) um conjunto finito I de símbolos de entrada.

(b) um conjunto finito S de estados.

(c) uma função próximo estado f: SxI $\rightarrow$ P(S).

(d) um subconjunto A de S de estados de aceitação.

(e) um estado inicial σ.

4. Autômatos Finitos Não Determinísticos

- **Exemplo 10.4.6**: desenhar o diagrama de transições do AF não determinístico com I={a,b}, S={σ,C,D}, A={C,D}, estado inicial σ e função próximo estado dado pela tabela abaixo:

I / S	a	b
σ	{σ,C}	{D}
C	ϕ	{C}
D	{C,D}	ϕ

4. Autômatos Finitos Não Determinísticos

- ***Definição (itens a-e)***: Seja At=(I,S,f,A,σ) um autômato finito não determinístico.

a) A cadeia nula é aceita por At sse $\sigma \in A$.

b) Se $\alpha = x_1 \ldots x_n$ é uma cadeia não nula sobre I e os estados $\sigma_0, \ldots, \sigma_n$ satisfazem: *(a)* $\sigma_0 = \sigma$; *(b)* $\sigma_i \in f(\sigma_{i-1}, x_i)$ com i = 1,.., n; *(c)* $\sigma_n \in A$, diz-se que α é aceita por At.

c) Denota-se Ac(At) ao conjunto de cadeias aceitas por At e diz-se que At aceita Ac(At).

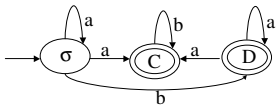
4. Autômatos Finitos Não Determinísticos

d) Se At e At' são autômatos finitos não determinísticos e Ac(At)=Ac(At'), então At e At' são ***equivalentes***.

e) Se $\alpha = x_1 \ldots x_n$ é uma cadeia sobre I e existem os estados $\sigma_0, \ldots, \sigma_n$ satisfazendo *(a)* e *(b) da Def.10.4.7*, o caminho $(\sigma_0, \ldots, \sigma_n)$ é chamado de ***caminho que representa α em At***.

4. Autômatos Finitos Não Determinísticos

- *Exercício*: a cadeia α = aabaabbb é aceita pelo autômato finito não determinístico abaixo? Se sim, localizar o caminho que representa α e o estado de aceitação.



4. Autômatos Finitos Não Determinísticos

- **Teorema:**

  Seja G = (N,T,P,σ) uma gramática regular.

  Seja I = T, S = N ∪ {F}, onde F∉ N∪T,
  f(S,x) = {S' | S→xS' ∈ P} ∪ {F | S →x ∈ P},
  A = {F} ∪ {S | S→λ ∈ P}.

  Desta forma, o autômato finito não determinístico At = (I, S, f, A, σ) aceita precisamente cadeias de L(G).

5. Relação entre linguagens e autômatos

- Na seção anterior vimos que, se A é um autômato finito, existe uma gramática regular G, com L(G)=Ac(A). Vimos também que, se G é uma gramática regular, existe um autômato finito não determinístico At, com L(G)=Ac(At).
- Nesta seção veremos que, se G é uma gramática regular, existe um autômato finito A, com L(G)=Ac(A). Isso será resultado da demonstração de que qualquer autômato finito não determinístico pode ser convertido num autômato finito equivalente.

5. Relação entre linguagens e autômatos

- **Teorema**: Seja um autômato finito não determinístico At=(I,S,f,A,σ). Seja:

(a) S'= P(S). (b) I'= I. (c) σ'= {σ}.

(d) A'= $\{X \subseteq S', \exists\, y \in X \mid y \cap A \neq \phi\}$.

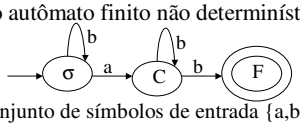
(e) Para $X \in S'$ e $x \in I$,

$$f'(X,x) = \begin{cases} \phi \text{ se } X=\phi \\ \bigcup_{S \in X} f(S,x) \text{ se } X \neq \phi. \end{cases}$$

Então o autômato finito At'= (I',S',f',A',σ') é equivalente a At.

5. Relação entre linguagens e autômatos

- ***Exemplo***: Encontrar o autômato finito equivalente ao autômato finito não determinístico dado abaixo:

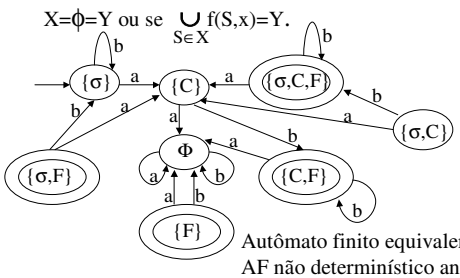


O conjunto de símbolos de entrada {a,b} não muda.

Os estados consistem de todos os subconjuntos do conjunto original S = {σ,C,F}: ϕ, {σ}, {C}, {F}, {σ,C}, {σ,F}, {C,F}, {σ,C,F}. O estado inicial é {σ}. Os estados de aceitação são todos os subconjuntos de S que contenham um estado de aceitação do AF não determinístico original: {F}, {σ,F}, {C,F}, {σ,C,F}.

5. Relação entre linguagens e autômatos

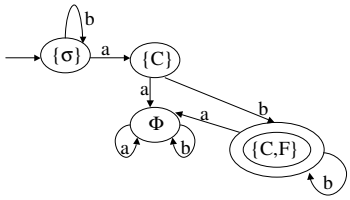
Uma aresta é desenhada de X para Y e rotulada de x se $X=\phi=Y$ ou se $\bigcup_{S \in X} f(S,x)=Y$.



Autômato finito equivalente ao AF não determinístico anterior.

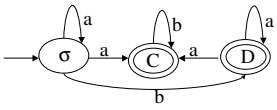
5. Relação entre linguagens e autômatos

Os estados {σ,C}, {σ,F}, {F}, {σ,C,F} não são nunca alcançados; portanto, pode-se apagá-los, resultando no autômato finito simplificado equivalente ao anterior:



5. Relação entre linguagens e autômatos

Exercício: Transforme o autômato finito não determinístico abaixo no seu equivalente autômato determinístico:



5. Relação entre linguagens e autômatos

- **Teorema:**

  Uma linguagem L é regular se e somente se existir um autômato finito que aceite precisamente as cadeias em L.

6. Autômatos de Pilha

Um **autômato de pilha (ou pushdown)**

Ap =(I, S, Γ, f, q_0, Z_0, A) é definido por:

- Um conjunto finito *I* de símbolos de entrada
- Um conjunto finito *S* de estados
- Um alfabeto Γ da pilha
- Um estado inicial $q_0 \in S$
- Um símbolo inicial na pilha $Z_0 \in \Gamma$
- Uma função f de transição de estado

  $f: S \times (I \cup \{\lambda\}) \times \Gamma \rightarrow P(S \times \Gamma^*)$
- Um conjunto $A \subseteq S$ de estados finais (ou de aceitação).

6. Autômatos de Pilha

A função de transição f do Ap:

$S \times (I \cup \{\lambda\}) \times \Gamma \rightarrow P(S \times \Gamma^*)$

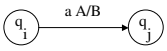
- S: estado
- $(I \cup \{\lambda\})$: símbolo lido na fita
- Γ: símbolo no topo da pilha
- $P(S \times \Gamma^*)$: possíveis configurações resultantes (estado, cadeia de símbolos colocados no topo da pilha)

Exemplo: $f(q_i, a, A) = \{[q_j, B], [q_k, C]\}$

Um AP admite transições não-determinísticas!

6. Autômatos de Pilha

- Representação por Diagrama de estados:



$f(q_i, a, A) = \{[q_j, B]\}$

Estado inicial q_i, símbolo a lido na fita, símbolo A no topo da pilha.

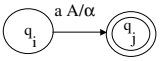
Estado resultante q_j, símbolo (**ou cadeia de símbolos**) B substitui A no topo da pilha.

6. Autômatos de Pilha

- Exemplos:

Nenhum símbolo lido, A no topo da pilha removido: $f(q_i, \lambda, A) = [q_i, \lambda]$

Nenhum símbolo lido, mantido A no topo da pilha: $f(q_i, \lambda, A) = [q_i, A]$



Símbolo **a** lido, símbolo **A** no topo da pilha substituído pela cadeia **α**: $f(q_i, a, A) = [q_j, \alpha]$

6. Autômatos de Pilha

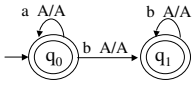
- ***Definição*:** Uma **configuração instantânea** de um AP é uma tripla $[q_i, w, \alpha]$, em que q_i é o estado, *w* é a cadeia não processada e α é o conteúdo da pilha (símbolo mais à esquerda no topo).
- ***Definição*:** A função $\vdash_M$ é definida por $[q_i, w, \alpha] \vdash_M [q_j, v, \beta]$ e indica que a configuração $[q_j, v, \beta]$ pode ser obtida a partir da configuração $[q_i, w, \alpha]$ por uma transição do AP *M*.

6. Autômatos de Pilha

- A notação $[q_i, w, \alpha] \vdash_M^* [q_j, v, \beta]$ indica que a configuração $[q_j, v, \beta]$ pode ser obtida a partir da configuração $[q_i, w, \alpha]$ por zero ou mais transições do AP *M*.
- Uma **computação** em um AP é uma seqüência de transições a partir do estado inicial com a pilha contendo apenas Z_0.

6. Configurações em APs: Exemplo 1

M: $S = \{q_0, q_1\}$
$I = \{a,b\}$
$\Gamma = \{A\}$
$Z_0 = \{A\}$
$A = \{q_0, q_1\}$

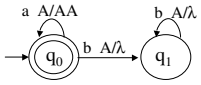


$[q_0, aabb, A] \vdash [q_0, abb, A] \vdash [q_0, bb, A] \vdash [q_1, b, A] \vdash [q_1, \lambda, A]$

Computação para a cadeia *aabb* terminou em um estado final → cadeia é **aceita (por estado final)** pelo AP.

6. Configurações em APs: Exemplo 2

M: $S = \{q_0, q_1\}$
$I = \{a,b\}$
$\Gamma = \{A\}$
$Z_0 = \{A\}$
$A = \{q_0\}$



$[q_0, abb, A] \vdash [q_0, bb, AA] \vdash [q_1, b, A] \vdash [q_1, \lambda, \lambda]$

Computação para a cadeia *abbb* terminou com a pilha vazia → cadeia é **aceita (por pilha vazia)** pelo AP.

6. Linguagens e APs

- **Definição:** Uma cadeia $w \in I^*$ é **aceita por estado final** por um AP $M=(S,I,\Gamma,f, q_0,Z_0,A)$ se existir computação $[q_0, w, Z_0] \overset{*}{\underset{M}{\vdash}} [q_i, \lambda, \alpha]$, com $q_i \in A$.
- **Definição:** Uma cadeia $w \in I^*$ é **aceita por pilha vazia** por um AP $M=(S,I,\Gamma,f, q_0,Z_0,A)$ se existir computação $[q_0, w, Z_0] \overset{*}{\underset{M}{\vdash}} [q_i, \lambda, \lambda]$.
- **Definição:** A linguagem $L(M)$ é o conjunto de cadeias aceitas por estado final pelo AP M.
- **Definição:** A linguagem $N(M)$ é o conjunto de cadeias aceitas por pilha vazia pelo AP M.

6. Linguagens e APs

- **Observação 1:**
  Podem existir computações que não terminam. As cadeias correspondentes são consideradas não aceitas.
- **Observação 2:**
  Aceitação como AFND's: aceita-se uma certa cadeia se existir **ao menos uma** computação que termine em estado final (ou pilha vazia).

6. Linguagens e APs: Exemplo 1

Um AP que aceita por pilha vazia a linguagem $\{wcw^R \mid w \in \{0,1\}^*\}$

OBS: W^R significa W reversa: se W=110, W^R =011 (se $W = \lambda$, $W^R = \lambda$)

M: $S=\{q_0, q_1\}$
$I=\{0,1,c\}$
$\Gamma=\{R,B,G\}$
$Z_0 = R$
$A = \varnothing$
q_0: estado inicial

$f(q_0,0,R)=\{[q_0,BR]\}$
$f(q_0,0,B)=\{[q_0,BB]\}$
$f(q_0,0,G)=\{[q_0,BG]\}$
$f(q_0,c,R)=\{[q_1,R]\}$
$f(q_0,c,B)=\{[q_1,B]\}$
$f(q_0,c,G)=\{[q_1,G]\}$
$f(q_1,1,R)=\{[q_0,GR]\}$
$f(q_1,1,B)=\{[q_0,GB]\}$
$f(q_1,1,G)=\{[q_0,GG]\}$
$f(q_1,\lambda,R)=\{[q_1,\lambda]\}$
$f(q_1,0,B)=\{[q_1,\lambda]\}$
$f(q_1,1,G)=\{[q_1,\lambda]\}$

$[q_0,01c10, R] \vdash [q_0,1c10,BR] \vdash [q_0,c10,GBR] \vdash [q_1,10,GBR] \vdash [q_1,0,BR]$
$\vdash [q_1,\lambda,R] \vdash [\mathbf{q_1}, \lambda, \lambda]$ **cadeia aceita!**

6. Linguagens e APs: Exemplo 2

Um APD que aceita por estado final a linguagem $\{wcw^R \mid w \in \{0,1\}^*\}$

M: $S=\{q_0, q_1, q_2\}$
$I=\{0,1,c\}$
$\Gamma=\{R,B,G\}$
$Z_0 = R$
$A=\{\mathbf{q_2}\}$
q_0: estado inicial

$f(q_0,0,R)=\{[q_0,BR]\}$
$f(q_0,0,B)=\{[q_0,BB]\}$
$f(q_0,0,G)=\{[q_0,BG]\}$
$f(q_0,c,R)=\{[q_1,R]\}$
$f(q_0,c,B)=\{[q_1,B]\}$
$f(q_0,c,G)=\{[q_1,G]\}$
$f(q_0,1,R)=\{[q_0,GR]\}$
$f(q_0,1,B)=\{[q_0,GB]\}$
$f(q_0,1,G)=\{[q_0,GG]\}$
$f(q_1,\lambda,R)=\{[q_2,R]\}$
$f(q_1,0,B)=\{[q_1,\lambda]\}$
$f(q_1,1,G)=\{[q_1,\lambda]\}$

$[q_0,01c10, R] \vdash [q_0,1c10,BR] \vdash [q_0,c10,GBR] \vdash [q_1,10,GBR] \vdash [q_1,0,BR]$
$\vdash [q_1,\lambda,R] \vdash [q_2,\lambda,R]$ **cadeia aceita!**

6. Linguagens e APs: Exemplo 3

Um APD que aceita por pilha vazia a linguagem $\{ww^R \mid w \in \{0,1\}^*\}$

M: $S=\{q_0, q_1\}$
$I=\{0,1\}$
$\Gamma=\{R,B,G\}$
$Z_0 = R$
$A = \varnothing$
q_0: estado inicial

$f(q_0,0,R)=\{[q_0,BR]\}$
$f(q_0,0,B)=\{[q_0,BB],[q_1,\lambda]\}$
$f(q_0,0,G)=\{[q_0,BG]\}$
$f(q_0,\lambda,R)=\{[q_1,\lambda]\}$
$f(q_1,\lambda,R)=\{[q_1,\lambda]\}$
$f(q_1,1,G)=\{[q_1,\lambda]\}$
$f(q_0,1,R)=\{[q_0,GR]\}$
$f(q_0,1,B)=\{[q_0,GB]\}$
$f(q_0,1,G)=\{[q_0,GG],[q_1,\lambda]\}$
$f(q_1,0,B)=\{[q_1,\lambda]\}$

Exemplo 3: AP **não-determinístico** - mais de uma transição possível para algumas configurações instantâneas.

Exemplos 1,2: APs **determinísticos** - no máximo uma transição possível para cada configuração instantânea

6. APs Determinísticos

- Formalmente, um AP $M=(S,I,\Gamma,f,q_0,Z_0,A)$ é determinístico se:

i) **$\forall$ q,Z f(q,λ,Z) não-vazio $\Rightarrow$ f(q,a,Z) vazio, $\forall$ a**
Elimino a possibilidade de escolha entre um movimento λ e um movimento dependente de um símbolo **a**.

ii) $\forall$ q,a,Z card(f(q,a,Z)) = 0 ou 1
Para qualquer f(q,a,Z) (incluindo-se a= λ), existe, no máximo, uma transição possível.

6. APs: observações importantes

- Não há equivalência entre APs não-determinísticos e APs determinísticos.
- Existe equivalência entre APs que aceitam linguagens por pilha vazia e APs que aceitam linguagens por estado final.

(Prova: Hopcroft/Ullman, págs, 114 e 115.)

6. APs e Linguagens Livres de Contexto

Teorema (*Chomsky, 1962*):

L é uma linguagem livre de contexto sss L é aceita por algum AP *M* que aceita por pilha vazia (ou por estado final).

(Prova: Hopcroft/Ullman, págs. 116-119)

7. Máquinas de Turing

- Na seção 3 vimos que $L=\{a^n b^n \mid n=1,2,\ldots\}$ não é uma linguagem regular (slide 38) e o teorema do slide 66 afirma que L não pode ser reconhecida por um autômato finito.
  - Para determinar se a cadeia 000000111111 faz parte de L, provavelmente contaremos o número de 0's na cadeia e, ao chegar o primeiro 1, anotaremos este número para futura comparação e então passaremos a contar o número de 1's. Desta forma, fizemos uso de alguma ***memória extra***, não disponível num autômato finito!

7. Máquinas de Turing

- Para simular procedimentos mais gerais dos que os permitidos pelas máquinas de estado finito, usamos uma ***Máquina de Turing***, proposta pelo matemático inglês Alan M. Turing em 1936.
- Uma máquina de Turing é essencialmente uma máquina de estado finito com a habilidade de **ler suas entradas mais de uma vez** e também de **apagar** ou **substituir** os valores de suas entradas; ela também tem uma **memória auxiliar ilimitada**.

7. Máquinas de Turing

- A Máquina de Turing consiste em: uma fita dividida em células, cada uma contendo no máximo um símbolo de um alfabeto finito disponível; um cursor que efetua leitura e gravação; uma máquina de estado finito, que controla o cursor (a fita é a memória externa).

A qualquer momento, apenas um número finito de células contém valor. O símbolo *b* denota célula sem valor. A máquina de estado finito -MEF- lê uma célula a cada momento.

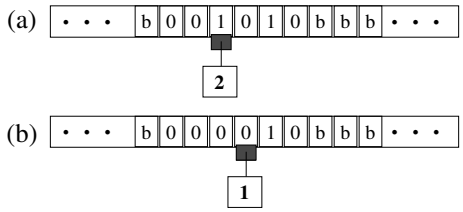
7. Máquinas de Turing

- ***Funcionamento***: uma célula é lida. No pulso de relógio seguinte, dependendo do estado corrente da MEF e do símbolo lido, a máquina pode:
  - (i) não realizar ação alguma (**parar**) ou
  - (ii) realizar três ações: 1) **imprimir um símbolo** do alfabeto na célula lida (pode ser o mesmo da célula); 2) **ir para o estado seguinte** (pode ser o mesmo que estava antes); 3) **mover o cursor** para a esquerda ou para a direita.

Ações: (s, i, i', s', d) sendo s: estado atual; i: símbolo que está sendo lido na fita; i': símbolo impresso, s': novo estado; d∈{D,E}: mover para direita ou esquerda.

7. Máquinas de Turing

- ***Ex***: Para a máquina em (a), executar a ação (2, 1, 0, 1, D), resultando em (b).



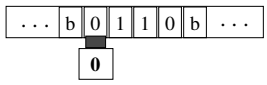
7. Máquinas de Turing

- **Definição:** Máquinas de Turing

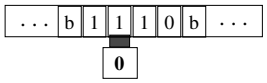
Seja S um conjunto finito de estados e I um conjunto finito de símbolos da fita (alfabeto da fita) incluindo o símbolo especial *b*. Uma máquina de Turing é um conjunto de quíntuplas da forma *(s, i, i', s', d)* onde *s, s'∈S; i, i'∈I; d∈{D,E}* e não há mais de uma quíntupla começando pelos mesmos símbolos *s* e *i*.

7. Máquinas de Turing

- Ex: uma máquina de Turing é definida pelo conjunto de quíntuplas: (0,0,1,0,D), (0,1,0,0,D), (0,*b*,1,1,E), (1,0,0,1,D), (1,1,0,1,D).

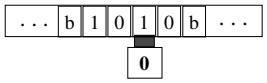
Início: 

(0,0,1,0,D)

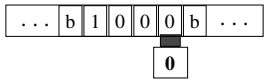


(0,1,0,0,D)

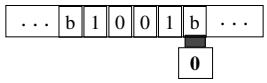
7. Máquinas de Turing



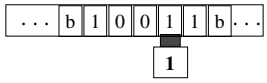
(0,1,0,0,D)



(0,0,1,0,D)

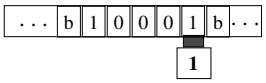


(0,b,1,1,E)

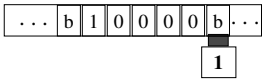


(1,1,0,1,D)

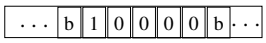
7. Máquinas de Turing



(1,1,0,1,D)



Como não há quíntuplas que definam a ação a ser tomada quando se está no estado 1 e lendo b, a máquina pára com a fita contendo:



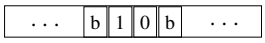
7. Máquinas de Turing

- A fita serve como memória para a Máquina de Turing.
- Como a MT também pode escrever na fita, a máquina dispõe de uma quantidade ilimitada, porém finita, de memória.
- Um autômato finito é um caso muito particular de MT, que sempre imprime o símbolo anterior na célula lida, sempre se move para a direita e sempre pára no símbolo *b*.

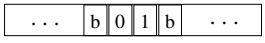
7. Máquinas de Turing

- Exercício: Considere a seguinte MT: (0,0,0,1,D), (0,1,0,0,D), (0,b,b,0,D), (1,0,1,0,D), (1,1,1,0,E). Qual o comportamento da MT quando iniciada com:

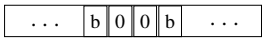
(a)



(b)



(c)



7. Máquinas de Turing

- Aplicações de MTs:
  - para reconhecer conjuntos (linguagens)
  - para computar funções
- Reconhecimento (aceitação) por MT
  - Uma máquina de Turing T com alfabeto da fita I reconhece (aceita) um subconjunto S de I* se T, começando em alguma configuração inicial em uma fita contendo uma cadeia α de símbolos de entradas, pára em um *estado final* se e somente se $\alpha \in S$.

7. Máquinas de Turing

- Um estado final em uma máquina de Turing é um estado que não seja o primeiro elemento de qualquer quíntupla.
- Pela definição de aceitação dada, T pode ter dois comportamentos quando aplicada a uma cadeia $\alpha \notin S$: *(i)* T pode parar em um estado não final ou (ii) T pode não parar.

7. Máquinas de Turing

- **Ex**: MT que reconhece $S=\{0^n1^n\}$, com $n \geq 0$. Alfabeto da fita={0,1,b,X}. O estado 8 é o único estado final.

(0,b,b,8,D) : reconhece a fita vazia, que pertence a S.
(0,0,X,1,D): apaga o 0 mais à esquerda e inicia mov para direita.
(1,0,0,1,D), (1,1,1,1,D), (1,b,b,2,E): move p/ direita, no estado 1, até que encontre o fim da cadeia inicial; então, move p/ esquerda, no estado 2.
(2,1,X,3,E): apaga o 1 mais à direita e inicia mov para esquerda.
(3,1,1,3,E): move para a esquerda por sobre os 1s.
(3,0,0,4,E): passa ao estado 4 ao encontrar um 0.
(3,X,X,7,D): passa ao estado 7 se não encontrar mais 0s.
(4,0,0,4,E): move para a esquerda por sobre os 0s.
(4,X,X,5,D): encontra o extremo esquerdo da cadeia e recomeça o processo.
(5,0,X,6,D): apaga o 0 mais à esquerda.
(6,0,0,6,D), (6,1,1,6,D), (6,X,X,2,E): move p/ direita, no estado 6, até que encontre o fim da cadeia binária; então, move p/ esquerda, no estado 2.
(7,X,X,8,D): não há mais 1s na cadeia, a máquina aceita a cadeia.

7. Máquinas de Turing

- MT para computar funções:
  - Dada uma máquina de Turing T e uma cadeia α de símbolos da fita, começamos com T na configuração inicial padrão em uma fita contendo α. Se T em algum momento pára deixando uma cadeia β na fita, podemos considerar β o valor de uma função avaliada em α. Assim, $T(\alpha) = \beta$. O domínio da função T consiste em todas as cadeias α para as quais T, em algum momento, pára.

8. Linguagens Formais e Dispositivos Computacionais

- Vimos que as cadeias aceitas por um dispositivo computacional de capacidade limitada coincide com a classe mais restrita de linguagens: uma linguagem regular (tipo3) é aceita por um autômato finito.
- O dispositivo computacional mais geral é a Máquina de Turing e a linguagem mais geral é a do tipo 0 (irrestrita). Por isso, conjuntos reconhecidos por máquinas de Turing são as linguagens de tipo 0.

8. Linguagens Formais e Dispositivos Computacionais

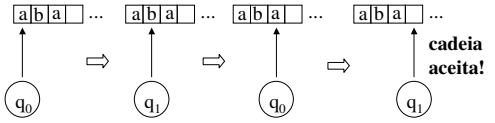
- Como vimos, existem dispositivos computacionais com capacidades intermediárias entre autômatos finitos e máquinas de Turing, que reconhecem linguagens do tipo 2 (livres de contexto) e tipo 1(sensíveis ao contexto).
- Os **autômatos de pilha** reconhecem linguagens do tipo 2 (livres de contexto).
- Os **autômatos limitados lineares** reconhecem linguagens do tipo 1(sensíveis ao contexto).

8. Linguagens Formais e Dispositivos Computacionais

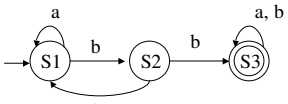
- Os autômatos finitos também podem ser vistos como uma máquina de estado finito, um cursor e uma fita. Neste caso, porém, o cursor somente efetua **leituras** e se movimenta para a **direita**.
  - fita: armazena uma cadeia de I (1 símbolo/célula).
  - cursor: lê célula da fita (um símbolo da cadeia).
  - máquina: altera estado de acordo com *f* e move o cursor para a direita (computação).
  - uma computação **termina** quando a cadeia "acaba".
  - uma cadeia é **aceita** pelo AF se a computação terminar em um estado $s \in A$.
  - uma cadeia é **rejeitada** pelo AF se a computação terminar em um estado $s \notin A$.

Exemplo 1 - AF:

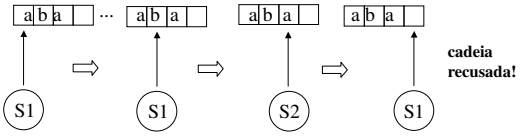
$S = \{q_0, q_1\}$ $f(q_0, a) = q_1$
$I = \{a, b\}$ $f(q_0, b) = q_0$
$A = \{q_1\}$ $f(q_1, a) = q_1$
$\sigma = q_0$ $f(q_1, b) = q_0$



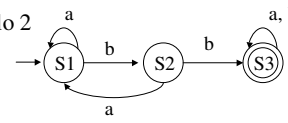
Exemplo 2:

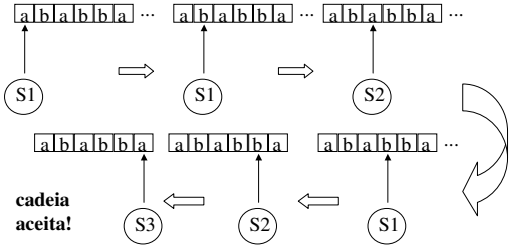


AF:
$S = \{S1,S2,S3\}$ $f(S1, a) = S1$ $f(S3,a) = S3$
$I = \{a, b\}$ $f(S1, b) = S2$ $f(S3,b) = S3$
$A = \{S3\}$ $f(S2, a) = S1$
$\sigma = S1$ $f(S2, b) = S3$



Continuação – exemplo 2





- **Autômato de Pilha (I)**:
  - consiste em uma máquina de estado finito que lê a entrada de uma fita e controla a atividade de uma pilha. Os símbolos do alfabeto podem ser empilhados ou desempilhados do topo da pilha.
  - O AP reconhece o conjunto de todas as entradas para as quais existe uma seqüência de movimentos que faça a pilha ficar ***vazia***.
  - A MEF tem um número finito de movimentos a realizar, os quais são escolhidos em função do símbolo de entrada lido, do estado atual e do símbolo do topo da pilha.

8. Linguagens Formais e Dispositivos Computacionais

- **Autômato de Pilha (II)**:

  Os movimentos podem ser:

  – Passa ao estado seguinte, retira o símbolo do topo da pilha e lê o próximo símbolo de entrada;

  – Passa ao estado seguinte, retira o símbolo do topo da pilha, inclui um número finito de símbolos no topo da pilha e lê o próximo símbolo de entrada; ou

  – Ignora o símbolo da entrada sendo lido, manipula a pilha como acima, mas não lê o próximo símbolo da entrada.

8. Linguagens Formais e Dispositivos Computacionais

- **Autômato Limitado Linear:**

  – é uma máquina de Turing cujo cursor de leitura/gravação está limitado à parte da fita que contém a entrada original; além disso, a cada passo ela tem uma escolha de quais movimentos realizar, da mesma forma que a MT.

  – Um autômato limitado linear aceita o conjunto de todas as entradas para as quais existe alguma seqüência de movimentos que faça a máquina de Turing parar em algum estado.

8. Linguagens Formais e Dispositivos Computacionais

Linguagem	**Gramática**	**Disp. Computacional**
irrestrita	irrestrita	Máquina de Turing
sensível ao contexto	sensível ao contexto	Autômato limitado linear
livre de contexto	livre de contexto	Autômato de Pilha
regular	regular	Autômato Finito

Bibliografia

[1] Johnsonbaugh, R. Discrete Mathematics. Prentice Hall International, London, UK, 4th. Ed. 1997. Cap. 10.

[2] Gersting, J.L. Fundamentos Matemáticos para a Ciência da Computação. LTC Editora, Rio de Janeiro, Brasil, 3a. Edição. 1995. Cap. 8.

[3] Neto, J.J. Introdução à Compilação. LTC Editora, Rio de Janeiro, Brasil. 1987. Cap. 2.

[4] Hopcroft, J.E. e J.D. Ullman. Introduction to Automata Theory, Languages and Computation. Addison-Wesley.

[5] Lewis, H. R. e C.H. Papadimitriou. Elements of the Theory of Computation. Prentice-Hall.