FUNDAMENTOS TEÓRICOS DA COMPUTAÇÃO

--- LINGUAGEM LIVRE DE CONTEXTO ---

Autômatos com Pilhas Determinísticos

Definição Formal

Um autômato de pilha determinístico (APD) é uma 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, onde:

- Q é um conjunto finito de estados
- Σ é o alfabeto de entrada
- Γ é o alfabeto da pilha
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow Q \times \Gamma^*$ é uma função de transição parcial e sem transições compatíveis
- $q_0 \in Q$ é o estado inicial
- $F \subseteq Q$ é o conjunto de estados de aceitação

Autômatos de Pilha Determinísticos

Os autômatos de pilha determinísticos (APDs) são especialmente importantes, já que lidam com uma classe de linguagens para as quais há reconhecedores eficientes

A definição de autômato de pilha determinístico, basicamente, acrescenta uma pilha a um AFD

Para que haja determinismo, não é possível mais de uma transição ser definida para uma mesma **configuração** instantânea

Em outras palavras, em um estado qualquer do APD, qualquer que seja o próximo símbolo de entrada, e qualquer que seja a situação atual da pilha, no máximo **uma transição** deverá ser possível.

Transições Compatíveis

Duas transições $\delta(e_1, a_1, b_1)$ e $\delta(e_2, a_2, b_2)$; $e_1, e_2 \in Q, a_1, a_2 \in (\Sigma \cup \varepsilon), b_1, b_2 \in (\Gamma \cup \varepsilon)$, são ditas **compatíveis** se e somente se:

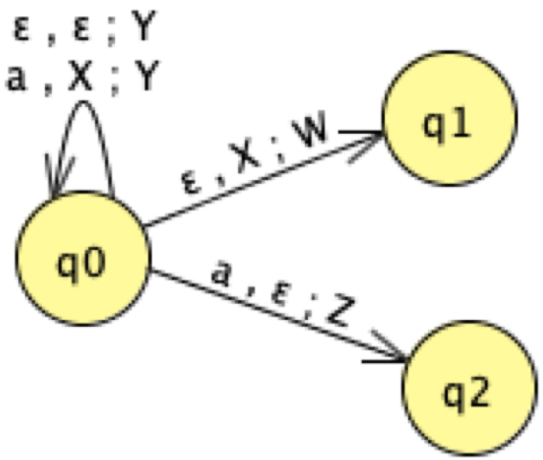
$$e_1 = e_2 \& (a_1 = a_2 | a_1 = \varepsilon | a_2 = \varepsilon) \& (b_1 = b_2 | b_1 = \varepsilon | b_2 = \varepsilon)$$

Veja que **não** pode:

$(e_1 = e_2) \& (a_1 = a_2) \& (b_1 = b_2)$ **ou**	$\boldsymbol{\delta(q_i, a, X)}$ & $\boldsymbol{\delta(q_i, a, X)}$
$(e_1 = e_2) \& (a_1 = \varepsilon) \& (b_1 = b_2)$ **ou**	$\boldsymbol{\delta(q_i, \varepsilon, X)}$ & $\boldsymbol{\delta(q_i, a, X)}$
$(e_1 = e_2) \& (a_1 = \varepsilon) \& (b_1 = \varepsilon)$ **ou**	$\boldsymbol{\delta(q_i, \varepsilon, \varepsilon)}$ & $\boldsymbol{\delta(q_i, a, X)}$
$(e_1 = e_2) \& (a_1 = \varepsilon) \& (b_2 = \varepsilon)$ **ou**	$\boldsymbol{\delta(q_i, \varepsilon, X)}$ & $\boldsymbol{\delta(q_i, a, \varepsilon)}$
$(e_1 = e_2) \& (a_2 = \varepsilon) \& (b_1 = b_2)$ **ou**	$\boldsymbol{\delta(q_i, a, X)}$ & $\boldsymbol{\delta(q_i, \varepsilon, X)}$
$(e_1 = e_2) \& (a_2 = \varepsilon) \& (b_1 = \varepsilon)$ **ou**	$\boldsymbol{\delta(q_i, a, \varepsilon)}$ & $\boldsymbol{\delta(q_i, \varepsilon, X)}$
$(e_1 = e_2) \& (a_2 = \varepsilon) \& (b_2 = \varepsilon)$	$\boldsymbol{\delta(q_i, a, X)}$ & $\boldsymbol{\delta(q_i, \varepsilon, \varepsilon)}$

Transições Compatíveis

$\omega = a$
Pilha $= X$



Qual transição executar?

Apesar de um APD admitir transições sob ε, ele **não admite transições compatíveis**. Logo, qualquer configuração instantânea que seja atingida terá no máximo uma sucessora.

Note que a compatibilidade **não depende** dos estados destino nem das palavras a empilhar.

Relação de Transição de Estados

As seguintes razões fazem que não haja uma função $\hat{\delta}$ similar às linguagens regulares:

- Pode haver computações que não terminam
- Além do estado final, tem-se saber o conteúdo da pilha

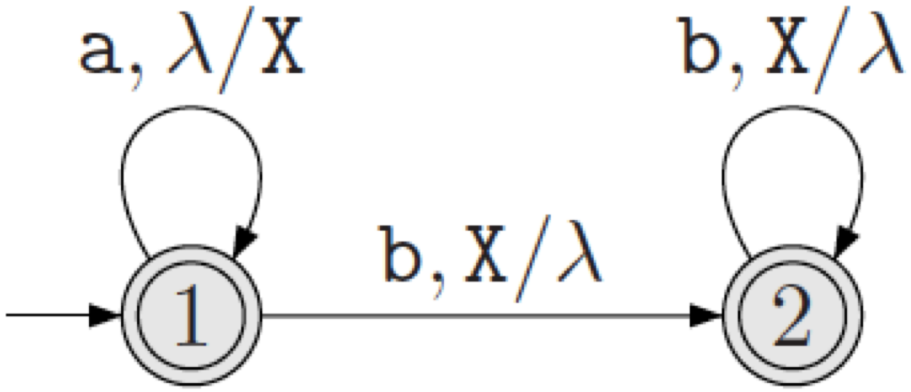
Em vez de uma função de transição estendida $\hat{\delta}$, será usada a relação $\vdash$ definida da seguinte forma:

Seja um APD $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, a relação $\vdash \subseteq (E, \Sigma^*, \Gamma^*)$ para M, é tal que, para todo $e, e' \in Q, a \in (\Sigma \cup \varepsilon), b \in (\Gamma \cup \varepsilon)$ e $x \in \Gamma^*$:

$[e, ay, bz] \vdash [e', y, xz]$ para todo $y \in \Sigma^*$ e $z \in \Gamma^*$ se e somente se, $\delta(e, a, b) = [e', x]$

Exemplo (1)

$\{a^n b^n : n \geq 0\}$



Funciona somente se a pilha está vazia **e** termina em estado de aceitação

Exemplo (1)

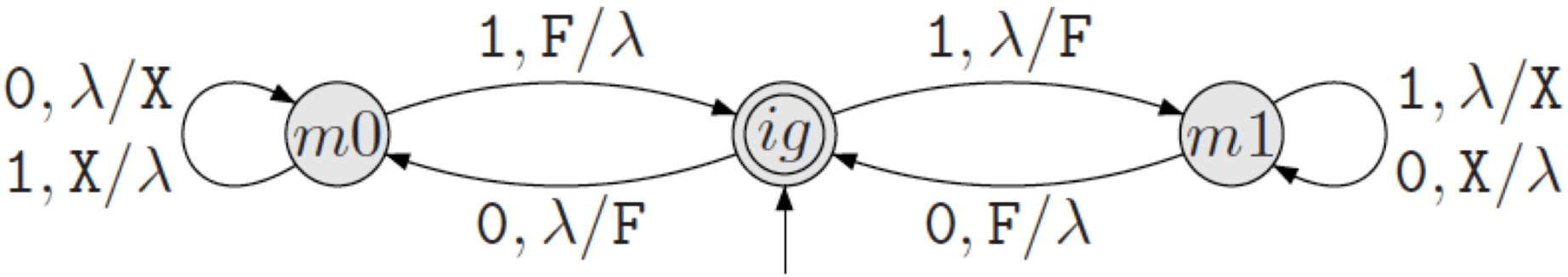
A seguinte computação mostra que "aabb" pertence à linguagem

$$[1, aabb, \varepsilon] \vdash [1, abb, X]$$
$$\vdash [1, bb, XX]$$
$$\vdash [2, b, X]$$
$$\vdash [2, \varepsilon, \varepsilon]$$

A cadeia foi **aceita** porque a pilha está vazia **e** o último estado é de aceitação

Exemplo (2)

$$\{\omega \in \{0,1\}^* : |\omega|_0 = |\omega|_1\}$$



A informação de que as quantidades são iguais (**estado** ig) ou a de 0s é maior (**estado** m0) ou a de 1s é maior (**estado** m1). Já a informação do excesso de um dos símbolos é dado pela **pilha**.

Exemplo (2)

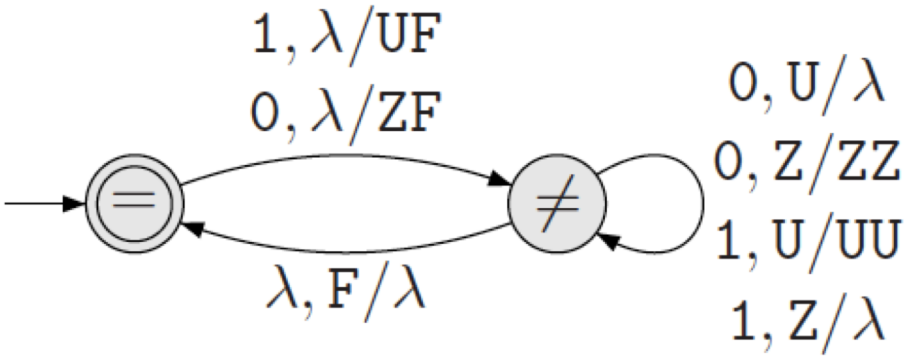
A seguinte computação mostra que ”001110" pertence à linguagem

$$[ig, 001110, \varepsilon] \vdash [m0, 01110, F]$$
$$\vdash [m0, 1110, XF]$$
$$\vdash [m0, 110, F]$$
$$\vdash [ig, 10, \varepsilon]$$
$$\vdash [m1, 0, F]$$
$$\vdash [ig, \varepsilon, \varepsilon]$$

A cadeia foi **aceita** porque a pilha está vazia **e** o último estado é de aceitação

Exemplo (3)

$\{\omega \in \{0,1\}^* : |\omega|_0 = |\omega|_1\}$



A informação de qual símbolo está em excesso é colocada na pilha

Exemplo (3)

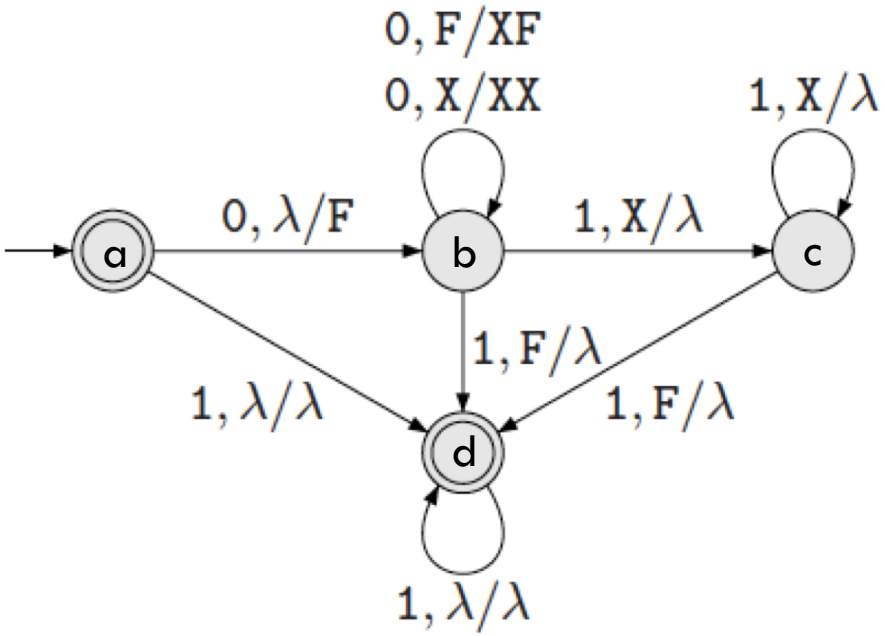
A seguinte computação mostra que ”001110" pertence à linguagem

$$[=, 001110, \varepsilon] \vdash [\neq, 01110, ZF]$$
$$\vdash [\neq, 1110, ZZF]$$
$$\vdash [\neq, 110, ZF]$$
$$\vdash [\neq, 10, F]$$
$$\vdash [=, 10, \varepsilon]$$
$$\vdash [\neq, 0, UF]$$
$$\vdash [\neq, \varepsilon, F]$$
$$\vdash [=, \varepsilon, \varepsilon]$$

A cadeia foi **aceita** porque a pilha está vazia **e** o último estado é de aceitação

Exemplo (4)

$\{0^m 1^n : m \leq n\}$



O marcador F é usado para detectar quando o fundo de pilha é atingido, que é quando, ao ler o próximo 1, o número de 1s se torna maior que o de 0s

Exemplo (4)

A seguinte computação mostra que ”00111" pertence à linguagem

$$[a, 00111, \varepsilon] \vdash [b, 0111, F]$$
$$\vdash [b, 111, XF]$$
$$\vdash [c, 11, F]$$
$$\vdash [d, 1, \varepsilon]$$
$$\vdash [d, \varepsilon, \varepsilon]$$

A cadeia foi **aceita** porque a pilha está vazia **e** o último estado é de aceitação

Autômato com Pilha Determinístico é o meu precioso!!!