Fundamentos Teóricos da Computação

CIÊNCIA DA COMPUTAÇÃO

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Sumário

* **Introdução**
* **Autômatos de Pilha Determinísticos**

Introdução

* **Introdução**

Introdução

* Veremos agora, uma extensão dos AFs, os denominados autômatos de pilha. São de grande importância, visto que constituem uma base para a obtenção de reconhecedores para muitas linguagens que ocorrem na prática.

* Em particular, alguns compiladores de linguagens de programação utilizam alguma variante de autômato de pilha na fase de análise sintática.

Introdução

- Ao contrário dos AFs, a versão não determinística desse tipo de autômato tem uma abrangência maior que a determinística.

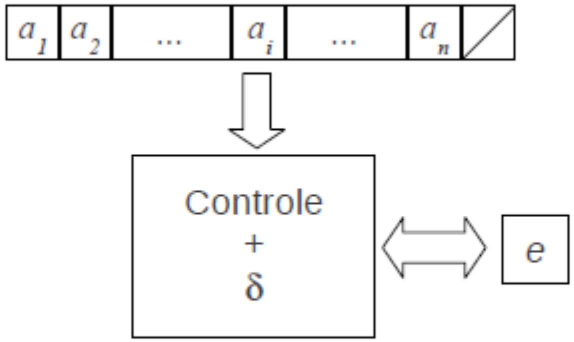
- No entanto, as linguagens que podem ser reconhecidas por autômatos de pilha determinísticos são especialmente importantes, já que admitem reconhecedores eficientes.

Um olhar para o futuro

* Depois de vermos as versões determinística e não determinística de autômatos de pilha, serão estudadas as gramáticas livres de contexto, que são um formalismo de grande utilidade prática para a especificação de linguagens reconhecíveis por autômatos de pilha.

Autômatos Finitos

- Um AFD pode ser visto como uma máquina que opera sobre uma fita somente de leitura, cujo cabeçote se movimenta somente para a direita.



- <u>Legenda</u>: a_1, a_2...a_i....a_n...:fita de leitura apenas, unidirecional e:registrador com estado atual

Limites dos AFs

* AFs reconhecem somente a classe das linguagens regulares

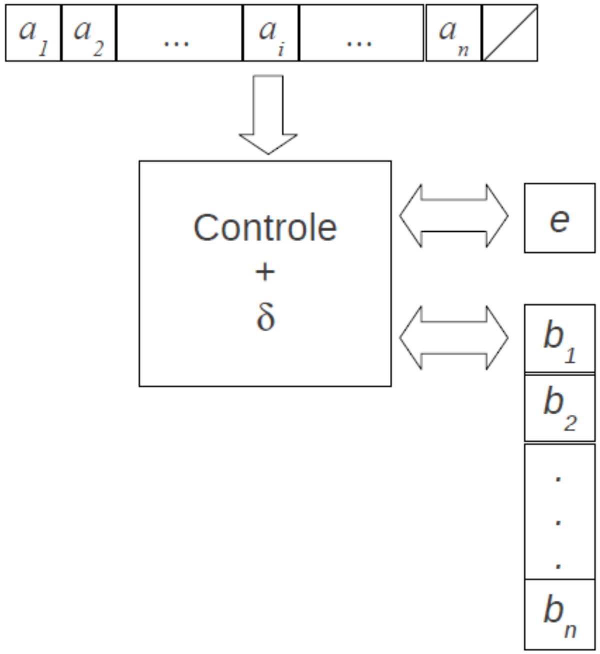
* Muitas linguagens interessantes não são regulares
  * Como é o caso, por exemplo, de algumas linguagens que contêm expressões aritméticas.

$$({}^n t_1 + t_2) + t_3)\ldots + t_{n+1})$$

    * Um AF não pode reconhecer a linguagem acima porque não tem uma memória poderosa o suficiente para "lembrar" que leu n ocorrências de certo símbolo, para n arbitrário.
    * O único modo de ler uma quantidade arbitrária de determinado símbolo, em um AF, é por meio de um ciclo. E, nesse caso, não há como contar o número de símbolos lidos.

Arquitetura de um AP

* Um AP pode ser visto como uma máquina semelhante à vinculada ao AF, porém com uma pilha adicional.



(b1 é o topo da pilha)

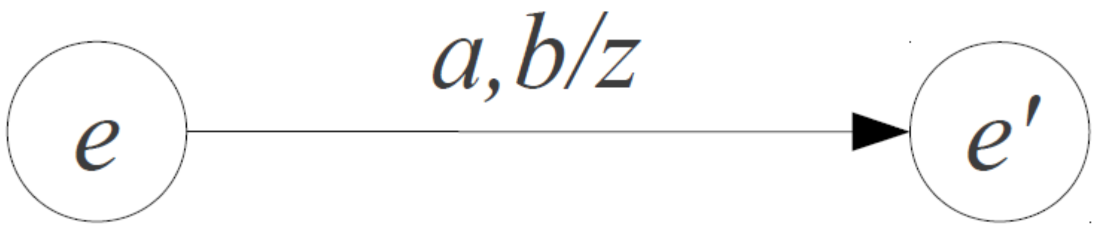
Função de Transição

* Em um AP com conjunto de estados E, alfabeto de entrada (da fita) Σ e alfabeto da pilha Γ, cada transição será da forma:

  * $\delta(e, a, b) = [e', z]$
    * Ou $[e', z] \in \delta(e, a, b)$ para AP não determinístico

Representação Gráfica de uma Transição

- A transição de e para e' ocorre se recebe-se um símbolo a e o topo da pilha é o símbolo b. O símbolo b é desempilhado e o símbolo z é empilhado.
  - z pode ser uma palavra. Neste caso, o símbolo mais a esquerda em z deve ficar no topo (convenção)



Transições

* Se $a = \lambda$, não é consumido símbolo de entrada
* Se $b = \lambda$, a pilha não é consultada e nada é desempilhado
* Se $z = \lambda$, nada é empilhado

Reconhecendo Palavras

- Informalmente:

  - Para um AP reconhecer uma palavra
    - A palavra tem que ser totalmente consumida
    - O AP terminar em um estado final
    - A pilha deve estar vazia ao fim da computação (λ)

Exemplo 1

* Seja a linguagem EA, de expressões aritméticas, definida recursivamente por:
  * $t \in EA$;
  * Se x, y $\in EA$, então pertencem à EA:
    * (x)
    * x + y
    * x – y

    O símbolo t representa expressões básicas, como número inteiros, reais ou variáveis (O reconhecimento de expressões básicas pode ser feito usando um AF, como já visto anteriormente)

Exemplo 1

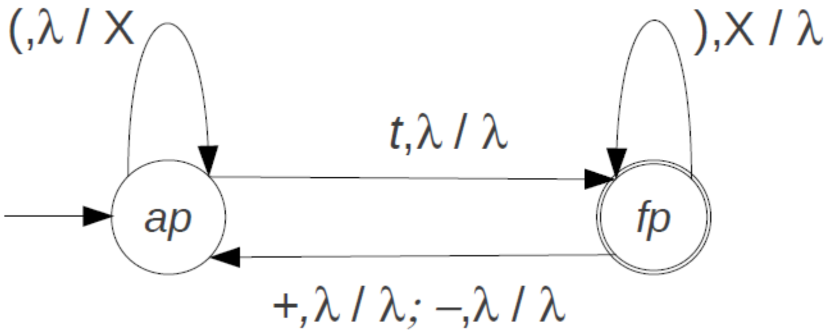
* Raciocínio
  * No estado inicial pode-se receber *t* ou “(”
    * Uma palavra de EA não pode começar com “)”
  * Sempre que se recebe um “(”, deve-se empilhar um símbolo na pilha para armazenar que ocorreu um abre parênteses. Vamos utilizar o símbolo “X”
  * Pode-se chegar a um estado final recebendo *t* ou “)”
    * Uma palavra de EA não pode terminar com “(”
  * Sempre que se recebe um “)”, deve-se desempilhar “X” da pilha
  * Depois de um “+” ou de um “–”, pode-se seguir um “(“ ou *t*

Exemplo 1

* Como ficou este AP que reconhece EA:

* *Conjunto de estados:* $E = \{ap, fp\}$
* *Alfabeto de entrada:* $\Sigma = \{t, (,), +, -\}$
* *Alfabeto da pilha:* $\Gamma = \{X\}$

Exemplo 1



1. $\delta(ap,\ (,\ \lambda) = [ap,\ X]$
2. $\delta(ap,\ t,\ \lambda) = [fp,\ \lambda]$
3. $\delta(fp,\ ),\ X) = [fp,\ \lambda]$
4. $\delta(fp,\ +,\ \lambda) = [ap,\ \lambda]$
5. $\delta(fp,\ -,\ \lambda) = [ap,\ \lambda]$

Exemplo 1

* Configuração instantânea
  * [estado, palavra, pilha]
  * Executamos até a palavra terminar ou nenhum símbolo poder ser consumido

[*ap*, (t - (t+t)), λ] |- [*ap*, t - (t+t)), *X*] por 1
|- [*fp*, - (t+t)), *X*] por 2
|- [*ap*, (t+t)), *X*] por 5
|- [*ap*, t+t)), *XX*] por 1
|- [*fp*, +t)), *XX*] por 2
|- [*ap*, t)), *XX*] por 4
|- [*fp*,)), *XX*] por 1
|- [*fp*,), *X*] por 3
|- [*fp*, λ, λ] por 3

1. δ(*ap*, (, λ) = [*ap*, X]
2. δ(*ap*, *t*, λ) = [*fp*, λ]
3. δ(*fp*,), X) = [*fp*, λ]
4. δ(*fp*, +, λ) = [*ap*, λ]
5. δ(*fp*, −, λ) = [*ap*, λ]

Exemplo 1

* Se executarmos o mesmo procedimento para a palavra: “t)”

* Veremos que não há transição que se aplique a $[fp, \text{)}, \lambda]$ (Esperávamos $[fp, \text{)}, X]$ por 3, não havia X para desempilhar)

* Isso mostra que o AP pode não consumir toda a palavra de entrada

* Sendo assim, diz-se que o AP pode “parar sem consumir toda a palavra de entrada”

Configuração Instantânea

* Em um AF, somente o estado e a palavra a ser processada eram o suficientes para determinar a configuração instantânea

* Em um AP, é necessário o estado, a palavra a ser processada e o conteúdo da pilha

Autômatos de Pilha Determinísticos

- **Autômatos de Pilha Determinísticos**

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.

Transições Compatíveis

* Uma pilha de símbolos de um alfabeto Γ será representada por meio de uma palavra w de Γ^*

* Seja a função de Transição

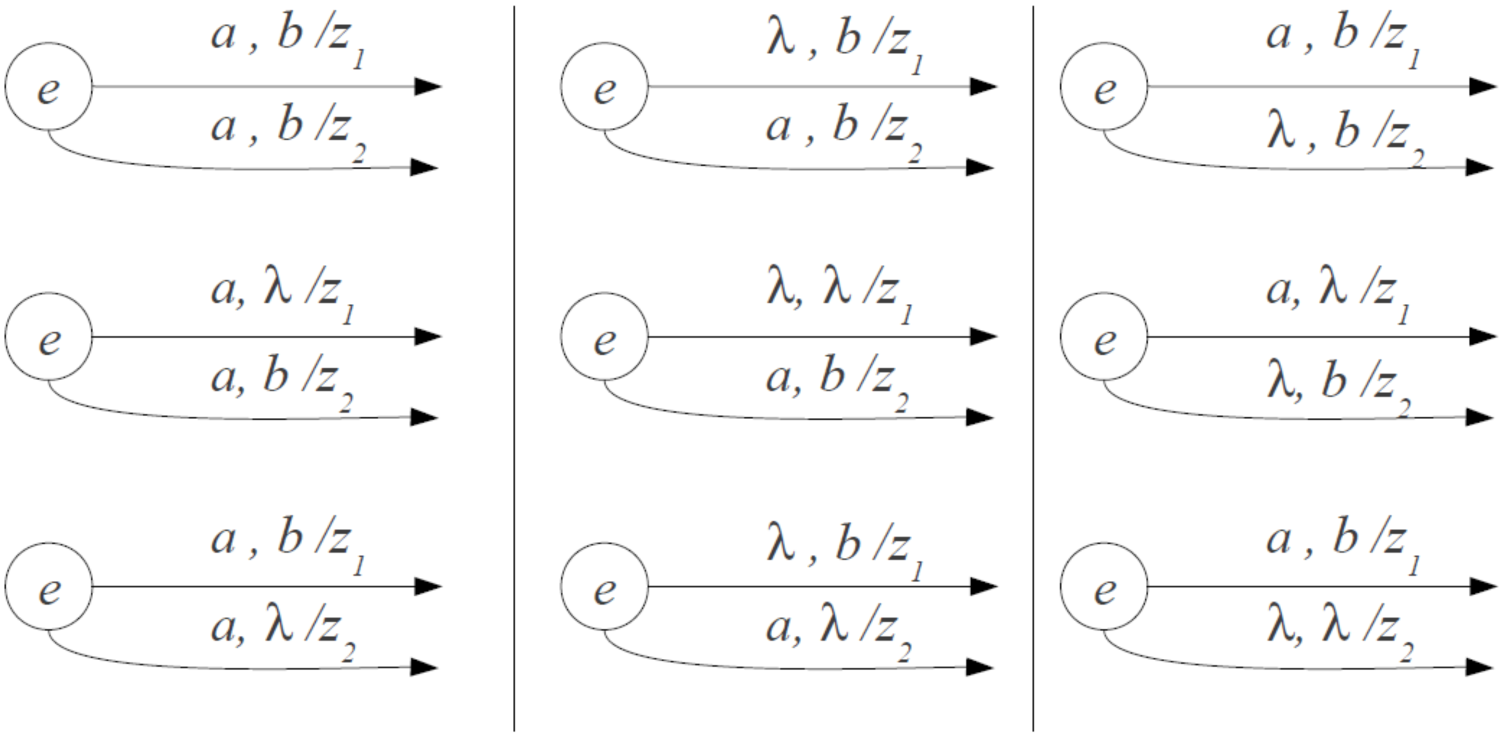
$$\delta : E \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \to E \times \Gamma^*$$

(Além dos estados atingidos, é importante saber o conteúdo da pilha)

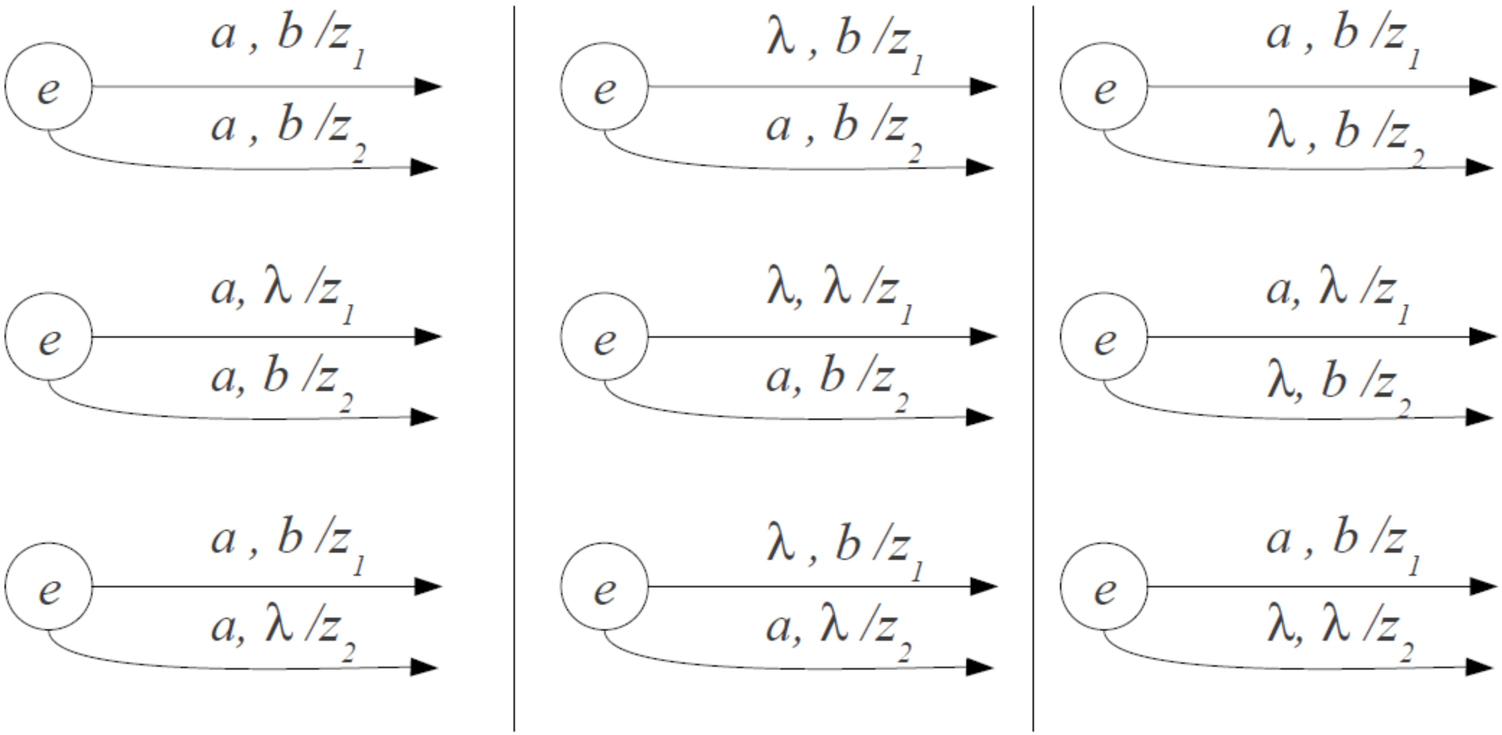
* Duas transições $\delta(e, a, b)$ e $\delta(e, a', b')$ são ditas compatíveis se, e somente se:

$$(a = a' \text{ ou } a = \lambda \text{ ou } a' = \lambda) \text{ e } (b = b' \text{ ou } b = \lambda \text{ ou } b' = \lambda)$$

Transições Compatíveis



Transições Compatíveis



Não determinismo nas transições

Transições Compatíveis

* Um autômato de pilha <u>determinístico</u> tem no máximo uma transição possível para uma mesma combinação de estado, símbolo de entrada, e símbolo no topo da pilha.

* Isto é o que o difere de um autômato de pilha <u>não determinístico</u>.

Problema de Equivalência

* Géraud Sénizergues (1997) provou que o problema de equivalência para autômatos de pilha determinísticos é decidível.

* Isto é, dados dois APDs A e B, é possível dizer que $L(A) = L(B)$?

* Para AP não determinístico, o problema de equivalência é indecidível.

Fonte: Sénizergues G. (1997) The equivalence problem for deterministic pushdown automata is decidable. In: Degano P., Gorrieri R., Marchetti-Spaccamela A. (eds) Automata, Languages and Programming. ICALP 1997. Lecture Notes in Computer Science, vol 1256. Springer, Berlin, Heidelberg.

Problema de Equivalência

* Esta prova rendeu a Géraud Sénizergues, em 2002, um Gödel Prize.

* O Prêmio Gödel é um prêmio por artigos de destaque em teoria da ciência da computação, homenageando Kurt Gödel e concedido conjuntamente pela Associação Europeia de Ciência Computacional Teórica (EATCS) e pela ACM SIGACT.

* O prêmio é concedido anualmente desde 1993. Seu valor monetário é de 5 mil dólares. O prêmio é concedido durante o "Simpósio sobre Teoria da Computação" ou durante o "Colóquio Internacional sobre Autômatos, Linguagem e Programação". Para ser elegível ao prêmio, um artigo deve ter sido publicado em uma revista especializada com revisores nos 14 anos precedentes. Anteriormente o tempo era de 7 anos.

Definição de APD

- Um Autômato de Pilha Determinístico (APD) é uma sêxtupla $(E, \Sigma, \Gamma, \delta, i, F)$ em que:
  - E é um conjunto finito de um ou mais estados;
  - Σ é o alfabeto de entrada;
  - Γ é o alfabeto de pilha;
  - δ, a função de transição, é <u>parcial</u> e <u>sem transições compatíveis</u>;
  - i é o estado inicial;
  - F é conjunto de estados finais.

Definição da relação "Resulta"

* As seguintes razões fazem com que não haja como definir uma função de transição estendida $\hat{\delta}$, similar àquela que vimos anteriormente na aula de AFDs:
  * 1) Ficou claro que podem haver computações que não terminam
  * 2) Além do(s) estado(s) atingido(s), é importante saber o conteúdo da pilha
* Sendo assim, em vez de uma função de transição estendida $\hat{\delta}$, será usada a relação $\vdash$ definida a seguir

Definição da relação "Resulta"

- Seja o APD $M = (E, \Sigma, \Gamma, \delta, i, F)$.
- A relação $\vdash \subseteq (E \times \Sigma^* \times \Gamma^*)^2$ é tal que para todo

$$e, e' \in E, a \in \Sigma \cup \{\lambda\}, b \in \Gamma \cup \{\lambda\} \text{ e } x \in \Gamma^* :$$

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

- Utilizamos a relação $\vdash$ no exemplo 1 (slide 18)
- Relembre:

$[ap, (\text{t - (t+t)}), \lambda] \vdash [ap, \text{t - (t+t))}, X] \leftrightarrow \delta(ap, (, \lambda) = [ap, X]$

Linguagem Reconhecida

- Utilizando a relação $*\vdash$, define-se a seguir o que é a linguagem reconhecida (aceita) por um APD.

- Seja o APD $M = (E, \Sigma, \Gamma, \delta, i, F)$. A linguagem reconhecida por M é:

$$L(M) = \{w \in \Sigma^* | [i, w, \lambda] {*\vdash} [e, \lambda, \lambda] \text{ para algum } e \in F\}$$

- Informalmente: Começamos consumindo a palavra w a partir do estado inicial i com a pilha vazia (λ) e terminamos em um estado final e com nada a ser desempilhado (λ) e nada a empilhar (λ).

Exemplo 2

- Vimos na aula de "Projeto de AFDs" que o conjunto $\{a^n b^n \mid n \in \mathbf{N}\}$ não é uma linguagem regular. Dessa forma, é <u>impossível</u> construir um AF para a linguagem:

$$L = \{a^n b^n | n \geq 0\}$$

- Essa linguagem, porém, pode ser reconhecida por um autômato de pilha determinístico (APD):

$$M = (\{\mathrm{ca}, \mathrm{cb}\}, \{\mathrm{a,b}\}, \{\mathrm{X}\}, \delta, \mathrm{ca}, \{\mathrm{ca}, \mathrm{cb}\})$$

Exemplo 2

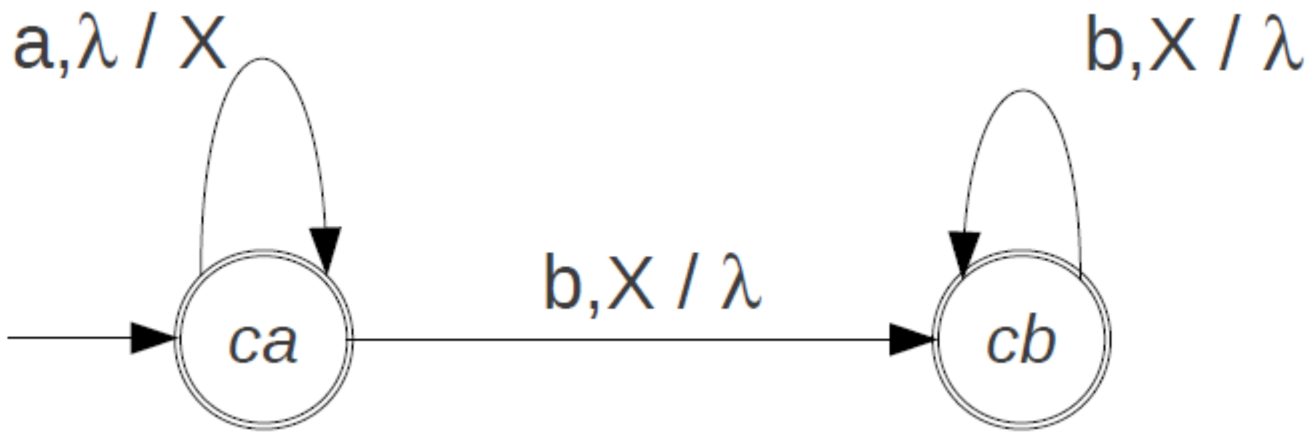
* Em que δ, é dada por:

1. $\delta(\text{ca}, a, \lambda) = [\text{ca}, \text{X}]$
2. $\delta(\text{ca}, b, \text{X}) = [\text{cb}, \text{X}]$
3. $\delta(\text{cb}, b, \text{X}) = [\text{cb}, \text{X}]$

* O APD M deve reconhecer a palavra vazia
* Deve-se contar o número de a's até chegar o primeiro b (empilhando um símbolo). Depois, deve-se contar o número de b's.
* A função de transição está representada graficamente na figura a seguir:

Exemplo 2

* APD para $a^n b^n$:



Exemplo 3

* Construir um APD que reconheça a seguinte linguagem, e simular o funcionamento do AP para algumas palavras

$$L(M) = \{\, w \in \{0,1\}^* \mid \text{o número de 0s em } w \text{ é igual ao de 1s}\}$$

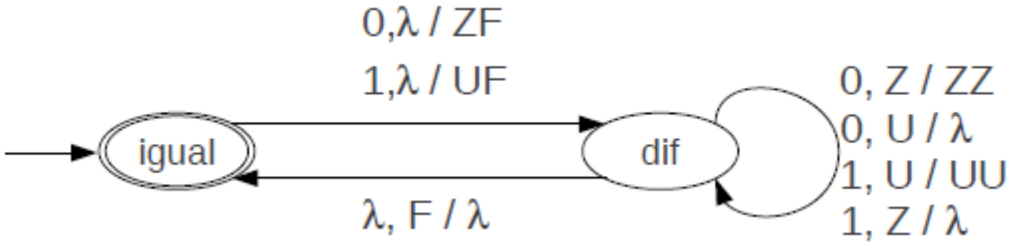
Exemplo 3

* Raciocínio
  * Sempre que se recebe um 0
    * Se o topo da pilha for Z, empilha ZZ:
    * Se o topo da pilha for U, não empilha nada.
  * Sempre que se recebe um 1
    * Se o topo da pilha for U, empilha UU:
    * Se o topo da pilha for Z, não empilha nada.
  * Ao fim da computação
    * A pilha terá Z^n se a palavra de entrada tiver *n* 0s a mais que 1s; ou
    * U^n se a palavra de entrada tiver *n* 1s a mais que 0s.
  * É necessário um símbolo para marcar que a pilha está vazia

Exemplo 3

- Construir um APD que reconheça a seguinte linguagem, e simular o funcionamento do AP para algumas palavras

$$L(M) = \{\, w \in \{0,1\}^* \mid \text{o número de 0s em } w \text{ é igual ao de 1s} \,\}$$



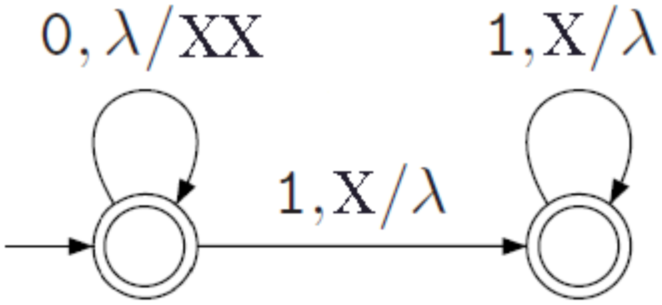
Dica: Lembre-se de quem está no topo da pilha.

Exercícios

- Construa APDs para as seguintes linguagens
- $\{ 0^n 1^{2n} \mid n >= 0 \}$
- $\{ w0w^r \mid w \in \{1,2\}^* \}$

Exercícios

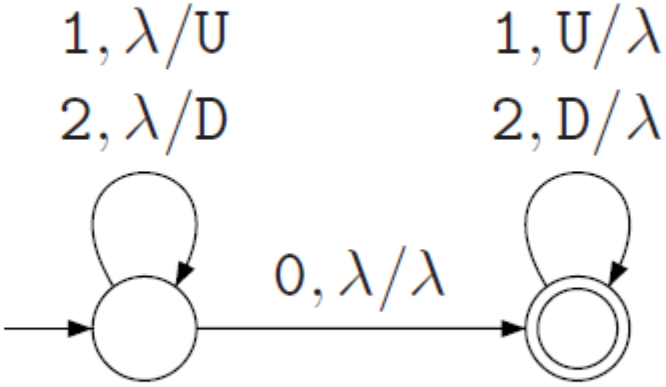
* $\{ 0^n 1^{2n} \mid n >= 0 \}$



* Exemplos de execução: 011, 001111, 000111111...

Exercícios

- $\{ w0w^r \mid w \in \{1,2\}^* \}$



- Exemplos de execução: 0, 101, 202, 11011, 12021…

Exercícios

* Resolução para $w0w^r$:
* LIFO: Last-in-first-out
* O último símbolo a entrar na pilha, deve ser o primeiro a sair dela:

w: 12 0 21

Passo 1) Empilha U [1]

Passo 2) Empilha D (que passa a ser o topo da pilha) [2]

(Empilhou DU, com D no topo)

Passo 3) Ocorre a transição sob 0 [0]

Passo 4) Desempilha D (Sai da pilha) [2]

Passo 5) Desempilha U (Sai da pilha, deixando-a vazia) [1]

Resposta: w reconhecida. É palíndromo.

O último símbolo a entrar na pilha (D) foi o primeiro a sair dela.

Obrigado.

joaopauloaramuni@gmail.com
joaopauloaramuni@fumec.br

UNIVERSIDADE FUMEC