Teoria da Computação

Aula Teórica 8:
Autómatos Finitos Deterministas

António Ravara

Departamento de Informática

1 de Abril de 2019

Estruturas para modelação de sistemas computacionais

"Tudo é um conjunto"

- A teoria de conjuntos permite modelar sistemas (computacionais)
  Ex: lâmpada, ficheiro, conta bancária, dropbox
- As estruturas obtidas são:
  - conjuntos de estados (os possíveis do sistema)
  - e de funções e/ou relações (transformações de estados).

Como definem um sistema?

- Estruturas são "declarativas" e "estáticas"
- Funções representam a evolução dos sistemas

Estruturas definem *especificações*

Declaram um conjunto dos estados e funções que os manipulam

De novo, a lâmpada

- O conjunto de estados: $SLAMP = \{0, 1\}$
- A função $turnOn \in SLAMP \to SLAMP$
  $turnOn \stackrel{\text{def}}{=} \{0 \mapsto 1, 1 \mapsto 1\}$
- A função $turnOff \in SLAMP \to SLAMP$
  $turnOff \stackrel{\text{def}}{=} \{0 \mapsto 0, 1 \mapsto 0\}$
- A estrutura que modela o sistema:
  $LAMP \stackrel{\text{def}}{=} (SLAMP, turnOn, turnOff)$

Como pode evoluir um sistema?

Resultado de aplicar sequência de funções tem que ser calculado

$$turnOn(turnOff(turnOn(turnOn(0)))) = ?$$

Efeito de executar sequência de funções dado por composição dessas funções:

$$\begin{aligned} turnOn(turnOff(turnOn(turnOn(0)))) &= \\ turnOn(turnOff(turnOn(1))) &= \\ turnOn(turnOff(1)) &= \\ turnOn(0) &= \\ 1 & \end{aligned}$$

Sequência "invertida".

Funções versus evoluções

Pretende-se um modelo centrado na evolução

Quer-se uma estrutura específica para:

- descrever sequências de execução;
- obter o estado em que fica o sistema após a execução de uma sequência;
- verificar se dada sequência de execução é ou não possível.

Autómatos finitos: um modelo "operacional" muito simples

- Definem-se os estados do sistema
- Define-se como passar (*transitar*) de um estado a outro

As funções são entendidas como *acções* (computacionais).

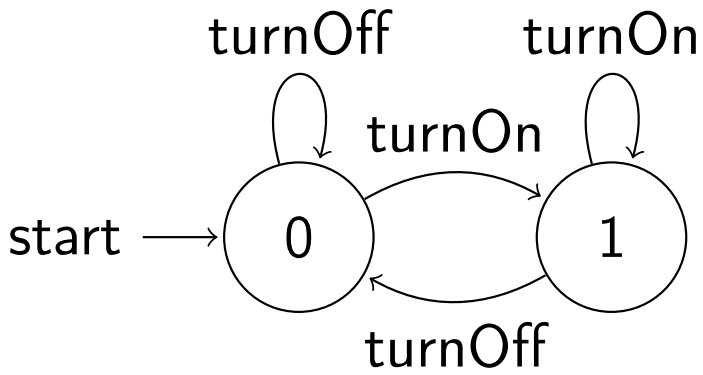
De novo a lâmpada

Sistema: $LAMP \stackrel{\text{def}}{=} (SLAMP, turnOn, turnOff)$

- O conjunto de estados: $SLAMP = \{0, 1\}$
- A função $turnOn \in SLAMP \to SLAMP$

$$turnOn \stackrel{\text{def}}{=} \{0 \mapsto 1, 1 \mapsto 1\}$$

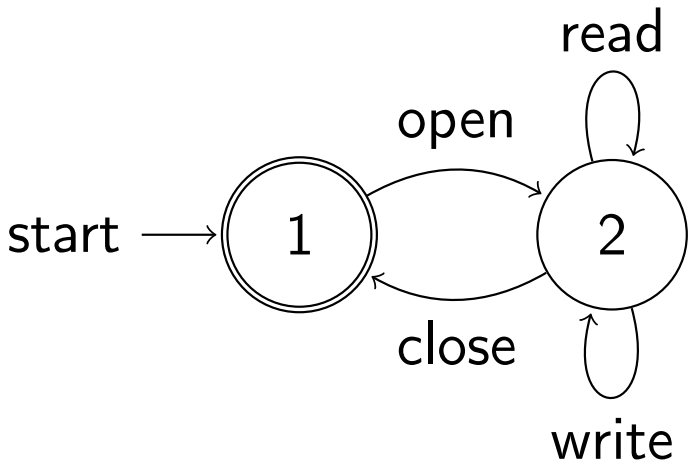
Autómato: mostra globalmente as transições do sistema



Autómatos finitos – componentes do modelo

- *Estados* (as circunferências) dos quais:
  - um é o inicial;
  - pode haver finais – duplas circunferências;
- *Transições* etiquetadas (setas com strings), que descrevem *acções* (funções parciais): transformações entre estados.

Exemplo: uso de um ficheiro

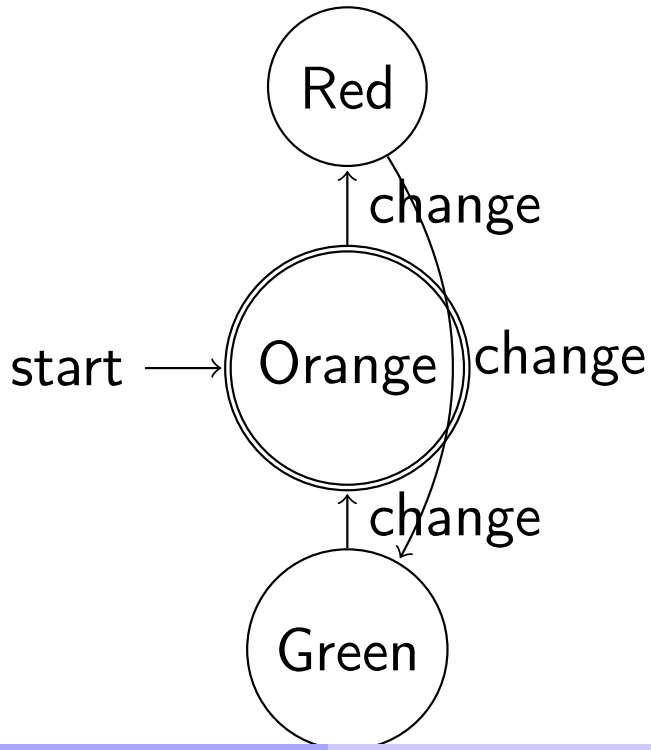


O semáforo: transição *vs.* acções como funções

Sistema

$Semaf = (\{orange, red, green\}, \{change\})$ com
$change = \{(orange, red), (red, green), (green, orange)\}$

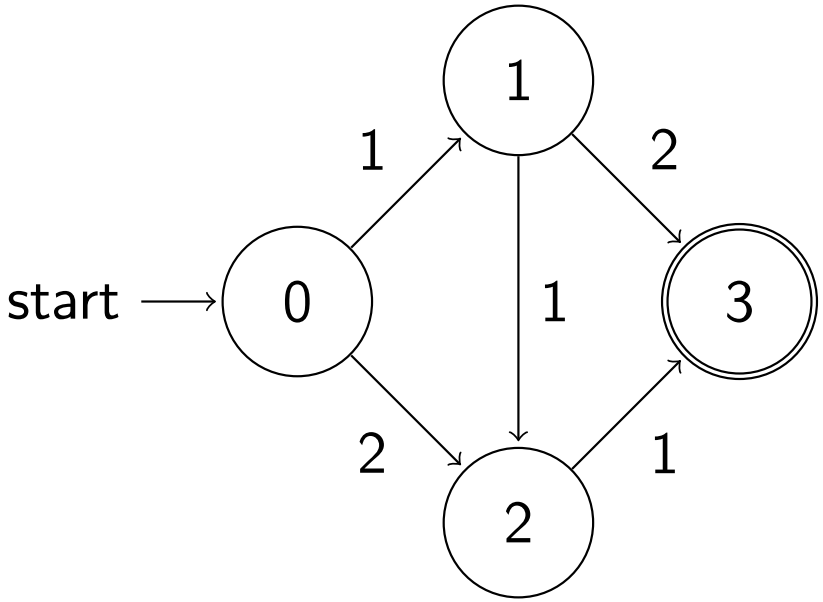
Autómato



Mais exemplos de autómatos finitos

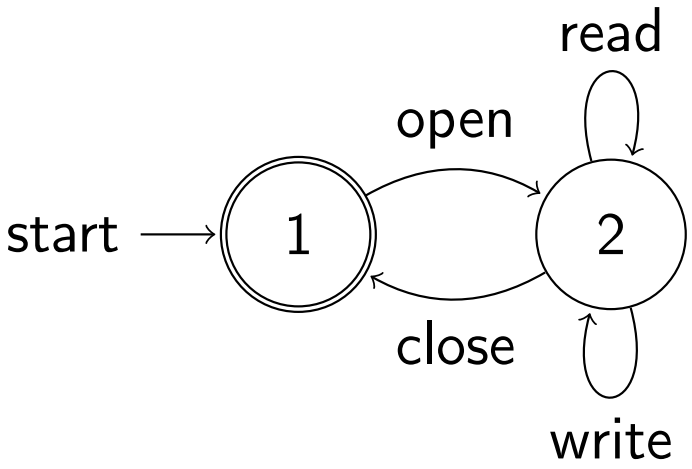
Máquina de chocolates

Aceita moedas de 1 e de 2 euros. Um chocolate custa 3 euros.



Podem haver vários caminhos.

Execuções dum sistema – de novo o ficheiro



Computação dum autómato

- Sequência de execução: uma "palavra" composta de acções `open read read write close`
- De 1, em que estado fica o sistema após a execução de `open`?
  No estado 2 (pode-se ler, escrever ou fechar).
- Sequência de execução `open open` é possível?
  No estado 2 não se pode fazer `open`.

Execuções dum autómato

Chama-se *palavra* ou *traço* a uma sequência de acções.

Computação por execução de "palavras"

- Uma computação começa num estado inicial.
- Uma computação termina num dos seus estados finais.

Computações dum autómato

Conjunto das palavra que podem ser "executadas" a partir do estado inicial e levam a um estado final.

Computações no ficheiro

- Sequência `open read close` é computação.
- Sequência `open read` não é computação.
- Sequência `open open` não é computação.

A matemática: definição rigorosa de autómato (como uma estrutura específica)

Autómato Finito Determinista

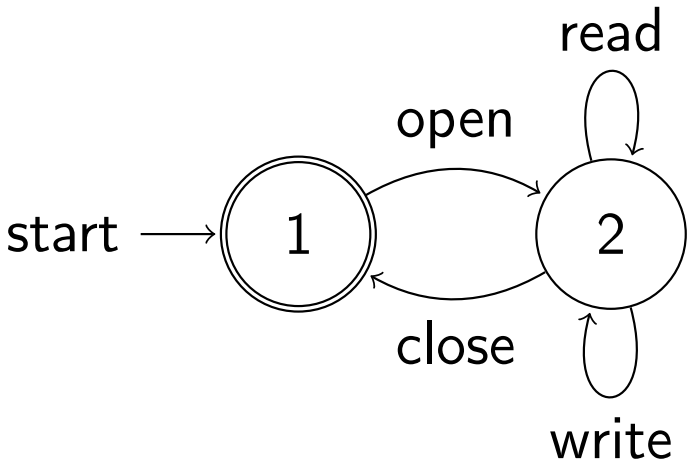
Um AFD é um quíntuplo $A = \langle S, \Sigma, s, \delta, F \rangle$, sendo:

- S um conjunto finito, dos *estados* de A.
- Σ um conjunto finito, das *acções* de A (o seu *alfabeto*).
- $s \in S$ é o *estado inicial* de A.
- $\delta \in S \times \Sigma \to S$ é a *função* (parcial) de *transição* de A.
- $F \subseteq S$ é o conjunto dos *estados finais* (ou de aceitação) de A.

- Considera-se normalmente S e Σ não vazios;
- O modelo é *determinista* porque δ é uma função (parcial: nem todas as acções se realizam em dado estado);
- Se F for vazio nenhuma computação termina.

O sistema AFILE

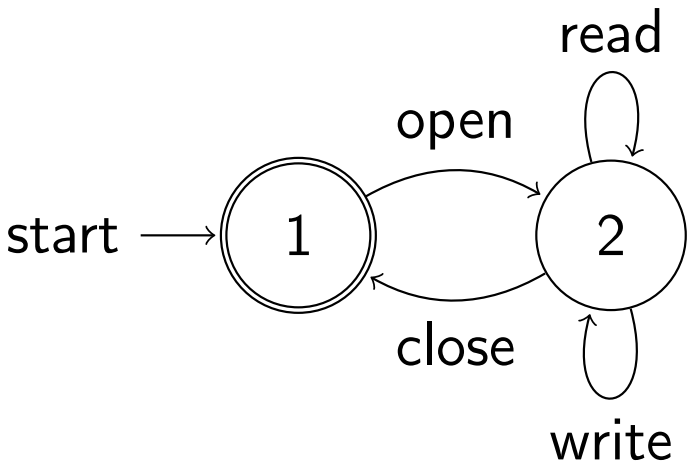
Informalmente



Formalmente, $A = \langle S, \Sigma, s, \delta, F \rangle$, sendo:

- $S = \{1, 2\}$
- $\Sigma = \{\texttt{open}, \texttt{close}, \texttt{read}, \texttt{write}\}$
- $s = 1$
- $F = \{1\}$

O sistema AFILE



δ	`open`	`close`	`read`	`write`
1	2			
2		1	2	2

ou seja, $\delta(1, \texttt{open}) = 2$, etc.

Sequências de elementos de um conjunto

- *Concatenação* de conjuntos:

$$A \cdot B \stackrel{\text{def}}{=} \{ab \mid a \in A \wedge b \in B\}$$

- Chama-se *palavra* a um elemento de um conjunto concatenação, e identifica-se normalmente pelo símbolo w.
- A palavra *vazia* (ϵ) é o elemento neutro da concatenação de palavras (à esquerda e à direita): $w\epsilon = w = \epsilon w$
- O conjunto das palavras de tamanho $n \in \mathbb{N}_0$ construidas com os símbolos de um conjunto A é definida indutivamente:
  - $A_0 \stackrel{\text{def}}{=} \{\epsilon\}$
  - $A_{n+1} \stackrel{\text{def}}{=} A_n \cdot A$
- O *fecho de Kleene* de um conjunto A é o conjunto de todas as sequências (finitas) de elementos de A:

$$A^* \stackrel{\text{def}}{=} \cup_{n \in \mathbb{N}_0} A_n$$

Configurações de um autómato

Intuições

- Chama-se *palavra* ou *traço* de um autómato a uma sequência de acções.
- Uma palavra *marcada* de um AFD é uma palavra do AFD com uma ocorrência do símbolo especial '|'.
  Exemplo: se ab for uma palavra, $|ab$, $a|b$ e $ab|$ são palavras marcadas.
- Uma *configuração* de um AFD é um par em que o primeiro elemento é uma palavra marcada e o segundo um estado.

Configurações de um autómato

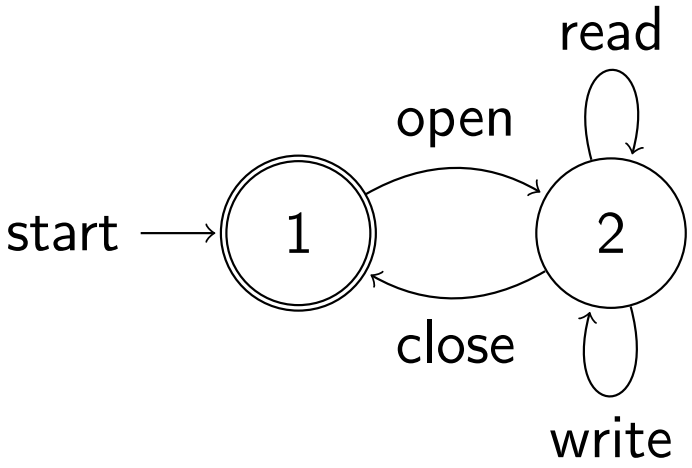
- A "marca" numa palavra distingue o que já se executou do que falta executar.
- Numa configuração, o estado indica onde se está após a execução da parte da palavra antes da marca.

Configurações do sistema AFILE

- Sequência `open read` é uma palavra do AFD:
  $\langle 1, |\, \texttt{open read}\rangle$ é configuração inicial,
  que por $\delta(1, \texttt{open}) = 2$ passa para $\langle 2, \texttt{open} \,|\, \texttt{read}\rangle$
  e por $\delta(2, \texttt{read}) = 2$ passa para $\langle 2, \texttt{open read}\,|\rangle$.
- Sequência `open open` não é uma palavra do AFD:
  $\langle 1, |\, \texttt{open open}\rangle$ é configuração inicial,
  que por $\delta(1, \texttt{open}) = 2$ passa para $\langle 2, \texttt{open} \,|\, \texttt{open}\rangle$
  mas agora $\delta(2, \texttt{open})$ não está definida!

Execução de palavras num autómato

Dado o sistema AFILE:



Exemplos de execução:

$$\langle 1, \mid \texttt{open read} \rangle \xrightarrow{\delta(1,\texttt{open})=2} \langle 2, \texttt{open} \mid \texttt{read} \rangle \xrightarrow{\delta(2,\texttt{read})=2} \langle 2, \texttt{open read} \mid \rangle$$

$$\langle 1, \mid \texttt{open open} \rangle \xrightarrow{\delta(1,\texttt{open})=2} \langle 2, \texttt{open} \mid \texttt{open} \rangle \not\rightarrow$$

Computações de um autómato

Computação de um AFD

Uma *computação* de um autómato finito determinista (AFD) é uma palavra que quando se começa a executar no estado inicial "leva" a um estado final.

Função de transição extendida, $\delta^* \in S \times \Sigma^* \to S$

Seja $\langle S, \Sigma, s, \delta, F \rangle$ um AFD; considere-se $a \in \Sigma$ e $w \in \Sigma^*$.
A função δ^* é definida indutivamente pelas seguintes regras:

- $\delta^*(t, \epsilon) = t$, para qualquer $t \in S$;
- $\delta^*(t, wa) = \delta(\delta^*(t, w), a)$, para qualquer $t \in S$ e $wa \in \Sigma^*$.

Linguagem de um autómato

Computação de um AFD

Dado um AFD $\langle S, \Sigma, s, \delta, F \rangle$, chama-se *computação* do autómato a uma palavra $w \in \Sigma^*$ tal que $\delta^*(s, w) = t$ e $t \in F$.

Uma computação diz-se também uma *palavra aceite* pelo autómato.

Linguagem de um AFD

Dado um AFD $A = \langle S, \Sigma, s, \delta, F \rangle$, chama-se *linguagem* do autómato, $\mathcal{L}(A)$ ao conjunto das palavras por ele aceites.

$$\{w \in \Sigma^* \mid \exists t \, (\delta^*(s, w) = t \wedge t \in F)\}$$

Exemplo de computação em AFILE

A sequência open read close é uma computação de AFILE:

$$
\begin{aligned}
\delta^*(1, \texttt{open read close}) &= \delta(\delta^*(1, \texttt{open read}), \texttt{close}) \\
&= \delta(\delta(\delta^*(1, \texttt{open}), \texttt{read}), \texttt{close}) \\
&= \delta(\delta(\delta(\delta^*(1, \epsilon), \texttt{open}), \texttt{read}), \texttt{close}) \\
&= \delta(\delta(\delta(1, \texttt{open}), \texttt{read}), \texttt{close}) \\
&= \delta(\delta(2, \texttt{read}), \texttt{close}) \\
&= \delta(2, \texttt{close}) \\
&= 1 \in F
\end{aligned}
$$

Mais exemplos em AFILE

open read e open open não são computações de AFILE:

$$\begin{aligned}
\delta^*(1, \texttt{open read}) &= \delta(\delta^*(1, \texttt{open}), \texttt{read}) \\
&= \delta(\delta(\delta^*(1, \epsilon), \texttt{open}), \texttt{read}) \\
&= \delta(\delta(1, \texttt{open}), \texttt{read}) \\
&= \delta(2, \texttt{read}) \\
&= 2 \notin F
\end{aligned}$$

$$\begin{aligned}
\delta^*(1, \texttt{open open}) &= \delta(\delta^*(1, \texttt{open}), \texttt{open}) \\
&= \delta(\delta(\delta^*(1, \epsilon), \texttt{open}), \texttt{open}) \\
&= \delta(\delta(1, \texttt{open}), \texttt{open}) \\
&= \delta(2, \texttt{open}) \\
&= \perp
\end{aligned}$$

Precisamos de um conjunto de estados finais?

- Precisamos de pelo menos um para que a linguagem do AFD não seja vazia.
- não basta ter um?
- Qual o AFD com acções $\{a, b\}$ que aceita palavras:
  - começadas por a e seguidas (só) por zero ou mais bs, ou
  - começadas por b e seguidas (só) por zero ou mais as?