Teoria da Computação

Aula Teórica 13:
Conversão de AFNs em AFDs

António Ravara

Departamento de Informática

17 de Abril de 2019

Relação entre Expressões Regulares e AFDs

Expressividade de um modelo computacional

Conjunto de programas que processa.

- Um programa implementa um algoritmo para calcular uma função.
- Considerando cada instrução for um símbolo, um programa é uma "palavra" e o conjunto de programas válidos que se podem escrever, uma "linguagem".

ERs vs. AFDs

L(ER) = L(AFD)

Vai-se mostrar que L(ER) $\subseteq$ L(AFD) e L(ER) $\supseteq$ L(AFD).

O resultado pretendido

Linguagens regulares $\subseteq$ Linguagens aceites por AFDs

Uma linguagem denotada por uma ER é linguagem de um AFD.

- A função $\mathit{Compile} \subseteq \mathit{RegExp}(\Sigma) \rightarrow \mathcal{AFN}_\Sigma$ fornece o resultado para a linguagem de um AFN.
- A linguagem de um AFN será "mais que regular"?
  Consegue-se representar com um AFN um sistema que não se consegue representar com um AFD?
- A diferença em termos de palavras processadas por AFNs e AFDs são as transições-ϵ.
  Mas o ϵ é o elemento neutro da concatenação de palavras...
- O conjunto das linguagens aceites por AFNs deve coincidir com o das linguagens aceites por AFDs.

Linguagens de $\mathcal{AFN}_\Sigma$ e $\mathcal{AFD}_\Sigma$ coincidem?

- Obviamente, qualquer AFD é também um AFN, ou seja, $\mathcal{AFD}_\Sigma \subseteq \mathcal{AFN}_\Sigma$.
  Logo qualquer linguagem aceite por um AFD é também uma linguagem aceite por algum AFN.
- O desafio é provar a inclusão inversa:
  qualquer linguagem aceite por um AFN é também uma linguagem aceite por algum AFD.
- É necessário ver se há uma forma de converter AFNs em AFDs.

Linguagens de $\mathcal{AFN}_\Sigma$ são linguagens de $\mathcal{AFD}_\Sigma$?

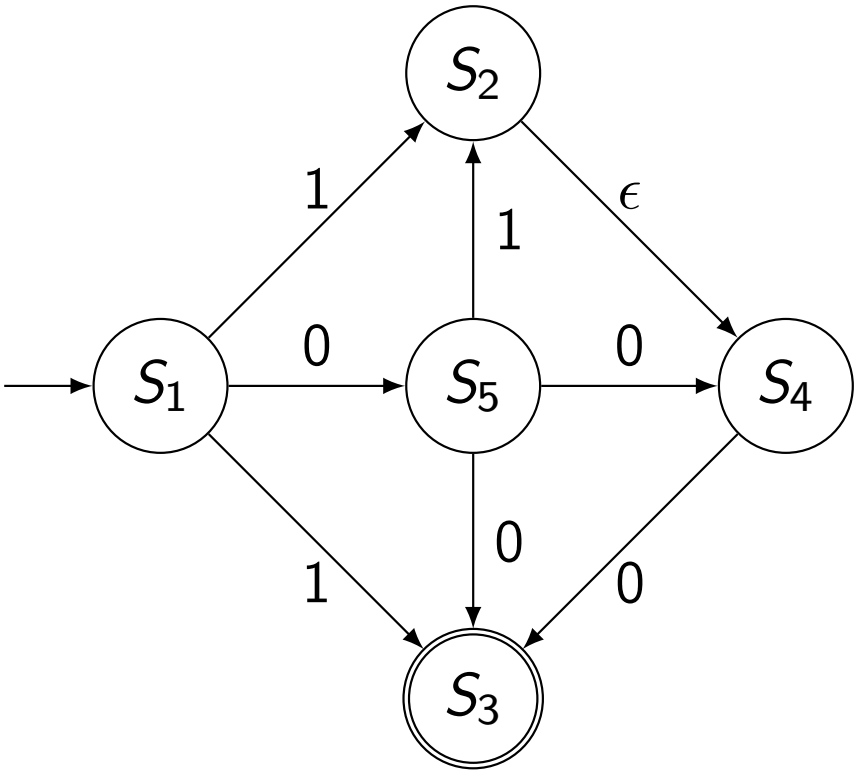
Como proceder?

Para ver se dada palavra w é aceite num AFN pode-se explorar simultaneamente todos os possíveis caminhos, em paralelo:

- Parte-se do símbolo inicial da palavra e do estado inicial do autómato, e regista-se o conjunto de estados para onde se pode transitar pela mesma acção.
  sendo $w = a\,w'$ então $\langle s, |a\,w'\rangle \longrightarrow \langle T, a \,|\, w'\rangle$ com
  $T = \{t \in S \,|\, \langle s, |a\,w'\rangle \longrightarrow \langle t, a \,|\, w'\rangle\}$
- Itera-se o processo para cada estado no conjunto obtido, consumindo os símbolos da palavra.
- Os conjuntos de estados são tratados como estados "únicos" – o nome do estado é o conjunto.

Processamento em "paralelo" de uma palavra num AFN

Um exemplo: o AFN BOB



BOB aceita 100 e 010?

Processamento de 100 no AFN BOB

- $\langle S_1, |100\rangle \xrightarrow{1} \langle S_2, 1|00\rangle$ porque $(S_1, 1, S_2) \in \Delta$
- $\langle S_1, |100\rangle \xrightarrow{1} \langle S_3, 1|00\rangle$ porque $(S_1, 1, S_3) \in \Delta$
- $\langle S_1, |100\rangle \xrightarrow{1} \langle S_4, 1|00\rangle$ porque $1 = 1\,\epsilon$, $(S_1, 1, S_2) \in \Delta$ e $(S_2, \epsilon, S_4) \in \Delta$

Logo, $\langle \{S_1\}, |100\rangle \xrightarrow{1} \langle \{S_2, S_3, S_4\}, 1|00\rangle$

- $\langle S_2, 1|00,\rangle \xrightarrow{0}\!\!\!\!\!\!/\;\;$ porque não há transições por 0 de S_2
- $\langle S_3, 1|00\rangle \xrightarrow{0}\!\!\!\!\!\!/\;\;$ porque não há transições (por 0) de S_3
- $\langle S_4, 1|00\rangle \xrightarrow{0} \langle S_3, 10|0\rangle$ porque $(S_4, 0, S_3) \in \Delta$

Logo, $\langle \{S_2, S_3, S_4\}, 1|00\rangle \xrightarrow{0} \langle \{S_3\}, 10|0\rangle$.

Como S_3 não tem transições, a palavra não é aceite.

Processamento de 010 no AFN BOB

- $\langle S_1, |010\rangle \xrightarrow{0} \langle S_5, 0|10\rangle$ porque $(S_1, 0, S_5) \in \Delta$

Logo, $\langle \{S_1\}, |010\rangle \xrightarrow{0} \langle \{S_5\}0|10\rangle$

- $\langle S_5, 0|10\rangle \xrightarrow{1} \langle S_2, 01|0\rangle$ porque $(S_5, 1, S_2) \in \Delta$
- $\langle S_5, 0|10\rangle \xrightarrow{1} \langle S_4, 01|0\rangle$ porque $1 = 1\,\epsilon$, $(S_5, 1, S_2) \in \Delta$ e $(S_2, \epsilon, S_4) \in \Delta$

Logo, $\langle \{S_5\}, 0|10\rangle \xrightarrow{1} \langle \{S_2, S_4\}, 01|0\rangle$.

- $\langle S_2, 01|0\rangle \not\xrightarrow{0}$ porque não há transições por 0 de S_2
- $\langle S_4, 01|0\rangle \xrightarrow{0} \langle S_3, 010|\rangle$ porque $(S_4, 0, S_3) \in \Delta$

Logo, $\langle \{S_2, S_4\}, 01|0\rangle \xrightarrow{0} \langle \{S_3\}, 010|\rangle$.

Como S_3 é estado final, a palavra é aceite.

Como se procedeu?

- Exploraram-se todas as possíveis transições não deterministas *em paralelo*.
- Registaram-se os *conjuntos de estado acessíveis* por dada acção, em vez estados individuais (como na relação de transição).
- Ao definir as transições entre conjuntos de estados, *recuperou-se* o comportamento determinista (a relação de transição é de novo uma função).

Como proceder então para obter o AFD dBOB, equivalente ao AFN BOB?

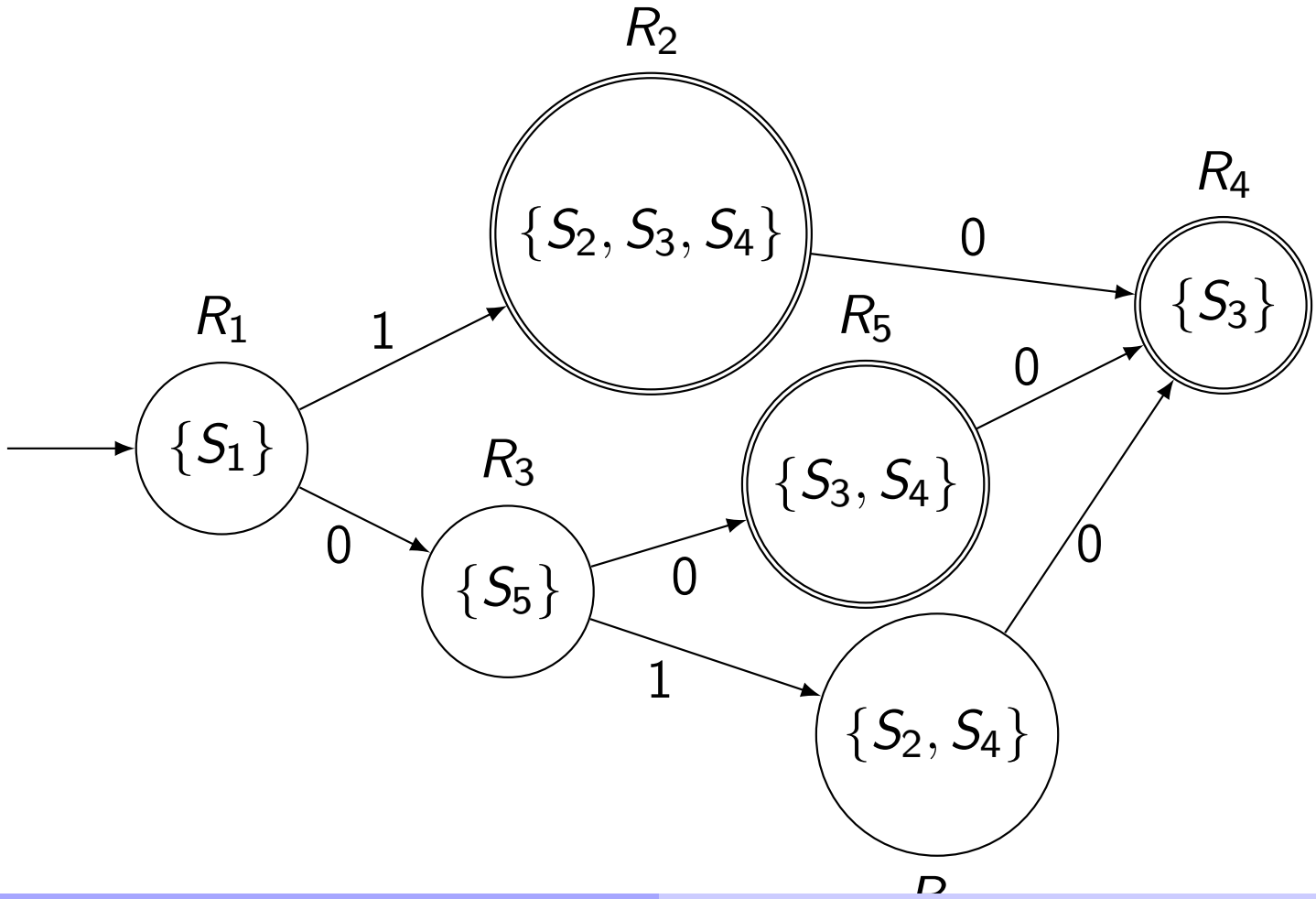
"Determinização" de BOB - dBOB

Os estados de dBOB são conjuntos de estados de BOB.

- O estado inicial de dBOB é um conjunto com os estados de BOB acessíveis do seu estado inicial por transições-ϵ.
  Então seja $R_1 = \{S_1\}$ o estado inicial.
- Que transições partem de R_1?
  Por 1 chega-se a $R_2 = \{S_2, S_3, S_4\}$. Logo, $\delta_{dBOB}(R_1, 1) = R_2$.
  Por 0 chega-se a $R_3 = \{S_5\}$. Logo $\delta_{dBOB}(R_1, 0) = R_3$.
- Que transições partem de R_2?
  Por 0 chega-se a $R_4 = \{S_3\}$. Logo, $\delta_{dBOB}(R_2, 0) = R_4$.
  De S_2, S_3 e S_4 não há transições por 1 — o conjunto de estados acessíveis é o vazio. Fica $\delta_{dBOB}(R_2, 1)$ indefinido.
- Que transições partem de R_3?
  Por 0 chega-se a $R_5 = \{S_3, S_4\}$. Logo, $\delta_{dBOB}(R_3, 0) = R_5$.
  Por 1 chega-se a $R_6 = \{S_2, S_4\}$. Logo,$\delta_{dBOB}(R_3, 1) = R_6$.
- Analisam-se igualmente as transições de R_5 e de R_6.

O AFD dBOB, equivalente ao AFN BOB

Os estados finais de dBOB são os que contém algum estado final de BOB (quando algum é alcançado, termina-se o processamento).



A construção de "determinização"

- O processo visa obter um AFD de um AFN, equivalente no sentido que reconhece a mesma linguagem.
- A conversão chama-se "Rabin-Scott powerset construction" e foi proposta em 1959.
- Usa duas noções auxiliares:
  - Fecho-ϵ de um conjunto de estados ($U \subseteq S_A$) de um AFN A: conjunto de estados acessíveis dos estados em U por transições-ϵ.

$$\begin{array}{lcl} \mathit{closeempty} & \subseteq & \wp(S) \to \wp(S) \\ \mathit{closeempty} & = & \{U \mapsto V \mid V = \{s \mid \exists t.t \in U \wedge (t, \epsilon, s) \in \Delta_A^*\}\} \end{array}$$

  - Fecho-a de um conjunto de estados ($U \subseteq S_A$) de um AFN A_Σ, com $a \in \Sigma$: conjunto de estados acessíveis dos estados em U por transições-a.

$$\begin{array}{lcl} \mathit{move} & \subseteq & \wp(S) \times \Sigma \to \wp(S) \\ \mathit{move} & = & \{U \times a \mapsto V \mid V = \{s \mid \exists t.t \in U \wedge (t, a, s) \in \Delta_A\}\} \end{array}$$

Rabin-Scott Powerset Construction

Dado um $A = \langle S_A, \Sigma_A, s_A, \Delta_A, F_A \rangle$ obtém-se um AFD $D = \langle S_A, \Sigma_D, s_D, \delta_D, F_D \rangle$ da seguinte forma:

- $S_D = \wp(S_A)$.
  Os estados de D são subconjuntos de estados de A (mas basta considerar os acessíveis do estado inicial).
- $\Sigma_D = \Sigma_A$
- $s_D = \mathit{closeempty}(\{s_A\})$
- $\delta_D = \{(s, a) \mapsto s' \in S_D \times \Sigma_D \times S_D \mid s' = \mathit{closeempty}(\mathit{move}(\{s\}, a))\}$
- $F_D = \{s \in S_D \mid s \cap F_A \neq \emptyset\}$
  Os estados finais de D são os que contêm pelo menos um estado final de A.

Conversão de AFNs em AFDs

- É fácil mostrar que um AFD D que resulta da conversão de um AFN A reconhece exactamente a linguagem de A.
  Prova-se que A e D se simulam mutuamente para cada possível palavra.
- A construção gera um AFD potencialmente com um grande número de estados:
  O número de estados de um AFD D obtido por conversão de um AFN A é exponencialmente maior: $\#S_D = 2^{\#S_A}$.
- Por um lado, um AFN expressa uma linguagem regular de forma muito compacta, sendo um bom mecanismo de especificação.
- Um AFN não é prático de implementar, mas um AFD, que se implementa facilmente, pode usar muito mais memória.