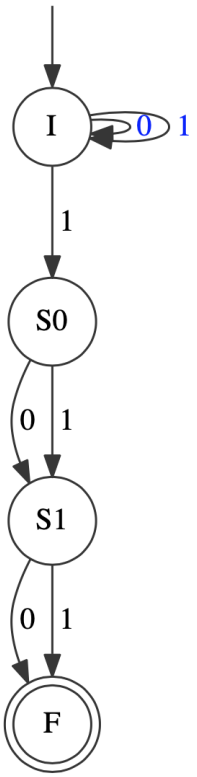
3. Autómatos Não-Determinístic

Copiar link

Seja $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, e seja $\mathcal{P}(Q)$ o conjunto de todos os subconjuntos de Q (i.e. um conjunto de conjuntos, conhecido como as *partes* ou "powerset" de Q, contendo $\emptyset$, o próprio Q, e todos os outros subconjuntos de Q)

Um **Autómato Não-Determinístico de Estados Finitos** (NFA) é um tuplo $(Q, \Sigma, \delta, Q_0, F)$ em que:

- Q é um conjunto *finito* e *não vazio* de estados
- Σ é um alfabeto (igualmente *finito* e *não vazio*)
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ é uma função de transição (total, ou seja, está definida para todos os *inputs*)
- $Q_0 \subseteq Q$ é um conjunto de estados iniciais do autómato
- $F \subseteq Q$ é um conjunto *finito* e *possivelmente vazio* de estados finais, também conhecidos por *estados de aceitação*



Ao contrário dos DFA, num NFA define-se um conjunto possível de estados para os quais o autómato pode transitar a partir de um dado estado, lendo um determinado símbolo. Em cada instante ele encontra-se num (e um só) dos seus estados! Não existem vários estados simultâneos ativos.

Por exemplo neste autómato, logo a partir do estado inicial I, ao ser lido um 1 o autómato pode transitar quer para o estado S0 quer para o mesmo estado I. A escolha é não-determinística, ou seja, em cada execução do autómato, sendo lido o mesmo símbolo num determinado estado, podem ocorrer transições diferentes.

Havendo mais do que um estado inicial, cada execução do autómato poder iniciar-se num estado diferente, sendo a escolha, novamente, não-determinística.

Considera-se habitualmente que pode também haver transições expontâneas, que não consomem símbolos, o que justifica a utilização de Σ_ε na definição.

O autómato representado acima aceita a palavra 100, porque existe uma execução que termina no estado F e lê os símbolos 1, 0, 0. Aceita também por exemplo 111, 0101, ou 10110. Mas não aceita a palavra 1000, porque não existe nenhuma execução que acabe em F e leia essa palavra.

Exercício:

1. Qual é a linguagem reconhecida pelo autómato representado em cima?
2. O que se pode concluir sobre vantagens de utilização de não-determinismo?

Interpretação Alternativa da Execução de um Autómato ND

Existe uma visão alternativa, equivalente, da execução destes autómatos, que tem algumas vantagens do ponto de vista formal. Nesta visão, em cada execução pode haver em cada momento mais do que um estado ativo: imagine-se que existe um *token* que viaja entre estados através das transições, e que se **divide** quando existem várias transições possíveis. Os estados activos são aqueles onde se encontra um *token.*

Por exemplo, a execução do autómato acima para a palavra 100 será a seguinte, em que se escreve o conjunto de estados activos em cada instante:

$$\{I\} \to 1 \to \{I, S0\} \to 0 \to \{I, S1\} \to 0 \to \{I, F\}$$

Existe um *token* que permanece no estado inicial I, enquanto um outro avança até ao estado F. Cada um deles corresponde a uma execução na visão inicial apresentada.

Vejamos mais exemplos. A palavra 111 leva à seguinte execução:

$$\{I\} \to 1 \to \{I, S0\} \to 1 \to \{I, S0, S1\} \to 1 \to \{I, S0, S1, F\}$$

E quanto à palavra 1000:

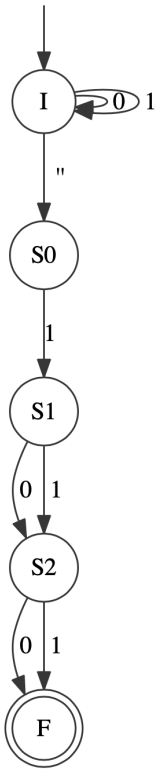
$$\{I\} \to 1 \to \{I, S0\} \to 0 \to \{I, S1\} \to 0 \to \{I, F\} \to 0 \to \{I\}$$

Note-se que neste último exemplo o estado F desaparece do conjunto de estados activos na última transição, porque o símbolo 0 não pode ser lido naquele estado. É como se o *token* que se encontrava naquele estado "saísse" do autómato.

Como identificar as duas primeiras palavras como aceites pelo autómato, e a terceira como rejeitada? É simples: nas duas primeiras o conjunto de estados obtidos depois da leitura da palavra contém um estado final (neste exemplo é único, mas em geral, se houvesse mais do que um, bastaria que o conjunto contivesse um deles).
No caso da leitura de 1000, o conjunto $\{I\}$ não contém o estado F.

Transições ε

Os autómatos ND podem ter transições associadas à leitura da palavra ε, ou seja, de nenhum símbolo. Trata-se de transições espontâneas, que podem ocorrer em qualquer estado activo.



Estas transições não são necessárias, mas permitem por vezes estruturar os autómatos de forma mais inteligível.

Por exemplo, o NFA anterior poderia ser substituído por este, equivalente, com um estado adicional, em que as duas fases (leitura de um prefixo qualquer, seguida de leitura de um 1 e dois outros símbolos quaisquer) se encontram separadas, e a transição ε separa as duas.

Sempre que existe um token no estado I, esse token espontaneamente divide-se em dois, transitando um dos tokens para o estado $S0$. Ou seja, sempre que I está activo, $S0$ também está.

As palavras 100, 111, e 1000 levam agora às seguintes execuções:

100:
$\{I, S0\} \to 1 \to \{I, S0, S1\} \to 0 \to \{I, S0, S2\} \to 0 \to \{I, S0, F\}$

111:
$\{I, S0\} \to 1 \to \{I, S0, S1\} \to 1 \to \{I, S0, S1, S2\} \to 1 \to \{I, S0, S1, S2, F\}$

1000:
$\{I, S0\} \to 1 \to \{I, S0, S1\} \to 0 \to \{I, S0, S2\} \to 0 \to \{I, S0, F\} \to 0 \to \{I, S0\}$

Linguagem de um Autómato Não-Determinístico

Seja $D = (Q, \Sigma, \delta, Q_0, F)$ um NFA.

Designemos por $\text{close}_\varepsilon(q)$ o conjunto de estados que são alcançados a partir do estado q realizando-se apenas transições ε (possivelmente uma sequência destas transições). Definimos então, para um conjunto de estados,

$$\text{close}_\varepsilon(Q) = \bigcup_{q \in Q} \text{close}_\varepsilon(q)$$

Por forma a definir a linguagem reconhecida por um NFA generalizamos a função δ para receber um *conjunto* de estados (manteremos o nome da função):

$$\delta : \mathcal{P}(Q) \times \Sigma_\varepsilon \to \mathcal{P}(Q)$$

$$\delta(Q, x) = \bigcup_{q \in Q} \delta(q, x)$$

e definimos depois a função $\hat{\delta}$ que recebe um conjunto de estados e uma palavra, em vez de um símbolo, e calcula o conjunto de estados em que a execução do autómato se encontra depois de consumir todos os símbolos da palavra, a partir de um conjunto inicial de estados.

$$\hat{\delta} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$$

$$\begin{aligned} \hat{\delta}(Q, \varepsilon) &= \text{close}_\varepsilon(Q) \\ \hat{\delta}(Q, xs) &= \hat{\delta}(Q', s), \qquad \text{com} \qquad Q' = \text{close}_\varepsilon(\delta(\text{close}_\varepsilon(Q), x)) \end{aligned}$$

O NFA N aceita a palavra $s \in \Sigma^*$ sse o conjunto $\hat{\delta}(Q_0, s)$ contém algum (pelo menos um!) estado final.
A linguagem L_N reconhecida por N é definida como

$$L_N = \{s \mid s \in \Sigma^* \wedge \hat{\delta}(Q_0, s) \cap F \neq \emptyset\}$$

Teorema

Seja L uma linguagem qualquer. Então existe um NFA que reconhece L se e só se existe um DFA que reconhece L.

Prova:
Basta mostrar que:

1. Para todo o DFA D, existe um NFA que reconhece a mesma linguagem que D.
   Trivial: um DFA é um caso particular de NFA, logo D é um NFA

2. Para todo o NFA N, existe um DFA D que reconhece a mesma linguagem que N.
   É possível construir D a partir de N, tendo em conta que as transições de N são vistas como sendo entre conjuntos de estados:
   - Os estados de D serão *conjuntos de estados* de N, fechados por close_ε, que em alguma execução de N estão simultaneamente activos (i.e. têm um token).
   - O estado inicial de D será dado pelo conjunto de estados iniciais de N
   - Um estado de D será final se, visto como conjunto de estados de N, contiver pelo menos um estado final de N
   - As transições de D são identificadas como nas simulações acima.