4.1.2
Extending the transition function to strings

Extending the transition function to strings

1. NFA $N = (Q, \Sigma, \delta, s, A)$
2. $\delta(q, a)$: set of states that N can go to from q on reading $a \in \Sigma \cup \{\varepsilon\}$.
3. Want transition function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$
4. $\delta^*(q, w)$: set of states reachable on input w starting in state q.

Extending the transition function to strings

1. NFA $N = (Q, \Sigma, \delta, s, A)$
2. $\delta(q, a)$: set of states that N can go to from q on reading $a \in \Sigma \cup \{\varepsilon\}$.
3. Want transition function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$
4. $\delta^*(q, w)$: set of states reachable on input w starting in state q.

Extending the transition function to strings

1. NFA $N = (Q, \Sigma, \delta, s, A)$
2. $\delta(q, a)$: set of states that N can go to from q on reading $a \in \Sigma \cup \{\varepsilon\}$.
3. Want transition function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$
4. $\delta^*(q, w)$: set of states reachable on input w starting in state q.

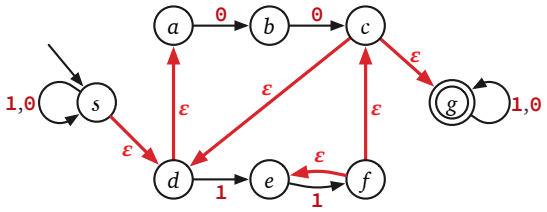
Extending the transition function to strings

1. NFA $N = (Q, \Sigma, \delta, s, A)$
2. $\delta(q, a)$: set of states that N can go to from q on reading $a \in \Sigma \cup \{\varepsilon\}$.
3. Want transition function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$
4. $\delta^*(q, w)$: set of states reachable on input w starting in state q.

Extending the transition function to strings

Definition

For NFA $N = (Q, \Sigma, \delta, s, A)$ and $q \in Q$ the ϵreach(q) is the set of all states that q can reach using only ε-transitions.



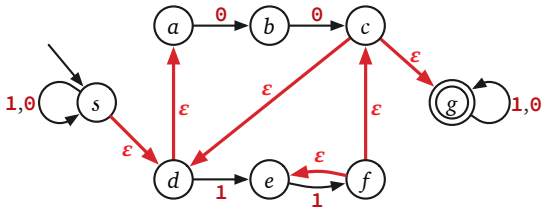
Definition

For $X \subseteq Q$: $\epsilon\text{reach}(X) = \bigcup_{x \in X} \epsilon\text{reach}(x)$.

Extending the transition function to strings

Definition

For NFA $N = (Q, \Sigma, \delta, s, A)$ and $q \in Q$ the ϵreach(q) is the set of all states that q can reach using only ε-transitions.



Definition

For $X \subseteq Q$: $\epsilon\text{reach}(X) = \bigcup_{x \in X} \epsilon\text{reach}(x)$.

Extending the transition function to strings

$\epsilon\text{reach}(q)$: set of all states that q can reach using only ε-transitions.

Definition

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

- if $w = \varepsilon$, $\delta^*(q, w) = \epsilon\text{reach}(q)$
- if $w = a$ where $a \in \Sigma$: $\delta^*(q, a) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \delta(p, a)\right)$
- if $w = ax$: $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$

Extending the transition function to strings

$\epsilon\text{reach}(q)$: set of all states that q can reach using only ε-transitions.

Definition

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

- if $w = \varepsilon$, $\delta^*(q, w) = \epsilon\text{reach}(q)$
- if $w = a$ where $a \in \Sigma$: $\delta^*(q, a) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \delta(p, a)\right)$
- if $w = ax$: $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$

Extending the transition function to strings

$\epsilon\text{reach}(q)$: set of all states that q can reach using only ε-transitions.

Definition

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

- if $w = \varepsilon$, $\delta^*(q, w) = \epsilon\text{reach}(q)$
- if $w = a$ where $a \in \Sigma$: $\delta^*(q, a) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \delta(p, a)\right)$
- if $w = ax$: $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$

Transition for strings: w = ax

Translation...

$$\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$$

1. $R = \epsilon\text{reach}(q) \implies \delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in R} \bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)$
2. $N = \bigcup_{p \in R} \delta^*(p, a)$: All the states reachable from q with the letter a.
3. $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{r \in N} \delta^*(r, x)\right)$

Transition for strings: w = ax

Translation...

$$\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$$

1. $R = \epsilon\text{reach}(q) \implies \delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in R} \bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)$
2. $N = \bigcup_{p \in R} \delta^*(p, a)$: All the states reachable from q with the letter a.
3. $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{r \in N} \delta^*(r, x)\right)$

Transition for strings: w = ax

Translation...

$$\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$$

1. $R = \epsilon\text{reach}(q) \implies \delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in R} \bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)$
2. $N = \bigcup_{p \in R} \delta^*(p, a)$: All the states reachable from q with the letter a.
3. $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{r \in N} \delta^*(r, x)\right)$

Transition for strings: w = ax

Translation...

$$\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in \epsilon\text{reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$$

1. $R = \epsilon\text{reach}(q) \implies \delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{p \in R} \bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)$
2. $N = \bigcup_{p \in R} \delta^*(p, a)$: All the states reachable from q with the letter a.
3. $\delta^*(q, w) = \epsilon\text{reach}\left(\bigcup_{r \in N} \delta^*(r, x)\right)$

Formal definition of language accepted by N

Definition

A string w is accepted by NFA N if $\delta_N^*(s, w) \cap A \neq \emptyset$.

Definition

The language $L(N)$ accepted by a NFA $N = (Q, \Sigma, \delta, s, A)$ is

$$\{w \in \Sigma^* \mid \delta^*(s, w) \cap A \neq \emptyset\}.$$

Important: Formal definition of the language of NFA above uses δ^* and not δ. As such, one does not need to include ε-transitions closure when specifying δ, since δ^* takes care of that.

Formal definition of language accepted by N

Definition

A string w is accepted by NFA N if $\delta_N^*(s, w) \cap A \neq \emptyset$.

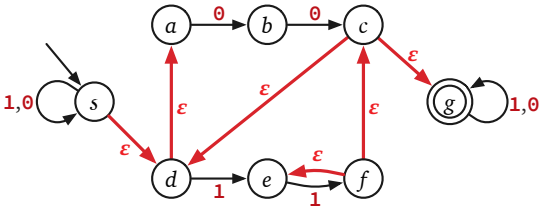
Definition

The language $L(N)$ accepted by a NFA $N = (Q, \Sigma, \delta, s, A)$ is

$$\{w \in \Sigma^* \mid \delta^*(s, w) \cap A \neq \emptyset\}.$$

Important: Formal definition of the language of NFA above uses δ^* and not δ. As such, one does not need to include ε-transitions closure when specifying δ, since δ^* takes care of that.

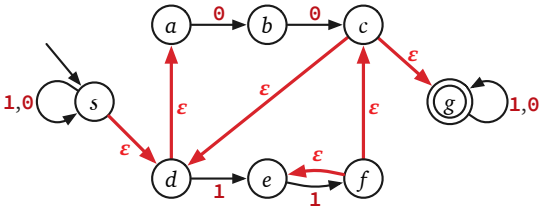
Example



What is:

- $\delta^*(s, \epsilon)$
- $\delta^*(s, 0)$
- $\delta^*(c, 0)$
- $\delta^*(b, 00)$

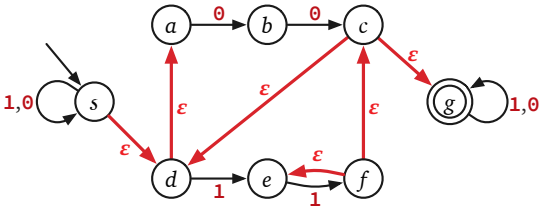
Example



What is:

- $\delta^*(s, \epsilon)$
- $\delta^*(s, 0)$
- $\delta^*(c, 0)$
- $\delta^*(b, 00)$

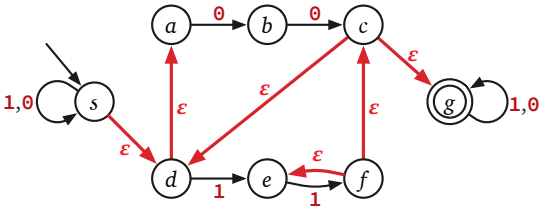
Example



What is:

- $\delta^*(s, \epsilon)$
- $\delta^*(s, 0)$
- $\delta^*(c, 0)$
- $\delta^*(b, 00)$

Example



What is:

- $\delta^*(s, \epsilon)$
- $\delta^*(s, 0)$
- $\delta^*(c, 0)$
- $\delta^*(b, 00)$

Another definition of computation

Definition

$q \xrightarrow{w}_N p$: State p of NFA N is **reachable** from q on $w \iff$ there exists a sequence of states $r_0, r_1, \ldots, r_k$ and a sequence $x_1, x_2, \ldots, x_k$ where $x_i \in \Sigma \cup \{\varepsilon\}$, for each i, such that:

- $r_0 = q$,
- for each i, $r_{i+1} \in \delta^*(r_i, x_{i+1})$,
- $r_k = p$, and
- $w = x_1 x_2 x_3 \cdots x_k$.

Definition

$$\delta_N^*(q, w) = \left\{ p \in Q \;\middle|\; q \xrightarrow{w}_N p \right\}.$$

Why non-determinism?

- Non-determinism adds power to the model; richer programming language and hence (much) easier to "design" programs
- Fundamental in **theory** to prove many theorems
- Very important in **practice** directly and indirectly
- Many deep connections to various fields in Computer Science and Mathematics

Many interpretations of non-determinism. Hard to understand at the outset. Get used to it and then you will appreciate it slowly.

THE END

...

(for now)