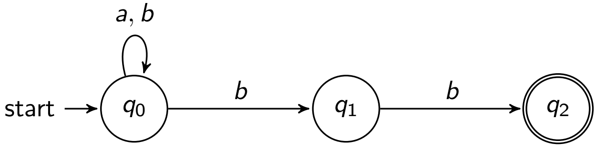
Non-deterministic Finite Automata

Outline

Non-deterministic Finite Automata (NFA)

Regular Languages, NFAs and DFAs

Non-deterministic finite automaton (NFA)



$\delta(q, a)$ is not one state, but a set of states.

δ	a	b
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

Non-deterministic Finite Automata (NFA)

M is a NFA over Σ if $M = (Q, q_0, \delta, F)$ with

Q	is a finite set of states
$q_0 \in Q$	is the initial state
$F \subseteq Q$	is a finite set of final states
$\delta : Q \times \Sigma \to \mathcal{P}(Q)$	is the transition function
	[$\mathcal{P}(Q)$ denotes the collection of subsets of Q]

Reading function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$ (multi-step transition)

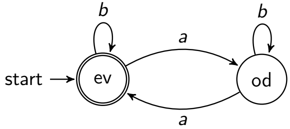
$$\begin{aligned} \delta^*(q, \lambda) &= \{q\} \\ \delta^*(q, aw) &= \{q' \mid q' \in \delta^*(p, w) \text{ for some } p \in \delta(q, a)\} \\ &= \bigcup_{p \in \delta(q,a)} \delta^*(p, w) \end{aligned}$$

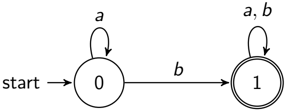
The language accepted by M, notation $\mathcal{L}(M)$, is:

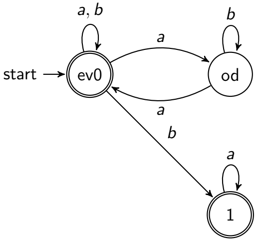
$$\mathcal{L}(M) = \{w \in \Sigma^* \mid \exists q \in \delta^*(q_0, w) \text{ such that } q \in F\}$$

Union of languages of NFAs

Example Suppose we want to have an NFA for $L_1 \cup L_2 = \{w \mid |w|_a \text{ is even or } |w|_b \geq 1\}$
First idea: put the two machines "non-deterministically in parallel"



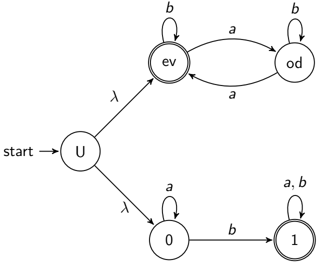




But this is wrong: The NFA accepts *aaa*.

NFAs with silent steps: NFA-λ

We add λ-transitions or 'silent steps' to NFAs
The correct union of M_1 and M_2 is:



In an NFA-λ we allow

$$\delta(q, \lambda) = q'$$

for $q \neq q'$. That means

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$$

NFA-λ (definition)

M is an NFA-λ over Σ if $M = (Q, q_0, \delta, F)$ with

Q	is a finite set of states
$q_0 \in Q$	is the initial state
$F \subseteq Q$	is a finite set of final states
$\delta : Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$	is the transition function

The λ-closure of a state q, $\lambda\text{-closure}(q)$, is the set of states reachable with only λ-steps.

Reading function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$ (multi-step transition)

$$\begin{aligned}
\delta^*(q, \lambda) &= \lambda\text{-closure}(q) \\
\delta^*(q, aw) &= \{q' \mid \exists p \in \lambda\text{-closure}(q)\, \exists r \in \delta(p, a)\, (q' \in \delta^*(r, w))\} \\
&= \bigcup_{p \in \lambda\text{-closure}(q)} \; \bigcup_{r \in \delta(p,a)} \delta^*(r, w)
\end{aligned}$$

The language accepted by M, notation $\mathcal{L}(M)$, is:

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid \exists q \in \delta^*(q_0, w) \text{ such that } q \in F\}$$

Kleene's Theorem (announced last lecture)

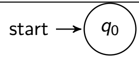
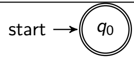
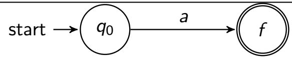
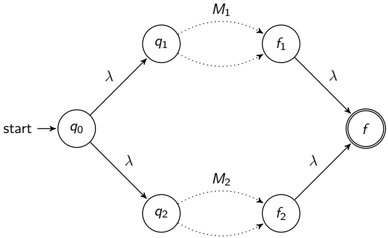
Theorem
The languages accepted by DFAs are exactly the regular languages

We will prove this as follows:

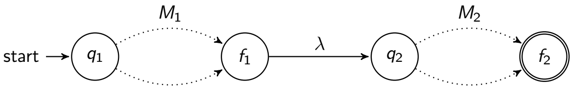
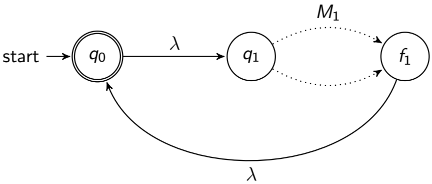
1. For every regular expression e there is an NFA-λ M such that $\mathcal{L}(e) = \mathcal{L}(M)$
2. For every NFA-λ M there is an NFA M' such that $\mathcal{L}(M) = \mathcal{L}(M')$
3. For every NFA M there is a DFA $\overline{M}$ such that $\mathcal{L}(M) = \mathcal{L}(\overline{M})$
4. For every DFA M there is a regexp e such that $\mathcal{L}(M) = \mathcal{L}(e)$.

So: reg expr, DFA, NFA, NFA-λ all characterise the same languages!

From regular expression to NFA-λ

e	M such that $\mathcal{L}(M) = \mathcal{L}(e)$
0	
1	
a (for $a \in \Sigma$)	
$e = e_1 + e_2$ with $\mathcal{L}(M_1) = \mathcal{L}(e_1)$ $\mathcal{L}(M_2) = \mathcal{L}(e_2)$	

Regexp to NFA-λ (continued)

e	M such that $\mathcal{L}(M) = \mathcal{L}(e)$
$e = e_1 e_2$ with $\mathcal{L}(M_1) = \mathcal{L}(e_1)$ $\mathcal{L}(M_2) = \mathcal{L}(e_2)$	
$e = (e_1)^*$ with $\mathcal{L}(M_1) = \mathcal{L}(e_1)$	

Regular languages accepted by an NFA-λ

Proposition. For every regular expression e there is an NFA-λ M_e such that

$$\mathcal{L}(M_e) = \mathcal{L}(e).$$

Proof. Apply the toolkit. M_e can be found *by induction on the structure of e*: First do this for the simplest regular expressions. For a composed regular expression compose the automata. ☺

Corollary. For every regular language L there is an NFA-λ M that accepts L (so $\mathcal{L}(M) = L$).

From NFA-λ to NFA

- if there is a path

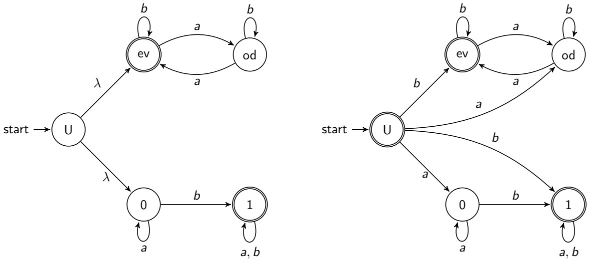
$$q_1 \xrightarrow{\lambda} q_2 \xrightarrow{\lambda} \ldots \xrightarrow{\lambda} q_n \xrightarrow{a} q'$$

then add

$$q_1 \xrightarrow{a} q'$$

- a state is accepting if reaches an accepting state via λ-steps

Example:



From NFA-λ to NFA

Given an NFA-λ $M = (Q, \delta, q_0, F)$ we build the NFA

$$M' = (Q, \bar{\delta}, q_0, \bar{F})$$

where

- $\bar{\delta}(q, a) = \bigcup_{p \in \lambda\text{-closure}(q)} \delta(p, a).$
- $\bar{F} = \{q \in Q \mid \lambda\text{-closure}(q) \cap F \neq \emptyset\}$ and

Theorem
Given an NFA-λ $M = (Q, \delta, q_0, F)$, the corresponding automaton $M' = (Q, \bar{\delta}, q_0, \bar{F})$ after elimination of λ-transitions accepts the same language.

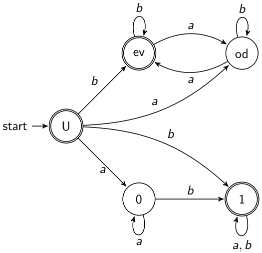
From NFA to DFA

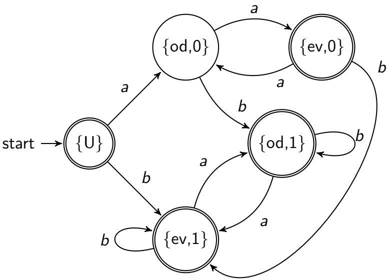
We can transform any NFA into a DFA that accepts the same language.

Idea:

- Keep track of the set of all states you can go to!
- States of the DFA are sets-of-states from the original NFA.
- A set of states is final if one of the members is final.

Example $L = \{w \mid |w|_a \text{ is even or } |w|_b \geq 1\}$





Eliminating non-determinism

Let M be a NFA given by (Q, q_0, δ, F)
Define the DFA $\overline{M}$ as $(\overline{Q}, \overline{q_0}, \overline{\delta}, \overline{F})$ where

$$
\begin{aligned}
\overline{Q} &= \mathcal{P}(Q) \\
\overline{q_0} &= \{q_0\} \\
\overline{\delta}(H, a) &= \bigcup_{q \in H} \delta(q, a), && \text{for } H \subseteq Q, \\
\overline{F} &= \{H \subseteq Q \mid H \cap F \neq \emptyset\}
\end{aligned}
$$

Correctness

Given M, an NFA, we have defined the DFA $\overline{M}$ by

$$\begin{array}{rcll} \overline{Q} & = & \mathcal{P}(Q) & \\ \overline{q_0} & = & \{q_0\} & \\ \overline{\delta}(H,a) & = & \displaystyle\bigcup_{q \in H} \delta(q,a), & \qquad\qquad \text{for } H \subseteq Q, \\ \overline{F} & = & \{H \subseteq Q \mid H \cap F \neq \emptyset\} & \end{array}$$

Theorem M and $\overline{M}$ accept the same languages.

Proof: This follows from

Lemma

$$\delta^*(q,w) \cap F \neq \emptyset \iff \overline{\delta}^*(\{q\},w) \in \overline{F}$$

(Take $q := q_0$)

Proof of the Lemma: induction on w, considering the cases $w = \lambda$ and $w = au$.

Equivalence of regexp, DFA, NFA and NFA-λ

Theorem
The class of regular languages is (equivalently) characterized as

1. *The languages described by a regular expression*
2. *The languages accepted by an NFA-λ*
3. *The languages accepted by an NFA*
4. *The languages accepted by a DFA*