Non-deterministic Finite Automata

H. Geuvers and A. Kissinger

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: fall 2015

Outline

Previous Weeks

Regular Expressions and Regular Languages

$$\boxed{\texttt{rexp}_\Sigma ::= 0 \mid 1 \mid s \mid \texttt{rexp}_\Sigma\ \texttt{rexp}_\Sigma \mid \texttt{rexp}_\Sigma + \texttt{rexp}_\Sigma \mid \texttt{rexp}_\Sigma^*}$$

with $\texttt{s} \in \Sigma$
$L \subseteq \Sigma^*$ is regular if $L = \mathcal{L}(e)$ for some regular expression e.

Deterministic Finite Automata, DFA

Proposition Closure under complement, union, intersection

If L_1, L_2 are accepted by some DFA, then so are

- $\overline{L_1} = \Sigma^* - L_1$
- $L_1 \cup L_2$
- $L_1 \cap L_2$.

Kleene's Theorem (announced last lecture)

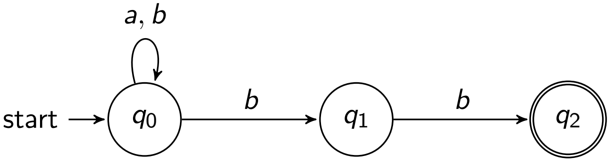
Theorem

The languages accepted by DFAs are exactly the regular languages

We prove this by

1. If $L = \mathcal{L}(M)$ for some DFA M, then there is a regular expression e such that $L = \mathcal{L}(e)$ (Previous lecture)
2. If $L = \mathcal{L}(e)$, for some regular expression e, then there is a non-deterministic finite automaton with λ-steps (NFA-λ) M such that $L = \mathcal{L}(M)$. (This lecture)
3. For every NFA-λ, M, there is a DFA M' such that $\mathcal{L}(M) = \mathcal{L}(M')$ (This lecture)

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	$\emptyset$	$\emptyset$

in shorthand

δ	a	b
q_0	q_0	q_0, q_1
q_1		q_2
q_2		

Non-deterministic Finite Automata: NFA (definition)

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)

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

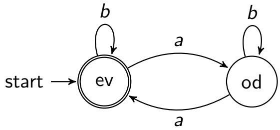
The language accepted by M, notation $\mathcal{L}(M)$, is:

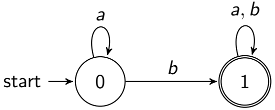
$$\mathcal{L}(M) = \{w \in \Sigma^* \mid \exists q_f \in F(q_f \in \delta^*(q_0, w))\}$$

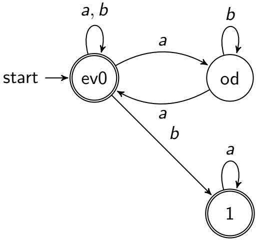
For the union of languages we can put NFAs in parallel

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"





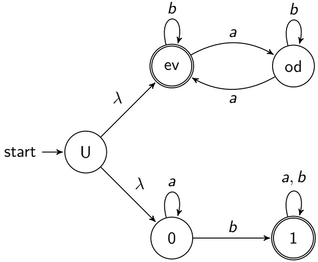


But this is wrong: The NFA accepts *aaa*.

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

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

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, λ-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)

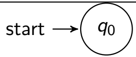
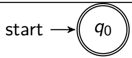
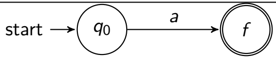
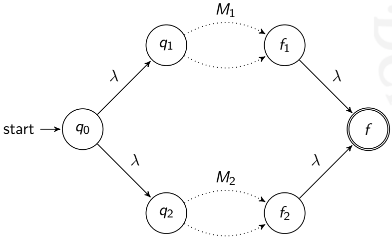
$$\begin{aligned}
\delta^*(q, \lambda) &= \lambda\text{-closure}(q) \\
\delta^*(q, aw) &= \{q' \mid \exists p \in \lambda\text{-closure}(q)\, \exists r \in \delta(p, a)\, (q' \in \delta^*(r, w))\} \\
&= \bigcup_{p \in \lambda\text{-closure}(q)} \; \bigcup_{r \in \delta(p,a)} \delta^*(r, w)
\end{aligned}$$

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

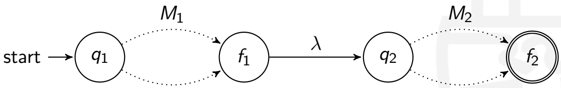
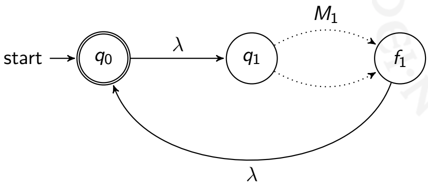
$$\mathcal{L}(M) = \{w \in \Sigma^* \mid \exists q_f \in F (q_f \in \delta^*(q_0, w))\}$$

Toolkit for building an NFA-λ from a regular expression

For each regular expression, we construct an 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 $\mathcal{L}(M_1) = \mathcal{L}(e_1)$ $\mathcal{L}(M_2) = \mathcal{L}(e_2)$	

Toolkit (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 a 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. ☻

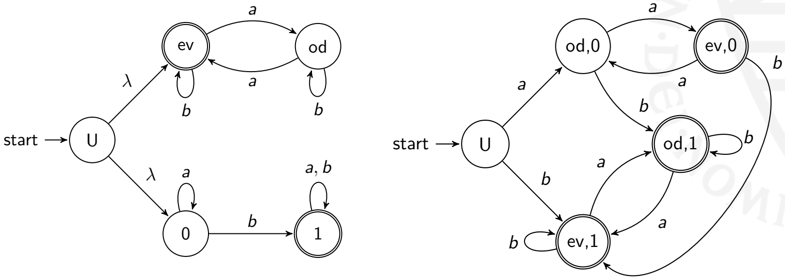
Corollary. For every regular language L there is an NFA-λ M that accepts L (so $\mathcal{L}(M) = L$).

Avoiding non-determinism

We can transform any NFA (and NFA-λ) 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-λ.
- 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 and λ-steps

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

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

If M is an NFA-λ, we define

$$\begin{aligned}
\overline{\delta}(H, a) &= \bigcup_{q \in H} \bigcup_{p \in \lambda\text{-closure}(q)} \lambda\text{-closure}(\delta(p, a)) \\
\overline{F} &= \{H \subseteq Q \mid \lambda\text{-closure}(H) \cap F \neq \emptyset\}
\end{aligned}$$

Correctness

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

$$\begin{array}{rcl} \overline{q_0} & = & \{q_0\} \\ \overline{\delta}(H, a) & = & \bigcup_{q \in H} \bigcup_{p \in \lambda\text{-closure}(q)} \lambda\text{-closure}(\delta(p, a)) \\ \overline{F} & = & \{H \subseteq Q \mid \lambda\text{-closure}(H) \cap F \neq \emptyset\} \end{array}$$

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 DFA, NFA and NFA-λ

Conclusion. Every NFA-λ (or NFA) M can be turned into a DFA $\overline{M}$ accepting the same language.

Corollary. For every regular language L there is a DFA M that accepts L (so $\mathcal{L}(M) = L$).

Proof. Given a regular expression e, first construct an NFA-λ M such that $\mathcal{L}(M) = \mathcal{L}(e)$. Then change it into a *DFA* preserving the language that is accepted. ☻

Rephrasing of Kleene's Theorem:

The class of regular languages is (equivalently) characterized as

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