Finite Automata Theory and Formal Languages

TMV027/DIT321– LP4 2018

Lecture 7

Ana Bove

April 12th 2018

Recap: Non-deterministic Finite Automata

- Defined by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$;
- Why "non-deterministic"?;
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}ow(Q)$;
- Easier to define for some problems;
- Accept set of words x such that $\hat{\delta}(q_0, x) \cap F \neq \emptyset$;
- Given a NFA N we apply the subset construction to get a DFA D ...
- ... such that $\mathcal{L}(N) = \mathcal{L}(D)$;
- Hence, NFA also accept the so called regular language.

Overview of Today's Lecture

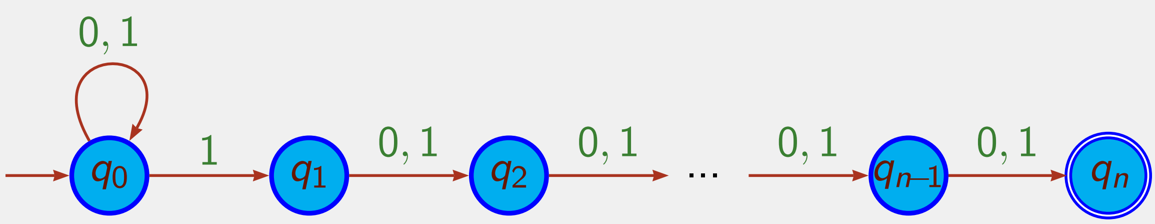
- More on NFA;
- NFA with ϵ-Transitions;
- Equivalence between DFA and ϵ-NFA;

Contributes to the following learning outcome:

- Explain and manipulate the different concepts in automata theory and formal languages;
- Have a clear understanding about the equivalence between (non-)deterministic finite automata and regular expressions;
- Understand the power and the limitations of regular languages and context-free languages;
- Design automata, regular expressions and context-free grammars accepting or generating a certain language;
- Describe the language accepted by an automata or generated by a regular expression or a context-free grammar;
- Determine if a certain word belongs to a language;
- Differentiate and manipulate formal descriptions of languages, automata and grammars.

A Bad Case for the Subset Construction

Proposition: *Any DFA recognising the same language as the NFA below has at least 2^n states:*



This NFA recognises strings over $\{0, 1\}$ such that the nth symbol from the end is a 1.

Proof: Let $\mathcal{L}_n = \{x1u \mid x \in \Sigma^*, u \in \Sigma^{n-1}\}$ and $D = (Q, \Sigma, \delta, q_0, F)$ a DFA.

We want to show that if $|Q| < 2^n$ then $\mathcal{L}(D) \neq \mathcal{L}_n$.

A Bad Case for the Subset Construction (Cont.)

Lemma: *If $\Sigma = \{0, 1\}$ and $|Q| < 2^n$ then there exist $x, y \in \Sigma^*$ and $u, v \in \Sigma^{n-1}$ such that $\hat{\delta}(q_0, x0u) = \hat{\delta}(q_0, y1v)$ in the DFA D.*

Proof: Let us define a function $h : \Sigma^n \to Q$ such that $h(z) = \hat{\delta}(q_0, z)$.

h cannot be *injective* because $|Q| < 2^n = |\Sigma^n|$.

So h sends 2 different words to the same image: $a_1 \dots a_n \neq b_1 \dots b_n$ but

$$h(a_1 \dots a_n) = \hat{\delta}(q_0, a_1 \dots a_n) = \hat{\delta}(q_0, b_1 \dots b_n) = h(b_1 \dots b_n)$$

Let us assume that $a_i = 0$ and $b_i = 1$.

Let $x = a_1 \dots a_{i-1}$, $y = b_1 \dots b_{i-1}$, $u = a_{i+1} \dots a_n 0^{i-1}$, $v = b_{i+1} \dots b_n 0^{i-1}$.

Hence (recall that for a DFA, $\hat{\delta}(q, zw) = \hat{\delta}(\hat{\delta}(q, z), w)$):

$$\hat{\delta}(q_0, x0u) = \hat{\delta}(q_0, a_1 \dots a_n 0^{i-1}) = \hat{\delta}(\hat{\delta}(q_0, a_1 \dots a_n), 0^{i-1}) =$$
$$\hat{\delta}(\hat{\delta}(q_0, b_1 \dots b_n), 0^{i-1}) = \hat{\delta}(q_0, b_1 \dots b_n 0^{i-1}) = \hat{\delta}(q_0, y1v)$$

A Bad Case for the Subset Construction (Cont.)

Lemma: *If $|Q| < 2^n$ then $\mathcal{L}(D) \neq \mathcal{L}_n$.*

Proof: Assume $\mathcal{L}(D) = \mathcal{L}_n$.

Let $x, y \in \Sigma^*$ and $u, v \in \Sigma^{n-1}$ as in previous lemma.

Then, $y1v \in \mathcal{L}(D)$ but $x0u \notin \mathcal{L}(D)$,

Hence it should be that $\hat{\delta}(q_0, y1v) \in F$ and $\hat{\delta}(q_0, x0u) \notin F$.

However, this contradicts the previous lemma that says that $\hat{\delta}(q_0, x0u) = \hat{\delta}(q_0, y1v)$.

Product Construction for NFA

Definition: Given 2 NFA $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ over the same alphabet Σ, we define the product $N_1 \otimes N_2 = (Q, \Sigma, \delta, q_0, F)$ as follows:

- $Q = Q_1 \times Q_2$;
- $\delta((p_1, p_2), a) = \delta_1(p_1, a) \times \delta_2(p_2, a)$;
- $q_0 = (q_1, q_2)$;
- $F = F_1 \times F_2$.

Lemma: $(t_1, t_2) \in \hat{\delta}((p_1, p_2), x)$ *iff* $t_1 \in \hat{\delta}_1(p_1, x)$ *and* $t_2 \in \hat{\delta}_2(p_2, x)$.

Proof: By (structural) induction on x.

Proposition: $\mathcal{L}(N_1 \otimes N_2) = \mathcal{L}(N_1) \cap \mathcal{L}(N_2)$.

Variation of Product Construction for NFA?

Recall: Given 2 DFA D_1 and D_2, then $\mathcal{L}(D_1 \oplus D_2) = \mathcal{L}(D_1) \cup \mathcal{L}(D_2)$.

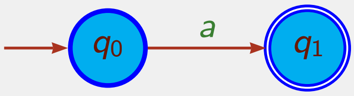
Given 2 NFA N_1 and N_2, do we need to define $N_1 \oplus N_2$?

Not really since union of languages can be modelled by the nondeterminism!

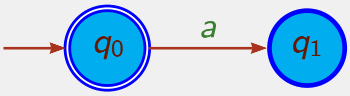
Complement of a NFA?

OBS: Given NFA $N = (Q, \Sigma, \delta, q, F)$ and $N' = (Q, \Sigma, \delta, q, Q - F)$, in general we do *not* have that $\mathcal{L}(N') = \Sigma^* - \mathcal{L}(N)$.

Example: Let $\Sigma = \{a\}$ and N and N' as follows:



$\mathcal{L}(N) = \{a\}$

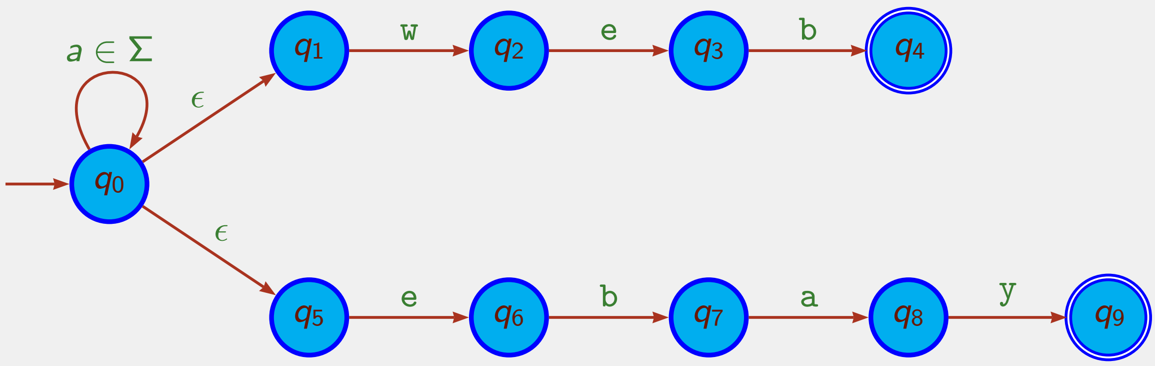


$\mathcal{L}(N') = \{\epsilon\} \neq \Sigma^* - \{a\}$

NFA with ϵ-Transitions

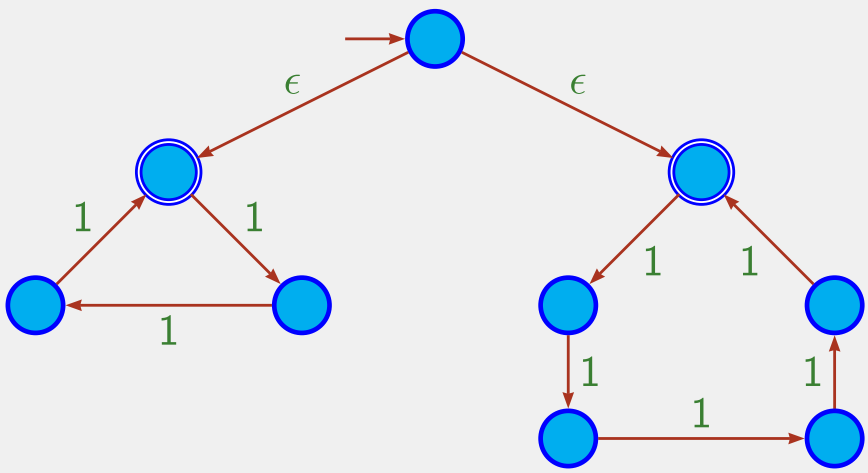
We could allow *ϵ-transitions*: transitions from one state to another *without* reading any input symbol.

Example: The following ϵ-NFA searches for the keyword `web` and `ebay`:



ϵ-NFA Accepting Words of Length Divisible by 3 or by 5

Example: Let $\Sigma = \{1\}$.



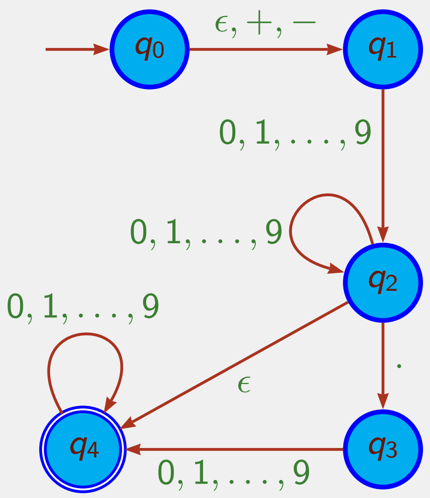
NFA with ϵ-Transitions

Definition: A *NFA with ϵ-transitions* (ϵ-NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ consisting of:

1. A finite set Q of *states*;
2. A finite set Σ of *symbols* (alphabet);
3. A "partial" transition function $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}ow(Q)$;
4. A *start state* $q_0 \in Q$;
5. A set $F \subseteq Q$ of *final* or *accepting* states.

Exercise: ϵ-NFA Accepting Decimal Numbers

Define a NFA accepting number with an optional +/- symbol and an optional decimal part.



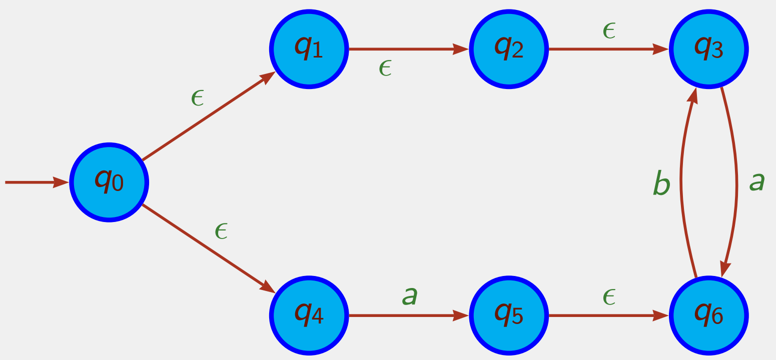
	+,-	.	0,1,...,9	ϵ
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$	$\emptyset$	$\{q_1\}$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\emptyset$
q_2	$\emptyset$	$\{q_3\}$	$\{q_2\}$	$\{q_4\}$
q_3	$\emptyset$	$\emptyset$	$\{q_4\}$	$\emptyset$
$*q_4$	$\emptyset$	$\emptyset$	$\{q_4\}$	$\emptyset$

The ϵ-transitions take care of the *optional* symbol +/- and the *optional* decimal part.

ϵ-Closure

Informally, the *ϵ-closure* of a state q is the set of states we can reach by doing nothing or by *only* following paths labelled with ϵ.

Example: For the automaton



the ϵ-closure of q_0 is $\{q_0, q_1, q_2, q_3, q_4\}$.

ϵ-Closure

Definition: Formally, we define the *ϵ-closure* of a set of states as follows:

- If $q \in S$ then $q \in \text{ECLOSE}(S)$;
- If $q \in \text{ECLOSE}(S)$ and $p \in \delta(q, \epsilon)$ then $p \in \text{ECLOSE}(S)$.

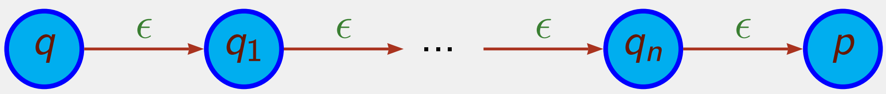
Note: Alternative formulation

$$\frac{q \in S}{q \in \text{ECLOSE}(S)} \qquad\qquad \frac{q \in \text{ECLOSE}(S) \qquad p \in \delta(q, \epsilon)}{p \in \text{ECLOSE}(S)}$$

Definition: We say that S is *ϵ-closed* iff $S = \text{ECLOSE}(S)$.

Remarks: ϵ-Closure

- Intuitively, $p \in \text{ECLOSE}(S)$ iff there exists $q \in S$ and a sequence of ϵ-transitions such that



- The ϵ-closure of a single state q can be computed as $\text{ECLOSE}(\{q\})$;

- $\text{ECLOSE}(\emptyset) = \emptyset$;

- S is ϵ-closed iff $q \in S$ and $p \in \delta(q, \epsilon)$ implies $p \in S$.

Exercise: Implement the ϵ-closure!

Extending the Transition Function to Strings

Definition: Given an ϵ-NFA $E = (Q, \Sigma, \delta, q_0, F)$ we define

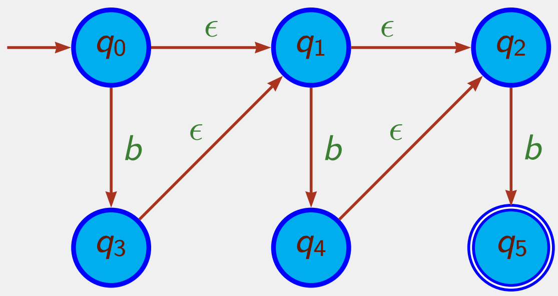
$$\begin{aligned}
&\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}ow(Q) \\
&\hat{\delta}(q, \epsilon) = \text{ECLOSE}(\{q\}) \\
&\hat{\delta}(q, ax) = \bigcup_{p \in \Delta(\text{ECLOSE}(\{q\}), a)} \hat{\delta}(p, x) \\
&\qquad \text{where } \Delta(S, a) = \cup_{p \in S} \delta(p, a)
\end{aligned}$$

Remark: By definition, $\hat{\delta}(q, a) = \text{ECLOSE}(\Delta(\text{ECLOSE}(\{q\}), a))$.

Language Accepted by a ϵ-NFA

Definition: The *language* accepted by the ϵ-NFA $(Q, \Sigma, \delta, q_0, F)$ is the set $\mathcal{L} = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \cap F \neq \emptyset\}$.

Example: Let $\Sigma = \{b\}$.



The automaton accepts the language $\{b, bb, bbb\}$.

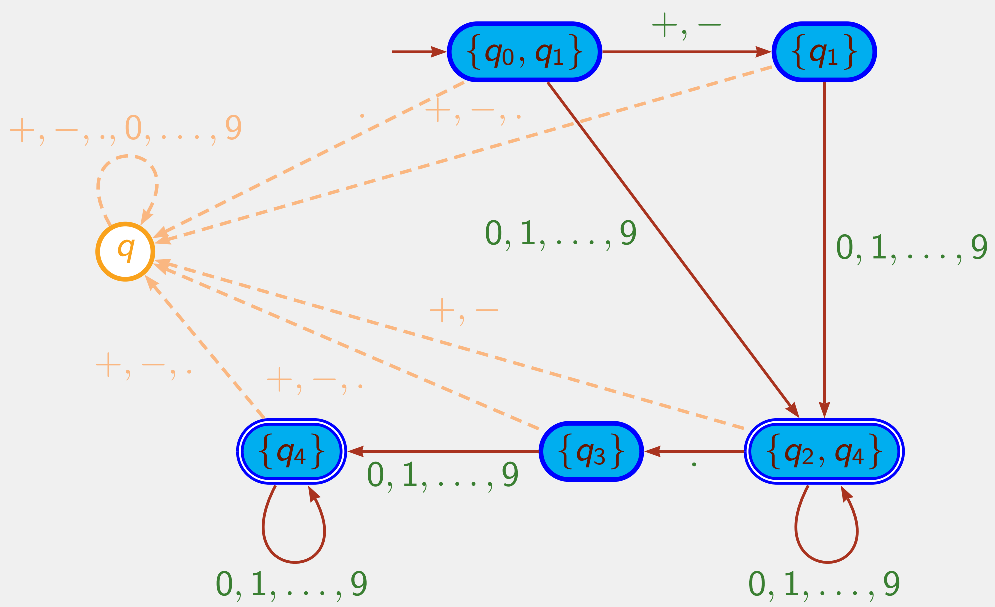
Note: Yet again, we could write a program that simulates a ϵ-NFA and let the program tell us whether a certain string is accepted or not.

Exercise: Do it!

Example: Eliminating ϵ-Transitions

Let us eliminate the ϵ-transitions in ϵ-NFA that recognises numbers in slide 12.

We obtain the following DFA:



Eliminating ϵ-Transitions

Definition: Given an ϵ-NFA $E = (Q_E, \Sigma, \delta_E, q_E, F_E)$ we define a DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ as follows:

- $Q_D = \{\text{ECLOSE}(S) \mid S \in \mathcal{P}ow(Q_E)\}$;
- $\delta_D(S, a) = \text{ECLOSE}(\Delta(S, a))$ with $\Delta(S, a) = \cup_{p \in S} \delta(p, a)$;
- $q_D = \text{ECLOSE}(\{q_E\})$;
- $F_D = \{S \in Q_D \mid S \cap F_E \neq \emptyset\}$.

Note: This construction is similar to the subset construction but now we need to ϵ-close after each step.

Exercise: Implement this transformation!

Eliminating ϵ-Transitions

Let E be an ϵ-NFA and D the corresponding DFA after eliminating ϵ-transitions.

Lemma: $\forall x \in \Sigma^*.\ \hat{\delta}_E(q_E, x) = \hat{\delta}_D(q_D, x)$.

Proof: By (structural) induction on x.

Proposition: $\mathcal{L}(E) = \mathcal{L}(D)$.

Proof: $x \in \mathcal{L}(E)$ iff $\hat{\delta}_E(q_E, x) \cap F_E \neq \emptyset$
iff $\hat{\delta}_E(q_E, x) \in F_D$ by definition of F_D
iff $\hat{\delta}_D(q_D, x) \in F_D$ by previous lemma
iff $x \in \mathcal{L}(D)$.

Finite Automata and Regular Languages

We have shown that DFA, NFA and ϵ-NFA are equivalent in the sense that we can transform the one into the other.

Hence, a language is *regular* iff there exists a finite automaton (DFA, NFA or ϵ-NFA) that accepts the language.

Overview of Next Lecture

Sections 3.1, brief on 3.4, 3.2.2:

- Regular expressions.
- Brief on algebraic laws for regular expressions;
- Equivalence between FA and RE: from FA to RE.

  Note: One of the methods is *not* in the book!

Overview of next Week

Mon 16	Tue 17	Wed 18	Thu 19	Fri 20
	Ex 10-12 EB ϵ-NFA, RE.		**10-12 ES61 Individual help**	
Lec 13-15 HB3 RE, FA$\to$RE.			**Lec 13-15 HB3** RE$\to$FA, RL.	
Ex 15-17 EA ϵ-NFA, RE.		**15-17 EL41 Consultation**		

Assignment 3: ϵ-NFA, RE.
Deadline: Sunday 22nd April 23:59.