

# Talen en Automaten

Lecture 2: Deterministic Finite Automata

Jurriaan Rot



# Outline

Finite Automata

Manipulating finite automata

Finite automata and regular languages



# Regular languages

## Last lecture

- Regular expressions  $e$ , describe a language  $L(e)$
- $L \subseteq A^*$  is regular if  $L = \mathcal{L}(e)$  for some regular expression  $e$ .

## Today

- Automata: 'machines' that accept languages
- Fact (this and next lecture): a language is described by a regular expression if and only if it is accepted by a finite deterministic automaton
- Correspondence between specification language and machine model



## Some context: decidability problems

- Q1 Can we decide (algorithmically) if  $w \in \mathcal{L}(e)$ ?
- Q2 If yes, what is the **complexity** of this decision procedure?
- Q3 Can we decide if  $\mathcal{L}(e_1) = \mathcal{L}(e_2)$ ?

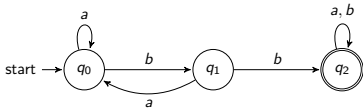
### Some answers

- A1 Yes, we can give a **deterministic finite automaton** for  $e$  to decide  $w \in \mathcal{L}(e)$ . (This lecture)
- A2 time is **linear** in  $|w|$ ; memory is constant (independent of  $w$ ).
- A3 Yes: via *automata constructions* or via *equations*, see Exercise 2.3.6 of the course notes.



## Deterministic Finite Automaton (DFA)

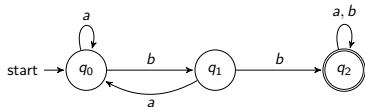
Intuition. Let  $A = \{a, b\}$ . Consider the DFA  $M$ :



Letters  $a, b$  are the moves in the graph  
Words  $w \in A^*$  give sequences of moves  
**Start** state is indicated by 'start→',  
**Accepting/final** states by double circle  
(there can be several accepting states)

The word  $abba$  is **accepted**, but  $baab$  is not accepted (rejected)

## Deterministic Finite Automaton (DFA)



$M = (Q, q_0, \delta, F)$  with  $Q = \{q_0, q_1, q_2\}$ ,  $F = \{q_2\}$  and  $\delta$  given by

$\delta$	$a$	$b$
$q_0$	$q_0$	$q_1$
$q_1$	$q_0$	$q_2$
$q_2$	$q_2$	$q_2$



## Deterministic Finite Automata formally

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

$A$  is a finite alphabet

$Q$  is a finite set of **states**

$q_0 \in Q$  is the **initial** state

$F \subseteq Q$  is a finite set of **final** states

$\delta : Q \times A \rightarrow Q$  is the **transition** function

The **multi-step transition**  $\delta^* : Q \times A^* \rightarrow Q$  is defined inductively by

$$\delta^*(q, \lambda) = q$$

$$\delta^*(q, aw) = \delta^*(\delta(q, a), w)$$

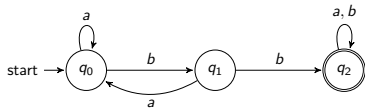
The **language accepted by**  $M$ , notation  $\mathcal{L}(M)$ , is:

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



## Reading words $w \in A^*$

Computation for  $\delta^*(q_0, w)$  in the example DFA.



Take  $w = abba$ :

$$\begin{aligned}\delta^*(q_0, abba) &= \delta^*(\delta(q_0, a), bba) = \delta^*(q_0, bba) \\ &= \delta^*(\delta(q_0, b), ba) = \delta^*(q_1, ba) \\ &= \delta^*(\delta(q_1, b), a) = \delta^*(q_2, a) \\ &= \delta^*(\delta(q_2, a), \lambda) = \delta^*(q_2, \lambda) \\ &= q_2\end{aligned}$$

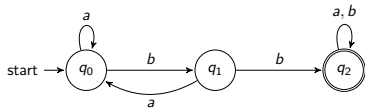
So *abba* is **accepted**.





## Reading words $w \in A^*$

Computation for  $\delta^*(q_0, w)$  in the example DFA.



Take  $w = aba$ :

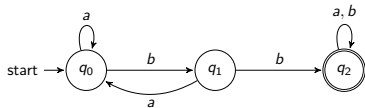
$$\begin{aligned}\delta^*(q_0, aba) &= \delta^*(\delta(q_0, a), ba) = \delta^*(q_0, ba) \\ &= \delta^*(\delta(q_0, b), a) = \delta^*(q_1, a) \\ &= \delta^*(\delta(q_1, a), \lambda) = \delta^*(q_0, \lambda) \\ &= q_0\end{aligned}$$

So  $aba$  is **not accepted**.



## Reading words $w \in A^*$

Computation for  $\delta^*(q_0, w)$  in the example DFA.



The (regular) language accepted by the automaton is:

$$\mathcal{L}((a + b)^* bb(a + b)^*).$$



## From transition table to state diagram

Consider the automaton  $M$  over  $A = \{a, b\}$  with

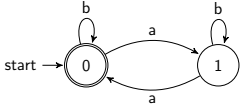
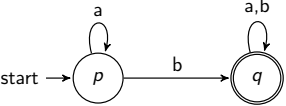
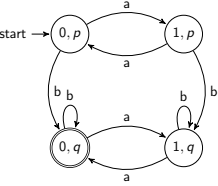
- $Q = \{0, 1, 2, 3, 4\}$ ,
- $q_0 = 0$ ,
- $F = \{4\}$
- 

$\delta$	$a$	$b$
0	1	0
1	1	2
2	1	3
3	4	0
4	4	4

1. Which of the following words is accepted?  $abba$ ,  $baba$ ,  $bba$
2. Is it the case that  $\{w \mid |w|_b \text{ is even}\} \subseteq \mathcal{L}(M)$ ?
3. Is it the case that  $\{w \mid w \text{ contains } aabbaa\} \subseteq \mathcal{L}(M)$ ?



# Manipulating Finite Automata: products for intersection

$M$	$\mathcal{L}(M)$
	$L_1 = \{w \mid  w _a \text{ is even}\}$
	$L_2 = \{w \mid  w _b \geq 1\}$
	$L_1 \cap L_2 = \{w \mid  w _a \text{ is even and }  w _b \geq 1\}$



## Product of two DFAs

Given two DFAs over the same  $A$

$$M_1 = (Q_1, q_{01}, \delta_1, F_1)$$

$$M_2 = (Q_2, q_{02}, \delta_2, F_2)$$

Define

$$M_1 \times M_2 = (Q_1 \times Q_2, q_0, \delta, F)$$

with

$$q_0 := (q_{01}, q_{02})$$

$$\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_2, a))$$

Then with

$$F := F_1 \times F_2 := \{(q_1, q_2) \mid q_1 \in F_1 \text{ and } q_2 \in F_2\}$$

we have

$$\mathcal{L}(M_1 \times M_2) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$$




## Closure Properties

### Proposition Closure under complement

If  $L$  is accepted by some DFA, then so is

$$\bar{L} = A^* - L.$$


**Proof.** Suppose that  $L$  is accepted by  $M = (Q, q_0, \delta, F)$ .

Then  $\bar{L}$  is accepted by  $\bar{M} = (Q, q_0, \delta, \bar{F})$ . 

### Proposition Closure under intersection and union

If  $L_1$ , and  $L_2$  are accepted by some DFA, then so are  $L_1 \cap L_2$  and  $L_1 \cup L_2$ .

**Proof.** For the intersection, this follows from the product construction on the previous slide.

For the union, this can be seen by the product construction, taking a different  $F$  (which one?) or by noticing that  $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ . 



## Kleene's Theorem

**Theorem** The languages accepted by DFAs are exactly the **regular languages**

We will prove this as follows:

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



## From DFAs to regular expressions

Given the DFA  $M = (Q, q_0, \delta, F)$ , we construct a regular expression  $e$  such that

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

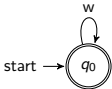
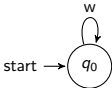
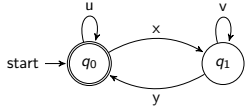
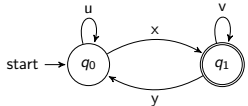
Procedure:

- We remove states, replacing symbols from  $A$  by regular expressions over  $A$ ,
- until we end up with a “simple automaton” from which we can read off  $e$ .





## Simple automata

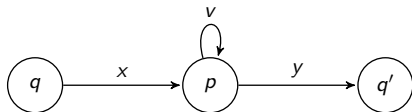
$M$	$e$ such that $\mathcal{L}(e) = \mathcal{L}(M)$
 <p>start <math>\rightarrow</math> <math>q_0</math></p>	$w^*$
 <p>start <math>\rightarrow</math> <math>q_0</math></p>	$0$
 <p>start <math>\rightarrow</math> <math>q_0</math></p>	$(u + xv^*y)^*$
 <p>start <math>\rightarrow</math> <math>q_0</math></p>	$u^*x(v + yu^*x)^*$



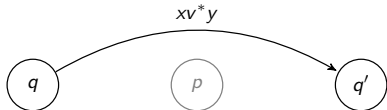
## Eliminating states

- Remove a state  $p$ ,
- while adding arrows  $q \xrightarrow{w} q'$  between other pairs of states.

Before:

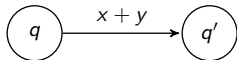
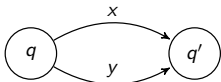


After:

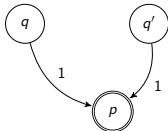


## Special cases

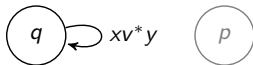
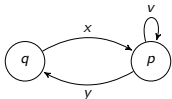
Join arrows using +



First create a new single final state



Beware of loops!



## From DFA to RegExp

The algorithm outlined in the previous slides produces for every deterministic finite automaton  $M$ , a regular expression  $e_M$ .

**Theorem** For  $M$  a DFA we have

$$\mathcal{L}(e_M) = \mathcal{L}(M).$$

