

# **Introduction to Computational Linguistics**

**Frank Richter**

**fr@sfs.uni-tuebingen.de.**

**Seminar für Sprachwissenschaft  
Eberhard Karls Universität Tübingen  
Germany**

# What is in a State

## Definition 4

Given a DFA  $M = (\Sigma, Q, i, F, \delta)$ ,

a *state of  $M$*  is triple  $(x, q, y)$

where  $q \in Q$  and  $x, y \in \Sigma^*$

# The *directly derives* relation

## Definition 5 (directly derives)

Given a DFA  $(\Sigma, Q, i, F, \delta)$ ,

a state  $(x, q, y)$  *directly derives* state  $(x', q', y')$ :

$(x, q, y) \vdash (x', q', y')$  iff

1. there is  $\sigma \in \Sigma$  such that  $y = \sigma y'$  and  $x' = x\sigma$  (i.e. the reading head moves right one symbol  $\sigma$ )
2.  $\delta(q, \sigma) = q'$

# The *derives* relation

## Definition 6 (derives)

Given a DFA  $(\Sigma, Q, i, F, \delta)$ ,

a state  $A$  *derives* state  $B$ :

$(x, q, y) \vdash^* (x', q', y')$  iff

there is a sequence  $S_0 \vdash S_1 \vdash \dots \vdash S_k$

such that  $A = S_0$  and  $B = S_k$

# Acceptance

## Definition 7 (Acceptance)

Given a DFA  $M = (\Sigma, Q, i, F, \delta)$  and a string  $x \in \Sigma^*$ ,

$M$  *accepts*  $x$  iff

there is a  $q \in F$  such that  $(0, i, x) \vdash^*(x, q, 0)$ .

# Language accepted by $M$

## Definition 8 (Language accepted by $M$ )

Given a DFA  $M = (\Sigma, Q, i, F, \delta)$ , the language  $L(M)$  accepted by  $M$  is the set of all strings accepted by  $M$ .

# Example of String Acceptance

Let  $M = (\{a, b\}, \{q_0, q_1, q_2\}, q_0, \{q_1\}, \{((q_0, a), q_1), ((q_0, b), q_1), ((q_1, a), q_2), ((q_1, b), q_2), ((q_2, a), q_2), ((q_2, b), q_2), \})$ .

# Example of String Acceptance

Let  $M = (\{a, b\}, \{q_0, q_1, q_2\}, q_0, \{q_1\}, \{((q_0, a), q_1), ((q_0, b), q_1), ((q_1, a), q_2), ((q_1, b), q_2), ((q_2, a), q_2), ((q_2, b), q_2), \})$ .

$M$  accepts  $a$  and  $b$  and nothing else, i.e.  $L(M) = \{a, b\}$ , since

$(0, q_0, a) \vdash (a, q_1, 0)$     and  
 $(0, q_0, b) \vdash (b, q_1, 0)$

are the only derivations from a start state to a final state for  $M$ .



# More Properties of FSAs

Given the FSAs  $A$ ,  $A_1$ , and  $A_2$  and the string  $w$ , the following properties are decidable:

Membership:  $w \stackrel{?}{\in} L(A)$

Emptiness:  $L(A) \stackrel{?}{=} \emptyset$

Totality:  $L(A) \stackrel{?}{=} \Sigma^*$

Subset:  $L(A_1) \stackrel{?}{\subseteq} L(A_2)$

Equality:  $L(A_1) \stackrel{?}{=} L(A_2)$

# Regular Expressions and Automata (1)

Regular Expression:  $\emptyset$

Automaton: 

Regular Expression:  $\emptyset$

Automaton: 

Regular Expression:  $a$

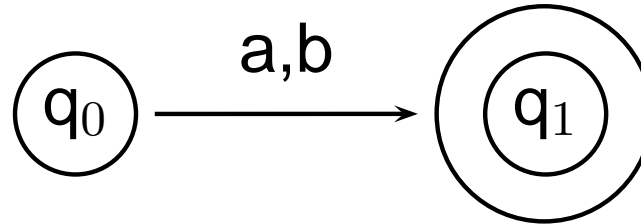
Automaton: 

# Regular Expressions and Automata (2)

Regular Expression:

$[a \mid b]$

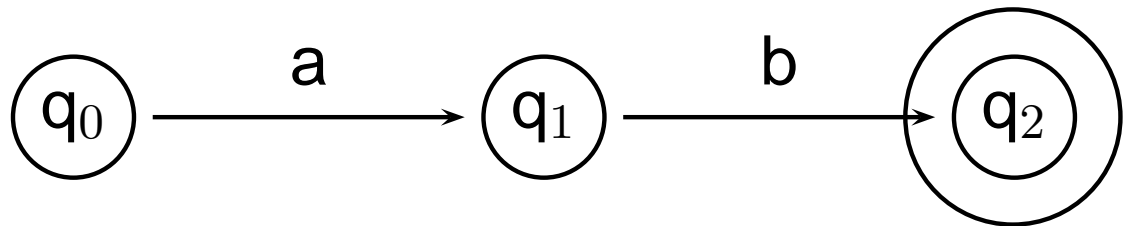
Automaton:



Regular Expression:

$[a b]$

Automaton:



# The Finite State Utilities

The FSA Utilities toolbox:

- a collection of utilities to manipulate regular expressions, finite-state automata (and finite-state transducers)
- implemented in Prolog by Gertjan van Noord, University of Groningen
- Home Page:  
`http://odur.let.rug.nl/~vannoord/Fsa/`
- command in the SfS network (on 'urobe'):  
`fsa -tk`

# Reg. Expressions: Syntactic Extensions

$\$A$       *contains*

$$\$A =_{def} [?^* A ?^*]$$

for example:  $\$[a \mid b]$  denotes all strings that contain at least one  $a$  or  $b$  somewhere.

$A \& B$       Intersection

$A - B$       Relative complement (minus)

$\sim A$       Complement (negation)

# The Bigger Picture

## Definition 9 (Regular Languages)

A language  $L$  is said to be *regular* or *recognizable* if the set of strings  $s$  such that  $s \in L$  are accepted by a DFA.

## Theorem (Kleene, 1956)

The family of regular languages over  $\Sigma^*$  is equal to the smallest family of languages over  $\Sigma^*$  that contains the empty set, the singleton sets, and that is closed under Kleene star, concatenation, and union.

$\Rightarrow$  The family of regular languages over  $\Sigma^*$  is equal to the family of languages denoted by the set of regular expressions.