

Deterministic & Nondeterministic Finite Automata

Martin Fränzle

Informatics and Mathematical Modelling
The Technical University of Denmark

02140 Languages and Parsing, MF, Fall 2004 – p.1/22

Preliminaries for Formal Languages:

Alphabets, strings, languages

02140 Languages and Parsing, MF, Fall 2004 – p.2/22

What you'll learn

You will be introduced to the basic notions of

Alphabet: A set of “symbols”,

String/word: a sequence of symbols from the alphabet,

Language: a set of strings.

02140 Languages and Parsing, MF, Fall 2004 – p.3/22

Alphabets

Def.: An alphabet is a *finite, nonempty set*.

Ex.:

- The binary alphabet $\{0, 1\}$,
- the set of all printable ASCII characters,
- the set $\{\text{red, green, blue, yellow, magenta, cyan}\}$ of basic colors.

N.B.: This notion of alphabet is distinctly different from everyday use:

- alphabets need not feature an ordering,
- the members of an alphabet need not be letters in the usual sense, but can well be compounds of some kind (e.g., tuples).

We will often denote alphabets by the greek letter Σ .

02140 Languages and Parsing, MF, Fall 2004 – p.4/22

Strings (a.k.a. words)

Def.: A string (or word) over an Alphabet Σ is a *finite* sequence of symbols from Σ .

The length of a string w , denoted $|w|$, is the number of positions for symbols in the string.

- Ex.:**
- 0100 (or, more formally, $\langle 0, 1, 0, 0 \rangle$) is a string of length 4 over alphabet $\{0, 1\}$.
 - $\langle \rangle$ is the string of length 0 over alphabet $\{a, \dots, z\}$ (as well as over any other alphabet). It is called the *empty string* and is generically denoted ε , irrespective of the particular alphabet.

Def.: Given strings $v = \langle v_1, v_2, \dots, v_n \rangle$ and $w = \langle w_1, w_2, \dots, w_m \rangle$, the concatenation of v and w is the string

$$vw \stackrel{\text{def}}{=} \langle v_1, v_2, \dots, v_n, w_1, w_2, \dots, w_m \rangle$$

of length $n + m$.

02140 Languages and Parsing, MF, Fall 2004 – p.5/22

Powers of an alphabet

Given an alphabet Σ and a natural number $n \in \mathbf{N}$,

- Σ^n is the set of all strings over Σ of length n ,
- $\Sigma^+ \stackrel{\text{def}}{=} \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{i=1}^{\infty} \Sigma^i$ is the set of all *nonempty* strings over Σ ,
- $\Sigma^* \stackrel{\text{def}}{=} \{\varepsilon\} \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \{\varepsilon\} \cup \Sigma^+$ is the set of all strings over Σ .

02140 Languages and Parsing, MF, Fall 2004 – p.6/22

Languages

Def.: A set of strings over an alphabet Σ (i.e., a subset of Σ^*) is called a language over Σ .

Ex.:

- The set of strings consisting of n 0's followed by n 1's for arbitrary $n \in \mathbf{N}$ forms a language over $\{0, 1\}$:

$$\{\varepsilon, 01, 0011, 000111, 00001111, \dots\} = \{0^n 1^n \mid n \in \mathbf{N}\}$$

- The set of strings over $\{0, \dots, 9\}$ representing a prime number forms a language over $\{0, \dots, 9\}$:

$$\{2, 3, 5, 7, 11, 13, 17, \dots\}$$

- Σ^* is a language over Σ ,
- $\emptyset$ is a language (over arbitrary alphabet),
- $\{\varepsilon\}$ is a language (over arbitrary alphabet).

Note: $\{\varepsilon\} \neq \emptyset$.

Note: Languages can be finite or infinite!

02140 Languages and Parsing, MF, Fall 2004 – p.7/22

Problems

Def.: Given a language L over some alphabet Σ , the problem L is:
Given a string w over Σ , decide whether or not $w \in L$.

- Ex.:**
- Problem of deciding whether some $w \in \{0, 1\}^*$ is a binary representation of a prime number.
 - Problem of deciding whether some $w \in \{a, \dots, z\}^*$ is the imperfect of an english verb.
 - Problem of deciding whether a string $w \in (\{A, \dots, H\} \times \{1, \dots, 8\} \times \{A, \dots, H\} \times \{1, \dots, 8\})^*$ describes
 - a game of chess where 'white' has won,
 - a game of chess that has reached a position where 'white' can enforce a win.

Questions concerning a given problem are in particular

- whether it can be solved by a computer,
- if so, what would be the best algorithm to do so,
- how much computational effort this requires.

02140 Languages and Parsing, MF, Fall 2004 – p.8/22

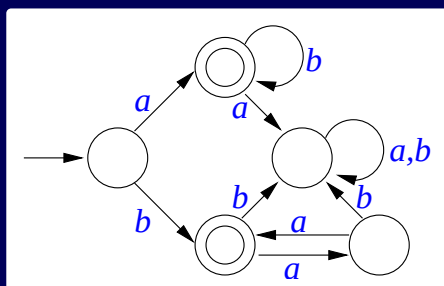
Finite Automata

The deterministic variant: DFAs

02140 Languages and Parsing, MF, Fall 2004 – p.9/22

DFAs — informally

- An **automaton** is a computational device that changes its state depending on the inputs it receives.
- A **finite automaton** has finitely many states for remembering its computation history.
- In a **deterministic finite automaton**, the current state is completely determined by the sequence of inputs seen so far.
- Automata define languages through the following mechanism:
 1. Some states are marked as being “**accepting**”;
 2. whenever the automaton is in an accepting state, the string of inputs seen so far is a member of the **language accepted by the automaton**.



⇒

$\{a, ab, abb, abbb, abbbb, \dots\} \cup$
 $\{b, baa, baaaa, baaaaaa, \dots\}$

02140 Languages and Parsing, MF, Fall 2004 – p.10/22

DFAs — formally

A **deterministic finite automaton (DFA)** $A = (Q, \Sigma, \delta, q_0, F)$ consist of

1. a **finite set of states** Q ,
2. a **finite set of input symbols** Σ , called the “**alphabet of A** ”,
3. a **transition function** $\delta : Q \times \Sigma \rightarrow Q$,
4. a **start state** $q_0 \in Q$,
5. a **set of final or accepting states** $F \subseteq Q$.

The automaton

1. starts in q_0 ,
2. proceeds from state q to state $\delta(q, a)$ if the current input is a ,
3. accepts a string (= sequence of symbols) iff that sequence of inputs drives it from q_0 to some $q \in F$.

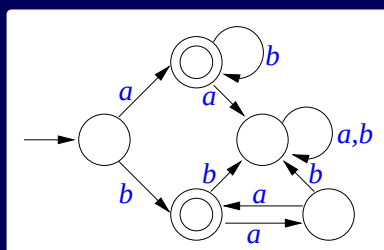
02140 Languages and Parsing, MF, Fall 2004 – p.11/22

Transition diagrams

are a graphical notation for FAs.

A **transition diagram** for a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a graph s.t.

1. For each state in Q there is a node,
2. if $\delta(q, a) = q'$ then (and only then) there is an arc from q to q' labelled a , (we allow ourselves to collate multiple arcs between the same nodes into a single one labelled with a list)
3. there is an arrow without source node into the start state,
4. nodes representing accepting states are marked with double circles.



02140 Languages and Parsing, MF, Fall 2004 – p.12/22

Transition tables

are **tabular representations of transition functions**, where information about initiality and finality of states is added:

	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

As the state space and the alphabet can be deduced from the table, transition tables are a **complete representation of DFAs!**

02140 Languages and Parsing, MF, Fall 2004 – p.13/22

Extending the trans. fct. to strings

Given $\delta : Q \times \Sigma \rightarrow Q$, we define a function $\hat{\delta} : Q \times (\Sigma^*) \rightarrow Q$ via recursion:

Base case: $\hat{\delta}(q, \varepsilon) \stackrel{\text{def}}{=} q$ for each $q \in Q$;

Recursion: $\hat{\delta}(q, xa) \stackrel{\text{def}}{=} \delta(\underbrace{\hat{\delta}(q, x)}_{\prec xa}, a)$ for each $q \in Q, x \in \Sigma^*, a \in \Sigma$.

N.B.: We could as well have defined

$$\hat{\delta}(q, ax) \stackrel{\text{def}}{=} \hat{\delta}(\delta(q, a), x)$$

without altering the function defined.

02140 Languages and Parsing, MF, Fall 2004 – p.14/22

The language of a DFA

Def: The language $L(A)$ of a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is

$$L(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\} .$$

The language $L(A)$ thus is the set of strings that drive A from the initial state to some accepting state.

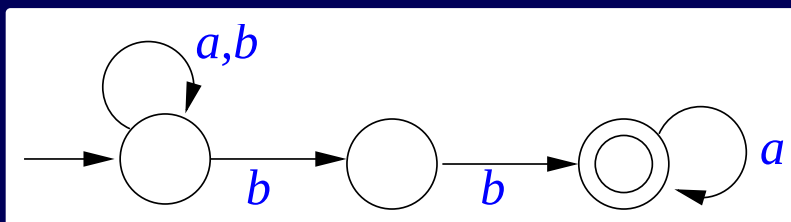
Def: If $L = L(A)$ for some DFA A then we call L a **regular language**. The regular languages are exactly those languages L which are definable by DFAs in the sense that $L = L(A)$ for some DFA A .

Finite Automata

The nondeterministic variant: NFAs

NFAs — informally

- Has the freedom to do various different moves when in a state and seeing some input,
- has the help of an “oracle” for performing the right guess,
- this is modeled mathematically as
 1. the ability to be in various states at once, and
 2. accepting a string whenever at least one of those states is accepting.



02140 Languages and Parsing, MF, Fall 2004 – p.17/22

NFAs — formally

An *nondeterministic finite automaton* (NFA) $A = (Q, \Sigma, \delta, q_0, F)$ consist of

1. a **finite set of states** Q ,
2. a **finite set of input symbols** Σ , called the “**alphabet of A** ”,
3. a **transition function** $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$, (which may be identified with a **transition relation** $\delta' \subset Q \times \Sigma \times Q$),
4. a **start state** $q_0 \in Q$,
5. a **set of final or accepting states** $F \subseteq Q$.

The automaton

1. starts in state set $\{q_0\}$,
2. proceeds from state set P to state set $\delta(P, a)$ if the current input is a ,
3. accepts a string (= sequence of symbols) iff that sequence of inputs drives it from q_0 to a state set *containing* some $q \in F$.

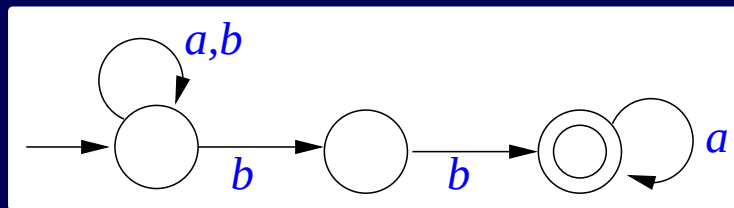
02140 Languages and Parsing, MF, Fall 2004 – p.18/22

Transition diagrams

are a graphical notation for FAs.

A **transition diagram** for a NFA $\bar{A} = (Q, \Sigma, \delta, q_0, F)$ is a graph s.t.

1. For each state in Q there is a node,
2. if $q' \in \delta(q, a)$ then (and only then) there is an arc from q to q' labelled a , (we allow ourselves to collate multiple arcs between the same nodes into a single one labelled with a list)
3. there is an arrow without source node into the start state,
4. nodes representing accepting states are marked with double circles.



02140 Languages and Parsing, MF, Fall 2004 – p.19/22

Transition tables

Do now assign **sets of states**:

	a	b
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\{q_2\}$	$\emptyset$

02140 Languages and Parsing, MF, Fall 2004 – p.20/22

Extending the trans. fct. to strings

Given $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$, we define a function $\hat{\delta} : Q \times (\Sigma^*) \rightarrow \mathcal{P}(Q)$ via recursion:

Base case: $\hat{\delta}(q, \varepsilon) \stackrel{\text{def}}{=} \{q\}$ for each $q \in Q$;

Recursion: $\hat{\delta}(q, xa) \stackrel{\text{def}}{=} \bigcup_{p \in \hat{\delta}(q, x)} \delta(p, a)$ for each $q \in Q, x \in \Sigma^*, a \in \Sigma$.

The language of an NFA

Def: The language $L(A)$ of a an NFA $A = (Q, \Sigma, \delta, q_0, F)$ is

$$L(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\} .$$

The language $L(A)$ thus is the set of strings that drive A from the initial state to a set of states containing some accepting state.