
Nondeterministic Finite Automata and their Determinization

Martin Fränze

Informatics and Mathematical Modelling
The Technical University of Denmark

02140 Languages and Parsing, MF, Fall 2003 – p.1/19

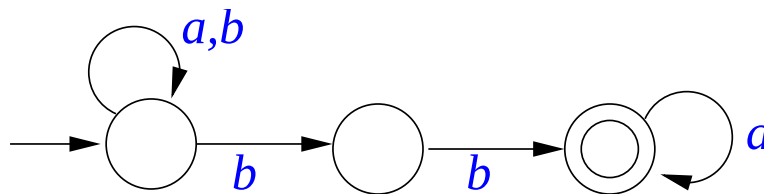
Finite Automata

The nondeterministic variant: NFAs

02140 Languages and Parsing, MF, Fall 2003 – p.2/19

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 2003 – p.3/19

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 $F \subseteq Q$.

02140 Languages and Parsing, MF, Fall 2003 – p.4/19

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 $\bigcup_{p \in P} \delta(p, a)$ 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 $q \in F$.

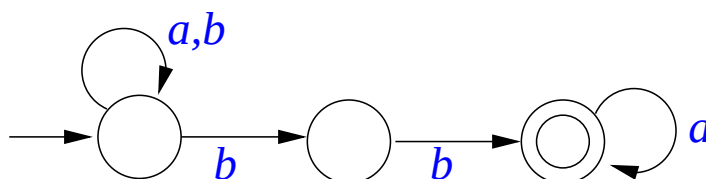
02140 Languages and Parsing, MF, Fall 2003 – p.4/19

Transition diagrams

are a graphical notation for NFAs.

A transition diagram for a NFA $A = (Q, \Sigma, \delta, q_0, F)$ 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 2003 – p.5/19

Transition tables

Do now assign sets of states:

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

Transition tables

Do now assign sets of states:

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

Through $\delta(q, a) = \emptyset$, an NFA can “forbid” an input a .

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.

NFA vs. DFA

The advantage of NFAs is that they are often much more concise:

- A DFA recognizing all words over $\{0, 1\}$ such that the n th-last symbol is 1 needs 2^n states;
- an NFA recognizing the same language needs $n + 1$ states.

Finite automata

Intersection and complementation

Intersection: Product construction

Thm: Given two DFA $A = (Q, \Sigma, \delta_A, q_0, F)$ and $B = (R, \Sigma, \delta_B, r_0, G)$, the DFA $A \times B = (S, \Sigma, \delta_{AB}, s_0, H)$ defined by

- $S \stackrel{\text{def}}{=} Q \times R$, States are pairs, $|S| = |Q| \cdot |R|$
- $\delta_{AB}((q, r), a) \stackrel{\text{def}}{=} (\delta_A(q, a), \delta_B(r, a))$ for each $s = (q, r) \in S$ and each $a \in \Sigma$,
- $s_0 \stackrel{\text{def}}{=} (q_0, r_0)$,
- $H \stackrel{\text{def}}{=} \{(q, r) \mid q \in F, r \in G\}$.

accepts the language $L(A \times B) = L(A) \cap L(B)$.

Def: $A \times B$ is called the “automaton product” of A and B .

Intersection: Product construction

Thm: Given two DFA $A = (Q, \Sigma, \delta_A, q_0, F)$ and $B = (R, \Sigma, \delta_B, r_0, G)$, the DFA $A \times B = (S, \Sigma, \delta_{AB}, s_0, H)$ defined by

- $S \stackrel{\text{def}}{=} Q \times R$, States are pairs, $|S| = |Q| \cdot |R|$
- $\delta_{AB}((q, r), a) \stackrel{\text{def}}{=} (\delta_A(q, a), \delta_B(r, a))$ for each $s = (q, r) \in S$ and each $a \in \Sigma$,
- $s_0 \stackrel{\text{def}}{=} (q_0, r_0)$,
- $H \stackrel{\text{def}}{=} \{(q, r) \mid q \in F, r \in G\}$.

accepts the language $L(A \times B) = L(A) \cap L(B)$.

Def: $A \times B$ is called the “automaton product” of A and B .

N.B.: A similar construction can be given for intersecting the languages of two NFA.

But how about intersecting an NFA with a DFA?

Complementation

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a **DFA**.

Thm: The DFA $A' = (Q, \Sigma, \delta, q_0, Q \setminus F)$ accepts the language $\Sigma^* \setminus L(A)$.

I.e., complementing the set of accepting states complements the language recognized.

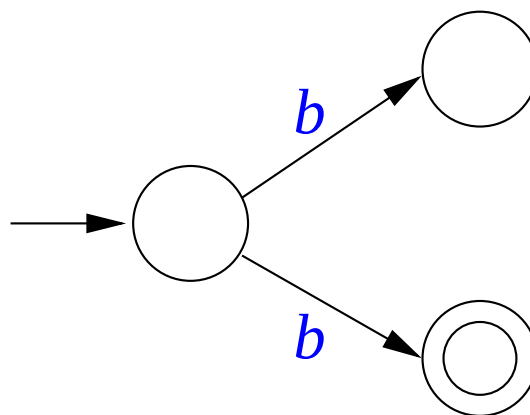
Prf:

$$\begin{aligned} L(A') &= \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in Q \setminus F\} \\ &= \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \notin F\} \\ &= \{w \in \Sigma^* \mid w \notin L(A)\} \\ &= \Sigma^* \setminus L(A) \end{aligned}$$

02140 Languages and Parsing, MF, Fall 2003 – p.12/19

Complementation

fails on NFAs:



maps to an automaton accepting the same language if the accepting state set is complemented.

02140 Languages and Parsing, MF, Fall 2003 – p.13/19

NFA vs. DFA

The advantage of NFAs is that they are often much more concise:

- A DFA recognizing all words over $\{0, 1\}$ such that the n th-last symbol is 1 needs 2^n states;
- an NFA recognizing the same language needs $n + 1$ states.

The advantage of DFA is that we can easily complement them.

NFA vs. DFA

The advantage of NFAs is that they are often much more concise:

- A DFA recognizing all words over $\{0, 1\}$ such that the n th-last symbol is 1 needs 2^n states;
- an NFA recognizing the same language needs $n + 1$ states.

The advantage of DFA is that we can easily complement them.

Questions arising:

- Do DFA and NFA define the same set of languages?
- Is there a way to convert one into the other?

Deterministic finite automata

A special case of NFAs

02140 Languages and Parsing, MF, Fall 2003 – p.15/19

Converting DFAs to NFAs

Thm: Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$, a corresponding NFA $A' = (Q', \Sigma', \delta', q'_0, F')$ accepting the same language can be defined by

- $Q' \stackrel{\text{def}}{=} Q$,
- $\Sigma' \stackrel{\text{def}}{=} \Sigma$,
- $\delta'(q, a) \stackrel{\text{def}}{=} \{\delta(q, a)\}$ for each $q \in Q$ and each $a \in \Sigma$,
- $q'_0 \stackrel{\text{def}}{=} q_0$,
- $F' \stackrel{\text{def}}{=} F$.

02140 Languages and Parsing, MF, Fall 2003 – p.16/19

Converting DFAs to NFAs

Thm: Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$, a corresponding NFA $A' = (Q', \Sigma', \delta', q'_0, F')$ accepting the same language can be defined by

- $Q' \stackrel{\text{def}}{=} Q$,
- $\Sigma' \stackrel{\text{def}}{=} \Sigma$,
- $\delta'(q, a) \stackrel{\text{def}}{=} \{\delta(q, a)\}$ for each $q \in Q$ and each $a \in \Sigma$,
- $q'_0 \stackrel{\text{def}}{=} q_0$,
- $F' \stackrel{\text{def}}{=} F$.

Prf: It is easy to show (by induction) that $\hat{\delta}'(q, w) = \{\hat{\delta}(q, w)\}$ for each $q \in Q$ and $w \in \Sigma^*$. Hence,

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

Converting DFAs to NFAs

Thm: Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$, a corresponding NFA $A' = (Q', \Sigma', \delta', q'_0, F')$ accepting the same language can be defined by

- $Q' \stackrel{\text{def}}{=} Q$,
- $\Sigma' \stackrel{\text{def}}{=} \Sigma$,
- $\delta'(q, a) \stackrel{\text{def}}{=} \{\delta(q, a)\}$ for each $q \in Q$ and each $a \in \Sigma$,
- $q'_0 \stackrel{\text{def}}{=} q_0$,
- $F' \stackrel{\text{def}}{=} F$.

Prf: It is easy to show (by induction) that $\hat{\delta}'(q, w) = \{\hat{\delta}(q, w)\}$ for each $q \in Q$ and $w \in \Sigma^*$. Hence,

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

Cor: If L is a regular language (i.e., L is the language of a DFA) then L is also the language of an NFA.

Nondeterministic finite automata

Determinization

02140 Languages and Parsing, MF, Fall 2003 – p.17/19

Converting NFAs to DFAs: Subset construction

Thm: Given an NFA $A = (Q, \Sigma, \delta, q_0, F)$, a corresponding DFA $A' = (Q', \Sigma', \delta', q'_0, F')$ accepting the same language can be defined by

- $Q' \stackrel{\text{def}}{=} \mathcal{P}(Q)$,
- $\Sigma' \stackrel{\text{def}}{=} \Sigma$,
- $\delta'(q', a) \stackrel{\text{def}}{=} \bigcup_{q \in q'} \delta(q, a)$ for each $q' \in Q'$ and each $a \in \Sigma$,
- $q'_0 \stackrel{\text{def}}{=} \{q_0\}$,
- $F' \stackrel{\text{def}}{=} \{q' \subseteq Q \mid q' \cap F \neq \emptyset\}$.

02140 Languages and Parsing, MF, Fall 2003 – p.18/19

Converting NFAs to DFAs: Subset construction

Thm: Given an NFA $A = (Q, \Sigma, \delta, q_0, F)$, a corresponding DFA $A' = (Q', \Sigma', \delta', q'_0, F')$ accepting the same language can be defined by

- $Q' \stackrel{\text{def}}{=} \mathcal{P}(Q)$,
- $\Sigma' \stackrel{\text{def}}{=} \Sigma$,
- $\delta'(q', a) \stackrel{\text{def}}{=} \bigcup_{q \in q'} \delta(q, a)$ for each $q' \in Q'$ and each $a \in \Sigma$,
- $q'_0 \stackrel{\text{def}}{=} \{q_0\}$,
- $F' \stackrel{\text{def}}{=} \{q' \subseteq Q \mid q' \cap F \neq \emptyset\}$.

Exponential blow-up: $|Q'| = 2^{|Q|}$

02140 Languages and Parsing, MF, Fall 2003 – p.18/19

Converting NFAs to DFAs: Subset construction

Thm: Given an NFA $A = (Q, \Sigma, \delta, q_0, F)$, a corresponding DFA $A' = (Q', \Sigma', \delta', q'_0, F')$ accepting the same language can be defined by

- $Q' \stackrel{\text{def}}{=} \mathcal{P}(Q)$,
- $\Sigma' \stackrel{\text{def}}{=} \Sigma$,
- $\delta'(q', a) \stackrel{\text{def}}{=} \bigcup_{q \in q'} \delta(q, a)$ for each $q' \in Q'$ and each $a \in \Sigma$,
- $q'_0 \stackrel{\text{def}}{=} \{q_0\}$,
- $F' \stackrel{\text{def}}{=} \{q' \subseteq Q \mid q' \cap F \neq \emptyset\}$.

Exponential blow-up: $|Q'| = 2^{|Q|}$

Prf: It is easy to show (by induction) that $\hat{\delta}'(q', w) = \bigcup_{q \in q'} \hat{\delta}(q, w)$ for each $q' \subseteq Q$ and $w \in \Sigma^*$. Hence,

$$\begin{aligned} L(A') &= \{w \in \Sigma^* \mid \hat{\delta}'(q'_0, w) \in F'\} &&= \{w \in \Sigma^* \mid \hat{\delta}'(q'_0, w) \cap F \neq \emptyset\} \\ &= \{w \in \Sigma^* \mid \bigcup_{q \in \{q_0\}} \hat{\delta}(q, w) \cap F \neq \emptyset\} &&= \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\} = L(A) \end{aligned}$$

02140 Languages and Parsing, MF, Fall 2003 – p.18/19

Equivalence of DFA and NFA

Thm: *DFA and NFA define the same languages.*

Let L be a language. Then there is a DFA A_D with $L = L(A_D)$ iff there is an NFA A_N with $L = L(A_N)$.

Equivalence of DFA and NFA

Thm: *DFA and NFA define the same languages.*

Let L be a language. Then there is a DFA A_D with $L = L(A_D)$ iff there is an NFA A_N with $L = L(A_N)$.

Cor: The languages definable/recognizable by NFA are *exactly* the regular languages.

Equivalence of DFA and NFA

Thm: *DFA and NFA define the same languages.*

Let L be a language. Then there is a DFA A_D with $L = L(A_D)$ iff there is an NFA A_N with $L = L(A_N)$.

Cor: The languages definable/recognizable by NFA are *exactly* the regular languages.

Things to remember:

- DFA and NFA define the same languages.
- There are algorithms for converting NFA into DFA and vice versa.
- Converting an NFA into a DFA, an exponential blow-up in size may be incurred.