Mathematics for linguists

Gerhard Jäger
gerhard.jaeger@uni-tuebingen.de

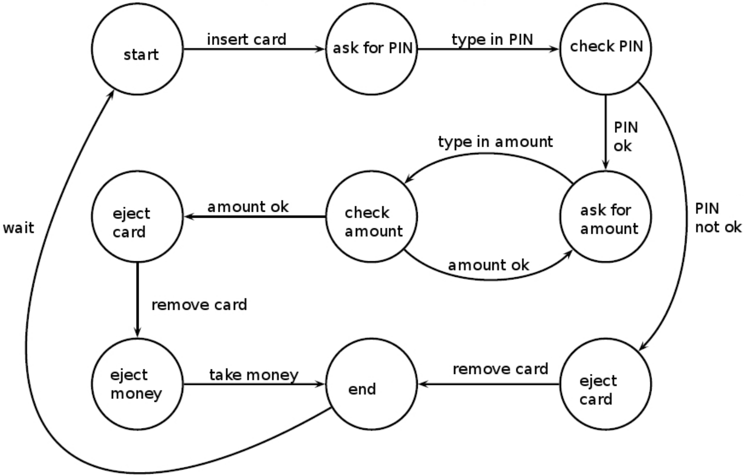
Uni Tübingen, WS 2009/2010

November 19, 2009

Automata (informally)

- imaginary machine/abstract model of a machine
- behaves according to certain rules.
- behavior of the automata depends on information, that the automate receives from the environment
- automata "make decisions"

An example

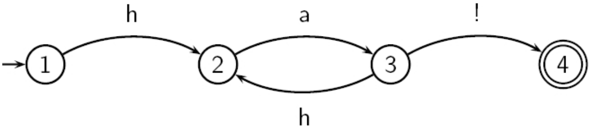


start
insert card
ask for PIN
type in PIN
check PIN
PIN
ok
PIN
not ok
type in amount
eject
card
amount ok
check
amount
ask for
amount
amount ok
wait
remove card
eject
money
take money
end
remove card
eject
card

Language automata

- automaton receives *input* from it environment (for instance key stroke by user)
- input can be represetned as string of symbols from an alaphabet (in the simplest case, these are just “0” and “1”)
- automaton produces *output*
- can also be represented as string of symbols

The laughing automaton (according to Stefan Müller)



Finite automata

- A **finite automaton**
  - has finitely many states,
  - receives as input strings over some alphabet Σ,
  - returns as output either "yes" or "no"
- A finite automaton thus defines a formal language — the set of inputs for which it returns the symbol "yes"

Finite automata

Definition (Deterministic finite automaton)

A *deterministic finite automaton* (DFA) M is a 5-tuple

$$M = \langle K, \Sigma, \delta, q_0, F \rangle$$

Here K is the set of *states* and Σ the *input alphabet*, $K \cap \Sigma = \emptyset$. K and Σ are finite sets. $q_0 \in K$ is the *initial state*, $F \subseteq K$ is the set of *final states*, and $\delta : K \times \Sigma \to K$ is the *transition function*.

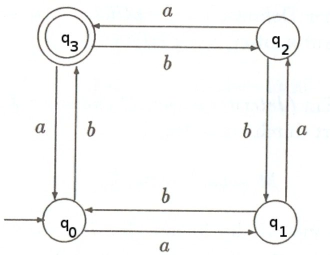
Finite automata: example

Let $M = \langle K, \Sigma, \delta, q_0, F \rangle$, where

$$
\begin{aligned}
K &= \{q_0, z_1, z_2, z_3\} \\
\Sigma &= \{a, b\} \\
F &= \{z_3\} \\
\delta(q_0, a) &= z_1 \\
\delta(q_0, b) &= z_3 \\
\delta(z_1, a) &= z_2 \\
\delta(z_1, b) &= q_0 \\
\delta(z_2, a) &= z_3 \\
\delta(z_2, b) &= z_1 \\
\delta(z_3, a) &= q_0 \\
\delta(z_3, b) &= z_2
\end{aligned}
$$

Finite automata: example

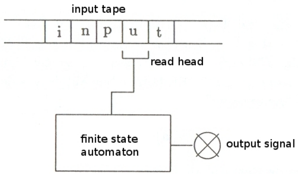
Finite automata can be represented as graphs:



- initial state is represented by an arrow
- final states are marked by double circle
- transition function is represented by labeled directed edges

Finite automata

- intuition:
  - automaton starts at initial state
  - input is written on some input tape (like a punchcard)
  - Per temporal unit, the automaton reads a symbol α on the input tape and moves along an arrow with the label α towards a new state
  - If the automaton is in a final state after reading the entire input tape, the string on the input tape is *accepted* (output: "yes")
  - else the string is not accepted (output: "no")



Question: which language is accepted by the automaton from the example?

Finite automata and formal languages

Definition

For a given DFA $M = \langle K, \Sigma, \delta, q_0, F \rangle$ we define a function $\hat{\delta} : K \times \Sigma^* \to K$ via a recursive definition as follows:

$$\begin{aligned} \hat{\delta}(z, \epsilon) &= z \\ \hat{\delta}(z, a\vec{x}) &= \hat{\delta}(\delta(z, a), \vec{x}) \end{aligned}$$

Here it holds that $z \in K, \vec{x} \in \Sigma^*$ and $a \in \Sigma$.
The language that is *accepted* by M is

$$L(M) = \{\vec{x} \in \Sigma^* | \hat{\delta}(q_0, \vec{x}) \in F\}$$

Finite automata and formal languages

- definition of $\hat{\delta}$ extends definition of δ from single symbols to strings of symbols
- for single symbols, it holds that: $\hat{\delta}(z, a) = \delta(z, a)$
- it also holds that

$$\hat{\delta}(z, a_1 a_2 \dots a_n) = \delta(\dots \delta(\delta(z, a_1), a_2) \dots, a_n)$$

Theorem

Every language that is accepted by a deterministic finite automaton is regular (Type 3 in the Chomsky hierarchy).

Idea of proof

Let

$$M = \langle K, \Sigma, \delta, q_0, F \rangle$$

be a DFA. We construct a regular grammar

$$G = \langle V_T, V_N, S, R \rangle$$

as follows:

- $V_T = \Sigma$
- $V_N = K$
- $S = q_0$

Idea of proof

- For every transition

$$\delta(z_1, a) = z_2$$

there is a rule

$$z_1 \longrightarrow a z_2$$

- If $z_2 \in F$, there is the additional rule

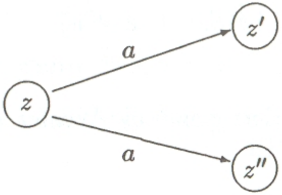
$$z_1 \longrightarrow a$$

- If $q_0 \in F$, there is the additional rule

$$q_0 \longrightarrow \epsilon$$

Non-deterministic automata

- With a *deterministic automaton*, it is uniquely determined for each state and each input symbol, into which state the automaton moves
- With a *non-deterministic* automaton it may be due to chance into which state the automaton moves
- In a non-deterministic automaton, δ need not be a function, but it is a relation.



Non-deterministic automaton

Definition (Non-deterministic finite automaton[1])

A *non-deterministic finite automaton* (NFA) M is a ein 5-tuple

$$M = \langle K, \Sigma, \delta, q_0, F \rangle$$

Here

- K is a finite set, the set of *states*,
- Σ is a finite set, the *input alphabet*, with $K \cap \Sigma = \emptyset$,
- $\delta \subseteq K \times \Sigma \times K$ is a relation, the *transition relation*,
- q_0 is the initial state, and
- $F \subseteq K$ is the set of final states.

[1] Differs in an inessential way from PtMW.

Non-deterministic automata

The non-deterministic transition relation can also be extended to a relation $\hat{\delta} \subseteq K \times \Sigma^* \times K$ for strings of symbols:

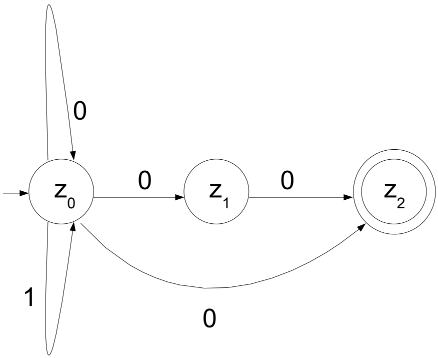
$$\begin{array}{lll} \hat{\delta}(q, \epsilon, q) & & \text{for all } q \in K \\ \hat{\delta}(q_1, a\vec{x}, q_2) & \text{iff} & \delta(q_1, a, q_3), \hat{\delta}(q_3, \vec{x}, q_2) \text{ for some } q_3 \in K \end{array}$$

The language $L(M)$ that is *accepted* by a NFA M is defined as

$$L(M) = \{\vec{x} \in \Sigma^* | \text{there is a } q \in F \text{ such that } \hat{\delta}(q_0, \vec{x}, q)\}$$

Non-deterministic automata

- example:
  - the following NFA accepts all words $\vec{x}$ over $\{0, 1\}$ that end in 0.



Non-deterministic automata

Theorem

Every language that is accepted by a NFA is also accepted by some DFA.

Idea of proof

Let

$$M_1 = \langle K, \Sigma, \delta, q_0, F \rangle$$

be a non-deterministic finite automaton. We construct a corresponding finite automaton

$$M' = \langle K', \Sigma', \delta', q'_0, F' \rangle$$

in the following way:

- $K' = \wp(K)$
- $\Sigma' = \Sigma$
- $\delta'(q'_1, a) = \{q \in K | \text{there is a } q_1 \in q'_1 \text{ such that } \delta(q_1, a, q)\}$
- $q'_0 = \{q_0\}$
- $F' = \{q' \in \wp(K) | q' \cap F \neq \emptyset\}$

M' accepts the same language as M.

Finite automata and regular grammars

Theorem

For every regular grammar

$$G = \langle V_T, V_N, S, R \rangle$$

there is a NFA

$$M = \langle K, \Sigma, \delta, q_0, F \rangle$$

with

$$L(G) = L(M)$$

Idea of proof

We assume that every rule R has the form $A \to aB$, $A \to a$ or $S \to \epsilon$. Every regular grammar can be transformed into this form.
We construct M as follows:

- $K = V_N \cup \{z_\omega\}$
- $\Sigma = V_T$
- $\delta(z_1, a, z_2)$ if $z_1 \to az_2 \in R$
- $\delta(z_1, a, z_\omega)$ if $z_1 \to a \in R$
- $q_0 = S$
- If $S \to \epsilon \in R$, $F = \{q_0, z_\omega\}$; otherwise $F = \{z_\omega\}$

M accepts exactly the language that is generated by G.

Finite automata and regular languages

Theorem

Both deterministic and non-deterministic finite automata accept exactly the regular languages.