Finite Automata

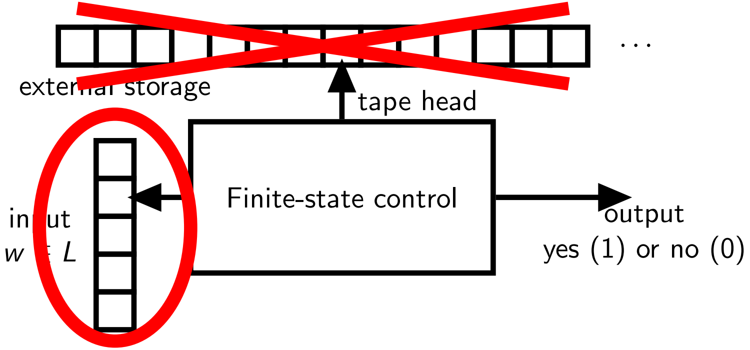
5DV037 — Fundamentals of Computer Science

Umeå University
Department of Computing Science

Stephen J. Hegner
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

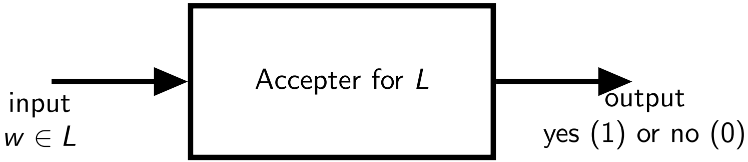
The Idea of Deterministic Finite Automata

- Recall the general form of an accepter.
- In a finite automaton, there is no external storage.
- The input is consumed left-to-right, one character at a time, with no possibility to move left and re-read.



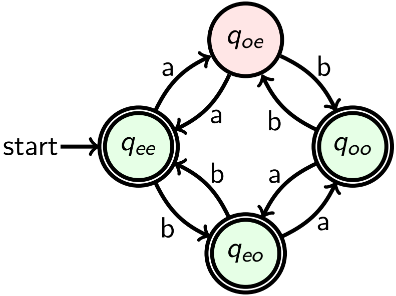
The Idea of Deterministic Finite Automata

- Recall the general form of an accepter.
- In a finite automaton, there is no external storage.
- The input is consumed left-to-right, one character at a time, with no possibility to move left and re-read.
- This picture is thus more representative.



An Example to Illustrate the Idea

- Let $\Sigma = \{a, b\}$.
  $L = \{w \in \Sigma^* \mid \text{Count}\langle a, w\rangle \text{ is even or Count}\langle b, w\rangle \text{ is odd}\}$
- Design a deterministic finite-state accepter for L.



State	Count$\langle a, u\rangle$	Count$\langle b, u\rangle$	Accept
q_{ee}	even	even	yes
q_{oe}	odd	even	no
q_{eo}	even	odd	yes
q_{oo}	odd	odd	yes

u = part of input already processed.

- States are represented as labelled circles.
- Transitions between states are represented as labelled arrows.
- The start state is identified by an inward arrow.
- Accepting states are identified by concentric circles.

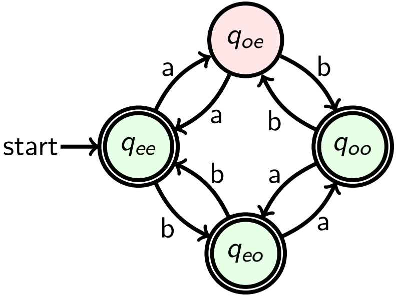
Formalization of Deterministic Finite Automata

A *deterministic finite-state automaton* or *deterministic finite-state accepter* (*DFA*) is a five-tuple

$$M = (Q, \Sigma, \delta, q_0, F)$$

in which

- Q is finite set of *states*;
- Σ is an alphabet, called the *input alphabet*;
- $\delta : Q \times \Sigma \to Q$ is a total function, the *state-transition function*;
- $q_0 \in Q$ is the *initial state*;
- $F \subseteq Q$ is the set of *final* or *accepting states*.



$Q = \{q_{ee}, q_{eo}, q_{oe}, q_{oo}\}$; $q_0 = q_{ee}$.

State q	$\delta(q, a)$	$\delta(q, b)$	$q \in F$
q_{ee}	q_{oe}	q_{eo}	yes
q_{oe}	q_{ee}	q_{oo}	no
q_{eo}	q_{oo}	q_{ee}	yes
q_{oo}	q_{eo}	q_{oe}	yes

The Extended Transition Function and Acceptance

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. The *extended transition function* or *run map*
$$\delta^* : Q \times \Sigma^* \to Q$$
extends $\delta : Q \times \Sigma \to Q$ to input strings.
- It is defined inductively as follows.
  - $\delta^*(q, \lambda) = q$ for any $q \in Q$;
  - $\delta^*(q, \alpha \cdot a) = \delta(\delta^*(q, \alpha), a)$ for any $q \in Q$, $\alpha \in \Sigma^*$, and $a \in \Sigma$.
- The *language accepted by M* is the set of all strings which drive M from its initial state to an accepting state.
- Formally,
$$\mathcal{L}(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$$
- Given $L \subseteq \Sigma^*$, M is called a *deterministic finite-state accepter* for L if $\mathcal{L}(M) = L$.

A Larger Example

- Let $\Sigma = \{0, 1\}$, and define
  $L = \{\alpha \in \Sigma^* \mid \text{Length}(\alpha) \geq 10$
  and the 10^{th} element from the right is a $1\}$.
- Design a DFA which accepts L.
- Such an accepter must have (at least) $2^{10} = 1024$ states.
- Define:
  - $Q = \{q_\beta \mid \beta \in \{0, 1\}^* \text{ and Length}(\beta) = 10\}$;
  - $q_0 = q_{0000000000}$;
- The transition function operates as shift left and append:
  - $\delta(q_\beta, x) = q_{\text{Rest}\langle\beta\rangle \cdot x}$.
- The accepting states are $F = \{q_\beta \in Q \mid \text{First}\langle\beta\rangle = 1\}$ with $\text{First}\langle\beta\rangle$ the leftmost element of β.
- Then $(Q, \Sigma, \delta, q_0, F)$ is a deterministic finite-state accepter for L.

Instantaneous Descriptions and the Move Relation

- An *instantaneous description* (or *machine configuration* or *ID*) for the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is a pair $(q, \alpha) \in Q \times \Sigma*$ in which:
  - q represents the current state;
  - α represents the part of the input string which has not yet been read.
- $\mathrm{ID}\langle M \rangle = Q \times \Sigma^*$; the set of all possible IDs of M.
- The *move relation* $\vdash_M \subseteq \mathrm{ID}\langle M \rangle \times \mathrm{ID}\langle M \rangle$ represents one step of M and is defined by $(q_1, \alpha_1) \vdash_M (q_2, \alpha_2)$ iff
  - $\alpha_2 = \mathrm{Rest}\langle \alpha_1 \rangle$; and
  - $\delta(q_1, \mathrm{First}\langle \alpha_1 \rangle) = q_2$.
- Thus $(q, a_1 a_2 \ldots a_k) \vdash_M (\delta(q, a_1), a_2 \ldots a_k)$.
- $\vdash_M^*$ is the reflexive and transitive closure of $\vdash_M$:
  - $(q, \alpha) \vdash_M^* (q, \alpha)$;
  - $(q_1, \alpha_1) \vdash_M^* (q_2, \alpha_2),\ (q_2, \alpha_2) \vdash_M^* (q_3, \alpha_3) \ \Rightarrow\ (q_1, \alpha_1) \vdash_M^* (q_3, \alpha_3)$.
- Thus $(q, \alpha_1 \alpha_2) \vdash_M^* (\delta^*(q, \alpha_1), \alpha_2)$.
- For a DFA, $\vdash_M$ and $\vdash_M^*$ are functions.

Computations and the Language Accepted by a DFA

- The *computation* of M on $\alpha \in \Sigma^*$ is the sequence

$$(q_0, \alpha) = (q_0, \alpha_0) \vdash_M (q_1, \alpha_1) \vdash_M \ldots \vdash_M (q_m, \alpha_m) = (q_m, \lambda)$$

- In the above, $\alpha_{i+1} = \text{Rest}\langle \alpha_i \rangle$ and $q_{i+1} = \delta(q_i, \text{First}\langle \alpha_i \rangle)$.
- The language of a DFA may be characterized succinctly using computations.

Observation: For any DFA $M = (Q, \Sigma, \delta, q_0, F)$,

$$\mathcal{L}(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha) \vdash_M^* (q_f, \lambda) \text{ with } q_f \in F\}. \quad \square$$

- This flavor of representation of the language of a machine will prove very useful in the more complex models of computation which will follow.

The Class of Languages Accepted by DFAs

Question: How is the class of languages which are accepted by DFAs characterized?

- Begin with a definition.
- The class of all languages (over a given alphabet Σ) which are accepted by some DFA is called the *regular languages (over Σ)*.
- The next task is to look for alternate characterizations for regular languages. There are several.
  - Alternate forms of finite automata:
    - nondeterministic finite automata
    - finite automata with λ-transitions
  - Other types of language characterization:
    - regular expressions
    - regular grammars

Nondeterministic Finite Automata

A *nondeterministic finite-state automaton* or *nondeterministic finite-state accepter* (*NFA*) is a five-tuple

$$M = (Q, \Sigma, \delta, q_0, F)$$

in which everything is the same as in a DFA except that

- $\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$.

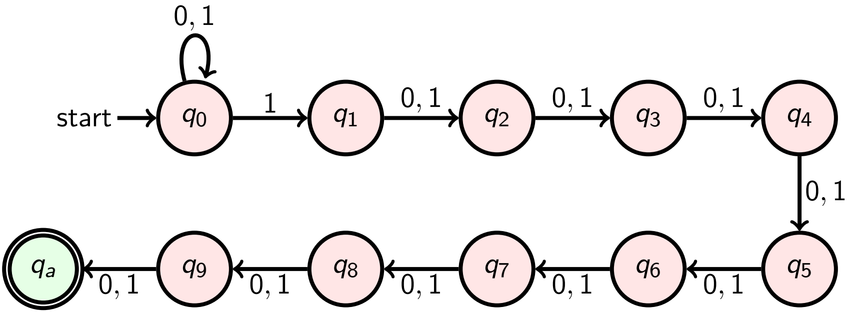
- Note that there are three significant differences between a DFA and an NFA:
  - The transition function is *nondeterministic*; that is, there is a set of possible next states as opposed to a single possibility.
  - The set of possible next states may in fact be empty, so there is not necessarily even one possible next state.
  - So-called *λ-transitions* are allowed in which no input symbol is consumed.
- Every DFA may be viewed as an NFA:
  - $M = (Q, \Sigma, \delta, q_0, F) \rightsquigarrow \tilde{M} = (Q, \Sigma, \tilde{\delta}, q_0, F)$ with $\tilde{\delta} : Q \times \Sigma \to 2^Q$ given by $(q, a) \mapsto \{\delta(q, a)\}$.

The Run Map and Acceptance for NFAs

- To define δ^* for an NFA $M = (Q, \Sigma, \delta, q_0, F)$, it is convenient to define and use the move relation.
- Define $(q_1, \alpha_1) \vdash_M (q_2, \alpha_2)$ to hold if either
  - $\alpha_2 = \mathrm{Rest}\langle\alpha_1\rangle$ and $q_2 \in \delta(q_1, \mathrm{First}\langle\alpha_1\rangle)$; or
  - $\alpha_2 = \alpha_1$ and $q_2 \in \delta(q_1, \lambda)$.
- Define $\vdash_M^*$ to be the reflexive and transitive closure of $\vdash_M$, just as for the DFA case.
- Note that $\vdash_M$ and $\vdash_M^*$ are not necessarily functions in the case of an NFA.
- Define $\delta^* : Q \times \Sigma^* \to 2^Q$ via $q' \in \delta^*(q, \alpha)$ iff $(q, \alpha) \vdash_M^* (q' \lambda)$.
- Define $\mathcal{L}(M) = \{\alpha \in \Sigma^* \mid \delta^*(q_0, \alpha) \cap F \neq \emptyset\}$.
- Thus, the NFA M accepts a string $\alpha \in \Sigma^*$ if *some* computation reads the entire input and winds up in an accepting state, and rejects that string if *no* computation has that property.

An Example of Acceptance by an NFA

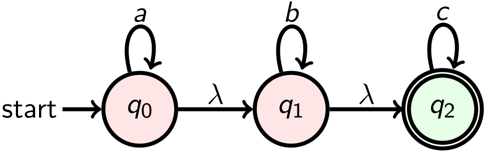
- Let $\Sigma = \{0, 1\}$, and define
  $L = \{\alpha \in \Sigma^* \mid \text{Length}(\alpha) \geq 10$
  and the 10^{th} element from the right is a $1\}$.
- Design an NFA which accepts L.



- Note that this nondeterministic accepter has only 10 states, as opposed to 1024 for the deterministic version.

An Example with λ-Transitions

- Let $\Sigma = \{a, b, c\}$ and let $L = \{a^i b^j c^k \mid i, j, k \in \mathbb{N}\}$.
- Here is a simple NFA accepter for L which makes use of λ-transitions.



Formulation of the Equivalence Theorem

Theorem: Given any NFA M, there is a DFA M' with $\mathcal{L}(M') = \mathcal{L}(M)$. $\square$

- In other words, NFAs and DFAs are equal in accepting power.
- The idea of the proof is rather simple.
  - Let $M = (Q, \Sigma, \delta, q_0, F)$ be the given NFA.
  - The set of states of M' is 2^Q.
  - There is a transition

    $$\delta'(S, a) = S'$$

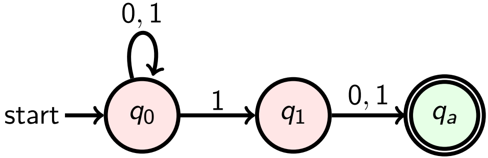
    in the DFA iff there are $q \in S$ and $q' \in S'$ with the property that $q' \in \delta^*(q, a)$.
- The algorithm also eliminates *unreachable* states.
- It is summarized on the next slide.

The NFA-to-DFA Conversion Algorithm

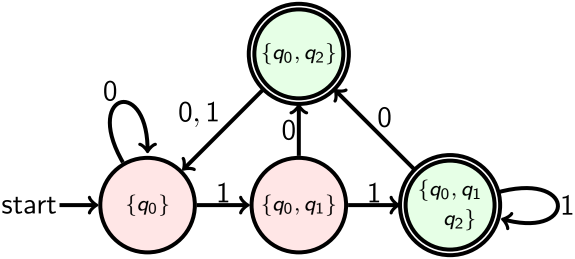
Input : An NFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: An equivalent DFA $M' = (Q', \Sigma, \delta', \{q_0\}, F')$
Pool $\leftarrow \{\{q_0\}\}$; Q' $\leftarrow \emptyset$; DFA_Transitions $\leftarrow \emptyset$;
while Pool $\neq \emptyset$ **do**
  choose $S \in$ Pool;
  Pool $\leftarrow$ Pool $\setminus \{S\}$; Q' $\leftarrow$ Q' $\cup \{S\}$;
  foreach $x \in \Sigma$ **do**
    NewState $\leftarrow \bigcup\{\delta^*(s, x) \mid s \in S\}$;
    DFA_Transitions $\leftarrow$ DFA_Transitions $\cup \{\delta'(S, x) =$ NewState$\}$;
    if NewState $\notin$ Q' $\cup$ Pool **then** Pool $\leftarrow$ Pool $\cup$ {NewState};
  end
end
$\delta' \leftarrow$ DFA_Transitions; $F' \leftarrow \{S \in Q' \mid S \cap F \neq \emptyset\}$;
if $\delta^*(q_0, \lambda) \cap F \neq \emptyset$ **then** $F' \leftarrow F' \cup \{\{q_0\}\}$;

Example of Conversion of an NFA to an Equivalent DFA

- Let $\Sigma = \{0,1\}$, and define
  $L = \{\alpha \in \Sigma^* \mid \text{Length}(\alpha) \geq 2,\ 2^{\text{nd}} \text{ element from the right is a } 1\}$.
- Here is an NFA which accepts L.

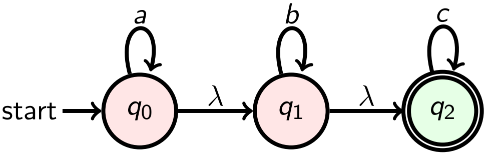


- The corresponding DFA according to the construction:



Example of Conversion with λ-Transitions

- Let $\Sigma = \{a, b, c\}$ and let $L = \{a^i b^j c^k \mid i, j, k \in \mathbb{N}\}$.



- The corresponding DFA according to the construction.