Finite Automata and Formal Languages

TMV026/DIT321– LP4 2012

Lecture 4
Ana Bove

March 19th 2012

Overview of today's lecture:

- Deterministic Finite Automata

Deterministic Finite Automata

Definition: A *deterministic finite automaton* (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ consisting of:

1. A finite set Q of *states*;
2. A finite set Σ of *symbols* (alphabet);
3. A *transition function* $\delta : Q \times \Sigma \to Q$ (total function that takes as argument a state and a symbol and returns a state);
4. A *start state* $q_0 \in Q$;
5. A set $F \subseteq Q$ of *final* or *accepting* states.

Example: DFA

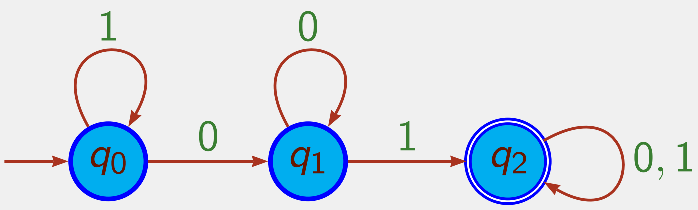
Let the DFA $(Q, \Sigma, \delta, q_0, F)$ be given by:

$$Q = \{q_0, q_1, q_2\}$$
$$\Sigma = \{0, 1\}$$
$$F = \{q_2\}$$
$$\delta : Q \times \Sigma \to Q$$
$$\delta(q_0, 0) = q_1 \qquad \delta(q_1, 0) = q_1 \qquad \delta(q_2, 0) = q_2$$
$$\delta(q_0, 1) = q_0 \qquad \delta(q_1, 1) = q_2 \qquad \delta(q_2, 1) = q_2$$

What does it do?

How to Represent a DFA?

Transition Diagram: As we have seen before.



Transition Table:

δ	0	1
$\to q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

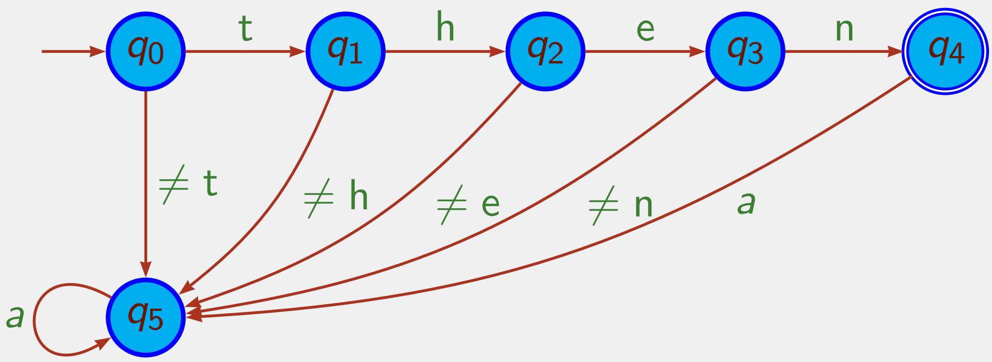
The start state is indicated with $\to$.
The final states are indicated with $*$.

When Does a DFA Accept a Word?

When reading the word the automaton moves according to δ.

Definition: If after reading the input it stops in a final state, it *accepts* the word.

Example:

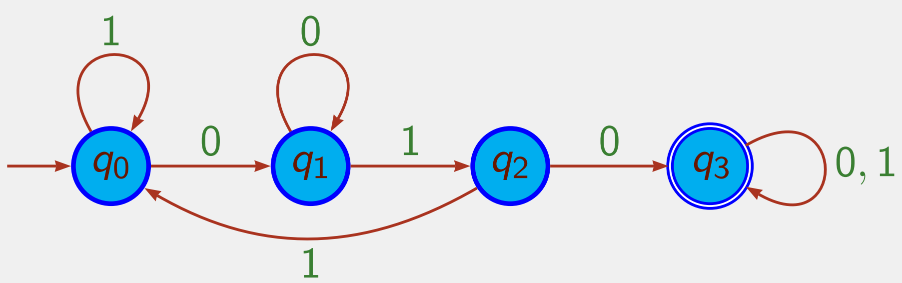


Only the word "then" is accepted.
We have a (non-accepting) *stop* or *dead* state q_5.

Example: DFA

Let us build an automaton that accepts the words that contain 010 as a subword.
That is, given $\Sigma = \{0, 1\}$ we want to accept words in
$\mathcal{L} = \{x010y \mid x, y \in \Sigma^*\}$.

Solution: $(\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$ given by



δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

Extending the Transition Function to Strings

How can we compute what happens when we read a certain word?

Definition: We extend δ to strings as $\hat{\delta} : Q \times \Sigma^* \to Q$.
We define $\hat{\delta}(q, x)$ by recursion on x.

$$\begin{array}{l} \hat{\delta}(q, \epsilon) = q \\ \hat{\delta}(q, ax) = \hat{\delta}(\delta(q, a), x) \end{array}$$

Note: $\hat{\delta}(q, a) = \delta(q, a)$ since the string $a = a\epsilon$.
$\hat{\delta}(q, a) = \hat{\delta}(q, a\epsilon) = \hat{\delta}(\delta(q, a), \epsilon) = \delta(q, a)$

Example: In the previous example, what are $\hat{\delta}(q_0, 10101)$ and $\hat{\delta}(q_0, 00110)$?

Some Properties

Proposition: *For any words x and y, and for any state q we have that* $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$.

Proof: We prove the result by induction on x.

Basis case: $\hat{\delta}(q, \epsilon y) = \hat{\delta}(q, y) = \hat{\delta}(\hat{\delta}(q, \epsilon), y)$.

Inductive step: Our IH is that $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ for any word y and any state q. We should prove that $\hat{\delta}(q, (ax)y) = \hat{\delta}(\hat{\delta}(q, ax), y)$.

$$\begin{array}{lll} \hat{\delta}(q, (ax)y) & = \hat{\delta}(q, a(xy)) & \text{by def of concat} \\ & = \hat{\delta}(\delta(q, a), xy) & \text{by def of } \hat{\delta} \\ & = \hat{\delta}(\hat{\delta}(\delta(q, a), x), y) & \text{by IH} \\ & = \hat{\delta}(\hat{\delta}(q, ax), y) & \text{by def of } \hat{\delta} \end{array}$$

Another Definition of $\hat{\delta}$

Recall that we have 2 descriptions of words: $a(b(cd)) = ((ab)c)d$.

We can define $\hat{\delta}'$ as follows:

$$\hat{\delta}'(q, \epsilon) = q$$
$$\hat{\delta}'(q, xa) = \delta(\hat{\delta}'(q, x), a)$$

Proposition: $\forall x.\forall q.\ \hat{\delta}(q, x) = \hat{\delta}'(q, x)$.

Proof: Observe that xa is a special case of xy where $y = a$.
Basis case is trivial.
The inductive step goes as follows:

$$\begin{array}{llll} \hat{\delta}(q, xa) & = \hat{\delta}(\hat{\delta}(q, x), a) & & \text{by previous prop} \\ & = \delta(\hat{\delta}(q, x), a) & & \text{by def of } \hat{\delta} \\ & = \delta(\hat{\delta}'(q, x), a) & & \text{by IH} \\ & = \hat{\delta}'(q, xa) & & \text{by def of } \hat{\delta}' \end{array}$$

Language Accepted by a DFA

Definition: The *language* accepted by the DFA $(Q, \Sigma, \delta, q_0, F)$ is the set $\mathcal{L} = \{x \mid x \in \Sigma^*, \hat{\delta}(q_0, x) \in F\}$.

Example: In the example on slide 5, 10101 is accepted but 00110 is not.

Note. We could write a program that simulates a DFA and let the program tell us whether a certain string is accepted or not.

Functional Representation of a DFA Accepting $x010y$

```
data Q = Q0 | Q1 | Q2 | Q3
data S = O | I

final :: Q -> Bool
final Q3 = True
final _ = False

delta :: Q -> S -> Q
delta Q0 O = Q1
delta Q0 I = Q0
delta Q1 O = Q1
delta Q1 I = Q2
delta Q2 O = Q3
delta Q2 I = Q0
delta Q3 _ = Q3
```


Functional Representation of a DFA Accepting $x010y$

```
run :: Q -> [S] -> Q
run q [] = q
run q (a:xs) = run (delta q a) xs

accepts :: [S] -> Bool
accepts xs = final (run Q0 xs)
```

Alternatively, given that

```
run q [x1,...,xn] = delta (... (delta Q0 x1) ...) xn
```

then

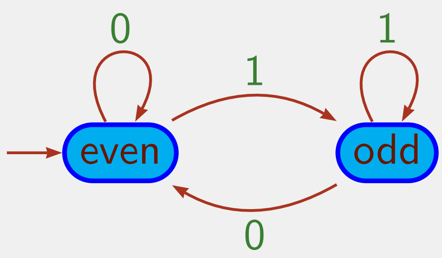
```
run :: Q -> [S] -> Q
run = foldl delta

accepts :: [S] -> Bool
accepts = final . run Q0
```

Accepting by End of String

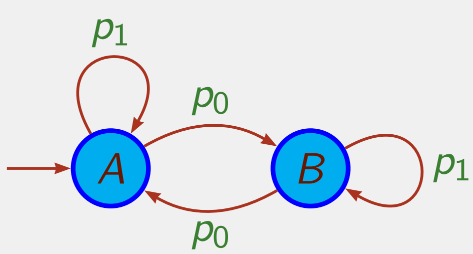
Sometimes we use an automaton to identify properties of a certain string. In these cases, the important thing is the state the automaton is in when we finish reading the input.
Here, the the set of final states is actually not needed and can be omitted.

Example: The following automaton determines whether a binary number is even or odd.

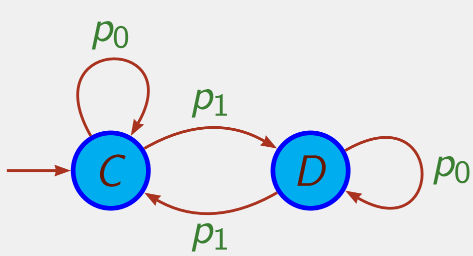


Example: Product of Automata

Given an automaton that determines whether the number of p_0's is even or odd



and an automaton that determines whether the number of p_1's is even or odd



How can we combine them to keep track of the parity of *both* p_0 and p_1?

Product Construction

Definition: Given two DFA $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ with the *same alphabet* Σ, we can define the *product* $D = D_1 \times D_2$ as follows:

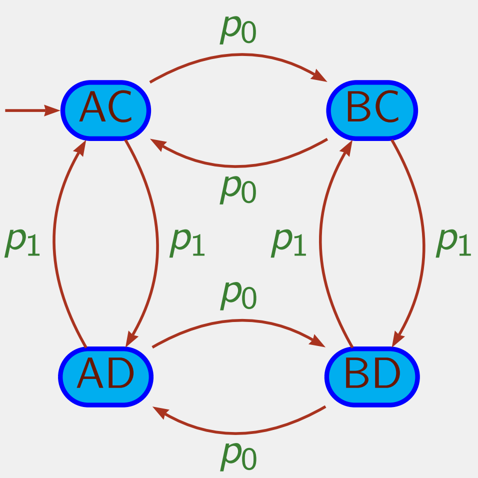
- $Q = Q_1 \times Q_2$
- $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- $q_0 = (q_1, q_2)$
- $F = F_1 \times F_2$

Proposition: $\hat{\delta}((r_1, r_2), x) = (\hat{\delta}_1(r_1, x), \hat{\delta}_2(r_2, x))$.

Proof: By induction on x.

Example: Product of Automata (cont.)

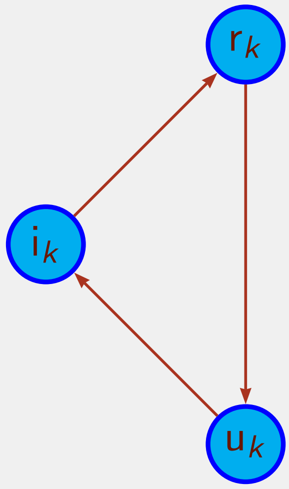
The product automaton that keeps track of the parity of *both* p_0 and p_1 is:



If after reading the word w we are in the state AD we know that w contains an even number of p_0's and an odd number of p_1's.

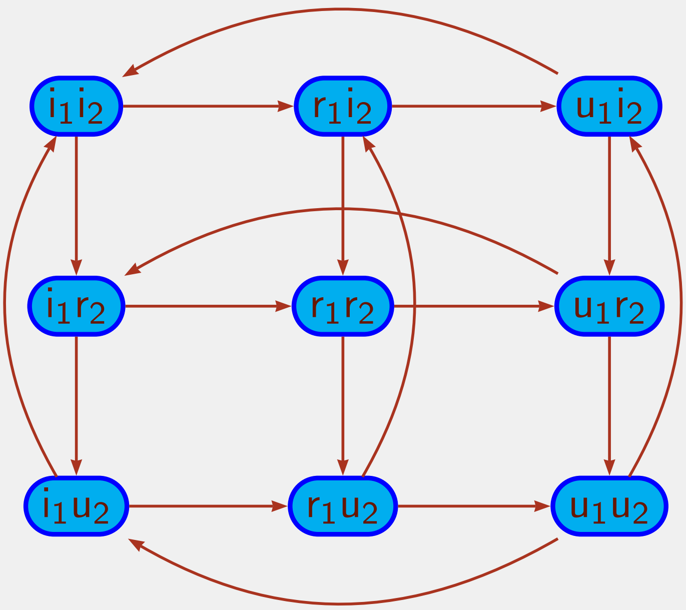
Example: Product of Automata

Let us model a system where users have three states: *idle*, *requesting* and *using*.
Let us assume we have 2 users.
Each user is represented by a simple automaton, for $k = 1, 2$:



Example: Product of Automata (cont.)

The complete system is represented by the product of these 2 automata and it has 3 * 3 = 9 states.



Language Accepted by a Product Automaton

Proposition: *Given two DFA D_1 and D_2, then* $\mathcal{L}(D_1 \times D_2) = \mathcal{L}(D_1) \cap \mathcal{L}(D_2)$.

Proof: $\hat{\delta}(q_0, x) = (\hat{\delta}_1(q_1, x), \hat{\delta}_2(q_2, x)) \in F$ iff $\hat{\delta}_1(q_1, x) \in F_1$ and $\hat{\delta}_2(q_2, x) \in F_2$, that is, $x \in \mathcal{L}(D_1)$ and $x \in \mathcal{L}(D_2)$ so $x \in \mathcal{L}(D_1) \cap \mathcal{L}(D_2)$.

Example: Let M_k be an automaton that accepts multiples of k such that $\mathcal{L}(M_k) = \{a^n \mid k \text{ divides } n\}$.
Then $M_6 \times M_9$ is M_{18} (6 divides k and 9 divides k iff 18 divides k.)

Note: It can be quite difficult to directly build an automaton accepting the intersection of two languages.

Example: Build a DFA for the language that contains the subword *abb* twice and an even number of *a*'s.

Variation of the Product

Definition: We define $D_1 \oplus D_2$ similarly to $D_1 \times D_2$ but with a different notion of accepting state:

a state (r_1, r_2) is accepting iff $r_1 \in F_1$ *or* $r_2 \in F_2$

Proposition: *Given two DFA D_1 and D_2, then* $\mathcal{L}(D_1 \oplus D_2) = \mathcal{L}(D_1) \cup \mathcal{L}(D_2)$.

Example: We define the automaton accepting multiples of 3 or of 5 by taking $M_3 \oplus M_5$.

Complement

Definition: Given the automaton $D = (Q, \Sigma, \delta, q_0, F)$ we define the *complement* $\overline{D}$ of D as the automaton $\overline{D} = (Q, \Sigma, \delta, q_0, Q - F)$.

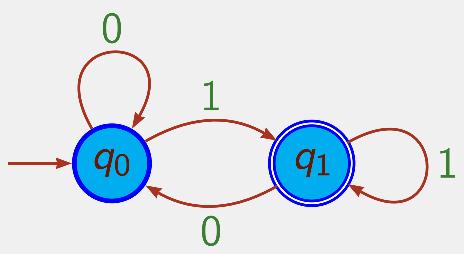
Proposition: *Given a DFA D we have that $\mathcal{L}(\overline{D}) = \Sigma^* - \mathcal{L}(D)$.*

Remark: We have that $D_1 \oplus D_2 = \overline{\overline{D_1} \times \overline{D_2}}$.

Accessible Part of a DFA

Consider the DFA $(\{q_0, \ldots, q_3\}, \{0, 1\}, \delta, q_0, \{q_1\})$ given by

This is clearly equivalent to the DFA



which is the *accessible* part of the DFA. The states q_2 and q_3 are not accessible from the start state and can be removed.

Accessible States

Definition: The set $\text{Acc} = \{\hat{\delta}(q_0, x) \mid x \in \Sigma^*\}$ is the set of *accessible* states (from the state q_0).

Proposition: *If $D = (Q, \Sigma, \delta, q_0, F)$ is a DFA, then $D' = (Q \cap \text{Acc}, \Sigma, \delta', q_0, F \cap \text{Acc})$, where δ' is the function δ restricted to the states in $Q \cap \text{Acc}$, is a DFA such that $\mathcal{L}(D) = \mathcal{L}(D')$.*

Proof: Notice that D' is well defined and that $\mathcal{L}(D') \subseteq \mathcal{L}(D)$.
If $x \in \mathcal{L}(D)$ then $\hat{\delta}(q_0, x) \in F$. Observe that by definition $\hat{\delta}(q_0, x) \in \text{Acc}$.
Hence $\hat{\delta}(q_0, x) \in F \cap \text{Acc}$ and then $x \in \mathcal{L}(D')$.

Regular Languages

Recall: Given an alphabet Σ, a *language* $\mathcal{L}$ is a subset of Σ^*, that is, $\mathcal{L} \subseteq \Sigma^*$.

Definition: A language $\mathcal{L} \subseteq \Sigma^*$ is *regular* iff there exists a DFA D on the alphabet Σ such that $\mathcal{L} = \mathcal{L}(D)$.

Proposition: *If $\mathcal{L}_1$ and $\mathcal{L}_2$ are regular languages then so are $\mathcal{L}_1 \cap \mathcal{L}_2$, $\mathcal{L}_1 \cup \mathcal{L}_2$ and $\Sigma^* - \mathcal{L}_1$.*

Proof: ...

Automatic Theorem Proving

Recall the example $f, g, h : \mathbb{N} \to \{0,1\}$ such that:

$$f(0) = 0 \qquad g(0) = 1 \qquad h(0) = 0$$
$$f(n+1) = g(n) \qquad g(n+1) = f(n) \qquad h(n+1) = 1 - h(n)$$

We can prove $\forall n.\ h(n) = f(n)$ automatically using a DFA.

- $Q = \{0,1\} \times \{0,1\} \times \{0,1\}$
- $\Sigma = \{1\}$ (The number n is represented by 1^n and 0 by $1^0 = \epsilon$)
- $q_0 = (f(0), g(0), h(0)) = (0, 1, 0)$.
- $\hat{\delta}((0,1,0), 1^n) = (f(n), g(n), h(n))$
  A transition goes from $(f(n), g(n), h(n))$ to $(f(n+1), g(n+1), h(n+1))$ and then $\delta((a, b, c), s) = (b, a, 1 - c)$

We check that all accessible states (a, b, c) satisfy $a = c$, that is, the property $a = c$ is an invariant for each transition of the automata

Automatic Theorem Proving

A more complex example:

$$f(0) = 0 \qquad f(1) = 1 \qquad f(n+2) = f(n) + f(n+1) - 2f(n)f(n+1)$$

We have

$$f(0) = 0 \quad f(1) = 1 \quad f(2) = 1 \quad f(3) = 0 \quad f(4) = 1 \quad f(5) = 1 \quad \ldots$$

Show that $f(n+3) = f(n)$ by using

- $Q = \{0,1\} \times \{0,1\} \times \{0,1\}$
- $\Sigma = \{1\}$
- $q_0 = (f(0), f(1), f(2)) = (0, 1, 1)$
- $\delta((a,, b, c), s) = (b, c, b + c - 2bc)$

Application: Automatic Theorem Proving

Assume $\Sigma = \{a, b\}$.
Let $\mathcal{L}$ be the set of $x \in \Sigma^*$ such that any a in x is followed by a b.
Let $\mathcal{L}'$ be the set of $x \in \Sigma^*$ such that any b in x is followed by a a.

How to prove that $\mathcal{L} \cap \mathcal{L}' = \{\epsilon\}$?

Intuitively:

- if $x \neq \epsilon$ in $\mathcal{L}$ we have that if $x = \ldots a \ldots$ then it should actually be $x = \ldots a \ldots b \ldots$
- if $x \neq \epsilon$ in $\mathcal{L}'$ we have that if $x = \ldots b \ldots$ then it should actually be $x = \ldots b \ldots a \ldots$

Hence a non-empty word in $\mathcal{L} \cap \mathcal{L}'$ should be infinite.

Application: Automatic Theorem Proving (cont.)

Formally we can automatically prove that $\mathcal{L} \cap \mathcal{L}' = \{\epsilon\}$ with an automaton.

Define a DFA D such that $\mathcal{L}(D) = \mathcal{L}$.
Define a DFA D' such that $\mathcal{L}(D') = \mathcal{L}'$.

Now we can compute $D \times D'$ and check that

$$\mathcal{L} \cap \mathcal{L}' = \mathcal{L}(D \times D') = \{\epsilon\}$$