TDDD14/TDDD85
Slides for Lecture 5
Minimization of Automata

Christer Bäckström, 2018

Equivalence Relations

A binary relation R on a set S is an equivalence relation if it satisfies the following three properties:
- reflexive: $R(x,x)$ for all $x \in S$
- symmetric: $R(x,y) \Rightarrow R(y,x)$
- transitive: $R(x,y)$ and $R(y,z) \Rightarrow R(x,z)$

Example: Let $\Sigma = \{0,1\}$. Def. relation R on Σ^* such that $R(x,y)$ iff $|x| = |y|$.

- For all x, $|x| = |x|$. Reflexive
- If $|x| = |y|$, then $|y| = |x|$. Symmetric.
- If $|x| = |y|$ and $|y| = |z|$, then $|x| = |z|$. Transitive.

Each string $x \in \Sigma^*$ has an associated equivalence class $[x]$, defined as $[x] = \{y \in \Sigma^* \mid R(x, y)\}$.

In the example, $[x]$ is the set of all strings that have the same length as x, including x.

$[\varepsilon] = \{\varepsilon\}$
$[0] = [1] = \{0, 1\}$
$[00] = [01] = [10] = [11] = \{00, 01, 10, 11\}$
etc.

An infinite number of equivalence classes in this case.

It follows from the definition that each element belongs to exactly one equivalence class.

Let S be a set and R an equivalence relation on S.
Let P be the set of all equivalence classes for R.
Then P is a partition of S, i.e.
- Each equivalence class is non-empty
- P covers S, i.e. every $x \in S$ belongs to some equivalence class.
- If X and Y are equivalence classes s.t. $X \neq Y$, then $X \cap Y = \varnothing$.

In the example, R gives a partition with one equivalence class P_i for each $i \in \mathbb{N}$, such that $P_i = \{|x| \in \Sigma^* \mid |x| = i\}$.

For instance $P_0 = [\varepsilon]$ and $P_3 = [001]$.

Let $\widehat{\delta}$ be the extension of δ to strings, defined such that for all states $p \in Q$:

- $\widehat{\delta}(p, \varepsilon) = p$

- $\widehat{\delta}(p, xa) = \delta(\widehat{\delta}(p, x), a)$ for all $x \in \Sigma^*$ and all $a \in \Sigma$

Quotient Automata

Consider collapsing two states p and q to one state in a DFA.

1. We cannot collapse p and q if $p \in F$ and $q \notin F$ (we must be distinguish between accept and reject).

2. If we collapse p and q and there is some $a \in \Sigma$ such that $\delta(p,a) \neq \delta(q,a)$, then we must collapse also $\delta(p,a)$ and $\delta(q,a)$ to one state. Otherwise we have two choices on symbol a.

Combining 1 and 2 gives that we can collapse p and q to one state, unless there is some string $x \in \Sigma^*$ such that $\widehat{\delta}(p,x) \in F$ and $\widehat{\delta}(q,x) \notin F$.

Define the binary relation $\approx$ on the set Q of states such that

$p \approx q$ holds if and only if for all $x \in \Sigma^*$ $(\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$.

Then $\approx$ has the properties:
1. $p \approx p$ for all p (reflexive)
2. if $p \approx q$, then $q \approx p$ (symmetric)
3. if $p \approx q$ and $q \approx r$, then $p \approx r$ (transitive)

That is, $\approx$ is an equivalence relation on Q.

This defines an equivalence class $[p]$ for every state p as
$[p] = \{q \mid q \approx p\}$.

Recall that an equivalence relation defines a partition, so every state belong to exactly one equivalence class, i.e.
$p \approx q$ if and only if $[p] = [q]$.

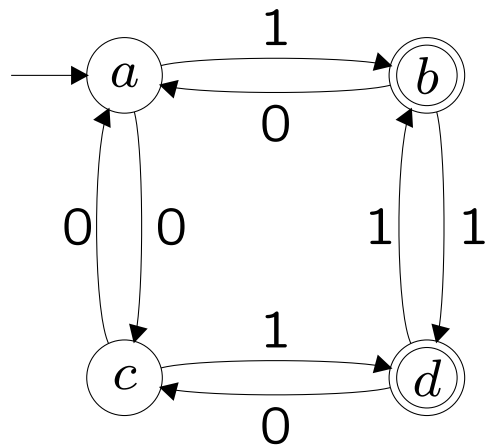
Let $M = \langle Q, \Sigma, \delta, s, F \rangle$ be a DFA.
Let $\approx$ be defined on Q as above.

We can then construct an equivalent DFA $M_{/\approx}$ that has one state for each equivalence class of $\approx$.

Define $M_{/\approx} = \langle Q', \Sigma, \delta', s', F' \rangle$ where

- $Q' = \{[p] \mid p \in Q\}$
- $\delta'([p], a) = [\delta(p, a)]$
- $s' = [s]$
- $F' = \{[p] \mid p \in F\}$

M

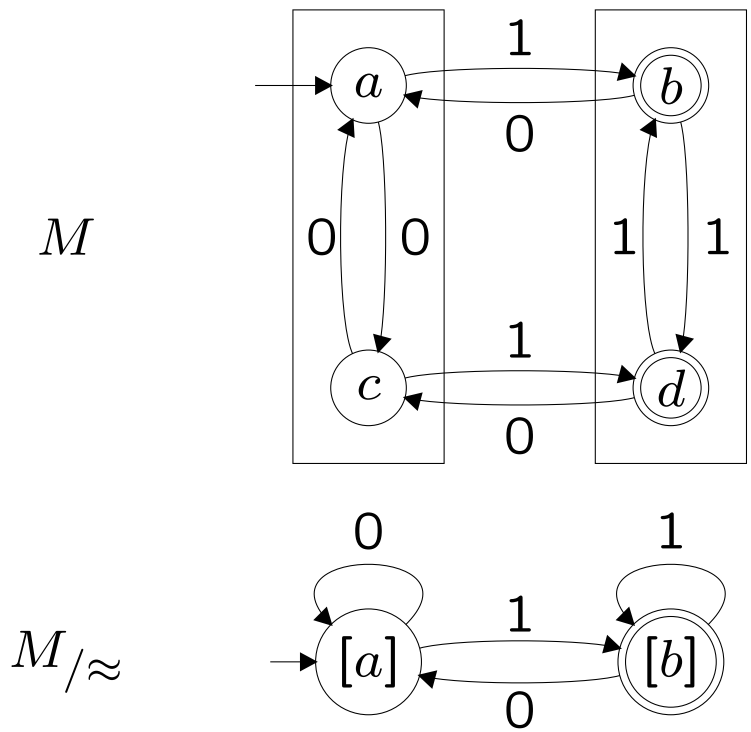


If we have just read 0, then we must be in a or c.
If we have just read 1, then we must be in b or d.

Hence, for all $x \in \Sigma^*$:
$\widehat{\delta}(a, x0) \notin F$ and $\widehat{\delta}(c, x0) \notin F$
$\widehat{\delta}(a, x1) \in F$ and $\widehat{\delta}(c, x1) \in F$

It follows that $\widehat{\delta}(a, y) \in F \Leftrightarrow \widehat{\delta}(c, y) \in F$ for all $y \in \Sigma^*$
so $a \approx c$.
We similarily get that $b \approx d$.

We have $[a] = [c]$ and $[b] = [d]$, so $M_{/\approx}$ has two states.



These two DFAs accept the same language.

Theorem: $L(M) = L(M_{/\approx})$

Proof: We claim that for all $p \in Q$ and all $x \in \Sigma^*$, it holds that $\widehat{\delta}(p, x) \in F$ iff $\widehat{\delta}'([p], x) \in F'$.
Proof by induction over the length of x.

Base case: $|x| = 0$, so $x = \varepsilon$.
We have $\widehat{\delta}(p, \varepsilon) = p$ and $\widehat{\delta}'([p], \varepsilon) = [p]$.
We have $p \in F$ iff $[p] \in F'$ by def. of $M_{/\approx}$.
Hence, $\widehat{\delta}(p, \varepsilon) \in F$ iff $\widehat{\delta}'([p], \varepsilon) \in F'$

Induction step: Suppose the claim holds for all strings of length n, for some $n \geq 0$. We must prove that it holds also for strings of length $n+1$.
Let $a \in \Sigma$ and $x \in \Sigma^n$. Then $|ax| = n+1$.
Let $q = \delta(p, a)$
Then $\delta'([p], a) = [\delta(p, a)] = [q]$.
It follows from the induction hypothesis that $\widehat{\delta}(q, x) \in F$ iff $\widehat{\delta}'([q], x) \in F'$.
Hence, $\widehat{\delta}(p, ax) \in F$ iff $\widehat{\delta}'([p], ax) \in F'$.

This proves the claim, so it follows that for all $x \in \Sigma^*$, it holds that $\widehat{\delta}(s, x) \in F$ iff $\widehat{\delta}'([s], x) \in F'$.
That is, $L(M) = L(M_{/\approx})$

Minimization Algorithm

Recall these observations:

1. We cannot collapse p and q if $p \in F$ and $q \notin F$ (we must be distinguish between accept and reject).

2. If we collapse p and q and there is some $a \in \Sigma$ such that $\delta(p, a) \neq \delta(q, a)$, then we must collapse also $\delta(p, a)$ and $\delta(q, a)$ to one state. Otherwise we have two choices on symbol a.

Note that 2 implies the following:
if $\delta(p, a)$ and $\delta(q, a)$ cannot be collapsed, then we cannot collapse p and q either.

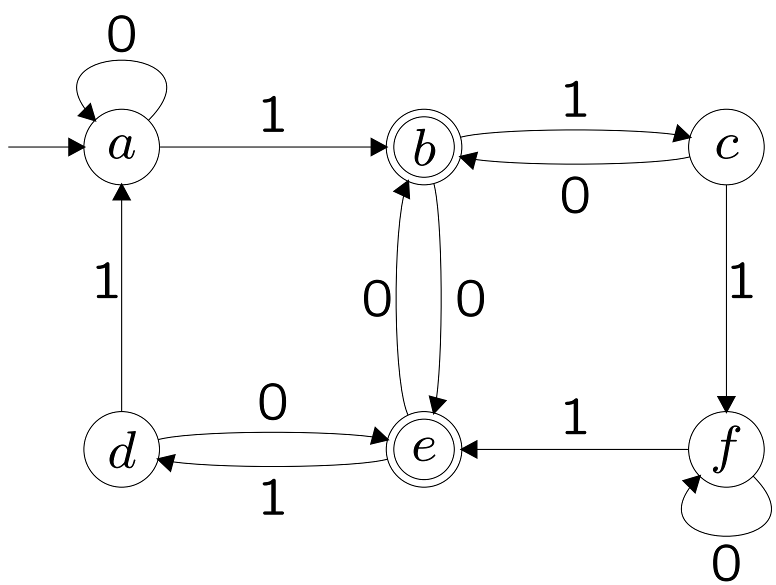
The idea for the algorithm is to iteratively mark all pairs that cannot be collapsed.

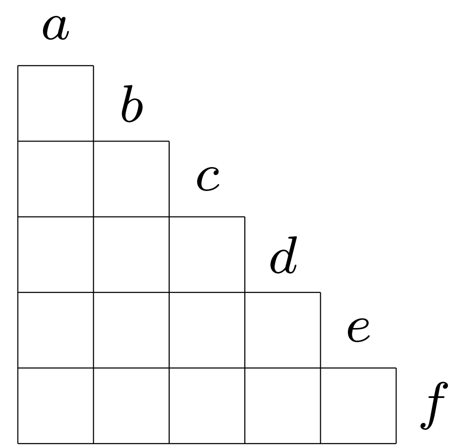
First mark all pairs p and q that break rule 1.

Then work backwards from the marked pairs. If a pair p and q is unmarked but rule 2 requires that we also collapse a pair that is already marked, then we mark also the pair p and q since it cannot be collapsed.

Make a table with one entry for each combination of two different states. (Note, there is no order on the states in an entry).

M





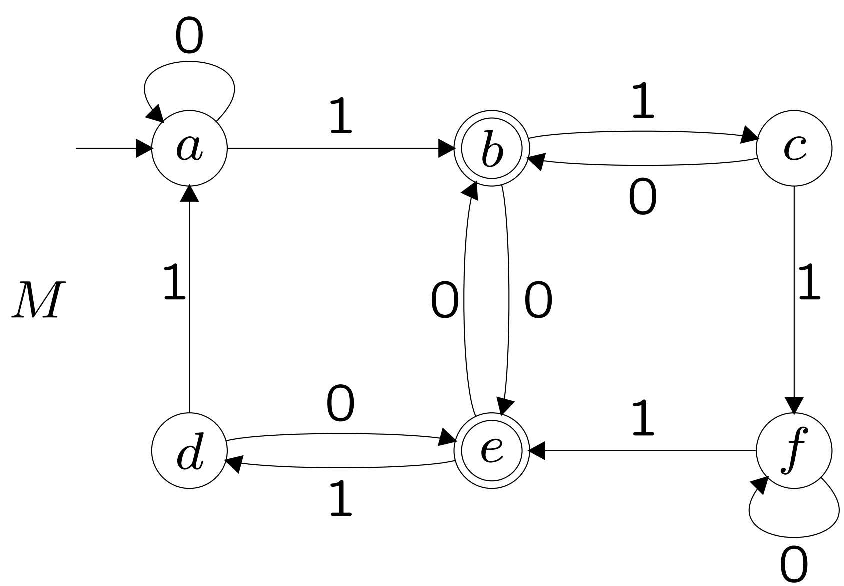
Marking Algorithm:

1. For all pairs of states $\{p, q\}$
   if $p \in F$ and $q \notin F$, then mark $\{p, q\}$
2. For all unmarked pairs of states $\{p, q\}$
   if there is some $a \in \Sigma$ such that $\{\delta(p, a), \delta(q, a)\}$ is marked
   then mark $\{p, q\}$.
3. Repeat 2 until no new pair is marked.

If $\{p, q\}$ is still unmarked, then $p \approx q$.

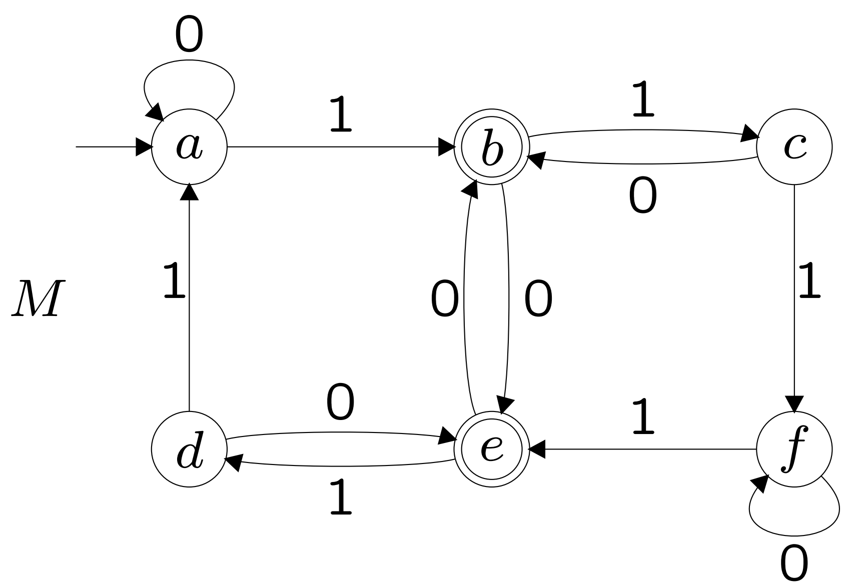
Step 1 (iteration 0).

Mark all pairs $\{p, q\}$ such that $p \in F$ and $q \notin F$.



	a	b	c	d	e
b	0				
c		0			
d		0			
e	0		0	0	
f		0			0

Step 2, iteration 1:



	a	b	c	d	e
b	0				
c	1	0			
d	1	0			
e	0		0	0	
f		0			0

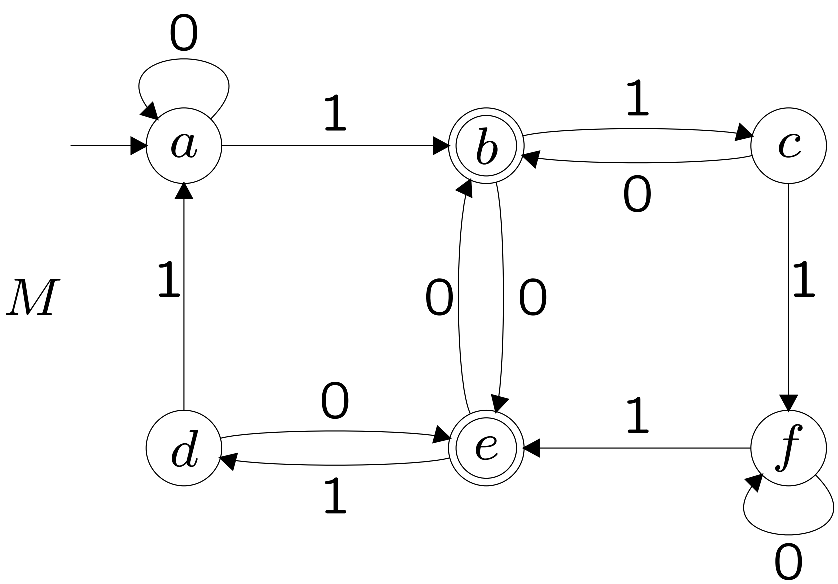
$\{a,c\}$: $\{delta(a,0),\delta(c,0)\} = \{a,b\}$ is marked, so mark $\{a,c\}$

$\{a,d\}$: $\{delta(a,0),\delta(d,0)\} = \{a,e\}$ is marked, so mark $\{a,d\}$

$\{a,f\}$: $\{delta(a,0),\delta(f,0)\} = \{a,f\}$ is unmarked, so check also 1

$\{a,f\}$: $\{delta(a,1),\delta(f,1)\} = \{b,e\}$ is unmarked, so don't mark!

Step 2, iteration 1 cont'd:



	a	b	c	d	e
b	0				
c	1	0			
d	1	0			
e	0		0	0	
f		0	1	1	0

$\{b, e\}$: $\{\delta(b, 0), \delta(e, 0)\} = \{b, e\}$ is unmarked, so check also 1

$\{b, e\}$: $\{\delta(b, 1), \delta(e, 1)\} = \{c, d\}$ is unmarked, so don't mark!

$\{c, d\}$: $\{delta(c, 0), \delta(d, 0)\} = \{b, e\}$ is unmarked, so check also 1

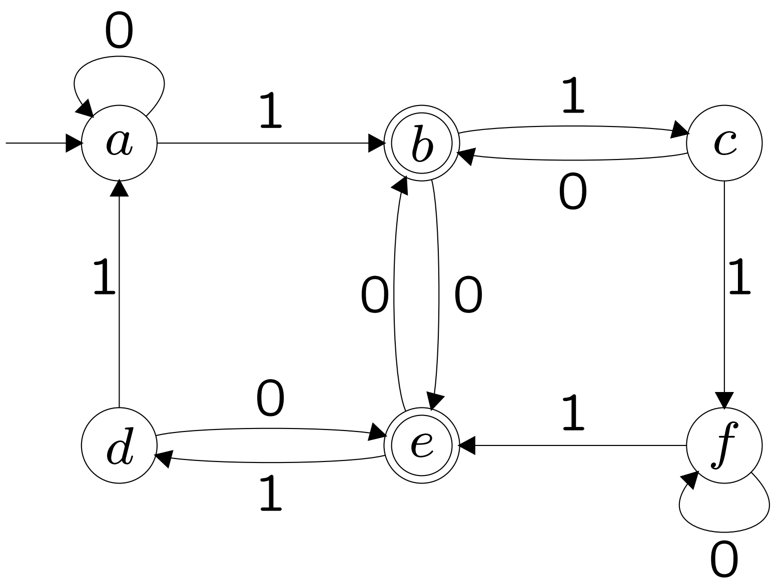
$\{c, d\}$: $\{delta(c, 1), \delta(d, 1)\} = \{f, a\}$ is unmarked, so don't mark!

$\{c, f\}$: $\{delta(c, 0), \delta(f, 0)\} = \{b, f\}$ is marked, so mark $\{c, f\}$

$\{d, f\}$: $\{delta(d, 0), \delta(f, 0)\} = \{e, f\}$ is marked, so mark $\{d, f\}$

Step 2, iteration 2:

M



	a	b	c	d	e
b	0				
c	1	0			
d	1	0			
e	0		0	0	
f		0	1	1	0

$\{a, f\} : \{\delta(a, 0), \delta(f, 0)\} = \{a, f\}$ is unmarked, so check also 1

$\{a, f\} : \{\delta(a, 1), \delta(f, 1)\} = \{b, e\}$ is unmarked, so don't mark!

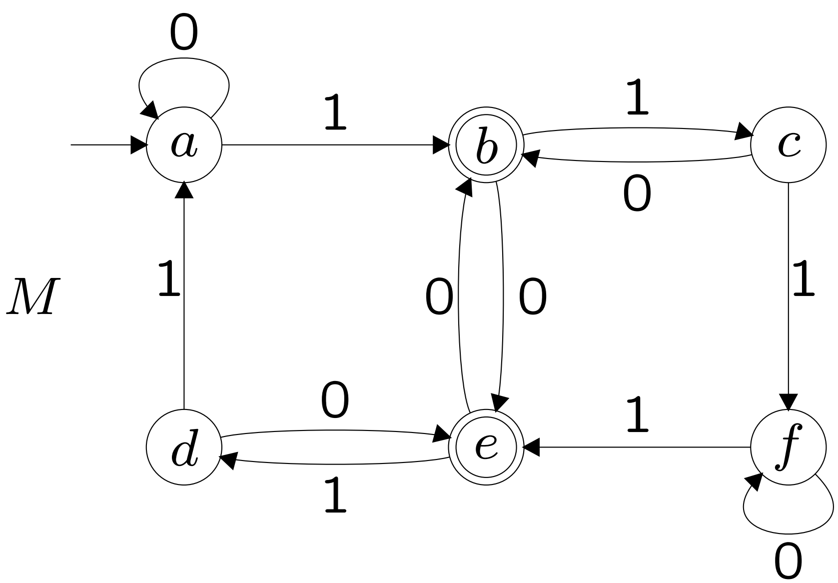
$\{b, e\} : \{\delta(b, 0), \delta(e, 0)\} = \{b, e\}$ is unmarked, so check also 1

$\{b, e\} : \{\delta(b, 1), \delta(e, 1)\} = \{c, d\}$ is unmarked, so don't mark!

$\{c, d\} : \{delta(c, 0), \delta(d, 0)\} = \{b, e\}$ is unmarked, so check also 1

$\{c, d\} : \{delta(c, 1), \delta(d, 1)\} = \{f, a\}$ is unmarked, so don't mark!

Step 2, iteration 2:



	a	b	c	d	e
b	0				
c	1	0			
d	1	0			
e	0		0	0	
f		0	1	1	0

Nothing more was marked in iteration 2, so we terminate.
The pairs $\{a, f\}$, $\{b, e\}$ and $\{c, d\}$ are unmarked.

We get $a \approx f$, $b \approx e$ and $c \approx d$.

We get a minimal DFA