COURSE: THEORY OF AUTOMATA COMPUTATION

TOPICS TO BE COVERED

- Conversion of NFA to DFA
- Inaccessible states
- How to find all accessible states
- Minimization process

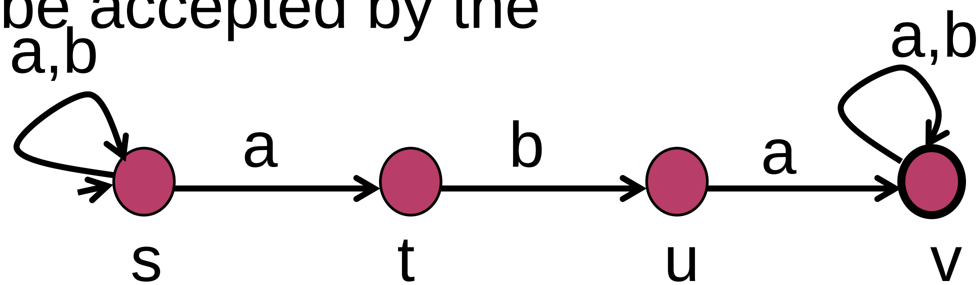
INTRODUCTION

MOTIVATIONS

Problems:

1. Given a DFA M with k states, is it possible to find an equivalent DFA M’ (I.e., L(M) = L(M’)) with state number fewer than k ?
2. Given a regular language A, how to find a machine with minimum number of states ?

Ex: A = L((a+b)*aba(a+b)*) can be accepted by the following NFA:



By applying the subset construction, we can construct a DFA M2 with 2^4=16 states,
 of which only 6 are accessible from the initial state {s}.

INACCESSIBLE STATES

- A state $p \in Q$ is said to be inaccessible (or unreachable) [from the initial state] if there exists no string x in Σ^* s.t. $\Delta(s,x) = p$ (I.e., $p \notin \{q \mid \exists x \in \Sigma^*, \Delta(s,x) = q\}$.)

Theorem: Removing inaccessible states from a machine M does not affect the language it accepts.

Pf: $M = \langle Q, \Sigma, \delta, s, F \rangle$: a DFA; p : an inaccessible state

Let $M' = \langle Q \setminus \{p\}, \Sigma, \delta', s, F \setminus \{p\} \rangle$ be the DFA M with p removed. Where $\delta' : (Q \setminus \{p\}) \times \Sigma \rightarrow Q \setminus \{p\}$ is defined by

$\delta'(q,a) = r$ if $\delta(q, a) = r$ and $q, r \in Q \setminus \{p\}$.

For M and M' it can be proved by induction on x that

for all x in Σ^*, $\Delta(s,x) = \Delta'(s,x)$.

Hence for all $x \in \Sigma^*$, $x \in L(M)$ iff $\Delta(s,x) = q \in F$

iff $\Delta'(s,x) = q \in F \setminus \{p\}$ iff $x \in L(M')$.

INACCESSIBLE STATES ARE REDUNDANT

- M : any DFA with n inaccessible states $p_1,p_2,\ldots,p_n$.

Let $M_1,M_2,..,M_{n+1}$ are DFAs s.t. DFA M_{i+1} is constructed from M_i by removing p_i from M_i. I.e.,

M -rm(p_1)-> M_1 -rm(p_2)-> M_2 - … M_n -rm(p_n)-> M_n

By previous lemma: $L(M) = L(M_1) = \ldots = L(M_n)$ and

M_n has no inaccessible states.

- Conclusion: Removing all inaccessible sates simultaneously from a DFA will not affect the language it accepts.
- In fact the conclusion holds for all NFAs we well.

Pf: left as an exercise.

- Problem: Given a DFA (or NFA), how to find all inaccessible states ?

HOW TO FIND ALL ACCESSIBLE STATES

- A state is said to be accessible if it is not inaccessible.

Note: the set of accessible states A(M) of a NFA M is

$\{q \mid \exists\, x \in \Sigma^*, q \in \Delta(S,x)\}$

and hence can be defined by induction.

- Let A_k be the set of states accessible from initial states of M by at most k steps of transitions.

  I.e., $A_k = \{q \mid \exists\, x \in \Sigma^* \text{ with } |x| \leq k \text{ and } q \in \Delta(S,x)\}$
- What is the relationship b/t A(M) and A_ks ?
  - sol: $A(M) = \bigcup_{k\geq 0} A_k$. Moreover $A_k \subseteq A_{k+1}$
- What is A_0 and the relationship b/t A_k and A_{k+1} ?

Formal definition: $M=\langle Q,\Sigma,\delta, S,F\rangle$: any NFA.

- Basis: Every start state $q \in S$ is accessible.($A_0 \subseteq A(M)$)
- Induction: If q is accessible and p in $\delta(q,a)$ for some $a \in \Sigma$, then p is accessible.

  ($A_{k+1}=A_k \cup \{p \mid p\in \delta(q,a) \text{ for some } q\in A_k \text{ and } a \in \Sigma.\}$)

AN ALGORITHM TO FIND ALL ACCESSIBLE STATES:

- REACH(M) { // M = <Q,Σ,δ, S ,F>

1. A = S; // $A = A_0$
2. B = Δ (A) - A ; // $B = A_1 - A_0$
3. For k = 0 to |Q| do { // $A = A_k$; $B = A_{K+1} - A_k$
4. A = A U B ; // $A = A_{K+1}$

   B = Δ(B) - A; // $B = \Delta(B) - A = \Delta(A_{K+1} - A_k) - A_{K+1} = A_{K+2} - A_{k+1}$;

   if B = {} then break };
5. Return(A) }

Function Δ(S) { // = $\bigcup_{p \in S, a \in \Sigma,} q \in \delta(p,a)$

1. Δ = {};
2. For each q in Q do

   for each a in Σ do

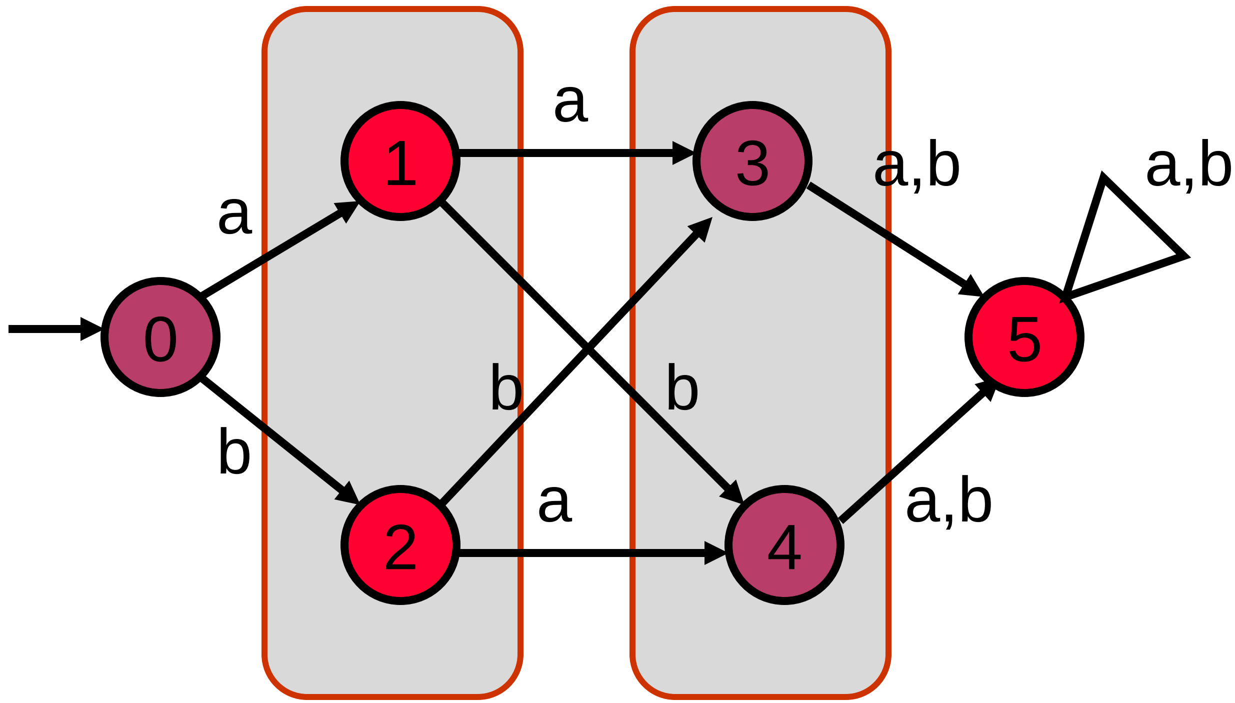
   Δ = Δ U δ (q,a);
3. Return(Δ) }

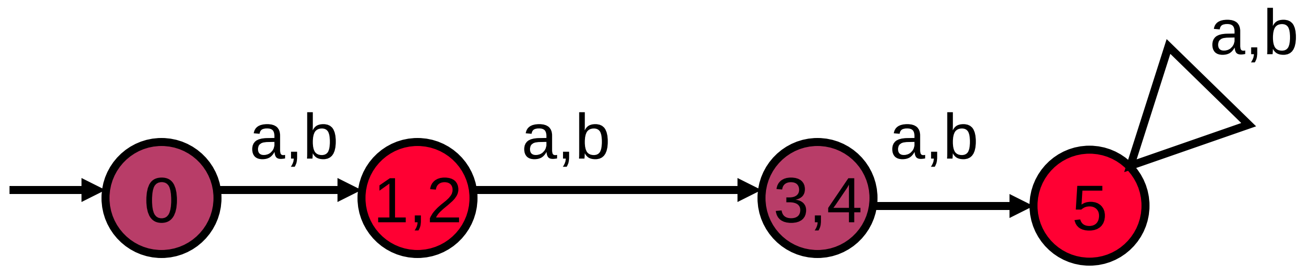
-

MINIMIZATION PROCESS

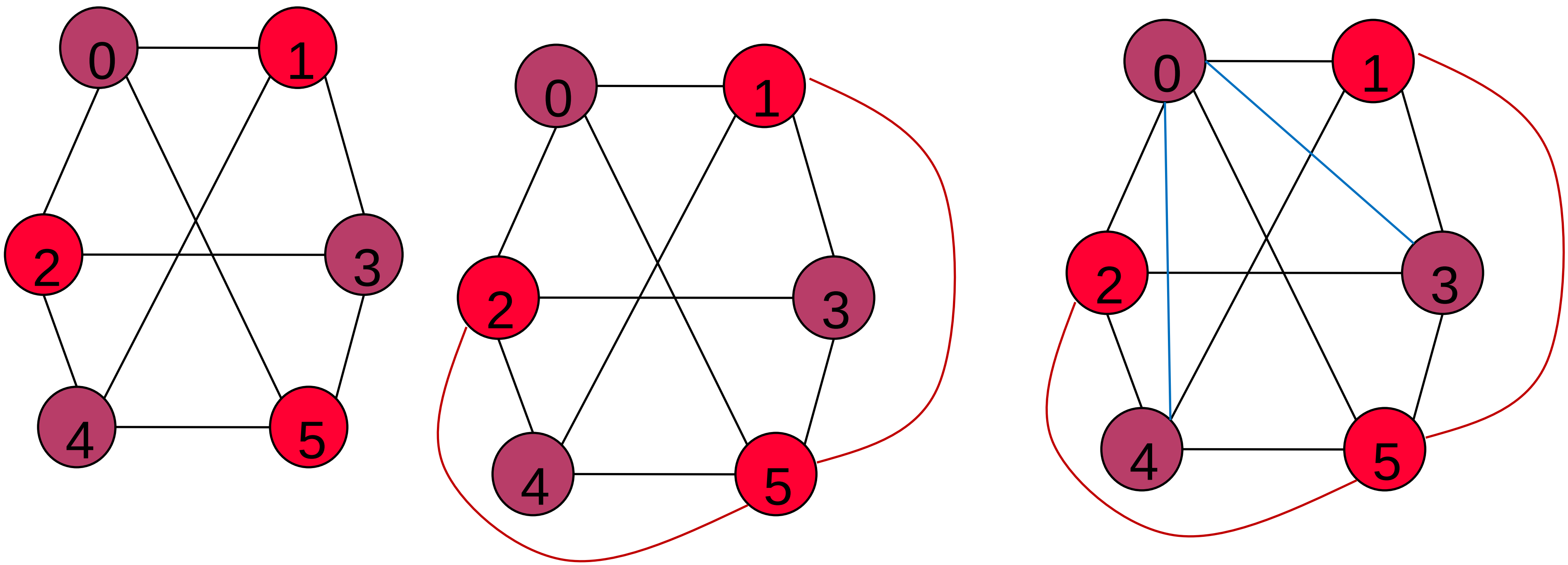
- Minimization process for a DFA:
  - 1. Remove all inaccessible states
  - 2. Merge all *equivalent* states
- What does it mean that two states are equivalent?
  - both *have the same observable behaviors* .i.e.,
  - there is no way to distinguish their difference.
- Definition: we say state p and q are *distinguishable* if there exists a string $x \in \Sigma^*$ s.t. $(\Delta(p,x) \in F \Leftrightarrow \Delta(q,x) \notin F)$.
  - If there is no such string, i.e. $\forall x \in \Sigma^* (\Delta(p,x) \in F \Leftrightarrow \Delta(q,x) \in F)$, we say p and q are equivalent (or indistinguishable).
- Example[13.2]: (next slide)
  - state 3 and 4 are equivalent.
  - States 1 and 2 are equivalent.
- Equivalents sates can be merged to form a simpler machine.

Example 13.2:





Example 13.2: Witness for states that are distinguishable



1. States b/t {0,3,4} and {1,2,5} can be distinguishsed by the empty string ε.
2. States b/t {1,2} and {5} can be distinguished by a or b.
3. States b/t {0} and {3,4} can be distinguished by aa,ab, ba or bb.
4. There is no way to distinguish b/t 1 and 2, and b/t 3 and 4.

QUOTIENT CONSTRUCTION

- M=(Q, Σ, δ, s, F): a DFA.
- ≈: a relation on Q defined by:

  $p \approx q \iff \forall x \in \Sigma^*\ \Delta(p,x) \in F \text{ iff } \Delta(q,x) \in F$
- Property: ≈ is an equivalence (i.e., reflexive, symmetric and transitive) relation.
- Hence it partitions Q into equivalence classes :
  - $[p] =_{def} \{q \in Q \mid p \approx q\}$ for $p \in Q$.
  - $Q/\approx\ =_{def} \{[p] \mid p \in Q\}$ is the quotient set.
  - Every $p \in Q$ belongs to exactly one class (which is [p])
  - $p \approx q$ iff [p]=[q] //why? since $p \approx q$ implies $p \approx r$ iff $q \approx r$.
- Ex: From Ex 13.2, we have 0, 1 ≈ 2, 3 ≈ 4, 5.
  - => [0] = {0}, [1] = {1,2}, [2]={1,2}, [3]={3,4},[4]={3,4} and
  - [5] = {5}. As a result, [1] = [2] = {1,2}, [3]=[4]= {3,4} and
  - Q/≈ = { {0},{1,2},{3,4},{5}} = { [0],[1],[2],[3],[4],[5] } = {[0],[1],[3],[5] }.

- Define a DFA called the quotient machine $M/\approx$ = <Q',Σ, δ',s',F'> where

THE FUNCTION Δ' IS WELL-DEFINED.

  - Q'=Q/≈ ; s'=[s]; F'={[p] | p ∈ F}; and
  - δ'([p],a)=[δ (p,a)] for all p∈Q and a∈Σ. But well-defined?

Lem 13.5. if p ≈ q then δ (p,a) ≈ δ (q,a).

Hence [p]=[q] ⇒ p≈q ⇒ δ(p,a) ≈ δ(q,a) ⇒ [δ (p,a)] = [δ (q,a)]

Pf: By def. [δ (p,a)] = [δ(q,a)] iff δ(p,a) ≈ δ (q,a)

iff ∀y∈Σ* Δ(δ (p,a),y) ∈ F ⇔ Δ(δ (q,a),y) ∈ F

iff ∀ y ∈ Σ* Δ (p, ay) ∈ F ⇔ Δ (q,ay) ∈ F

if p ≈ q.

Lemma 13.6. p ∈ F iff [p] ∈ F'.

pf: => : trival.

<=: need to show that if q ≈ p and p ∈ F, then q ∈ F.

But this is trivial since p = Δ(p,ε) ∈ F iff Δ (q, ε) = q ∈ F

PROPERTIES OF THE QUOTIENT MACHINE.

Lemma 13.7: $\forall$ x $\in$ Σ*, Δ'([p],x) = [Δ(p,x)].

Pf: By induction on |x|.

Basis x = ε: Δ'([p], ε] = [p] = [Δ(p, ε)].

Ind. step: Assume Δ'([p],x) = [Δ(p,x)] and let a $\in$ Σ.

Δ'([p],xa) = δ'(Δ'(p,x),a) = δ'([Δ(p,x)],a) --- ind. hyp.

=[δ(Δ (p,x),a)] -- def. of δ'

= [Δ (p,xa)]. -- def. of Δ.

Theorem 13.8: L(M/≈) = L(M).

Pf: $\forall$ x $\in$ Σ*,

x $\in$ L(M/≈) iff Δ'(s',x) $\in$ F'

iff Δ'([s],x) $\in$ F' iff [Δ(s,x)] $\in$ F' --- lem 13.7

iff Δ (s,x) $\in$ F --- lem 13.6

iff x $\in$ L(M).

- Theorem: ((M/≈) / ≈) = M/≈

M/≈ NEED NOT BE MERGED FURTHER

Pf: Denote the second ≈ by ~. I.e.

[p] ~ [q] iff ∀ x ∈ Σ*, Δ'([p],x) ∈ F' ⇔ Δ'([q],x) ∈ F'

Now

[p] ~ [q]

iff ∀ x ∈ Σ*, Δ'([p],x) ∈ F' ⇔ Δ'([q],x) ∈ F' -- def.of~

iff ∀ x ∈ Σ*, [Δ(p,x)] ∈ F' ⇔ [Δ(q,x)] ∈ F' -- lem 13.7

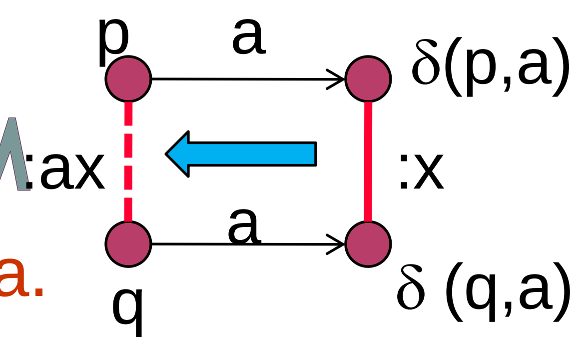
iff ∀ x ∈ Σ*, Δ (p,x) ∈ F ⇔ Δ (q,x) ∈ F -- lem 13.6

iff p ≈ q -- def of ≈

iff [p] = [q] -- property of equivalence ≈

A MINIMIZATION ALGORITHM

1. Write down a table of all pairs {p,q}, initially unmarked.



2. mark {p,q} if $p \in F$ and $q \notin F$ or vice versa.

3. Repeat until no additional pairs marked:

 3.1 if ∃ unmarked pair {p,q} s.t. {δ(p,q), δ(q,a) } is marked for some $a \in \Sigma$, then mark {p,q}.

4. When done, $p \approx q$ iff {p,q} is not marked.

Let M_k ($k \geq 0$) be the set of pairs marked after the k-th iteration of step 3. [and M_0 is the set of pairs before step 3.]

Notes: (1) $M = \cup_{k \geq 0} M_k$ is the final set of pairs marked by the alg. (2) The algorithm must terminate since there are totally only C(n,2) pairs and each iteration of step 3 must mark at least one pair for it to not terminate..

AN EXAMPLE:

- The DFA: (Ex 13.2)

	a	b
>0	1	2
1F	3	4
2F	4	3
3	5	5
4	5	5
5F	5	5

INITIAL TABLE

1	-				
2	-	-			
3	-	-	-		
4	-	-	-	-	
5	-	-	-	-	-
	0	1	2	3	4

AFTER STEP 2 (M_0)

1	M				
2	M	-			
3	-	M	M		
4	-	M	M	-	
5	M	-	-	M	M
	0	1	2	3	4

AFTER FIRST PASS OF STEP 3 (M_1)

1	M				
2	M	-			
3	-	M	M		
4	-	M	M	-	
5	M	M	M	M	M
	0	1	2	3	4

2ND PASS OF STEP 3. (M_2 & M_3)

- The result : 1 ≈ 2 and 3 ≈ 4.

1	M				
2	M	-			
3	M2	M	M		
4	M2	M	M	-	
5	M	M1	M1	M	M
	0	1	2	3	4

CORRECTNESS OF THE MINIMIZATION ALGORITHM

Let M_k ($k \geq 0$) be the set of pairs marked after the k-th itration of step 3. [and M_0 is the set of pairs befer step 3.]

Lemma: $\{p,q\} \in M_k$ iff $\exists x \in \Sigma^*$ of length $\leq k$ s.t. $\Delta(p,x) \in F$ and $\Delta(q,x) \notin F$ or vice versa,

Pf: By ind. on k. Basis $k = 0$. trivial.

Ind. step: $\exists\ x \in \Sigma^*$ of length $\leq k+1$ s.t. $\Delta\ (p,x) \in F \Leftrightarrow \Delta\ (q,x) \notin F$,

iff $\exists\ y \in \Sigma^*$ of length $\leq k$ s.t. $\Delta\ (p,y) \in F \Leftrightarrow \Delta\ (q,y) \notin F$, or

$\exists\ ay \in \Sigma^*$ of length $\leq k+1$ s.t. $\Delta(\delta\ (p,a),y) \in F \Leftrightarrow \Delta(\delta(q,a),y) \notin F$,

iff $\{p\ ,\ q\} \in M_k$ or $\{\delta\ (p,a),\ \delta\ (q,a)\} \in M_k$ for some $a \in \Sigma$.

iff $\{p,q\} \in M_{k+1}$.

Theorem 14.3: The pair $\{p,q\}$ is marked by the algorithm iff not($p \approx q$) (i.e., $\exists\ x \in \Sigma^*$ s.t. $\Delta\ (p,x) \in F \Leftrightarrow \Delta\ (q,x) \notin F$)

Pf: not($p \approx q$) iff $\exists\ x \in \Sigma^*$ s.t. $\Delta\ (p,x) \in F \Leftrightarrow \Delta\ (q,x) \notin F$

iff $\{p,q\} \in M_k$ for some $k \geq 0$

iff $\{p,q\} \in M = \bigcup_{k \geq 0} M_k$.