

CSE 105—Theory of Computability

Lecture 5—April 18, 2006
DFA State Minimization

Instructor: Neil Rhodes

Contradiction?

All finite sets are regular

All regular sets are pumpable

Finite sets are not regular

Example (1.57)

If A is a language, A' is the set of all first halves of strings in A :

- $A' = \{x \mid \text{for some } y, |x| = |y| \text{ and } xy \in A\}$

Show that if A is regular, A' is regular

Example (1.58)

If A is any language, let A'' be the set of all strings in A with their middle thirds removed so that:

- $A'' = \{xz \mid \text{for some } y, |x| = |y| = |z| \text{ and } xyz \in A\}$

Show that if A is regular, A'' is not necessarily regular

How can you distinguish one state from another?

If state A accepts, and state B rejects

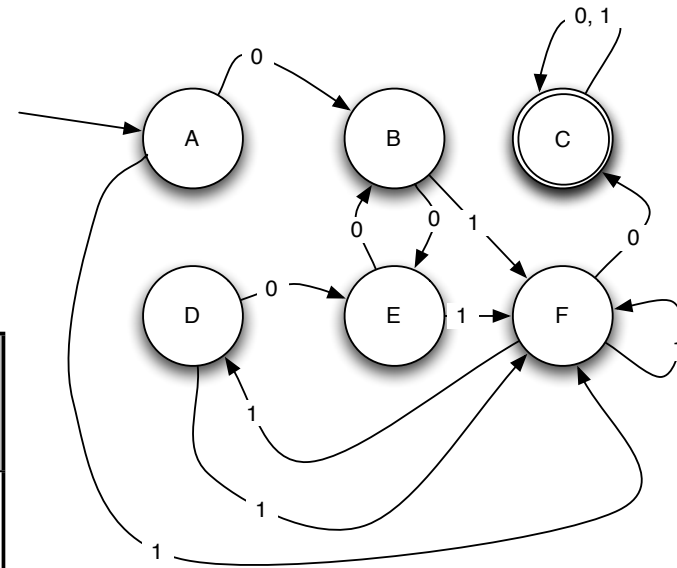
Otherwise, if, on symbol x , A goes to A' , B goes to B' and

- A' is distinguishable from B'

For an n -state FSA, if A and B can be distinguished, they can be distinguished in _____ symbols

Distinguishing example

	A	B	C	D	E	F
B						
C						
D						
E						
F						



Procedure

Mark all accepting/non-accepting state pairs as distinguishable

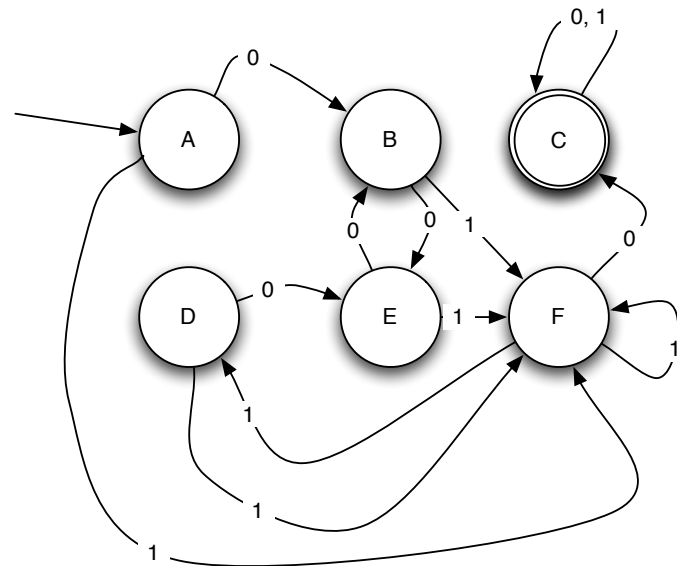
Round: foreach pair (A, B)

- Foreach symbol s
 - Check if transition from A on s and transition of B on s lead to a distinguishable state. If so, mark A and B as distinguishable

Continue doing rounds until no new distinguishable pairs have been found

More efficient distinguishing

	A	B	C	D	E	F
B						
C						
D						
E						
F						



More Efficient Procedure

Foreach pair (A, B)

- Keep list, $L(A, B)$, of states that *depend* on (A, B)

Mark all accepting/non-accepting state pairs as distinguishable

- If a state pair (A, B) is marked, recursively mark state pairs that depend on (A, B)

Fallout of Distinguishing procedure

Minimization of DFAs

- Eliminate states unreachable from start state
- Run distinguishing procedure
 - Collapse undistinguished states into new states

Easy way to tell whether two FSAs accept the same language