

cse 311: foundations of computing

Spring 2015

Lecture 23: State minimization and NFAs



state minimization

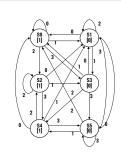
- · Many different FSMs (DFAs) for the same problem
- Take a given FSM and try to reduce its state set by combining states
 - Algorithm will always produce the unique minimal equivalent machine (up to renaming of states) but we won't prove this

state minimization algorithm

- 1. Put states into groups based on their outputs (or whether they are final states or not)
- 2. Repeat the following until no change happens
 - a. If there is a symbol s so that not all states in a group G agree on which group s leads to, split G into smaller groups based on which group the states go to on s



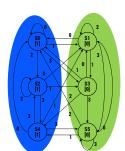
state minimization example



present state	0 n	ext st	ate 2	3	utput
					
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S1 S2	S1	S3 S3	S2	S4	1
S3 S4 S5	S1	S0	S4	S5	0
S4	S0	S1 S4	S2	S5	1
S5	S1	S4	S0	S5	0
		state sition	table		

Put states into groups based on their outputs (or whether they are final states or not)

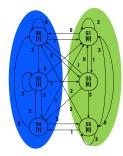
state minimization example



present state	0	next 1	state 2	3	utput
S0 S1 S2 S3 S4 S5	S0 S1 S1 S0 S1	\$1 \$3 \$3 \$0 \$1 \$4	S2 S1 S2 S4 S2 S0	S3 S5 S4 S5 S5 S5	1 0 1 0 1 0
	trans	state	table		

Put states into groups based on their outputs (or whether they are final states or not)

state minimization example



present state	l۵	next	state	3	utput
					
SU	SO	ST	S2	23	
S1	S0	S3	S1	S5	0
S0 S1 S2 S3 S4 S5	S1	S3 S3	S2	S4	1
S3	S1	S0 S1	S4 S2	S5 S5	0
S4	l SO	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state transition table

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol ${\bf s}$ so that not all states in a group G agree on which group ${\bf s}$ leads to, split G based on which group the states go to on ${\bf s}$

state minimization example

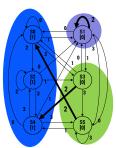
present	ı	next	state	(utput
state	0	1	2	3	l '
S0	SU	ST	S2	23	
S1	S0	S3	S1	S5	0
S2	S0 S1	S3 S3	S2	S4	1
S1 S2 S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S4 S5	Sī	Š4	ŠŪ	S5	Ó

state transition tab

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol ${\bf s}$ so that not all states in a group G agree on which group ${\bf s}$ leads to, split G based on which group the states go to on ${\bf s}$

state minimization example



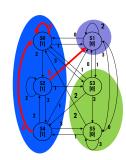
present state	0	next 1	state 2	3	utput
SO	SU	ST	52	23	\neg
S1	l SO	S3	S1	S5	l 0
S2	S1	S3 S3	S2	S4	1
\$2 \$3	S1	SO	S4	\$5	0
S4	S0	S1	S2	S5	1
S4 S5	Šī	Š4	SO	Š5	ΙÓ

state

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol ${\bf s}$ so that not all states in a group ${\bf G}$ agree on which group ${\bf s}$ leads to, split ${\bf G}$ based on which group the states go to on ${\bf s}$

state minimization example



present state	0	next 1	state 2	3	utput
S0	SU	ST	S2	23	
S1	l SO	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S2 S3	S1	SO	S4	S5	l 0
S4	S0	S1	S2	S5	1
S5	S1	S4	SO	S5	Ò

state transition table

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol ${\it s}$ so that not all states in a group G agree on which group ${\it s}$ leads to, split G based on which group the states go to on ${\it s}$

state minimization example

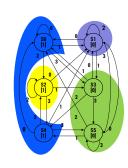


present state	0	next 1	state 2	3	utput
S0 S1 S2 S3 S4 S5	S0 S1 S1 S0 S1	\$1 \$3 \$3 \$0 \$1 \$4	S2 S1 S2 S4 S2 S0	S3 S5 S4 S5 S5 S5	1 0 1 0 1 0
	tran	state sition	table		

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol ${\it s}$ so that not all states in a group G agree on which group ${\it s}$ leads to, split G based on which group the states go to on ${\it s}$

state minimization example



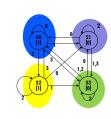
present state	0	next 1	state 2	3	utput
SO	SO	S1	S2	23	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S3 S4	S0	\$1	S2	S5	1
\$5	S1	S4	S0	S5	0

state transition table

Can combine states S0-S4 and S3-S5.

In table replace all S4 with S0 and all S5 with S3

minimized machine

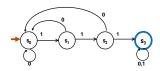


present	١.	next	state		utput	
state	0	1	2	3		
SO	SO	SI	S2	53		
S1	SO	S3	S1	S3	0	
S2	S1	S3	S2	SO	Ιi	
S3	S1	SO	SO	S3	Ó	
	-					
		state				
transition table						

another way to look at DFAs

Definition: The label of a path in a DFA is the concatenation of all the labels on its edges in order

Lemma: x is in the language recognized by a DFA iff x labels a path from the start state to some final state



goal: NFA to recognize...

binary strings that have even # of 1's or contain the substring 111

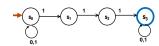
NFAs and regular expressions

Theorem: For any set of strings (language) A described by a regular expression, there is an NFA that recognizes A.

Proof idea: Structural induction based on the recursive definition of regular expressions...

nondeterministic finite automaton (NFA)

- Graph with start state, final states, edges labeled by symbols (like DFA) but
 - Not required to have exactly 1 edge out of each state labeled by each symbol--- can have 0 or >1
 - Also can have edges labeled by empty string ε
- Definition: x is in the language recognized by an NFA if and only if x labels a path from the start state to some final state



three ways of thinking about NFAs

- Outside observer: Is there a path labeled by x from the start state to some final state?
- Perfect guesser: The NFA has input x and whenever there is a choice of what to do it magically guesses a good one (if one exists)
- Parallel exploration: The NFA computation runs all possible computations on x step-by-step at the same time in parallel

regular expressions over Σ

- · Basis:
 - Ø, ε are regular expressions
 - ${\it a}$ is a regular expression for any ${\it a} \in \Sigma$
- Recursive step:
 - If A and B are regular expressions then so are:

(A ∪ B) (AB)

Α*

base case base case

- Case Ø:
- Case ε:
- Case **a**:

• Case Ø:



- Case ε:





inductive hypothesis

• Suppose that for some regular expressions ${\bf A}$ and ${\bf B}$ there exist NFAs N_A and N_B such that N_A recognizes the language given by ${\bf A}$ and N_B recognizes the language given by ${\bf B}$





inductive step

Case (A \cup B):



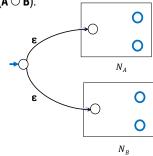


 N_B

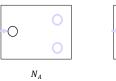
inductive step

inductive step





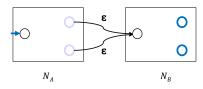
Case (AB):





inductive step inductive step

Case (AB):



Case A*



inductive step

Case A*

