CSE 105
THEORY OF COMPUTATION

Fall 2016

http://cseweb.ucsd.edu/classes/fa16/cse105-abc/

Thanks for your patience with the tech challenges today!
Today's learning goals

- Design NFA to recognize a given language
- Compare properties of regular and NFA-recognizable languages
- Convert an NFA (with or without epsilon transitions) to a DFA recognizing the same language
- Decide whether or not a string is described by a given regular expression
- Design a regular expression to describe a given language
- Convert between regular expressions and automata
General proof structure/strategy

**Theorem:** For any $L$ over $\Sigma$, if $L$ is regular then [the result of some operation on $L$] is also regular.

**Proof:**

*Given* name variables for sets, machines assumed to exist.

*WTS* state goal and outline plan.

*Construction* using objects previously defined + new tools working towards goal. Give formal definition and explain.

*Correctness* prove that construction works.

*Conclusion* recap what you've proved.
The regular operations

For $A$, $B$ languages over same alphabet, define:

$A \cup B = \{ x | x \in A \text{ or } x \in B \}$

$A \circ B = \{ xy | x \in A \text{ and } y \in B \}$

$A^* = \{ x_1 x_2 \ldots x_k | k \geq 0 \text{ and each } x_i \in A \}$

How can we prove that the concatenation of two regular languages is a regular language?
A **nondeterministic finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\) where

1. \(Q\) is a finite set called the states
2. \(\Sigma\) is a finite set called the alphabet
3. \(\delta : Q \times \Sigma_e \rightarrow \mathcal{P}(Q)\) is the transition function
4. \(q_0 \in Q\) is the start state
5. \(F \subseteq Q\) is the set of accept states.
Differences between NFA and DFA

- **DFA**: unique computation path for each input
- **NFA**: allow several (or zero) alternative computations on *same input*
  - $\delta(q,x)$ may specify *more than one* possible next states
  - $\epsilon$ transitions allow the machine to *transition between states spontaneously*, without consuming any input symbols

Types of components of formal definition

- **DFA** $\delta : Q \times \Sigma \rightarrow Q$
- **NFA** $\delta : Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$
Nondeterministic finite automata

- "Guess" some stage of input at which switch modes
- "Guess" one of finite list of criteria to meet

Input

Accept if either (or both) accepts
…structure i was using to display the output was a TreeCtrl, so I figured some kind of weird tree traversal system would work and i made it into this implicit state machine that used a stack to traverse up and down into different levels. In short, it was an absolute monster of a structure (which i am immensely proud of for continuing to work as new requirements were added every couple of hours :p). Needless to say, all that code is now gone. Tuesday I came in on a mission to reimplement the tree without any interruption in the continual updating of my project. This was made possible by mentor deciding to just disappear into thin air for the day. I designed am NFA(thanks based Shacham) that could cover every case and also allow the timeline to be completely extensible. Note, I needed an NFA because there are a few input strings that require up to 5 transitions (the tree gets really deep) and the states couldn't really be just jumped to because of the way the TreeCtrl works. After lunch, I implemented this NFA (as it turns out, there was a slight amount of copy and paste) and had it working and fully functional within an hour. On Wednesday, I arranged a short meeting with my mentor to show it off and discuss some requirements for a side project I was to begin working on. When my mentor showed up to the meeting, he brought with him a Qualstar for me! Qualstars are internal awards given to employees who exceed expectations and provide excellent work (or something like that) and I got one for completing my first project so quickly and saving engineer time. Needless to say, my mentor was not impressed with my NFA and gave me some more work to do…

I FINALLY USED SOMETHING I LEARNED AT UCSD WHICH I NEVER THOUGHT I WOULD USE!!!!! i used CSE 105 to design my NFA and knew that an NFA would be a good way to solve this problem only because of that class. Also, that Qualstar i got kinda feels like a real world A+, except it actually means something :p

Chris Miranda (CSE 197)
Simulating NFA with DFA

Not quite a closure proof, but …

Proof:
Given name variables for sets, machines assumed to exist.
WTS state goal and outline plan.
Construction using objects previously defined + new tools working towards goal. Give formal definition and explain.
Correctness prove that construction works.
Conclusion recap what you've proved.
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

**Proof:**

*Given* $A$, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

*WTS* there is some DFA $M$ with $L(M) = A$

**Construction**

**Correctness**

**Conclusion**
From NFA to DFA

What is the tree of computation paths?
Subset construction

Given A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

WTS there is some DFA $M$ with $L(M) = A$

Construction Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q' = \text{the power set of } Q = \{ X | X \text{ is a subset of } Q \}$
- $q_0' = \{ q_0 \}$
- $F' = \{ X | X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \}$
- $\delta' ( \quad ) =$
Subset construction

**Given** A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

**WTS** there is some DFA $M$ with $L(M) = A$

**Construction** Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q' = \text{the power set of } Q = \{ X | X \text{ is a subset of } Q \}$
- $q_0' = \{ q_0 \}$
- $F' = \{ X | X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \}$
- $\delta'(X, x) = \bigcup_{q \in X} \delta(q, x)$

What are the arguments of $\delta'$?

A. $\delta'(q, x)$ where $q$ in $Q$ and $x$ in $\Sigma$
B. $\delta'(q, x)$ where $q$ in $Q$ and $x$ in $\Sigma \epsilon$
C. $\delta'([q], x)$ where $q$ in $Q$ and $x$ in $\Sigma \epsilon$
D. $\delta'(X, x)$ where $X$ is a subset of $Q$ and $x$ in $\Sigma$
E. I don't know
Subset construction

Given A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

WTS there is some DFA $M$ with $L(M) = A$

Construction Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q' = \text{the power set of } Q = \{ X | X \text{ is a subset of } Q \}$
- $q_0' = \{ q_0 \}$
- $F' = \{ X | X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \}$
- $\delta' (X, x) = \{ q \in Q | q \text{ is in } \delta(r, x) \text{ for some } r \text{ in } X \}$
Subset construction example

How big is $Q'$?

A. 2  
B. 4  
C. 5  
D. 16  
E. I don't know
What is the initial state $q_0'$?

A. $q_0$
B. $q_3$
C. $\{q_0,q_1,q_2,q_3\}$
D. $\{q_0\}$
E. I don't know
Subset construction example
Subset construction example

NFA

DFA

inaccessible states
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

Proof:
Given $A$, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

WTS there is some DFA $M$ with $L(M) = A$

Construction Define $M = (Q', \Sigma, \delta', q_0', F')$ with $Q' = P(Q)$,...
$q_0' = \{q_0\}$, $\delta' (X, x) = \{ q \in Q | q \text{ is in } \delta(r, x) \text{ for some } r \in X \}$

Correctness ??

Conclusion
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

**Proof:**

*Given* A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

*WTS* there is some DFA $M$ with $L(M) = A$

*Construction* Define $M = (Q', \Sigma, \delta', q_0', F')$ with …

**Correctness** ?? • • •

**Conclusion**

Details, with epsilon transitions: Sipser 55-56
Application

A language $A$ over $\Sigma$ is **regular** if and only if

- it is recognized by a DFA
- it is recognized by a NFA

To prove that the class of regular languages is closed under operation .... :

Let $A$ be a regular language, so recognized by DFA $M$. Build a **NFA** that recognizes the result of .... on $A$. Conclude this result is also a regular language.
The regular operations

For A, B languages over same alphabet, define:

\[ A \cup B = \{ x | x \in A \text{ or } x \in B \} \]

\[ A \circ B = \{ xy | x \in A \text{ and } y \in B \} \]

\[ A^* = \{ x_1x_2 \ldots x_k | k \geq 0 \text{ and each } x_i \in A \} \]

How can we prove that the concatenation of two regular languages is a regular language?
Concatenation

- "Guess" some stage of input at which switch modes

Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ build

$$N = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$$ with $\delta$...
Concatenation

\[ \delta(q, x) = \]

if q is in Q1, x is in \( \Sigma \)

if q is in Q2, x is in \( \Sigma \)

?
Concatenation

\[ \delta( q, x ) = \begin{cases} \delta_1( q, x ) & \text{if } q \text{ is in } Q_1, \ x \text{ is in } \Sigma \\ \delta_2( q, x ) & \text{if } q \text{ is in } Q_2, \ x \text{ is in } \Sigma \end{cases} \]
Concatenation

\[ \delta(q, x) = \begin{cases} 
\delta_1(q, x) & \text{if } q \text{ is in } Q_1, x \text{ is in } \Sigma \\
\delta_2(q, x) & \text{if } q \text{ is in } Q_2, x \text{ is in } \Sigma \\
\{q_2\} & \text{if } q \text{ is in } F_1, x = \varepsilon \\
\emptyset & \text{otherwise} 
\end{cases} \]

Correctness proof in the book (page 61)
Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, build

$$N = (Q_1 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup \{q_0\})$$

and $\delta(q,x) = \ldots$

*Construction in the book (page 63)*
Regular languages

To prove that a set of strings over the alphabet $\Sigma$ is regular,

- Build a **DFA** whose language is this set.
- Build an **NFA** whose language is this set.
- Use the **closure properties** of the class of regular languages to construct this set from others known to be regular.
  - Union
  - Intersection
  - Complementation
  - Concatenation
  - Flip bits
  - Kleene star
Inductive application of closure

R is a regular expression over Σ if

1. \( R = a \), where \( a \in \Sigma \)
2. \( R = \varepsilon \)
3. \( R = \emptyset \)
4. \( R = (R_1 \cup R_2) \), where \( R_1, R_2 \) are themselves regular expressions
5. \( R = (R_1 \circ R_2) \), where \( R_1, R_2 \) are themselves regular expressions
6. \( (R_1^*) \), where \( R_1 \) is a regular expression.
Regular expressions

Conventions:
- Σ is shorthand for (0 U 1) if Σ = {0,1}
- Parentheses may be omitted
  - Precedence: star, then concatenation, then union
- R^+ is shorthand for RR^*, R^k is shorthand for R concatenated with itself k times
- Circle indicated concatenation may be omitted

Which of the following is not a regular expression over {0,1}?
A. (ΣΣΣΣΣ)^*   B. Σ ∩ 1   C. 1^*∅0   D. εε   E. I don't know
The language described by a regular expression, $L(R)$:

- $L(\varepsilon) = \{\varepsilon\}$
- $L(\varepsilon \omega) = \{\varepsilon \omega\}$
- $L(\omega) = \{\omega\}$
For next time

Haskell project 1 due tomorrow
Discussion sections tomorrow
Homework 3 due next week

**Budget for invalid regrade requests**

Exam 1: one week from today!