CSE 105
THEORY OF COMPUTATION

Friday, January 27, 2017

Winter 2017

http://cseweb.ucsd.edu/classes/wi17/cse105-ab/
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
Regular Expressions

$R$ is a **regular expression** over $\Sigma$ 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.

Watch out for overloaded symbols!
Theorem: A language is regular if and only if some regular expression describes it.

Lemma 1.55: If a language is described by a regular expression, then it is regular.

Lemma 1.60: If a language is regular, then it is described by some regular expression.
Idea: basic regular expressions are easy to implement as DFA, for inductive step of definition, use closure under regular operations.

E.g.: build NFA recognizing the language described by \((00 \cup 11)^*\)
DFA to regular expression

- Idea: use intermediate model GNFA whose labels are regular expressions
DFA to regular expression  Example 1.68, page 76
All roads lead to ... regular sets?

Are there any languages over \{0,1\} that are not regular?

A. Yes: a language that is recognized by an NFA but not any DFA.
B. Yes: there is some infinite language of strings over \{0,1\} that is not described by any regular expression.
C. No: all languages over \{0,1\} are regular because that's what it means to be a language.
D. No: for each set of strings over \{0,1\}, some DFA recognizes that set.
E. I don't know.
Counting

• **Fact:** a countable union of countable sets is countable.
• **Fact:** $\{0,1\}^*$ is countably infinite. $X^*$ is countably infinite when $X$ is finite.
• **Fact:** the set of subsets of a countably infinite set is uncountable.

• **Fact:** there are countably many DFA with $\Sigma=\{0,1\}$
• **Fact:** there are countably many regular languages over $\{0,1\}$
Counting

- **Fact:** a countable union of countable sets is countable.
- **Fact:** \( \{0,1\}^* \) is countably infinite. \( X^* \) is countably infinite when \( X \) is finite.
- **Fact:** the set of subsets of a countably infinite set is uncountable.
- **Fact:** there are countably many DFA with \( \Sigma = \{0,1\} \).
- **Fact:** there are countably many regular languages over \( \{0,1\} \).
- **Fact:** uncountably many languages over \( \{0,1\} \).
Proving nonregularity

How can we prove that a language is nonregular?

A. Design a DFA to recognize it and prove that it doesn’t work (that it doesn’t actually recognize the language).
B. Prove that it's a strict subset of some regular language.
C. Prove that it's the union of two regular languages.
D. Prove that its complement is not regular.
E. I don't know.
Where we stand

• There exist nonregular languages.

• If we know that some languages are not regular, we can conclude others are also not regular judiciously reasoning using closure properties of class of regular languages.

• No example of a specific nonregular language ... yet.
Bounds on DFA

- In DFA, memory = states

- Automata can only "remember"…
  - …finitely far in the past
  - …finitely much information

- If a computation path visits the same state more than once, the machine can't tell the difference between the first time and future times it visited that state.
Example!

\{ 0^n1^n \mid n \geq 0 \}

What are some strings in this set?  
What are some strings not in this set?  

Compare to $L(0^*1^*)$  
Design a DFA? NFA?
Example!

\[ \{ 0^n1^n \mid n \geq 0 \} \]

What are some strings in this set?
What are some strings not in this set?

Compare to \( L(0^*1^*) \)
Design a DFA? NFA?
Pumping

- Focus on computation path through DFA
Pumping

- Focus on computation path through DFA
Pumping

- Focus on computation path through DFA

Idea: if one long string is accepted, then many other strings have to be accepted too
Pumping Lemma

If $A$ is a regular language, then there is a number $p$ (the pumping length) where, if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into three pieces, $s = x y z$ such that

- $|y| > 0$, and
- for each $i \geq 0$, $xy^i z \in A$,
- $|xy| \leq p$. 

Sipser p. 78 Theorem 1.70
Pumping Lemma

If $A$ is a regular language, then there is a number $p$ (the pumping length) where, if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into three pieces, $s = x y z$ such that

- $|y| > 0$, and
- for each $i \geq 0$, $xy^iz \in A$,
- $|xy| \leq p$. 

Sipser p. 78 Theorem 1.70

# states in DFA recognizing $A$

Transition labels along loop