CSE 105
THEORY OF COMPUTATION

Winter 2022

https://cseweb.ucsd.edu/classes/wi22/cse105-a/
Today's learning goals

- 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

DFA $\leftrightarrow$ regular exp.
Inductive application of closure

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.

\( \Sigma \) is shorthand for \( (0 \cup 1) \) if \( \Sigma = \{0, 1\} \), Parentheses may be omitted, \( R^+ \) means \( RR \), \( R^k \) means \( R \) concatenated with itself \( k \) times

\[ R^3 = RR \cdot RR \cdot RR \]
Syntax 🙌 Languages

The language described by a regular expression, L(R):

• $L \left( (0 \cup 1) \cup 1 \right) = \{ \emptyset, 1 \}$

• $L \left( (\Sigma \Sigma \Sigma \Sigma)^* \right) = \{ w \in \Sigma^*: \text{length divisible by 4} \}$

• $L \left( 1^* \emptyset 0 \right) = \{ \emptyset, 1, 11, 111, \ldots \}$

1. $R = a$, where $a \in \Sigma$
2. $R = \varepsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$
5. $R = (R_1 \circ R_2)$
6. $(R_1^*)$
Which of the following strings is not in the language described by

\[
\left( (00)^* (11) \right)^* \cup 01^*
\]

A. 00
B. 01
C. 1101
D. \( \varepsilon \)
E. I don't know
Let $L$ be the language over \{a,b\} described by the regular expression

$$a \cup \emptyset \quad a \quad ((a \cup \emptyset) \ b^*)^* \quad (a b^*)^*$$

Which of the following is not true about $L$?

A. Some strings in $L$ have equal numbers of a's and b's
B. $L$ contains the string aaaaaa
C. a's never follow b's in any string in $L$
D. $L$ can also be represented by the regular expression $(a b^*)^*$
E. More than one of the above.
Regular expressions in practice

- **Compilers**: first phase of compiling transforms Strings to Tokens *keywords, operators, identifiers, literals*
  - One regular expression for each token type

- **Other software tools**: grep, Perl, Python, Java, Ruby, …
"Regular = regular"  

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.
L(R) to NFA (to DFA)

• 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

Book version

- Idea: use intermediate model GNFA whose labels are regular expressions

- E.g.: build regular expression describing language recognized by
DFA to regular expression

My version

- Idea: directly from NFAs to regular expressions

- Use induction on number of edges in NFA

- Lemma: for any $r$, if $r$ is a NFA with $r$ edges, then there is a regular expression such that
DFA to regular expression
My version

• Lemma: for any , if is a NFA with edges, then there is a regular expression such that

• Notations:
  • Start node of is
  • Accept nodes of are

• Base case: . There are two cases:
  • : In this case and
  • : In this case and
DFA to regular expression

My version

• Lemma: for any \( M \) is a NFA with \( n \) edges, then there is a regular expression \( \epsilon \) such that

• Notation: Start node of \( M \) is \( s \), accept nodes of \( M \) are \( \{a_1, a_2, \ldots, a_k\} \)

• Induction case: Assume true for \( n-1 \), prove for \( n \).
• Pick edge \( e \) in \( M \) labeled by \( a \) (that is: \( (s, a) \))
• Let \( M' \) be with the edge \( e \) removed
• Note: \( M' \) has \( n-1 \) edges
• Induction: exists regular language \( \mathcal{L} \) such that
• Notice:
• Next goal: complete the difference
DFA to regular expression

My version

• Induction case: Assume true for n-1, prove for n.
• Notation: start node of , accept nodes of
• is with some edge labeled by removed
• Define 3 new NFAs:
  • : same as , but with start node and accept nodes
  • : same as , but with start node and accept nodes
  • : same as , but with start node and accept nodes
• All NFAs with n-1 edges, so can apply induction for each
• Completing the proof:
DFA to regular expression

My version

1. \[ M_{\text{start}}: q \rightarrow q' \]
   \[ R_{\text{start}} \]

2. \[ M_{\text{end}}: q'' \rightarrow F \]
   \[ \text{Read} \]

3. \[ M_{\text{middle}}: q'' \rightarrow q' \]

\[ L(C_{R_i}) \cup L(C_{R_{\text{start}}}) \cdot a \cdot L(R_{\text{end}}) \]

\[ L(R_{\text{start}}) \cdot (a \cdot L(R_{\text{middle}}))^* \cdot a \cdot L(R_{\text{end}}) \]

\[ L(M_i) \subseteq L(M) \]
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 sets 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 \(\{0,1\}\).

Uncountably many languages over \(\{0,1\}\)

Countably many regular languages over \(\{0,1\}\)
Proving nonregularity

How can we prove that a set is non-regular?

A. Try to design a DFA that recognizes it and, if the first few attempts don't work, conclude there is none that does.

B. Prove that it's a strict subset of some regular set.

C. Prove that it's the union of two regular sets.

D. Prove that its complement is not regular.

E. I don't know.
Where we stand

• There exist non-regular sets.

• If we know that some sets 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 regular set ... 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.