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

• Give an example of a non-regular language
• Outline two strategies for proving that there are non-regular languages
Reminders

• **HW 3** due Friday

• **Exam 1** next week
  • Wednesday April 20 evening
  • 8pm-9:50pm
  • **SOLIS 107** (seating chart will be released shortly)
  • One (double-sided) handwritten 3" by 5" index card for notes

• **Review session for exam 1 on Monday April 18, Peterson 108, 8-10**
Proofs of correctness

• Is $w$ in $L(M)$?
• Need to consider sequence of states visited by machine
Proofs of correctness

For $M = (Q, \Sigma, \delta, q_0, F)$ a DFA, with $\delta : Q \times \Sigma \to Q$ its transition function, define $\delta^* : Q \times \Sigma^+ \to Q$ recursively as

$$\delta^*(q, a) = \delta(q, a) \quad \text{for } a \in \Sigma$$
$$\delta^*(q, aw) = \delta(\delta(q, a), w) \quad \text{for } a \in \Sigma, w \in \Sigma^+$$

"A nonempty string $w$ is accepted by $M$ iff $\delta^*(q_0, w) \in F$ "

A. True  C. I don't know.
B. False
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 \( \Sigma \) 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.

Sipser 1.52 p. 64

Watch out for overloaded symbols!
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*∅∅  D. εε  E. I don't know
Syntax $\rightarrow$ Languages

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

- $L(\epsilon) = \{\}$
- $L(\epsilon) = \{\}$
- $L(\epsilon) = \{\}$
- $L(\epsilon) = \{\}$
- $L(\epsilon) = \{\}$
Which of the following strings is not in the language described by 

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

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

\[
((a \cup \emptyset) \ b)^*\]

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 aaaaaaa
C. a's never follow b's in any string in \( L \)
D. \( L \) can also be represented by the regular expression \((ab^*)^*\)
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, …
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)  

Lemma 1.55, page 67

• 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 U 11)*
DFA to regular expression

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

• E.g.: build regular expression describing language recognized by
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.
Example!

\{ a^n b^n \mid n \geq 0 \}

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

Compare to \( L(a^*b^*) \)
Design a DFA?
Example!

\{ a^n b^n \mid n \geq 0 \}

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

Compare to \( L(a^*b^*) \)
Design a DFA? NFA?
What is the length of the shortest word accepted by this DFA?

What is the length of the longest word accepted by this DFA?

What is the language of this DFA?
What is the length of the longest word accepted by this DFA without visiting any state more than once?

A. 0  
B. 1  
C. 2  
D. 3  
E. 4
What feature do computation paths of longer words that are accepted by the DFA have?

A. They must visit q3 at least once.
B. They must visit q1 more than once.
C. They must start at q0.
D. More than one of the above.
E. I don't know.
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.
Reminders

• **HW 3** due Friday

• **Exam 1** next week
  • Wednesday April 20 evening
  • 8pm-9:50pm
  • **SOLIS 107** (seating chart will be released shortly)
  • One (double-sided) handwritten 3" by 5" index card for notes

• *Review session for exam 1 on Monday April 18, Peterson 108, 8-10*