Last Time: Definition of a Regular Expression

R is a regular expression iff
- R is a string over \( \Sigma \cup \{ \varepsilon, \emptyset, (, ), \cup, \ast \} \) and R is:
  1. Some symbol \( a \in \Sigma \), or
  2. \( \varepsilon \), or
  3. \( \emptyset \), or
  4. \((R_1 \cup R_2)\) where \( R_1 \) and \( R_2 \) are regular exps., or
  5. \( R_1R_2 = R_1 \circ R_2 \) where \( R_1 \) and \( R_2 \) are reg. exps., or
  6. \( R_1^* \) where \( R_1 \) is a regular expression.

Precedence: Evaluate \( \ast \) first, then \( \circ \), then \( \cup \)

E.g. \( 0 \cup 11^* = 0 \cup (1 \circ (1^*)) = \{0\} \cup \{1, 11, 111, \ldots\} \)

Examples

What is R for each of the following languages?

1. \( L(R) = \{ w \mid w \text{ contains exactly two 0's} \} \)
2. \( L(R) = \{ w \mid w \text{ contains at least two 0's} \} \)
3. \( L(R) = \{ w \mid w \text{ contains an even number of 0's} \} \)
4. \( L(R) = \{ w \mid w \text{ is a valid identifier in C} \} \)
5. \( L(R) = \{ w \mid w \text{ is what you hear in a Kevin Bacon movie} \} \)
6. \( L(R) = \{ w \mid w \text{ does not contain 00} \} \)

(Example on board)
Regular Expressions and Finite Automata

- What is the relationship between regular expressions and DFAs/NFAs?

- Specifically:
  1. **R \rightarrow NFA**? Given a reg. exp. R, can we create an NFA N such that L(R) = L(N)?
  2. **NFA \rightarrow R**? Given an NFA N (or its equivalent DFA M), can we come up with a reg. exp. R such that L(M) = L(R)?

  I think so...do you??

From Regular Expressions to NFAs

- Problem: Given *any* regular expression R, how do we construct an NFA N such that L(N) = L(R)?

- Soln.: Use the multi-part definition of regular expressions!!
  - Show how to construct an NFA for each possible case in the definition: R = a, or R = ε, or R = ∅, or R = (R1 ∪ R2), or R = R1°R2, or R = R1*.

  Told ya ‘twas possible!

- Example: Draw NFA for 01Σ*10
From NFAs/DFAs to Regular Expressions

✦ Problem: Given any NFA N, how do we construct a regular expression R such that \( L(N) = L(R) \)?

✦ Solution: First, convert NFA N to an equivalent DFA M to keep things simple. Then:
  ≫ Idea: Collapse 2 or more edges in M labeled with single symbols to a new edge labeled with an equivalent regular expression
  ≫ This results in a “generalized” NFA (GNFA)
  ≫ Our goal: Get a GNFA with 2 states (start and accept) connected by a single edge labeled with the required regular expression R

Steps for extracting regular expressions from DFAs:
1. Add new start state connected to old one via an \( \varepsilon \)-transition
2. Add new accept state receiving \( \varepsilon \)-transitions from all old ones
3. Keep applying 2 rules until only start and accept states remain:
   1. Collapse Parallel Edges:
   2. Remove “loopy” states:

(Example: On board and in textbook)