Definition of a Regular Expression

R is a regular expression iff
R is a string over Σ ∪ { ε, ∅, (, ), ∪, * } and R is:
1. Some symbol a ∈ Σ, or
2. ε, or
3. ∅, or
4. (R₁ ∪ R₂) where R₁ and R₂ are regular expressions, or
5. R₁R₂ = R₁°R₂ where R₁ and R₂ are regular expressions, or
6. R₁* where R₁ is a regular expression.

Precedence: Evaluate * first, then °, then ∪
E.g. 0 ∪ 11* = 0 ∪ (1° (1*)) = {0} ∪ {1, 11, 111, …}

Examples

What is R for each of the following languages?
1. L(R) = {w | w contains exactly two 0’s}
2. L(R) = {w | w contains at least two 0’s}
3. L(R) = {w | w contains an even number of 0’s}
4. L(R) = {w | w does not contain 00}
5. L(R) = {w | w is a valid identifier in C}
6. L(R) = {w | w is a word bleeped out in a Snoop Dogg song}
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 \( L(M) = L(R) \)?

Kevin Bacon: 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: \( R = a, or R = \epsilon, or R = \emptyset, or R = (R1 \cup R2), or R = R1^*R2, or R = R1^* \).

Kevin Bacon: Told ya 'twas possible!

✦ Example: Draw NFA for \( 10^*01 \)
From NFAs/DFAs to Regular Expressions

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

✦ Solution:

✦ Idea: Collapse 2 or more edges in N 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 NFAs/DFAs:
1. Add new start state connecting to old one via an ε-transition
2. Add new accept state receiving ε-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)