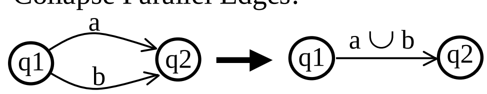
CSE 322: Regular Expressions and Finite Automata II

- **Question from Last Time**: Are regular expressions and NFAs/DFAs equivalent?
- **We showed:**
  - **R → NFA**: We can convert any reg. exp. R into an equivalent NFA N such that L(R) = L(N)
- How about showing the converse?
  - **NFA → R**? Given an NFA N (or its equivalent DFA M), is there a reg. exp. R such that L(M) = L(R)?

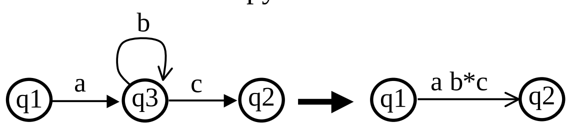
From DFAs to Regular Expressions

- Steps for extracting regular expressions from DFAs:
  1. Add new start state connected 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:



Note: Also applies to q1 = q2

     2. Remove "loopy" states:



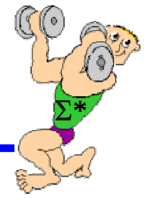
Note: Also applies to q1 = q2

(Example DFA: {w | # 0's in w is not divisible by 3) on the board

Beyond the Regular world…

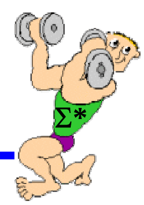
- Are there languages that are *not* regular?
- **Idea:** If a language violates a property obeyed by all regular languages, it cannot be regular!
  - **Pumping Lemma** for showing *non-regularity* of languages

The Pumping Lemma for Regular Languages



- **What is it?**
  - A statement (“lemma”) that is true for all regular languages
- **Why is it useful?**
  - Can be used to show that certain languages are *not regular*
  - How? *By contradiction*: Assume the given language is regular and show that it does not satisfy the pumping lemma

The Pumping Lemma for Regular Languages



- **What is the idea behind it?**
  - Any regular language L has a DFA M that recognizes it
  - If M has **p states** and accepts a string of length $\geq$ p, the sequence of states M goes through must contain a **cycle** (repetition of a state) due to the *pigeonhole principle*! Thus:
  - *All strings* that make M go through this cycle 0 or any number of times are also accepted by M and *should be in L.*

Formal Statement of the Pumping Lemma

- **Pumping Lemma**: If L is a regular language, then there exists a number p (the "pumping length") such that for all strings *s* in L such that $|s| \geq$ p, there exist *x*, *y*, and *z* such that $s = xyz$ and:
  1. $xy^iz \in$ L for all $i \geq 0$, and
  2. $|y| \geq 1$, and
  3. $|xy| \leq$ p.
- **More Plainly**: p = number of states of a DFA accepting L. Any string *s* in L of length ≥ p can be expressed as $s = xyz$ where *y* is not null (*y* is the cycle), $|xy| \leq$ p (cycle occurs within p state transitions), and any "pumped" string xy^iz is in L for all $i \geq 0$ (go through the cycle 0 or more times).
- Proved in 1961 by Bar-Hillel, Peries and Shamir.

The Pumping Lemma

- Proof on the board…(see page 79 in textbook)
  - See how it applies to {w | # 0's in w is not divisible by 3}
- In-Class Examples: Using the pumping lemma to show a language L is *not regular*
  - 5 steps for a proof by contradiction:
  1. Assume L is regular.
  2. Let p be the pumping length given by the pumping lemma.
  3. Choose cleverly an *s* in L of length at least p, such that
  4. For *any way* of decomposing *s* into *xyz*, where $|xy| \leq$ p and *y* isn't null,
  5. You can find an $i \geq 0$ such that xy^iz is not in L.

Weekend Exercise:

Try proving the following are not regular

using the 5 steps in the previous slide:

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

$\{0^n1^m \mid n > m\}$

$\{0^p \mid p \text{ is a prime number}\}$

Next Class: More on being Non-Regular

- Things to do over the weekend:
  - Download homework # 4 from course website:
    www.cs.washington.edu/education/courses/322/02au/assignments.html
  - Work on (and finish!) homework # 4 (due Friday, Nov 1)
  - Start reading Chapter 2 in the text
  - Have a great "pumping lemma" of a weekend!