Definition of a Regular Expression

R is a regular expression iff
R is a string over \( \Sigma \cup \{ \varepsilon, \emptyset, (, ), \cup, * \} \) 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 * 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 | 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} (or in Java)
  6. L(R ) = {w | w is a word heard on the MTV show “The Osbournes”}
Are u saying our language is regular??
Regular Expressions and Finite Automata

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

Specifically:

1. $R \rightarrow \text{NFA}$? Given a reg. exp. $R$, can we create an NFA $N$ such that $L(R) = L(N)$?

2. $\text{NFA} \rightarrow R$? 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 $R = \varepsilon$, or $R = \emptyset$, or $R = (R_1 \cup R_2)$, or $R = R_1 \circ R_2$, or $R = R_1^*$.  

✦ Example: Draw NFA for $10\Sigma^*01$
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 $R$
From NFAs/DFAs to Regular Expressions

Steps for extracting regular expressions from NFAs/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)
Beyond the Regular world…

Are there languages that are not regular?

How do we prove it?

Idea: If a language violates a property obeyed by all regular languages, it cannot be regular!

Pumping Lemma for showing non-regularity of languages

I love ze pumping lemma!

http://www.ipjnet.com/schwarzenegger2/pages/arnold_01.htm
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
More about the Pumping Lemma

- **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).
  - Why?
    - Due to the *pigeonhole principle*! $p$ states allow at most $p-1$ transitions before a state is repeated.
  - Therefore, *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 regular, then $\exists p$ such that $\forall s$ in $L$ with $|s| \geq p$, $\exists x, y, z$ with $s = xyz$ and:

1. $|y| \geq 1$, and
2. $|xy| \leq p$, and
3. $xy^iz \in L \ \forall \ i \geq 0$

- Proof on board…(also in the textbook)
- Proved in 1961 by Bar-Hillel, Peries and Shamir
Pumping Lemma in Plain English

- Let $L$ be a regular language and let $p = \text{“pumping length”} = \text{no. of states of a DFA accepting } L$

- Then, any string $s$ in $L$ of length $\geq p$ can be expressed as $s = xyz$ where:
  - $y$ is not empty ($y$ is the cycle)
  - $|xy| \leq p$ (cycle occurs within $p$ state transitions), and
  - any “pumped” string $xy^iz$ is also in $L$ for all $i \geq 0$ (go through the cycle 0 or more times)
Using The Pumping Lemma

- **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 *all ways* of decomposing s into xyz, where |xy| ≤ p and y is not null,
    5. There is an i ≥ 0 such that xy^i z is not in L.
Proving Non-Regularity using the Pumping Lemma

- In class examples: Show the following are not regular
  - $L_1 = \{0^n1^n \mid n \geq 0\}$ over the alphabet $\{0, 1\}$
  - $L_2 = \{ww \mid w \text{ in } \{0, 1\}^*\}$