Announcements

• Homework 8 is due Thursday in class.

• Homeworks 1-4 will be in front of my office (Lally 301) MTWF 12pm – 4pm this week.
Last Time – Regular Expressions

• **A regular expression** is a way of describing a regular language.
  
  • Uses the operations over languages: union, concatenation, Kleene-star

• **Theorem:** A language is regular if and only if there is a regular expression that describes it.
  
  • Implies all languages recognized by FA can be described by regular expressions
Last Time - NFAs

• NFA generalize FA by adding nondeterminism
  • Allows several alternative computations for the same input string.
  • Alternative computations execute in parallel on different copies of the machine.

• Two changes:
  • Allow $\delta(q, a)$ to specify more than one successor state

$$\delta(q_i, a) = \{ q_j, q_k \}$$

• Add $\lambda$-transitions: transitions made for “free” without consuming any input.
Equivalence of NFAs and DFAs

• Theorem: A language is regular if and only if it is accepted by some NFA.

• * DFAs and NFAs have the same computational power.
Bonus Quiz : 1 point on Final Exam
Work in groups of 2 or 3.

Construct an NFA that accepts that language described by the regular expression \(((01)^+0)^+\).
CONTEXT-FREE LANGUAGES
Context-Free Languages

\[ \{a^n b^n\} \quad \{ww^R\} \]

Regular Languages
Context-Free Languages

• A context free language is a language that can be recognized by a new type of computational model – a pushdown automata.
Context Free Grammars

• A grammar gives rules for expressing a language.
• A context free language can be described by a context-free grammar

• Context-free grammars can be used to describe:
  • Programming languages
    • Used by compilers
  • The English language
    • Used in Natural Language Processing, Grammar checkers
Example of a Context-Free Grammar

\[
\begin{align*}
\langle \text{sentence} \rangle & \rightarrow \langle \text{noun \_ phrase} \rangle \langle \text{predicate} \rangle \\
\langle \text{noun \_ phrase} \rangle & \rightarrow \langle \text{article} \rangle \langle \text{noun} \rangle \\
\langle \text{predicate} \rangle & \rightarrow \langle \text{verb} \rangle \\
\langle \text{article} \rangle & \rightarrow a \\
\langle \text{article} \rangle & \rightarrow \text{the} \\
\langle \text{noun} \rangle & \rightarrow \text{cat} \\
\langle \text{noun} \rangle & \rightarrow \text{dog} \\
\langle \text{verb} \rangle & \rightarrow \text{runs} \\
\langle \text{verb} \rangle & \rightarrow \text{walks}
\end{align*}
\]
\[\text{sentence} \rightarrow \text{noun \_ phrase} \text{ predicate}\]
\[\text{noun \_ phrase} \rightarrow \text{article} \text{ noun}\]
\[\text{predicate} \rightarrow \text{verb}\]
\[\text{article} \rightarrow a\]
\[\text{article} \rightarrow \text{the}\]
\[\text{noun} \rightarrow \text{cat}\]
\[\text{noun} \rightarrow \text{dog}\]
\[\text{verb} \rightarrow \text{runs}\]
\[\text{verb} \rightarrow \text{walks}\]
Formal definition of a CFG

\[ S \rightarrow aSb \]
\[ S \rightarrow \lambda \]

Variables are capital letters. Terminals are lower-case letters.
Language of a Grammar

- The sequence of substitutions used to obtain a particular string is called a **derivation**.
  - We write $S \Rightarrow^{*} w$
- The **language of a grammar**, denoted $L(G)$, is the set of all strings generated by derivations.

\[
S \rightarrow aSb \\
S \rightarrow \lambda
\]
Another Example Grammar

\[ S \rightarrow Ab \]
\[ A \rightarrow aAb \]
\[ A \rightarrow \lambda \]
Convenient Notation

• We can combine rules that have the same variable on the left-hand side.
• The “|” symbol acts as an “or”
Designing a CFG

Design a context-free grammar $G$ such that

$$L(G) = \{ww^R : w \in \{a, b\}^*\}$$
Designing a CFG (2)

• Let $\Sigma = \{a,b\}$. Design a CFG that generates the language of strings with an even number of $b$’s
• Let \( \Sigma = \{a, +, \times, (, )\} \)
  and let the rules be
  \[
  E \rightarrow E + E \mid E \times E \mid (E') \mid a
  \]
• Give a derivation for the string \( a + a \times a \)
Ambiguity

• Let $\Sigma = \{a, +, \times, (, )\}$ and let the rules be

$$E \rightarrow E + E \mid E \times E \mid (E) \mid a$$

• Give a derivation for the string $a + a \times a$

• The string has two leftmost derivations
  • At every step, we replace the leftmost variable.
• A CFG is ambiguous if some string $w \in L(G)$ has at least two different leftmost derivations
Inherent Ambiguity

- Ambiguity is bad for programming languages.
- It leads to two or more ways to parse statements
  - Makes compilers sad
  - They do not know how to resolve such ambiguities

Some context-free languages have only ambiguous grammars
- For example, \( L = \{ a^{i}b^{j}c^{k} : i,j,k \geq 0 \text{ and } i = j \text{ or } j = k \} \) is inherently ambiguous