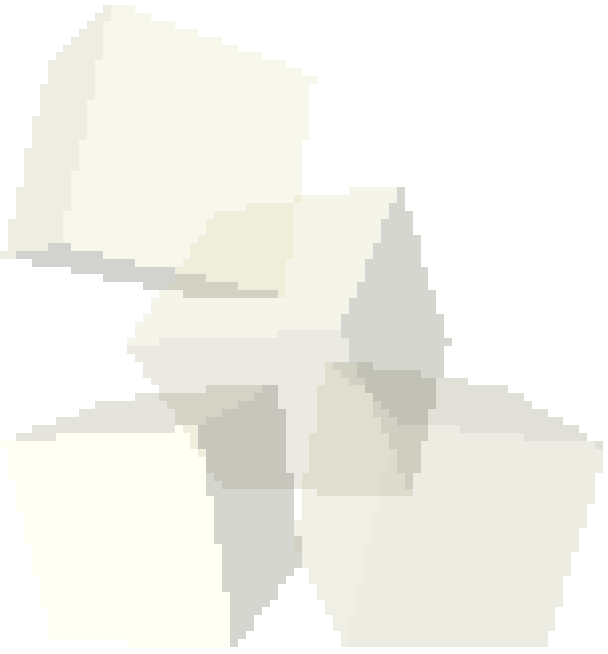
Grammars and Chompsky Hierarchy

4-7-2010



Opening Discussion

- What did we talk about last class?
- Was there anything in the reading you found interesting?
- Do you have any questions about the next assignment?

Grammars

- Grammars are a central concept in theoretical computer science. They are formal ways of specifying “languages” or sets of strings and how those strings can be produced.
- The general idea of grammars is that you have productions that map a string to another string. How these productions are applied varies between different types of grammars. Some styles of grammars also have limitations on the strings that can be on either side of a production.

Chomsky Grammars

- Noam Chomsky developed a hierarchy of grammars that have become standard models of different computational abilities.
- Chomsky grammars have sets of terminal and nonterminal characters along with a set of productions and a start symbol.
- By convention people generally represent nonterminal characters with capital letters and terminals with lower case letters.
- There are four classes of Chomsky grammars: regular, context free, context sensitive, and recursively enumerable.

Regular Expressions

- Regular grammars have productions of the following forms:
  - A -> a
  - A -> Ba or A -> aB
- There is a single nonterminal on the left and either a single terminal or a terminal and nonterminal on the right. All productions with both a terminal and nonterminal have to agree on which comes first.
- These grammars produce languages that can be generated with finite state automata. They have no memory.

Context Free

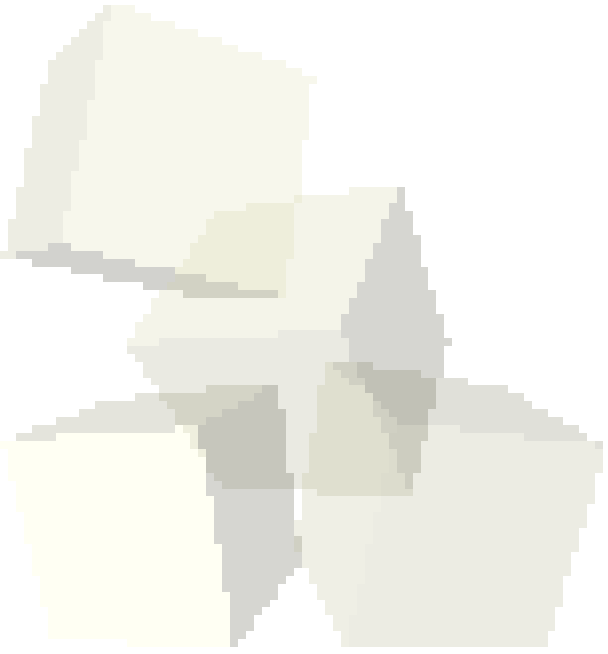
- Context Free (CF) grammars have productions with a single nonterminal on the left and any string of terminals and nonterminals on the right.
- The languages of CF grammars are those generated by pushdown automata.
- Most programming languages are defined by CF grammars.
- Have limited memory, but access to memory isn't random.

Context Sensitive

- Context Sensitive (CS) grammars have productions of the following form.
  - $\alpha A\beta \Rightarrow \alpha \gamma \beta$
- α , β , and γ are arbitrary strings of terminals and nonterminals.
- These are generated by a linear-bounded non-deterministic Turing machine.
- These are the least used of the grammars. Even their theory isn't all that well understood.

Recursively Enumerable

- Recursively enumerable grammars allow any type of production.
- These grammars are computationally equivalent to a full Turing machine so they can generate anything that you want within the bounds of what can be computed.



Other Grammars

- There are other types of grammars that aren't part of the Chomsky hierarchy.
- I am fond of L-systems. They are an example of a non-Chomsky set of grammars and they also have different levels of complexity.
- The primary difference between L-systems and Chomsky grammars is that Chomsky grammars replace one randomly selected nonterminal at a time while L-systems replace everything at each step. They also don't technically have terminal symbols.

Perl Regular Expressions

- Most languages these days have some type of support for regular expressions. In languages like Java this is done through libraries.
- Perl has language features that do regular expressions so that they are easy to include and require minimal typing.
- Appendix B in your book has a brief introduction to Perl as a whole and also contains more details about regular expressions.

Features

- Regular expressions basically give you three capabilities. Each is fairly simple. When combined they become reasonably powerful and allow us to do a lot with little effort.
- Repetition – Use the * to denote 0 or more of the previous element. Use the + to denote 1 or more.
- Alternation – The | can separate two possible options that you allow. Character classes also let you specify sets of possibilities.
- Concatenation – The default behavior is for one thing to follow the next.
- Parentheses can be used to group things.

Special Perl Variables

- There are a number of special variables in Perl that get set when a regular expression is matched.
- $& has the section of the string that matched.
- $` gives everything before the match and $' gives everything after the match.
- The pieces of the match, as set off by parentheses, are also stored. This is a very helpful feature. They are in $1, $2, ... They a numbered by the position of the opening parentheses.

Reminders

- I changed up due dates for assignments so you should look at the schedule.