Grammars and Chompsky Hierarchy

10-27-2006

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Have you caught up with the reading? Was there anything in the reading you found interesting?
- Do you have any questions about the assignment?

Code

- I want us to play with file access some. The Run dialog will let you specify command line arguments that could be read in using the <> operator. Make a file that has a name on a line followed by a number (like a phone number). Put in several entires. Now read that in and use a hash to look things up by name. Now look them up by number instead.

Grammars

- Grammars are a central concept in theoretical computer science. They are formal way 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.
  - $\alpha A \beta \to \alpha \gamma \beta$
- 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.
- We will talk about L-systems a bit later. 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.

Reminders

- Read the next few chapters. They talk about doing regular expressions in Perl. This is a big part of the reason that we are studying Perl in this class, the fact that regular expressions are built into the language itself.