Grammars

10-31-2011

Opening Discussion

- Minute essay comments:
  - Using a comparison that returns an Int instead of a Boolean.
  - Duplicate keys?
- Unbalanced trees.
- Code from last time.
- How many of you have used regular expressions before?

Formal Languages

- Today we will introduce the concept of formal languages and grammars.
- These are formal sets of rules for building strings. The rules determine what strings are in a particular language.
- There are many ways of specifying languages. We will focus on the one that is most broadly used today.

Chomsky Grammars

- Noam Chomsky developed a hierarchy of grammar types that could be used to specify different languages.
  - Regular
  - Context-Free
  - Context-Sensitive
  - Recursively Enumerable
- Each of these can also be associated with a different type of machine or automaton.

Nature of Chomsky Grammars

- Chomsky grammars have terminals and non-terminals. Normally a terminal is lowercase and a non-terminal is uppercase.
- There is a special non-terminal called the start symbol, S.
- A string in “complete” when it contains only terminals.
- Rules specify what a non-terminal can be replaced with.

Regular Grammars

- The simplest Chomsky grammar type is the regular grammars. There are only two types of allowed rules:
  - $A \rightarrow a$
  - $A \rightarrow aB$
- Note that 'A' and 'B' represent any non-terminals and 'a' is any terminal.
- Equivalent to a finite state automaton. Have no memory.

Context-Free Grammars

- Allow more general rules:
  - $A \rightarrow \gamma$
- Where γ is any combination of terminals and non-terminals.
- Equivalent to a pushdown automaton. Has memory, but only as a stack.
- These are how we specify the syntax of programming languages. Can describe almost all natural language.

Context-Sensitive Grammars

- Takes surrounding characters into account:
  - $\alpha A \beta \rightarrow \alpha \gamma \beta$
- Equivalent to a linear bounded non-determinstic Turing machine.
- Not used all that much because of challenges. Needed for some elements of natural language.

Recursively Enumerable Grammars

- Allows basically any transformation.
  - $\alpha \rightarrow \beta$
- There are no bounds on what these can be.
- This is equivalent to a Turing machine. That means that you could calculate anything you want using one of these.

Regular Expressions

- One if the applications of these formal systems is the use of regular expressions to perform String operations.
- Scala has a class called scala.util.matching.Regex. You can get one of these by calling the r method on a String.
- This wraps the functionality of java.util.regex.Pattern and provides Scala style functionality and pattern matching.
- Let's look at API entries.

Minute Essay

- Questions?
- There is an IcP next class.