Announcements

• Reading assignments
  — 7th Edition, Section 5.3 and pp. 878-880
  — 6th Edition, Section 4.3 and pp. 817-819
  — 5th Edition, Section 3.4 and pp. 766

Highlights from last lecture

• Recursively defined sets
  — Strings
    • Alphabet $\Sigma$: a finite set of characters
    • $\Sigma^*$ the set of all strings over the alphabet $\Sigma$
      — $\lambda \in \Sigma^*$
      — If $w \in \Sigma^*$, $a \in \Sigma$ then $wa \in \Sigma^*$
  • Palindromes
  • Rooted binary trees
• Functions on Recursively defined sets
  — Application of structural induction

Rooted Binary trees

• Basis: ● is a rooted binary tree
• Recursive Step: If $T_1$ and $T_2$ are rooted binary trees then so is:

Functions defined on rooted binary trees

• $\text{size}(\bullet) = 1$
• $\text{size}(\overbrace{T_1}^{T_1} \overbrace{T_2}^{T_2}) = 1 + \text{size}(T_1) + \text{size}(T_2)$
• $\text{height}(\bullet) = 0$
• $\text{height}(\overbrace{T_1}^{T_1} \overbrace{T_2}^{T_2}) = 1 + \max\{\text{height}(T_1), \text{height}(T_2)\}$

For every rooted binary tree $T$

$\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$
Languages: Sets of Strings

- Sets of strings that satisfy special properties are called languages. Examples:
  - English sentences
  - Syntactically correct Java/C/C++ programs
  - All strings over alphabet $\Sigma$
  - Palindromes over $\Sigma$
  - Binary strings that don’t have a 0 after a 1
  - Legal variable names, keywords in Java/C/C++
  - Binary strings with an equal # of 0’s and 1’s (HW6)

Regular Expressions over $\Sigma$

- Each is a “pattern” that specifies a set of strings
- Basis:
  - $\emptyset$, $\lambda$ are regular expressions
  - $a$ is a regular expression for any $a \in \Sigma$
- Recursive step:
  - If $A$ and $B$ are regular expressions then so are:
    - $(A \cup B)$
    - $(AB)$
    - $A^*$

Each regular expression is a “pattern”

- $\lambda$ matches the empty string
- $a$ matches the one character string $a$
- $(A \cup B)$ matches all strings that either $A$ matches or $B$ matches (or both)
- $(AB)$ matches all strings that have a first part that $A$ matches followed by a second part that $B$ matches
- $A^*$ matches all strings that have any number of strings (even 0) that $A$ matches, one after another

Examples

- $0^*$
- $0^*1^*$
- $(0 \cup 1)^*$
- $(0^*1)^*$
- $(0 \cup 1)^*0110(0 \cup 1)^*$
- $(0 \cup 1)^*(0110 \cup 100)(0 \cup 1)^*$

Regular expressions in practice

- Used to define the “tokens”: e.g., legal variable names, keywords in programming languages and compilers
- Used in grep, a program that does pattern matching searches in UNIX/LINUX
- Pattern matching using regular expressions is an essential feature of hypertext scripting language PHP used for web programming
  - Also in text processing programming language Perl

Regular Expressions in PHP

- int preg_match ( string $pattern , string $subject,...)
- $\$pattern syntax:
  - [01] a 0 or a 1 ^ start of string $ end of string
  - [0-9] any single digit \ . period \ , comma \ - minus
  - any single character
  - ab a followed by b (AB)
  - (a|b) a or b (A U B)
  - a? zero or one of a (A U $\lambda$)
  - a* zero or more of a A*
  - a+ one or more of a AA*
- e.g. ^[\-+] ? [0-9]* (\ . | \ ) ? [0-9]+ $ General form of decimal number e.g. 9.12 or -9.8 (Europe)
More examples

- All binary strings that have an even # of 1’s

- All binary strings that don’t contain 101

Regular expressions can’t specify everything we might want

- **Fact**: Not all sets of strings can be specified by regular expressions
  — One example is the set of binary strings with equal #’s of 0’s and 1’s from HW6