CSE 311 Foundations of Computing I

Lecture 17

Structural Induction: Regular Expressions, Regular Languages

Autumn 2011

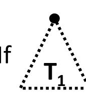
Announcements

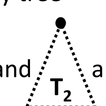
- Reading assignments
  - 7^{th} Edition, Section 5.3 and pp. 878-880
  - 6^{th} Edition, Section 4.3 and pp. 817-819
  - 5^{th} Edition, Section 3.4 and pp. 766

Highlights from last lecture

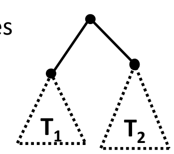
- Recursively defined sets
  - Strings
    - *Alphabet* Σ : a finite set of characters
    - Σ^* the set of all *strings* over the alphabet Σ
      - $\lambda \in \Sigma^*$
      - If $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$
    - Palindromes
  - Rooted binary trees
- Functions on Recursively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

Rooted Binary trees

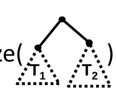
- Basis: ● is a rooted binary tree
- Recursive Step: If 

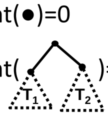
and 

are rooted binary trees then so is:



Functions defined on rooted binary trees

- size(●)=1
- size(

) = 1+size(T_1)+size(T_2)
- height(●)=0
- height(

)=1+max{height(T_1),height(T_2)}

For every rooted binary tree T 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 Σ
  - Palindromes over Σ
  - 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 Σ

- Each is a "pattern" that specifies a set of strings
- Basis:
  - $\varnothing$, λ 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** ∪ **B**)
    - (**AB**)
    - **A***

Each regular expression is a "pattern"

- λ matches the empty string
- ***a*** matches the one character string *a*
- (**A** ∪ **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*** ∪ ***1***)*
- (***0*1****)*
- (***0*** ∪ ***1***)* ***0110*** (***0*** ∪ ***1***)*
- (***0*** ∪ ***1***)* (***0110*** ∪ ***100***)(***0*** ∪ ***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** ∪ **B**)
a**?**	zero or one of a	(**A** ∪ λ)
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