University of Washington
Department of Computer Science and Engineering
CSE 311, Autumn 2011

November 10, 2011

Quiz Section, November 10, 2011

Homework Review

- Homework 6, Problem 1

  Prove that $f_1^2 + f_2^2 + \cdots + f_n^2 = f_n f_{n+1}$ when n is a positive integer.

  - Don't start by assuming P(k+1)
  - Two good methods:
    1. Start with P(k) and derive P(k+1)
    2. Start with LHS of goal and use P(k) to show it is equal to RHS of goal

  Let $P(x) = f_1^2 + f_2^2 + \cdots + f_x^2 = f_x f_{x+1}$ We will prove by mathematical induction that P(x) is true for all integers $x >= 1$.

  Base case:

  P(1) is true because $f_1^2 = f_1 f_2 = 1$

  Inductive hypothesis:

  Assume P(k) is true for some arbitrary integer $k >= 1$,

  That is, assume $f_1^2 + f_2^2 + \cdots + f_k^2 = f_k f_{k+1}$

  Inductive Step:

  We must show that P(k+1) is true, assuming P(k) is true.

  That is, we must show that $f_1^2 + f_2^2 + \cdots + f_{k+1}^2 = f_{k+1} f_{k+2}$:

$$\begin{aligned} f_1^2 + f_2^2 + \cdots + f_{k+1}^2 &= (f_1^2 + f_2^2 + \cdots + f_k^2) + f_{k+1}^2 \\ &= f_k f_{k+1} + f_{k+1}^2 \quad \text{by the inductive hypothesis} \\ &= f_{k+1}(f_k + f_{k+1}) \\ &= f_{k+1} f_{k+2} \end{aligned}$$

  Conclusion:

  By induction, $P(x)$ is true for all $x >= 1$.

- Homework 6, Problem 4

  Give a recursive definition of the set of bit strings that have the same number of zeros and ones.

  - You weren't required to give a proof, just a justification, but we provide the formal proof here.

  Basis step: $\lambda \in S$
  Recursive step: If $x \in S$ and $y \in S$, then:

  - $0x1 \in S$
  - $1x0 \in S$
  - $xy \in S$

  Proof:

  Let $B =$ the set of all balanced bit strings (bit strings having equal numbers of ones and zeroes). We must show that $S \subseteq B$ and $B \subseteq S$.

  To show that $S \subseteq B$, we use structural induction. First, the basis step specifies that λ is in S and we note that λ is also in B because it has 0 ones and 0 zeroes. Next we assume that all elements already in S are balanced bit strings, and we must show that all bit strings generated by the recursive step are also balanced bit strings. Both of the first two rules add 1 zero and 1 one to x, a bit string in S. Since x is already in S, it is a balanced bit string by the I.H. and so it has k zeroes and k ones for some integer k. Thus the resulting bit string has $k+1$ zeroes and $k+1$ ones and is therefore also in B. The third rule concatenates x and y, which are both in S. Because they are in S, we know by the I.H. that x has k zeroes and k ones and y has j zeroes and j ones for some integers k and j. Thus the resulting bit string has $j+k$ zeroes and $j+k$ ones and is therefore in B. We have proven that $S \subseteq B$.

  To show that $B \subseteq S$, we must show that every balanced bit string is in S. Let $P(n)$ be the statement that every balanced bit string of length n is in S. We will use strong induction to show that $P(n)$ holds for all integers $>= 0$.

  Base Case: $P(0)$ holds because the basis step of S's definition specifies that λ is in S.

  Inductive hypothesis: Assume $P(j)$ for $0 <= j <= k$ for some arbitrary integer k. That is, assume all balanced bit strings of length k or less are in S.

  Inductive step: We must show that $P(k+1)$ is true. That is, we must show that all balanced bit strings of length $k+1$ are in S.

  Let s be a balanced bit string of length $k+1$. There are two cases to consider:
  1) s starts and ends with different digits (0...1, 1...0)
  2) s starts and ends with the same digit (0...0, 1...1)

Case 1: Since s is balanced, we know that the middle part of the string (all the digits except the first and last digit) must also be balanced. Since this middle portion is a balanced bit string of length $k-1$, we know by the inductive hypothesis that it must be in S. Therefore, by either rule 1 or rule 2 of S's definition, s must also be in S.

Case 2: Assume, without loss of generality, that s starts and ends with 1. Consider the prefix of s that consists of just its first digit. This bit string has more 1's than 0's. Now consider the prefix of s that consists of all but its last digit. This bit string must have more 0's than 1's (otherwise, adding the final 1 would result in an unbalanced bit string).

Now consider each prefix of s in increasing order of length. Because the prefix of length 1 has a surplus of 1's, while the prefix of length k has a deficit of 1's, and because the difference between the 1's count and 0's count can change by only 1 with each added digit, there must be a "crossover" point. That is, there must be a prefix p of length m, where $2 <= m <= k-1$, such that p has an equal number of 0's and 1's. Since p is a balanced bit string with length no more than $k-1$, it must be in S by the inductive hypothesis.

Now consider f, the suffix of s that remains if we remove the prefix p. Because both s and p are balanced, f must also be balanced. And because p is of length at least 2, f's length can be no more than $k-1$. Thus, by the inductive hypothesis, f is also in S.

We have shown that s can be separated into two smaller balanced bit strings that are both in S. Thus, by rule 3 of S's definition, s must also be in S.

Conclusion:

By induction, $P(n)$ is true for all $n >= 0$.

We have shown that $S \subseteq B$ and $B \subseteq S$. Thus we have proven that S is the set of all bit strings with equal numbers of zeroes and ones.

New Stuff

1. Regular Expressions

   Express each of these sets using a regular expression:

   (a) The set of strings of odd length $(0 \cup 1)(00 \cup 01 \cup 10 \cup 11)^*$

   (b) The set of strings ending in 1 and not containing 000
   $(1 \cup 01 \cup 001)^*(1 \cup 01 \cup 001)$ (every group of 1 or 2 zeros must be followed by a 1)

   (c) The set of strings containing a string of 1's such that the number of 1's equals 2 modulo 3, followed by an even number of 0's.
   $11(111)^*(00)^*$

   (d) The set of binary strings with an equal number of 1's and 0's
   (trick question - not a regular language)

2. Context Free Grammars

   Give a grammar for each of these languages:

   (a) The set of all strings containing an equal number of 0's and 1's
   $S \to 0S1 \mid 1S0 \mid SS \mid \lambda$
   Note that there is another grammar given in the book (#15g in Section 12.1 or Section 13.1) which includes rules like $AB \to BA$ and $BA \to AB$. That one is NOT a context-free grammar. (Because the LHS of a CFG rule must consist of a single variable.)

   (b) The set of all strings containing more 0's than 1's
   $S_1 \to T0S_1 \mid T0T$
   $T \to 0T1 \mid 1T0 \mid TT \mid \lambda$

   (c) The set of all strings containing more 1's than 0's
   $S_2 \to U1S_2 \mid U1U$
   $U \to 0U1 \mid 1U0 \mid UU \mid \lambda$

   (d) The set of all strings containing an unequal number of 0's and 1's
   Combine the rules from parts b and c above and add one new rule:
   $S \to S_1 \mid S_2$
   $S_1 \to T0S_1 \mid T0T$
   $T \to 0T1 \mid 1T0 \mid TT \mid \lambda$
   $S_2 \to U1S_2 \mid U1U$
   $U \to 0U1 \mid 1U0 \mid UU \mid \lambda$

   Note that I intentionally used different variable names in parts b and c so I could combine them here. If you want to get the union of two languages by combining grammars that don't have disjoint sets of variable names, simply rename the variables in one grammar before combining the rules. (In this case, T and U are used identically, so i could combine them into one variable and simplify my combined grammar. But I did it this way to demonstrate the point about having disjoint sets of variable names.)

3. Combining CFGs

Let G_1 and G_2 be context-free grammars, generating the languages $L(G_1)$ and $L(G_2)$ respectively. Show that there is a CFG generating each of the following:

(a) $L(G_1) \cup L(G_2)$
This is the one we just gave an example of. In general you combine all the rules from G_1 and G_2 (renaming variables first if necessary), and you add a new rule: $S \to S_1 \mid S_2$. This gives us all strings that are generated by either G_1 or G_2.

(b) $L(G_1)L(G_2)$
Again you combine all the rules from both grammars and add one new rule. Here, the new rule is $S \to S_1 S_2$. This gives us all strings that consist of a string that G_1 generates, followed by a string that G_2 generates.

(c) $L(G_1)^*$
Here the new rule is $S \to S_1 S \mid \lambda$.
This gives us all strings that are the concatenation of any number of strings that G_1 generates.