CS 154 – Lecture 1 Notes (3/29/00)

Inductive Proofs

Prove a statement $S(X)$ about a family of objects X (e.g., integers, trees) in two parts:

1. *Basis*: Prove for one or several small values of X directly.
2. *Inductive step*: Assume $S(Y)$ for Y "smaller than" X; prove $S(X)$ using that assumption.

Example

A binary tree with n leaves has $2n-1$ nodes.

- Formally, $S(T)$: if T is a binary tree with n leaves, then T has $2n-1$ nodes.
- Induction is on the *size* = number of nodes of T.

Basis: If T has 1 leaf, it is a one-node tree. $1 = 2 \times 1 - 1$ so OK.

Induction: Assume $S(U)$ for trees with fewer nodes than T. In particular, assume for the subtrees of T.

- T must be a root plus two subtrees U and V.
- If U and V have u and v leaves, respectively, and T has t leaves, then $u+v=t$.
- By the inductive hypothesis, U and V have $2u-1$ and $2v-1$ nodes, respectively.
- Then T has $1+(2u-1)+(2v-1)$ nodes.
  - ❖ $= 2(u+v)-1$.
  - ❖ $= 2t-1$, proving the inductive step.

If-And-Only-If Proofs

Often, a statement we need to prove is of the form "X if and only if Y." We are then required to do two things:

1. Prove the *if-part*: Assume Y and prove X.
2. Prove the *only-if-part*: Assume X, prove Y.

Remember:

- The if and only-if parts are *converses* of each other.
- One part, say "if X then Y," says nothing about whether Y is true when X is false.

- An equivalent form to "if X then Y" is "if not Y then not X"; the latter is the *contrapositive* of the former.

Equivalence of Sets

Many important facts in language theory are of the form that two sets of strings, described in two different ways, are really the same set. To prove sets S and T are the same, prove:

- x is in S if and only if x is in T. That is:
  - ❖ Assume x is in S; prove x is in T.
  - ❖ Assume x is in T; prove x is in S.

Example: Balanced Parentheses

Here are two ways that we can define "balanced parentheses":

1. *Grammatically*:
   a) The empty string ϵ is balanced.
   b) If w is balanced, then (w) is balanced.
   c) If w and x are balanced, then so is wx.
2. *By Scanning*: w is balanced if and only if:
   a) w has an equal number of left and right parentheses.
   b) Every prefix of w has at least as many left as right parentheses.

- Call these GB and SB properties, respectively.
- Theorem: a string of parentheses w is GB if and only if it is SB.

If

An induction on $|w|$ (length of w). Assume w is SB; prove it is GB.

Basis: If $w = \epsilon$ (length = 0), then w is GB by rule (a).

- Notice that we do not even have to address the question of whether ϵ is SB (it is, however).

Induction: Suppose the statement "SB implies GB" is true for strings shorter than w.

1. Case 1: w is not ϵ, but has no nonempty prefix that has an equal number of (and). Then w must begin with (and end with); i.e., $w = (x)$.
   - ❖ x must be SB (why?).
   - ❖ By the IH, x is GB.

- By rule (b), (x) is GB; but $(x) = w$, so w is GB.

2. Case 2: $w = xy$, where x is the shortest, nonempty prefix of w with an equal number of (and), and $y \neq \epsilon$.

   - x and y are both SB (why)?
   - By the IH, x and y are GB.
   - w is GB by rule (c).

Only-If

An induction on $|w|$. Assume w is GB; prove it is SB.

Basis: $w = \epsilon$. Clearly w obeys the conditions for being SB.

Induction: Assume "GB implies SB" for strings shorter than w, and assume $w \neq \epsilon$.

1. Case 1: w is GB because of rule (b); i.e., $w = (x)$ and x is GB.

   - by the IH, x is SB.
   - Since x has equal numbers of ('s and)'s, so does (x).
   - Since x has no prefix with more ('s than)'s, so does (x).

2. Case 2: w is not ϵ and is GB because of rule (c); i.e., $w = xy$, and x and y are GB.

   - By the IH, x and y are SB.
   - (Aside) Trickier than it looks: we have to argue that neither x nor y could be ϵ, because if one were, the other would be w, and this rule application could not be the one that first shows w to be GB.
   - xy has equal numbers of ('s and)'s because x and y both do.
   - If w had a prefix with more)'s than ('s, that prefix would either be a prefix of x (contradicting the fact that x has no such prefix) or it would be x followed by a prefix of y (contradicting the fact that y also has no such prefix).
   - (Aside) Above is an example of *proof by contradiction.* We assumed our conclusion about w was false and showed it would imply something that we know is false.

Languages

- *Alphabet* = finite set of symbols, e.g., $\{0, 1\}$ (*binary* alphabet) or ASCII.
- *String* = finite sequence of symbols chosen from some alphabet, e.g., 01101 or `abracadabra`.
- *Language* = set of strings chosen from some alphabet.
  - ❖ Subtle point: the language may be infinite, but there is some finite set of symbols of which all its strings are composed.

Example; Languages

- The set of all binary strings consisting of some number of 0's followed by an equal number of 1's; that is, $\{\epsilon, 01, 0011, 000111, \ldots\}$.
- C (the set of compilable C programs).
- English.

Finite Automata

An important way to describe certain simple, but highly useful languages called "regular languages."

- A graph with a finite number of nodes, called *states.*
- Arcs are labeled with one or more symbols from some alphabet.
- One state is designated the *start state* or *initial state.*
- Some states are *final states* or *accepting states.*
- The language of the FA is the set of strings that label paths that go from the start state to some accepting state.

Example

- Below FA scans HTML documents, looking for a list of what could be title-author pairs, perhaps in a reading list for some literature course.
- It accepts whenever it finds the end of a list item.

- In an application, the strings that matched the title (before `' by '`) and author (after) would be stored in a table of title-author pairs being accumulated.