University of Washington
Department of Computer Science and Engineering
CSE 321

April 20, 2001

Induction Proofs for Recursively Defined Sets

These notes are intended to supplement the text by giving a definition of inductive proofs for recursively defined sets. A recursive definition of a set has three parts:
A **basis** telling what elements are in the set initially;
the **recursive** or **constructor** part: which tells how to add elements to the set given that a list of certain elements already are in the set;
and an **extremal clause** which is the legal fine print saying that all elements in the set follow from the basis and the recursive part.

Induction Proof Format

A typical recursive definition of a set S is of the following form:
Basis $a_1, a_2, \ldots, a_r$ are all members of S.
Constructors: If $x_1, x_2, \ldots, x_s$ are all members of S then $f_1(x_1, \ldots, x_s) \in S$, $f_2(x_1, \ldots, x_s) \in S$, $\ldots$, $f_k(x_1, \ldots, x_s) \in S$.
Extremal Clause Nothing else is in S but what follows from the basis and constructors.

Suppose that want to prove that a predicate $P(x)$ holds for all $x \in S$. The standard form of the induction proof of this is as follows:

Proof. By induction:
1. State what $P(x)$ is.
2. **Basis** Show that $P(a_1), \ldots, P(a_r)$ are all true.
3. **Inductive Hypothesis** Assume that $P(x_1), \ldots, P(x_s)$ are all true for some arbitrary members $x_1, \ldots, x_s \in S$.
4. **Inductive Step** Prove that it follows that P is true for $f_1(x_1, \ldots, x_s)$, $f_2(x_1, \ldots, x_s)$,..., $f_k(x_1, \ldots, x_s)$, (i.e. $P(f_1(x_1, \ldots, x_s))$, etc. are all true.)
5. Conclusion: For all x in S, $P(x)$ is true. □

That is, one first shows that P is true for the basis elements, and then shows that each constructor preserves the truth of P.

To illustrate a recursive proof, consider the set S defined as follows:
Basis: $4 \in S$, $10 \in S$
Constructors: If $x \in S$ and $y \in S$ then $x - y \in S$. If $x \in S$ and $y \in S$ then $x + y \in S$.
Extremal Clause Nothing else is in S but what follows from the basis and constructors.

Suppose we want to show all of the elements of S are divisible by 2.

Proof. By induction.
We let $P(x)$ be the predicate which is true if x is even.

Base case: 4 and 10 are both even.

Inductive hypothesis: Assume that $P(x)$ and $P(y)$ are true for some elements x and y in S. That is, we assume that x and y are both in S and even.

Inductive Step: We want to show that $P(x+y)$ and $P(x-y)$ are both true.

To do this we can use the definition of evenness to prove that the sum of two even numbers is always even and the difference of two even numbers is also always even. (This part is an ordinary proof of "x even and y even implies $x+y$ is even and $x-y$ is even". I omit the details.)

Therefore we have shown that $P(x+y)$ and $P(x-y)$ are both true.

Conclusion: We have proved by induction that $P(x)$ is true for all $x \in S$. □

Note that what we've actually proved taking the inductive hypothesis and the inductive step together is that $\forall x \in S \forall y \in S((P(x) \wedge P(y)) \rightarrow (P(x+y) \wedge P(x-y)))$

Ordinary Induction Ordinary induction can be viewed as a special case of this. The set of positive integers N can be defined recursively:
Basis $1 \in N$
Constructor If $n \in N$ then $n+1 \in N$.
Extremal Clause The usual

A recursive proof with respect to this definition of N consists of proving $P(1)$ and that for any $n \in N$ if $P(n)$ is true then $P(n+1)$ must be true, i.e. $\forall n \in N.(P(n) \rightarrow P(n+1))$ and this is exactly what we do for ordinary induction.

(Another piece of intuition as to why recursive proofs work is to think of the set

$$S_P = \{x \in S \mid P(x) \text{ is true }\}.$$

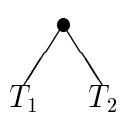
The above proof shows that 4 and 10 are in S_P and that for any x and y in S_P, both $x+y$ and $x-y$ are in S_P. Thus S_P satisfies essentially the same recursive definition as S does. Therefore $S_P = S$.)

Another example: Here we prove something about the functions defined on elements of a recursively defined set S.

We can define the set B of all binary trees by the following two rules:

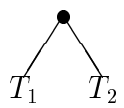
BASIS: "•" is a binary tree

CONSTRUCTOR: If T_1 and T_2 are binary trees then so is:



We can define a couple of functions on binary trees:

Define $\mathrm{height}()$ by: $\mathrm{height}(\bullet) = 1$ and $\mathrm{height}(T) = \max(\mathrm{height}(T_1), \mathrm{height}(T_2)) + 1$ and define $\mathrm{size}()$ by: $\mathrm{size}(\bullet) = 1$ and $\mathrm{size}(T) = \mathrm{size}(T_1) + \mathrm{size}(T_2) + 1$ where T is the tree:



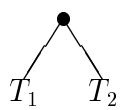
Now we want to prove that for all $T \in B$, $\mathrm{size}(T) \leq 2^{\mathrm{height}(T)} - 1$.

Proof. 1. Let $P(T)$ be $\mathrm{size}(T) \leq 2^{\mathrm{height}(T)} - 1$.

2. Basis: $\mathrm{size}(\bullet) = 1 = 2^1 - 1 = 2^{\mathrm{height}(\bullet)} - 1$

3. Ind. Hypothesis: Assume that $P(T_1)$ and $P(T_2)$ are true for some binary trees T_1 and T_2.

4. Ind. Step: We want to prove that $P(T)$ follows where T is the tree:



It is most convenient to break things up into two cases:

Case (a): $\mathrm{height}(T_1) \leq \mathrm{height}(T_2)$: In this case $\mathrm{height}(T) = \mathrm{height}(T_2) + 1$ and thus

$$\begin{aligned}
\mathrm{size}(T) &= \mathrm{size}(T_1) + \mathrm{size}(T_2) + 1 \\
&\leq (2^{\mathrm{height}(T_1)} - 1) + (2^{\mathrm{height}(T_2)} - 1) + 1 \qquad \text{by Ind. Hyp.} \\
&\leq (2^{\mathrm{height}(T_2)} - 1) + (2^{\mathrm{height}(T_2)} - 1) + 1 \\
&= 2^{\mathrm{height}(T_2)+1} - 1 \\
&= 2^{\mathrm{height}(T)} - 1
\end{aligned}$$

Case(b): $\mathrm{height}(T_2) \leq \mathrm{height}(T_1)$: This is just like the proof of case (a) with T_2 and T_1 reversed.

Therefore $P(T)$ follows and the claim is proved for all $T \in B$. $\square$