BU CAS CS 320 (Summer I, 2009)
Concepts of Programming Languages

Inductive Reasoning

Induction is a powerful tool for reasoning in mathematics. In the study of programming languages, it is frequent to encounter inductive reasoning in the form of inductive definitions and inductive proofs. In this section, we present some basics on inductive reasoning, which the reader may find useful for studing some later parts of the course.

1 Induction on Natural Numbers

We assume that it is clear to the reader what the set of natural numbers is. Suppose that we have a property P on natural numbers and we would like to prove $P(n)$ for every natural number n. For instance, $P(n)$ might be the property stating that $(2n+1)^2 - 1$ is a multiple of 8. A straightforward method to prove $P(n)$ for every natural number n is to simply prove $P(0)$ and then $P(1)$ and then $P(2)$ and so on. Evidently, this is not a feasible method given that the set of natural numbers is infinite.

A strategy to overcome the difficulty is to prove (a) $P(0)$ and (b) $P(n)$ implies $P(n+1)$ for every natural number n. Intuitively, we may imagine that we have infinitely many lined domino cards; (a) means that the first card is to fall and (b) states that if a card is to fall then the card next to it is also to fall; thus we infer that all cards are to fall. In mathematical terminology, (a) and (b) are called *base case* and *inductive step*, respectively.

Example 1 *There is a story claiming that as a young child, the great German mathematician Gauss derived the following formula*

$$0 + 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

for every natural number n.

Proof We use the Σ-notation $\Sigma_{i=0}^{n} i$ for $1 + 2 + \cdots + n$. Let us proceed by induction on n.

- **Base case** Obviously, $\Sigma_{i=0}^{0} = 0 = 0(0+1)/2$.
- **Induction step** We need to prove $\Sigma_{i=0}^{n+1} i = (n+1)(n+1+1)/2$ under the assumption that $\Sigma_{i=0}^{n} i = n(n+1)/2$. In mathematical terminology, such an assumption is called inductive hypothesis. Thus, we have the following.

  $$\Sigma_{i=0}^{n+1} i = (\Sigma_{i=0}^{n} i) + n + 1 = n(n+1)/2 + n + 1 = (n+1)(n+2)/2$$

  By mathematical induction, we have proven $\Sigma_{i=0}^{n} i = n(n+1)/2$.

■

In summary, given a property P on natural numbers, we can prove $P(n)$ for every natural number n by proving the following.

- **(a)** $P(0)$ holds.
- **(b)** $P(n)$ implies $P(n+1)$ for every natural number n.

In practice, there are some variations to this strategy. For instance, one can replace (b) with the following.

- **(b')** $P(0), \ldots, P(n)$ imply $P(n+1)$ for every natural number n.

The interested reader can readily find references that discuss the issue in depth.

Exercise 2 *Prove the following statement using mathematical induction.*

$$(2n+1)^2 - 1 \text{ is a multiple of } 8 \text{ for every natural number } n.$$

2 Inductive Definitions

It is assumed that the reader knows what the set of natural numbers is. Though natural numbers are often represented in base 10, they can certainly be represented in other bases. For example, let $\mathcal{N}$ be the set of unary numerals (i.e., numerals in base 1) for natural numbers. Then we can define $\mathcal{N}$ as follows.

- (1) $Z \in \mathcal{N}$.
- (2) If $n \in \mathcal{N}$ then $S(n) \in \mathcal{N}$.

This is an inductive definition, where (1) and (2) are called *rules*. Usually, a rule consists of many, possibly zero, premisses, and one conclusion. For instance, rule (1) has zero premiss and rule (2) has one premiss. Given a rule consisting of n premisses $P_1, \ldots, P_n$ and conclusion C, we can represent the rule in the following form.

$$\frac{P_1 \quad \cdots \quad P_n}{C}$$

For example, rule (1) and rule (2) can be represented as follows.

$$\frac{}{Z \in \mathcal{N}} \qquad\qquad \frac{n \in \mathcal{N}}{S(n) \in \mathcal{N}}$$

The induction definition allows us to determine what *are* unary numerals. For example, we know that $Z, S(Z), S(S(Z)), \ldots$ are unary numeral. However, the definition does not allow us to tell what *are not* numerals. For example, the following set $\mathcal{N}'$ also satisfies (1) and (2).

$$\mathcal{N}' = \{Z, Z', S(Z), S(Z'), S(S(Z)), S(S(Z')), \ldots\}$$

This prompts us to provide the following clause to complete the definition of $\mathcal{N}$.

- (3) nothing else is in $\mathcal{N}$.

Equivalently, one can say that $\mathcal{N}$ is the least set satisfying (1) and (2). In future presentation, a clause like (3) is always omitted for brevity. The following is a succinct form that is commonly used to represent the above inductive definition.

$$\text{unary numerals} \quad n \quad ::= \quad Z \mid S(n)$$

Let us now introduce another example.

Example 3 *We define the set of binary trees as follows.*

- *(1) The empty tree E is a binary tree.*
- *(2) If t_l and t_r are binary trees, then $B(t_l, t_r)$ is also a binary tree.*

In this case, the rule (1) and rule (2) can be represented as follows.

$$\frac{}{E \text{ is a binary tree}} \qquad \frac{t_l \text{ is a binary tree} \quad t_r \text{ is a binary tree}}{B(t_l, t_r) \text{ is a binary tree}}$$

The inductive definition can also be given in the following form, where it should noticed that the two t's in $B(t, t)$, merely meaning that they are binary trees, have no relation to each other (we may also use subscripts to make this clear by writing, say, $B(t_l, t_r)$).

$$\text{binary trees} \quad t \quad ::= \quad E \mid B(t, t)$$

It is soon to be clear that inductive definitions are ubiquitous in programming language study.

3 Structural Induction

Let us recall the set of unary numeral $\mathcal{N}$. Given a numeral $n \in \mathcal{N}$, we know that n is either Z or $S(n')$ for some numeral n'. Suppose we would like to prove that a property P holds for every $n \in \mathcal{N}$; we can first prove that $P(Z)$ holds; we then prove that $P(n)$ implies $P(S(n))$ for every $n \in \mathcal{N}$. This is a form of structural induction.

Example 4 *Given a binary tree t, the number of empty node E in t equals 1 plus the number of branch node B.*

Proof Given a binary tree t, let $B(t)$ and $E(t)$ stand for the number of branch node and the number of empty node in t, respectively. We proceed to prove the statement by structural induction on t.

- t is E. Then the statement is obviously true.
- t is $B(t_l, t_r)$ for some binary trees t_l and t_r. We have $B(t) = B(t_l)+1+B(t_r)$ and $E(t) = E(t_l)+E(t_r)$. By induction hypothesis, $E(t_l) = 1+B(t_l)$ and $E(t_r) = 1+B(t_r)$. Therefore, we have the following.

$$E(t) = E(t_l) + E(t_r) = 1 + B(t_l) + 1 + B(t_r) = 1 + B(t)$$

■

Let us define function H on binary trees inductively.

$$H(E) = 0 \qquad H(B(t_l, t_r)) = 1 + \max(H(t_l), H(t_r))$$

Evidently, $H(t)$ computes the height of t.

Example 5 *Given a binary tree t, we have $B(t) < 2^{H(t)}$.*

Proof We proceed by structural induction on t.

- $t = E$. Then $B(t) = 0 < 1 = 2^0 = 2^{H(t)}$.
- $t = B(t_l, t_r)$ for some binary trees t_l and t_r. By induction hypothesis, $B(t_l) < 2^{H(t_l)}$ and $B(t_r) < 2^{H(t_r)}$. Hence, we have the following.

$$\begin{aligned} B(t) &= 1 + B(t_l) + B(t_r) \leq 1 + 2^{H(t_l)} - 1 + 2^{H(t_r)} - 1 \\ &\leq 2^{\max(H(t_l),H(t_r))} + 2^{\max(H(t_l),H(t_r))} - 1 < 2^{1+\max(H(t_l),H(t_r))} = 2^{H(t)} \end{aligned}$$

■

Exercise 6 *Braun trees are binary trees defined as follows.*

- *E is a Braun tree.*
- *$B(t_l, t_r)$ is a Braun tree if t_1 and t_r are Braun trees and $B(t_r) \leq B(t_l) \leq B(t_r) + 1$.*

Show that for every nonempty Braun tree t, $2^{H(t)-1} \leq B(t) < 2^{H(t)}$. Also please answer the following questions.

- *How many distinct Braun trees t satisfy $B(t) = 100$?*
- *How many distinct Braun trees t satisfy $H(t) = 100$?*