# 2 Induction

In the preceding section we have stated a number of theorems which are not obvious to prove. The main complication arises from the fact that the proofs concern statements about *infinite sets* of objects. Perhaps one of the most common proof techniques used in Computer Science for statements involving infinite sets is *induction*. Indeed, in this course we shall be using this technique as our main proof tool. Thus we shall spend some time reviewing this technique and lay down some foundations before embarking on using it for more ambitious languages.

**Induction**

Induction is a *formal* technique for reasoning about *(possibly infinite)* sets of objects, whereby the objects are:

- *structured* by some fixed number of rules.
- *finite* (though arbitrarily large).

By *reasoning* we mean:

- determining whether a particular object is in our set of discourse.
- determine whether a property is satisfied by all the members of this set.

**Slide 25**

**Application of Induction**

There are a number of examples of sets that induction can be used to reason about:

- natural numbers.
- data structures such as trees and lists.
- programs of a particular language.
- proof derivation themselves.

In each of the above cases, there are an infinite number elements, each element is *arbitrarily large*, but always *finite*.

**Slide 26**

## 2.1 Mathematical Induction

One can find many forms of induction in the literature and often they are different manifestations of the same concept. We start by reviewing what is perhaps the most familiar form of induction: *mathematical induction i.e.,* induction on natural numbers. We will first review how it works and why it works. We shall later see that this mathematical tool is a specific instance of a more general technique called *rule induction*. We shall then sharpen our focus again on another manifestation of rule-induction called structural induction. This latter form of induction will be the main form of induction we shall use to prove the main results in this module.

**Mathematical Induction**

Given a property $P(x)$ concerning natural numbers, we can *prove* that $P(n)$ holds for every natural number $n \in \textsc{Nat}$ by showing that:

1. $P(0)$ holds.
2. For arbitrary number $k$, if $P(k)$ holds, then it must also be the case that $P(k+1)$ holds too.

**Slide 27**

Note that, in Slide 27, $k$ is universally quantified over natural numbers. Thus, whereas the first clause of mathematical induction deals with a specific instance of a natural number *i.e.,* 0, the second clause talks about a natural number *variable*, sometimes called the *induction parameter*. The two clauses are often referred to by the names of *base* case (or clause) and *inductive* case (or clause).

**How to prove by Mathematical Induction**

Given a property $P(x)$ we showing that:

1. $P(0)$ holds. How we go about this largely depends on $P(x)$.
2. Show that $P(k+1)$ holds after assuming that $P(k)$ holds. $P(k)$ is called the *inductive hypothesis* (*I.H.*).

Assuming that $P(k)$ holds is crucial because the proof that $P(k+1)$ holds often uses the fact that $P(k)$ holds.

**Slide 28**

Mathematical induction suffices to show that *all* the natural numbers satisfy the property $P(x)$ for two important reasons:

1. the entire set of natural numbers can be ordered by the relation $\prec_{\text{plusone}}$ as

   $$0 \prec_{\text{plusone}} 1 \prec_{\text{plusone}} 2 \prec_{\text{plusone}} 3 \prec_{\text{plusone}} \cdots$$

   without leaving any natural number out. That is, starting with the number 0 and equipped with the relation $\prec_{\text{plusone}}$, we can generate the *entire* set of natural numbers.

2. The inductive proof suggest an algorithm for proving the property for the *entire* set of natural numbers. More concretely:

$P(0)$ holds
Since $P(0)$ holds , $P(1)$ holds as well.
Since $P(1)$ holds , $P(2)$ holds as well.
Since $P(2)$ holds , $P(3)$ holds as well.
...

Let us consider a specific example where mathematical induction turns out to be a very elegant approach to solving mathematical problems. Slide 29 defines the property we need to verify about *all* the natural numbers. The main problem with a naive approach towards solving this problem is the fact that there is an infinite amount of natural numbers and if we start testing the property for every element in the set NAT, we would never terminate.

**Mathematical Induction Example(1)**

To prove that for any natural number $n \in \text{NAT}$ the property

$$P(x) \quad \stackrel{\text{def}}{=} \quad \sum_{i=0}^{x} i = \frac{x^2 + x}{2}$$

holds.

*Recall that a property is a predicate: for every $n$ substituted for $x$, $P(n)$ has to be either true or false.*

**Slide 29**

The proof for property $P(x)$ of Slide 29 is given on Slide 30 (the base case) and Slide 31.

**Mathematical Induction Example(2)**

**Base Case**
$P(0)$ holds (*i.e.*, is true) because:

$$\sum_{i=0}^{0} i \quad = \quad 0 \quad = \quad \frac{0^2 + 0}{2}$$

**Slide 30**

As is often the case, proving the inductive case for $P(x)$ of Slide 29 turns out to be the most involving, as shown in Slide 31. Here we highlight the most important steps for our exposition, namely the first and second steps of the derivation. More precisely, by expanding the definition of $\sum$, the first step allows us to *recast the problem for $k+1$ in terms of a problem for $k$*. The second step then uses the inductive hypothesis,

**Mathematical Induction Example (3)**

**Inductive Case**
Assuming $P(k)$ holds, *i.e.*, $\sum_{i=0}^{k} i = \frac{k^2+k}{2}$, then we can also show that $P(k+1)$ holds because:

$$\begin{aligned}
\sum_{i=0}^{k+1} i &= \Big(\sum_{i=0}^{k} i\Big) + k{+}1 && \textit{by definition of } \textstyle\sum \\
&= \Big(\frac{k^2+k}{2}\Big) + k{+}1 && \textit{by I.H.} \\
&= \Big(\frac{k^2+k}{2}\Big) + \frac{2k+2}{2} = \frac{k^2+k+2k+2}{2} \\
&= \frac{k^2+k+2k+1+1}{2} = \frac{(k^2+2k+1)+(k+1)}{2} \\
&= \frac{(k+1)^2+(k+1)}{2}
\end{aligned}$$

**Slide 31**

*i.e.*, the assumption that the property hold s for $k$, which then allows us to substitute $\sum_{i=0}^{k} i$ for $\frac{k^2+k}{2}$, emanating from the equality relation in $P(x)$. All the remaining steps use additional properties for the set of natural numbers, which we shall not concern ourselves with in this module.

## 2.2 Rule Induction

We now recast mathematical induction as a more general form of induction called *rule induction*. This recasting should also help us demistify some "magical" steps we used to prove the statement in Slide 29. (*cf.* the first two steps in Slide 31)

**Rule-based definition for Nat**

The *least* set $S$ satisfying:

- $0 \in S$
- $k \in S$ implies $k+1 \in S$

**Slide 32**

An alternative way how to specify a set, instead of specifically naming every element, is to state the conditions *i.e.*, rules, that it satisfies. For instance, assuming the mathematical operation +, it turns out that NAT is the *least* set satisfying the conditions in Slide 32. Note how, using a single element 0 and the operation +, we are *generating* (constructing) the set NAT. In a sense, this is the the flip-side of the interpretation sometimes given to mathematical induction, whereby one first had a set NAT and then used the relation $\prec_{\text{plusone}}$ to *totally-order* the elements of the set.

A crucial aspect of the definition in Slide 32 is the use of the word *"least"*. In fact there are other sets that satisfy these conditions *e.g.,* the set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$. However, these conditions have a special significance to the set NAT because they *characterise* this set exactly *i.e., every* element in NAT satisfies (either of) these conditions. Stated otherwise, for any other set $S'$ that satisfies these conditions, we know that $\text{NAT} \subseteq S'$ (hence the word *"least"*). In fact, $\text{NAT} \subseteq \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. Since these rules characterise the set NAT, we can use them for our induction proofs to prove things about this set.

**Rule-based definition for Even**

The *least* set $S$ satisfying:

- $0 \in S$
- $k \in S$ implies $k + 2 \in S$

**Slide 33**

Slide 33 illustrates a rule-based definition for the set of even numbers. Sometimes we prefer to write these rules as in Slide 34 *i.e.,* in the style of Slide 12.

**Writing Conditions as Rules**

**Rules defining Nat**

$$\frac{}{0 \in \text{NAT}} \qquad \frac{k \in \text{NAT}}{k + 1 \in \text{NAT}}$$

**Rules defining Even**

$$\frac{}{0 \in \text{EVEN}} \qquad \frac{k \in \text{EVEN}}{k + 2 \in \text{EVEN}}$$

**Slide 34**

Note again in Slide 33, and then again in Slide 34, the crucial role played by the constraint "least". For instance, NAT satisfies the conditions in Slide 33, but it is not the least set satisfying these conditions. In fact, from our prior knowledge of these two sets, we know that $\text{EVEN} \subseteq \text{NAT}$. As we shall see, we will now be also able to *prove* this property. But before, let us consider further examples of rule-based set definitions:

The rules in Slide 35 characterise the *divides* relation (*i.e.,* a *set* whereby the elements are tuples) over natural numbers whereby $\langle n, m\rangle \in \text{DIV}$ implies that there exists some natural number $k \in \text{NAT}$ such that $m \div n = k$, and viceversa, whenever there exists some natural number $k \in \text{NAT}$ such that $m \div n = k$, then $\langle n, m\rangle \in \text{DIV}$. Alternatively, this can be stated as

$$\text{DIV} = \big\{\langle n, m\rangle \mid \text{ there exists some } k \text{ whereby } n \times k = m\big\}$$

It is important to keep in mind that certain inductive definitions make little or no sense. For instance, it turns out that the least set characterised by the first inductive definition in Slide 36 is the empty set, $\emptyset$.

**Rule-based definition of Relations**

**Rules defining "Divides"**
$n$ Div $m$ is shorthand for $\langle n, m\rangle \in$ Div. Div is the *least* relation of the form Nat $\times$ Nat satisfying the following rules:

$$\frac{}{n \text{ Div } 0} \qquad \frac{n \text{ Div } k}{n \text{ Div } k+n}$$

**Slide 35**

**Inductive definitions with little or no sense**

What are the sets defined by the following rules?

**Inductive Definition 1**

$$\frac{n \in X}{n+2 \in X}$$

**Inductive Definition 2**

$$\frac{}{0 \in X} \qquad \frac{k \in X}{k+2 \in X} \qquad \frac{k+2 \in X}{k \notin X}$$

**Slide 36**

Let us examine why this is so. The empty set vacuously satisfies the rule in Slide 36 because no element is contained in $\emptyset$ and as a result, every element (*i.e.*, none) satisfy the rule

$$\frac{n \in X}{n+2 \in X}$$

Moreover, this set is (trivially) the least such set satisfying this rule because $\emptyset \subseteq S$ for any $S$.

The second inductive definition in Slide 36 is even more problematic! It turns out that this definition does not define *any* set because any set would contradict these rules. In fact, by the first rule

$$\frac{}{0 \in X}$$

a set $S$ satisfying these rules must contain the element 0, *i.e.*,

$$0 \in S \tag{5}$$

By the second rule,

$$\frac{k \in X}{k+2 \in X}$$

and (5) it must also be the case that

$$2 \in S \tag{6}$$

However, by the third rule

$$\frac{k+2 \in X}{k \notin X}$$

and (6), 0 cannot be in $S$, *i.e.,*

$$0 \notin S$$

which contradicts the statement (5). Since set membership is crisp *i.e.,* an entity is either inside a set or outside it (but never both) the second batch of rules on Slide 36 can never define a set.

**Inductive Definitions as Proof Systems**

| **Proving that** 6 **is even** | **Proving that** 17 **divides** 51 |
|---|---|
| $\dfrac{}{0 \in \textsc{Even}}$ | $\dfrac{}{17 \textsc{ Div } 0}$ |
| $\dfrac{}{2 \in \textsc{Even}}$ | $\dfrac{}{17 \textsc{ Div } 17}$ |
| $\dfrac{}{4 \in \textsc{Even}}$ | $\dfrac{}{17 \textsc{ Div } 34}$ |
| $\dfrac{}{6 \in \textsc{Even}}$ | $\dfrac{}{17 \textsc{ Div } 51}$ |

**Slide 37**

Despite these pitfalls, inductive definitions carry a number of advantages. One pleasing property is that they act as *proof-systems*. For example, assuming the inductive definition of Slide 34 for Even, we can *prove* (*i.e.,* provide a step-by-step argument) why 6 is an even number, as shown in Slide 37. More specifically, to prove that $6 \in \textsc{Even}$, we use the mathematical property of $+$ to express 6 as $4+2$ and then, use the second rule defining Even in Slide 34 *i.e.,*

$$\frac{k \in \textsc{Even}}{k+2 \in \textsc{Even}}$$

to derive the proof step that $4+2 \in \textsc{Even}$ if we can show that $4 \in \textsc{Even}$. Similar reasoning is required for showing $4 \in \textsc{Even}$ and $2 \in \textsc{Even}$ which leads us to the case $0 \in \textsc{Even}$, which is immediately true by the Even axiom in Slide 34, *i.e.,*

$$\frac{}{0 \in \textsc{Even}}$$

Perhaps less obviously, assuming the inductive definition of Slide 35 we can also prove that 17 divides 51 (see Slide 37 (right)). Note that, for this proof, instead of finding the witness number $n$ such that $17 \times n = 51$, we now provide the rule-based proof shown in Slide 37. In fact, as in both of the above two cases, and also for *any* inductively defined set, every member of such a set (be it a relation *etc.*) is backed up by a *proof* justifying its membership.

Another advantage of inductively defined sets is that they provide an easy and structured way how to defined functions over them. To define such functions, we once again inductively define the output of the function for *every rule case* of the inductive definition. Moreover, the inductive cases (*i.e.,* rules with premises) often employ the output of the function being define applied to their premises to define the ouput of the function for the rule conclusion.

**Inductive Definitions and Functions**

$$\sum(x \in \textsc{Nat}) \stackrel{\text{def}}{=} \begin{cases} 0 & x = 0 \\ (k{+}1) + \sum(k) & x = k+1 \end{cases}$$

$$\textbf{quotient}(x \textsc{ Div } y) \stackrel{\text{def}}{=} \begin{cases} 0 & y = 0 \\ 1 + \textbf{quotient}(x \textsc{ Div } y') & y = x + y' \end{cases}$$

**Slide 38**

Slide 38 illustrates this with two examples: the first, summation (*i.e.,* $\sum$) is defined over Nat whereas the second (*i.e.,* **quotient**) is defined over the relation Div and returns the quotient m for a pair $\langle n, k\rangle \in$ Div (*i.e.,* the number $m$ such that $n \times m = k$). Notice how, for example, in the case of $\sum(k+1)$, the definition assumes that $\sum(k)$ is defined and in turn uses it to give the definition for the summation of $k+1$ *i.e.,* $\sum(k+1) \stackrel{\text{def}}{=} (k{+}1) + \sum(k)$.

The final advantage we shall consider in this exposition is that inductive definitions provide a *proof technique* for proving properties about the elements of these inductively defined sets. This proof technique is called *proof by induction* (or inductive proof). Proofs by induction bear striking similarities to inductively defined functions. In fact, to give an inductive proof we need to provide:

1. a proof for the axiom rules *i.e.,* the *base* case.
2. a proof for the inductive rules *i.e.,* the *inductive* case. Similar to inductive function definitions, for the inductive case, we are allowed to assume that the proof holds for the premises of the rule.

If, for instance, we want to show that $x \in$ Even implies $x \in$ Nat *i.e.,* Even $\subseteq$ Nat we can show this by giving the inductive proof on Slides 39, 40 and 41. What is really happening there is that we are considering the set Nat and we are ensuring that it observes the two conditions required of the set Even (those of Slide 33), namely:

- $0 \in$ Nat
- $k \in$ Nat implies $k + 2 \in$ Nat

The reasoning then follows that, since Even is the least set satisfying these conditions, it is a subset of any other set that also satifies them. In this case, if Nat satifies these conditions, then Even must also be a subset of Nat.

Slide 40 gives a proof that the first condition of Slide 33, which corresponds to the *base case* of an inductive proof. Slide 41 then proves the second condition of Slide 33, which corresponds to the *inductive case* in a proof by induction.

In both Slides 40 and 41, we are just using the conditions that we know Nat satisfies in order to prove that it also satisfies the conditions characterising Even. In some textbooks, they would then refer to Even as being a property of Nat, since Even $\subseteq$ Nat.

**Inductive Definitions and Proofs by Induction**

$x \in \textsc{Even}$ implies $x \in \textsc{Nat}$ or $\textsc{Even} \subseteq \textsc{Nat}$

Recall that Even is the *least* set $S$ satisfying:

- $0 \in S$
- $k \in S$ implies $k + 2 \in S$

Thus for any other set $S'$ satisfying the above condition:

$$\textsc{Even} \subseteq S'$$

**Slide 39**

**Inductive Definitions and Proofs by Induction**

We therefore prove that Nat satisfies the conditions characterising Even:

**Condition 1** We have to show that $0 \in \textsc{Nat}$. This is immediate by the axiom rule for Nat:

$$\frac{}{0 \in \textsc{Nat}}$$

**Slide 40**

At this point we can also recall the earlier inductive proof of Slides 29, 30 and 31 and we can view the proof as an instance of a proof by Rule Induction where Nat, the set over which the property has to hold, can be defined inductively by the rules in Slide 34. Moreover, in the inductive case of Slide 42, the first line of the proof is a mere expansion of an inductively defined function (*i.e.,* the summation function $\sum$) over Nat (as shown earlier in Slide 38) and the second step in this proof is just using the inductive hypothesis in the substitution. Finally, we should also be able to appreciate why this inductive proof works. What we want to show is that

$$\textsc{Nat} = \left\{ x \mid \sum_{i=0}^{x} i = \frac{x^2 + x}{2} \right\}$$

Since we know that the property ranges over Nat, it immediately follows that $\left\{ x \mid \sum_{i=0}^{x} i = \frac{x^2+x}{2} \right\} \subseteq \textsc{Nat}$. Thus, the required result follows if we are able to show that $\textsc{Nat} \subseteq \left\{ x \mid \sum_{i=0}^{x} i = \frac{x^2+x}{2} \right\}$. On Slides 30 and 31, what we actually show is that the set $\left\{ x \mid \sum_{i=0}^{x} i = \frac{x^2+x}{2} \right\}$ observes the conditions characteristing Nat, *i.e.,* those on Slide 32. Then, by virtue of the fact that Nat is the least set satisfying these conditions, we infer that $\textsc{Nat} \subseteq \left\{ x \mid \sum_{i=0}^{x} i = \frac{x^2+x}{2} \right\}$ as required.

**Inductive Definitions and Proofs by Induction (Cont.)**

**Condition 2** We have to show that whenever $k \in \text{NAT}$ then $k+2 \in \text{NAT}$. Since $k+2 = (k+1)+1$ we can show this by the derivation:

$$\dfrac{\dfrac{k \in \text{NAT}}{k+1 \in \text{NAT}}}{(k+1)+1 \in \text{NAT}}$$

**Slide 41**

## 2.3 Structural Induction

Our definition of NAT, (and that of EVEN and DIV for that matter) relied on properties from the addition operation, +. We can refine our definition of NAT and treat our elements in NAT as purely *syntactic* elements (*i.e.,* when we have $k+1$ for some element k, we do not evaluate it into a number using + but leave it as the expression $k+1$).

Such a definition is shown in Slide 43, using textual notation zero and succ(-) to accentuate the difference. This instance of a (rule) inductive definition is called *structural* inductive definition because the elements of discourse are entirely (or mostly) syntactic (*i.e.,* syntax without any auxiliary meaning). The elements of structurally defined sets are thus *built* from *basic elements*, such as zero, and from *constructors* which take a number of (constructed) elements from the set and return some other element. succ(-) would be an example of a constructor, taking a number and returning another number, more precisely its successor. Thus, as shown on Slide 43, the number 3, which we could previously expand as the expression $1 + 1 + 1$, is now denoted by the syntactic term succ(succ(succ(zero))).

**Mathematical Induction as an instance of Rule Induction**

To prove that for any natural number $n \in \text{NAT}$

$$P(x) \quad \stackrel{\text{def}}{=} \quad \sum_{i=0}^{x} i = \frac{x^2 + x}{2}$$

Since NAT is *inductively defined*, we can use an *inductive proof*!!

**Base Case:** $P(0)$ holds because:

$$\sum_{i=0}^{0} i \quad = \quad 0 \quad = \quad \frac{0^2 + 0}{2}$$

**Inductive Case:** $P(k+1)$ holds because:

$$\begin{aligned} \sum_{i=0}^{k+1} i &= \Big(\sum_{i=0}^{k} i\Big) \; + \; k{+}1 && \textit{unfolding the ind. def. of } \textstyle\sum \\ &= \Big(\frac{k^2 + k}{2}\Big) \; + \; k{+}1 && \textit{by I.H.} \\ &= \ldots \end{aligned}$$

**Slide 42**

**Structural Inductive Definition for Nat**

**Rules defining Nat**

$$\frac{}{\mathsf{zero} \in \text{NAT}} \qquad \frac{n \in \text{NAT}}{\mathsf{succ}(n) \in \text{NAT}}$$

We thus write:

- 0 as $\mathsf{zero}$
- 1 as $\mathsf{succ}(\mathsf{zero})$
- 2 as $\mathsf{succ}(\mathsf{succ}(\mathsf{zero}))$
- ...

**Slide 43**

**Structural Inductive Definition for Binary Trees**

$$\frac{}{\mathsf{leaf} \in \text{BTREE}} \qquad \frac{t_1 \in \text{BTREE} \qquad t_2 \in \text{BTREE}}{\mathsf{node}(t_1, t_2) \in \text{BTREE}}$$

The following are syntactic representations of binary trees:

- leaf
- node(leaf,leaf)
- node(node(leaf,leaf),leaf)
- ...

Can you give a graphical representation for the above?

**Slide 44**

In this course we will encounter numerous cases of structurally defined sets. In fact, Slide 44 gives another example of a syntactically defined set, namely the set of Binary trees. Notice that this time the binary tree constructor, node(-,-), takes *two* binary trees to return another binary tree. Using this inductive definition, we can construct syntactic trees such as node(leaf,(node(leaf,leaf))). We stress once again that, although the number of trees that we can construct is infinite, every tree is finite along all of its branches.

**Structural Inductive Definitions and BNF**

NAT and BTREE can also be defined using BNF notation:

$$n \in \text{NAT} ::= \mathsf{zero} \mid \mathsf{succ}(n)$$

$$t \in \text{BTREE} ::= \mathsf{leaf} \mid \mathsf{node}(t, t)$$

Notice the tight correspondence to rule-based inductive definitions.

**Slide 45**

**Structurally Defined Functions**

$$\mathbf{add}(n, m) \stackrel{\mathbf{def}}{=} \begin{cases} n & m = \mathsf{zero} \\ \mathbf{add}(\mathsf{succ}(n), m') & m = \mathsf{succ}(m') \end{cases}$$

$$\mathbf{leaves}(t) \stackrel{\mathbf{def}}{=} \begin{cases} 1 & t = \mathsf{leaf} \\ \mathbf{leaves}(t_1) + \mathbf{leaves}(t_2) & t = \mathsf{node}(t_1, t_2) \end{cases}$$

$$\mathbf{branches}(t) \stackrel{\mathbf{def}}{=} \begin{cases} 0 & t = \mathsf{leaf} \\ 1 + \mathbf{branches}(t_1) + \mathbf{branches}(t_2) & t = \mathsf{node}(t_1, t_2) \end{cases}$$

**Slide 46**

When defining sets whose elements are of a structural nature, the more succinct BNF notation is usually used for convenience. Slide 45 shows how NAT and BTREE can be defined in BNF. It is nevertheless important to keep in mind that such notation has a more formal underpinning in terms of the syntactic rules inductively defining the set.

Since structural induction is a case of rule induction, we inherit all the pleasant properties we discussed earlier. In particular, we can define functions inductively over these sets and prove properties about these set by *structural induction* (*i.e.,* induction on the *structure* of the elements in a set). Slide 46 defines an addition function over elements of NAT and two functions over the set BTREE calculating the number of leaves and branches in a tree. Thus for instance we have:

- **add**(succ(succ(zero)), succ(succ(succ(zero)))), where succ(succ(zero)) and succ(succ(succ(zero))) denote 2 and 3 respectively, returns succ(succ(succ(succ(succ(zero))))), denoting 5.
- **leaves**(node(node(leaf, leaf), leaf)) returns 3.
- **branches**(node(node(leaf, leaf), leaf)) returns 2.

**Proofs by Structural Induction**

$$P(x) \quad \stackrel{\text{def}}{=} \quad \textbf{leaves}(x) = \textbf{branches}(x) + 1$$

Is $P(t)$ true for all $t \in$ BTree?

Proof by Induction on the *structure* of $t$

**Base Case:**
$t =$ leaf

$$\textbf{leaves}(\text{leaf}) = \textbf{branches}(\text{leaf}) + 1$$
$$1 = 0 + 1$$

**Slide 47**

We can also prove properties about structurally defined sets (of potentially inifnite size) once again using induction. This time, we however refer to it as *structural induction*, as the inductive principle is based on the inductive structure of the elements of discourse. Slide 47 and Slide 48 outline a proof by structural induction for a property enjoyed by every element in BTree. As before, it involves unfolding the definition of inductively defined functions on elements of BTree, which allows us to state the property to be proved in the inductive case, in terms of the property assumed in our inductive hypothesis.

Note that since our elements are syntactic, we could have easily defined elements of BTree in infix fashion, as shown in Slide 49. This change does not affect the semantic meaning of the set BTree in any major way (Remember that our elements are syntactic objects with no auxiliary meaning whatsoever). In fact, according to this slight change, if we cosmetically change the functions defined over BTree (*i.e.,* the case for node) as shown in Slide 49, then the property $P(x)$ (restated in Slide 49), proved earlier in Slides 47 and 48 should still hold.

**Proofs by Structural Induction (Cont.)**

**Inductive Case:**
$t = \mathsf{node}(t_1, t_2)$

$$\mathbf{leaves}(\mathsf{node}(t_1, t_2)) = \mathbf{branches}(\mathsf{node}(t_1, t_2)) + 1$$
$$\mathbf{leaves}(t_1) + \mathbf{leaves}(t_2) = 1 + \mathbf{branches}(t_1) + \mathbf{branches}(t_2) + 1$$

By Inductive Hypothesis we know that $P(t_1)$ and $P(t_2)$ hold, *i.e.,*:

- $\mathbf{leaves}(t_1) = \mathbf{branches}(t_1) + 1$
- $\mathbf{leaves}(t_2) = \mathbf{branches}(t_2) + 1$

Substituting these equalities in our proof gives us:

$$(\mathbf{branches}(t_1) + 1) + (\mathbf{branches}(t_2) + 1)$$
$$= 1 + \mathbf{branches}(t_1) + \mathbf{branches}(t_2) + 1$$

**Slide 48**

**Infix notation for trees**

We could have defined binary trees using *infix syntax*:

$$t \in \text{BTree} ::= \mathsf{leaf} \mid t\,\mathsf{node}\,t$$

The changes to functions defined over BTree are cosmetic

$$\mathbf{leaves}(t) \stackrel{\mathbf{def}}{=} \begin{cases} 1 & t = \mathsf{leaf} \\ \mathbf{leaves}(t_1) + \mathbf{leaves}(t_2) & t = t_1\,\mathsf{node}\,t_2 \end{cases}$$

$\ldots$

And the property $P(x) \stackrel{\mathbf{def}}{=} \mathbf{leaves}(x) = \mathbf{branches}(x) + 1$ still holds for BTree.

**Slide 49**

## 2.4 Proving properties about Languages

**$L_1$ expressions as trees**

$L_1$ expressions are, in essence, (parsed) trees:

$$e \in \text{Exp} ::= b \mid n \mid e \texttt{+} e \mid e \texttt{-} e \mid e \texttt{<=} e \mid e \texttt{\&\&} e \mid \texttt{~} e$$

We could define *inductively* functions over Exp:

$$\text{depth}(x) \stackrel{\text{def}}{=} \begin{cases} 0 & x = b \text{ or } x = n \\ 1 + \text{depth}(e) & x = \texttt{~}\, e \\ 1 + \max(\text{depth}(e_1), \text{depth}(e_2)) & x = e_1 \texttt{+} e_2 \\ \ldots & \end{cases}$$

$$\text{width}(x) \stackrel{\text{def}}{=} \begin{cases} 1 & x = b \text{ or } x = n \\ 1 + \text{width}(e) & x = \texttt{~}\, e \\ 1 + \text{width}(e_1) + \text{width}(e_2) & x = e_1 \texttt{+} e_2 \\ \ldots & \end{cases}$$

We could also define and prove (or disprove) properties for Exp

$$P(x) \quad \stackrel{\text{def}}{=} \quad \text{depth}(x) < \text{width}(x)$$

**Slide 50**

Let us recall the BNF definition for our expression language $L_1$ (*cf.* Slide 50). It should now be evident that our expressions are, in essence, trees whereby values are the leaf nodes and operators are the branch nodes (they are not exactly binary trees because `~` $e$ is a tree with one branch). Similar to the previous case for binary trees we can define functions based on the *structure* of expressions (*cf.* Slide 50 for depth($e$) and width($e$), returning the maximum depth and the width of an expression $e$ respectively), and then prove properties for Exp using *structural induction* (*cf.* Sec. 2.5).

**Proving Determinism of $\Downarrow$for $\mathbf{L}_1$ (1)**

**Theorem** (Determinacy)**.** *If* $e \Downarrow v_1$ *and* $e \Downarrow v_2$ *then* $v_1 = v_2$

**Proof** By *structural induction* on $e$.

**Case $e$ =$b$':** By case analysis, we know that only one rule could have been applied to derive both $b' \Downarrow v_1$ and $b' \Downarrow v_2$, namely:

$$\frac{}{b' \Downarrow b'}\ \textsc{eBool}$$

From this rule (schema), we deduce that $v_1 = b' = v_2$.

**Case $e$ = $n$:** Similar.

*These two cases constitute the base cases of our proof by induction.*

**Slide 51**

**Proving Determinism of $\Downarrow$for $\mathbf{L}_1$ (2)**

**Theorem** (Determinacy)**.** *If* $e \Downarrow v_1$ *and* $e \Downarrow v_2$ *then* $v_1 = v_2$

**Case $e$ = ~ $e'$:** By case analysis, we know that only eNot could have been applied to derive $\sim e' \Downarrow v_1$ and $\sim e' \Downarrow v_2$. Thus:

$$\frac{e' \Downarrow b'}{\sim e' \Downarrow v_1}\ \textsc{eNot} \qquad \frac{e' \Downarrow b''}{\sim e' \Downarrow v_2}\ \textsc{eNot}$$

For some $b'$ and $b''$ where

$$v_1 = \neg b' \qquad v_2 = \neg b'' \tag{7}$$

Since $e'$ is a subexpression of $\sim e'$, we could use inductive hypothesis (I.H.) and deduce from $e_1 \Downarrow b'$ and $e_1 \Downarrow b''$ that $b' = b''$. Thus from the rule side-conditions (7) we derive $v_1 = v_2$.

**Slide 52**

**Proving Determinism of $\Downarrow$for $L_1$ (3)**

**Theorem** (Determinacy). *If* $e \Downarrow v_1$ *and* $e \Downarrow v_2$ *then* $v_1 = v_2$

**Case $e = e' + e''$:** By case analysis, only eAdd could have been applied to derive $e' + e'' \Downarrow v_1$ and $e' + e'' \Downarrow v_2$. Thus:

$$\frac{e' \Downarrow n' \quad e'' \Downarrow n''}{e' + e'' \Downarrow v_1}\ \text{eAdd} \qquad \frac{e' \Downarrow n''' \quad e'' \Downarrow n''''}{e' + e'' \Downarrow v_2}\ \text{eAdd}$$

For some $n'$, $n''$, $n''$ and $n''$, where

$$v_1 = n' + n'' \qquad v_2 = n''' + n'''' \tag{8}$$

Since $e'$ is a subexpression of $e' + e''$, we could use I.H. and deduce from $e' \Downarrow n'$ and $e' \Downarrow n'''$ that $n' = n'''$. Similarly, by I.H. , from $e'' \Downarrow n''$ and $e'' \Downarrow n''''$ we deduce $n'' = n''''$. Thus from the rule side-conditions (8) we derive $v_1 = v_2$.

**Slide 53**

Using a similar proof technique, we can also prove the properties set out earlier in Sec. 1. For instance, let us consider the first Theorem we stated in Sec. 1 for $L_1$, *i.e.,* Determinacy for $\Downarrow$. This is restated in Slide 51. One way how to prove this theorem is to use structural induction on the structure of the expression $e$ over which the relation is defined. We stress again that this can be done because we know that every $e$ over which $e \Downarrow v$ is defined is:

1. finite (in structure).
2. has a regular structure, as defined by the BNF of Slide 50.

Our proof by structural induction has 2 bases cases, one for every kind of *value*; values constitute the base cases because, from the point of expressions, values cannot be decomposed any further. It is instructive to review the proof outlined in Slide 51, for the base case proof when our expression is a *boolean* kind of value. We assume that our expression is some arbitrary boolean value (*i.e.,* either `true` or `false`): since we do not actually care what this boolean value is exactly, we assign a metavariable to it, *i.e.,* $b'$.

Now that we have set the structure of $e$ we proceed to analyse which rules (rule schema) from Slides 10 and 11 could have been (immediately) used to derive $e \Downarrow v$ (or, more specifically, now that we have set the structure of $e$ to some $b'$, to derive $b' \Downarrow v$). Our analysis should lead us to conclude that only rule (schema) eBool could have been applied. We conclude this because other rules require the expression to be of a structure that is different from $b$' (*e.g.,* eAnd could *not* have been applied because it requires $e$ to be of the form $e$' `&&` $e$", which is clearly not of the form $b$', the form we assumed $e$ to be). Thus we have to use the rule schema:

$$\frac{}{b \Downarrow b}\ \text{eBool}$$

For both derivations of the theorem, $b' \Downarrow v_1$ and $b' \Downarrow v_2$ we have to *instantiate* the schema for the boolean value $b$' (*i.e.,* we substitute the rule metavariable $b$ for our boolean value $b$') and for both derivations we

obtain:

$$\frac{}{b' \Downarrow b'}\ \textsc{eBool}$$

This means that, for the first derivation, it turns out that $v_1$ is $b'$ and similarly, for the second derivation, $v_2$ is also $b'$. This means that the two values $v_1$ and $v_2$ are equivalent, as required by the theorem, and thus we have shown that the theorem holds for this particular case.

We now have to repeat the same procedure for each different case of structure for $e$ and produce a proof for every case. The proof for the other base case, *i.e.,* when $e$ is of the form $n'$ (for some numeral $n'$) is similar to the case proof we just outlined, and should be attempted by the reader.

We now shift our attention to the inductive cases of the proof. The first inductive case we consider regards the "not" expressions, *i.e.,* when $e$ is of the form $\sim e'$ for some subexpression $e'$; the proof for this case is outlined on Slide 52. Once again we perform a case analysis to determine which rules could have been applied from Slides 10 and 11 to derive $\sim e' \Downarrow v_1$ and $\sim e' \Downarrow v_2$. Due to the structure of $\sim e'$ it turns out that only one rule (schema) could have been applied and this is the rule schema:

$$\frac{e_1 \Downarrow b}{\sim e_1 \Downarrow v}\ \textsc{eNot} \qquad \text{where } v = \neg b$$

Thus after instantiating the above schema with $\sim e'$ we obtain:

$$\frac{e' \Downarrow b'}{\sim e' \Downarrow v_1}\ \textsc{eNot} \qquad \frac{e' \Downarrow b''}{\sim e' \Downarrow v_2}\ \textsc{eNot}$$

for some $b'$ and $b''$ where

$$v_1 = \neg b' \qquad v_2 = \neg b''$$

The left-hand derivation states that $\sim e' \Downarrow v_1$ is derived *after* the subexpression $e'$ has been evaluated to some boolean value $b'$, through the (sub) derivation $e' \Downarrow b'$, where $v_1 = \neg b'$ (side-condition). Similarly, the right-hand derivation relies on the assumption that the subexpression $e'$ has been evaluated to some other boolean value $b''$, through the (sub) derivation $e' \Downarrow b''$, where $v_2 = \neg b''$. Notice that the two assumption relating to the sub derivations $e' \Downarrow b'$ and $e' \Downarrow b''$ are distinct (*i.e.,* there are potentially two distinct derivations) and, a priori, we have no knowledge whether $b'$ is equal to $b''$ or not. However, since e' is a sub-expression of $\sim e'$, the expression we are trying to prove the property for, we can apply the inductive hypothesis. This means that we can assume that the property holds for the two sub-derivations, *i.e.,*

$$e' \Downarrow b' \text{ and } e' \Downarrow b'' \text{ implies } b' = b''$$

Subsequently, from the fact that $b' = b''$, the rule side condition facts $v_1 = \neg b'$ and $v_2 = \neg b''$, and the assumed fact that $\neg$ is a deterministic operation, we conclude that $v_1 = v_2$, thereby showing that the theorem holds for this particular case as well.

The final case we consider is another inductive case where $e$ is of the from e' + e", as outlined on Slide 53. From the structure of e' + e" we deduce that only one rule (schema) could have been applied,

$$\frac{e_1 \Downarrow n_1 \quad e_2 \Downarrow n_2}{e_1 + e_2 \Downarrow n_3}\ \textsc{eAdd} \qquad \text{where } n_3 = n_1 + n_2$$

and after instantiating it we obtain the two derivation steps outlined in Slide 53 each assuming two sub-derivations, one for $e'$ and another for $e''$. This time we apply the inductive hypothesis *twice* to obtain:

$$e' \Downarrow n' \text{ and } e' \Downarrow n''' \text{ implies } n' = n'''$$
$$e'' \Downarrow n'' \text{ and } e'' \Downarrow n'''' \text{ implies } n'' = n''''$$

**Proving Determinism of $\longrightarrow$ and $\longrightarrow^*$ for $\mathbf{L}_1$**

**Theorem** (Determinacy). *If* $e \longrightarrow e'$ *and* $e \longrightarrow e''$ *then* $e' = e''$

**Proof** By *structural induction* on $e$.

**Theorem** (Determinacy). *If* $e \longrightarrow^* v$ *and* $e \longrightarrow^* v'$ *then* $v = v'$

**Proof** How do we go about it?

**Slide 54**

and from the side conditions of the instantiated rules it is easy to conclude that $v_1 = v_2$.

Using induction we can also prove Determinism for the reduction relations we discussed in Sec. 1; these are restated on Slide 54. Whereas structural induction on $e$ suffices to prove the first Theorem on Slide 54, we need a different from of induction to prove the second Theorem.

Let us recall the definition of $\longrightarrow^*$ from Slide 21; this definition essentially states that $\longrightarrow^*$ is the transitive closure of $\longrightarrow$, *i.e.*, $e \longrightarrow^* e'$ holds if there exists a reduction sequence of length $n$ where $n \geq 0$ such that:

$$e \longrightarrow e_1 \longrightarrow e_2 \longrightarrow \ldots \longrightarrow e_{n-1} \longrightarrow e'$$

Clearly, when the reduction length $n = 0$, then $e = e'$. We often denote this sequence by the notation $e \longrightarrow^n e'$.

Note also that if the expression is a value then it cannot reduce further. This is stated as Lemma 6 and proved on Slide 55. This lemma uses a definition for terminal expressions, *i.e.*, expressions that cannot reduce further, $e \not\longrightarrow$, defined also on Slide 55. In certain language expositions, this property is sometimes expressed and defined in terms of *normal forms*. This need not concern us too much at this stage but we note that we shall refer back to this definition later on in Sec. 6.9.

**Proving Determinism of $\longrightarrow^n$ for $\mathbf{L}_1$**

**Definition** (Terminal Expressions)**.** $e \not\longrightarrow \stackrel{def}{=} \nexists e'.\, e \longrightarrow e'$

All values are terminal *i.e.*,, there is no $e$ such that $v \longrightarrow e$.

**Lemma 6** (Termination)**.** $v \not\longrightarrow$

**Proof** By rule inspection

**Slide 55**

**Proving Determinism of $\longrightarrow^n$ for $\mathbf{L}_1$ (1)**

**Theorem** (Determinacy)**.** *If* $e \longrightarrow^n v$ *and* $e \longrightarrow^m v'$ *then* $v = v'$ *and* $n = m$

**Proof** By mathematical induction on $n$

$n = 0$**:** Then by definition of $e \longrightarrow^0 v$ we deduce that

$$e = v \tag{9}$$

By (9) and Lemma 6, we know that, since $e$ is a value, it cannot reduce further, and thus we deduce that $m = 0$ as well, *i.e.*, by (9) and $e \longrightarrow^m v'$ we deduce

$$v \longrightarrow^0 v'$$

and by the definition of $\longrightarrow^0$, we deduce $v = v'$, as required.

**Slide 56**

**Proving Determinism of $\longrightarrow^n$ for $\mathbf{L}_1$ (2)**

$n = k + 1$**:** We expand $e \longrightarrow^{k+1} v$, for some $e_1$, to

$$e \longrightarrow e_1 \longrightarrow^k v \tag{10}$$

By (10) and Lemma 6 we know $e$ cannot be a value and hence $m = l + 1$ for some number $l$ and expression $e_2$, *i.e.,*

$$e \longrightarrow e_2 \longrightarrow^l v' \tag{11}$$

By $e \longrightarrow e_1$, $e \longrightarrow e_2$ and Theorem 3 we know $e_1 = e_2$. Substituting $e_1$ for $e_2$ in (11) we get $e_1 \longrightarrow^l v'$, and by $e_1 \longrightarrow^k v$ and I.H. we obtain $v = v'$ and $k = l$, which also means that $n = k + 1 = l + 1 = m$.

**Slide 57**

The proof for Theorem 4 is based on mathematical induction over the number of reductions leading to a value. As we have discussed at length in Sec. 2.2, mathematical induction follows the same structure as rule induction. The proof for Theorem 4 is given on Slides 56 and 57.

The final result that remains to be proved from Sec. 1 is Theorem 5. It should be intuitive by now that the first clause of this theorem is proved by structural induction on $e$ whereas the proof for the second clause (the other direction of the implication) is proved by mathematical induction on $e \longrightarrow^n v$. It is instructive for the reader to try both proofs out at this point (*cf.* Sec. 2.5.5).

## 2.5 Exercises

1. Prove that for any $n \in \textsc{Nat}$ the following property holds:

$$P(x) \stackrel{\text{def}}{=} \sum_{i=0}^{x} 2^i = 2^{x+1} - 1$$

2. Prove that for any $n \in \textsc{Nat}$ the following property holds:

$$P(x) \stackrel{\text{def}}{=} \left((x+1)^2 - 1\right) = x(x+2)$$

   Can we prove this property without using induction? Why?

3. Prove for all $n \in \textsc{Nat}$ that $2n \in \textsc{Even}$.

4. Prove that $\textsc{Div} = \left\{\langle n, m\rangle \mid \text{ there exists some } k \text{ whereby } n \times k = m\right\}$

5. Consider the inductive definition:

$$\frac{}{n \ \textsc{Div}' \ n} \qquad\qquad \frac{n \ \textsc{Div}' \ k}{n \ \textsc{Div}' \ k + n}$$

   (a) Prove by induction that $\textsc{Div}' \subseteq \textsc{Div}$.

   (b) Prove that $\textsc{Div}' \subset \textsc{Div}$.

6. Prove that whenever $n \in$ Even and $m \in$ Even then $n + m \in$ Even.
7. Use the previous result to prove that whenever $n$Div$m$ and $n \in$ Even then $m \in$ Even.
8. Complete the definitions for depth($x$) and width($x$) from Slide 50 and then prove that $P(e)$ holds for all $e \in$ Exp in $\mathrm{L}_1$.
9. Complete the remaining cases for the proof of Determinism for $\Downarrow$ (*cf.* Slides 51, 52 and 53.)
10. Prove Determinism for $\longrightarrow$(*cf.* Slide 54.)
11. Prove the Correspondence Theorem for $\mathrm{L}_1$ (*cf.* Slide 23)