2 Inductive Definitions

Inductive definitions are an indispensable tool in the study of programming languages. In this chapter we will develop the basic framework of inductive definitions and give some examples of their use. An inductive definition consists of a set of *rules* for deriving *judgments*, or *assertions*, of a variety of forms. Judgments are statements about one or more abstract binding trees of some sort. The rules specify necessary and sufficient conditions for the validity of a judgment, and hence fully determine its meaning.

2.1 Judgments

We start with the notion of a *judgment*, or *assertion*, about an abstract binding tree. We shall make use of many forms of judgment, including examples such as these:

n nat	n is a natural number
$n_1 + n_2 = n$	n is the sum of n_1 and n_2
τ type	τ is a type
$e : \tau$	expression e has type τ
$e \Downarrow v$	expression e has value v

A judgment states that one or more abstract binding trees have a property or stand in some relation to one another. The property or relation itself is called a *judgment form*, and the judgment that an object or objects have that property or stand in that relation is said to be an *instance* of that judgment form. A judgment form is also called a *predicate*, and the objects constituting an instance are its *subjects*. We write a J or J a, for the judgment asserting that J holds of the abt a. Correspondingly, we sometimes notate the judgment form J by $-$ J, or J $-$, using a dash to indicate the absence of an argument to J. When it is not important to stress the subject of the judgment, we write J to stand for an unspecified judgment, that is, an instance of some judgment form. For particular judgment forms, we freely use prefix, infix, or mix-fix notation, as illustrated by the above examples, in order to enhance readability.

2.2 Inference Rules

An *inductive definition* of a judgment form consists of a collection of *rules* of the form

$$\frac{J_1 \quad \ldots \quad J_k}{J} \tag{2.1}$$

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

in which J and $J_1, \ldots, J_k$ are all judgments of the form being defined. The judgments above the horizontal line are called the *premises* of the rule, and the judgment below the line is called its *conclusion*. If a rule has no premises (that is, when k is zero), the rule is called an *axiom*; otherwise, it is called a *proper rule*.

An inference rule can be read as stating that the premises are *sufficient* for the conclusion: to show J, it is enough to show $J_1, \ldots, J_k$. When k is zero, a rule states that its conclusion holds unconditionally. Bear in mind that there may be, in general, many rules with the same conclusion, each specifying sufficient conditions for the conclusion. Consequently, if the conclusion of a rule holds, then it is not necessary that the premises hold, for it might have been derived by another rule.

For example, the following rules form an inductive definition of the judgment form $-$ nat:

$$\frac{}{\texttt{zero nat}} \tag{2.2a}$$

$$\frac{a\ \texttt{nat}}{\texttt{succ}(a)\ \texttt{nat}} \tag{2.2b}$$

These rules specify that a nat holds whenever either a is zero, or a is succ(b) where b nat for some b. Taking these rules to be exhaustive, it follows that a nat iff a is a natural number.

Similarly, the following rules constitute an inductive definition of the judgment form $-$ tree:

$$\frac{}{\texttt{empty tree}} \tag{2.3a}$$

$$\frac{a_1\ \texttt{tree} \quad a_2\ \texttt{tree}}{\texttt{node}(a_1;a_2)\ \texttt{tree}} \tag{2.3b}$$

These rules specify that a tree holds if either a is empty, or a is node(a_1;a_2), where a_1 tree and a_2 tree. Taking these to be exhaustive, these rules state that a is a binary tree, which is to say it is either empty, or a node consisting of two children, each of which is also a binary tree.

The judgment form a is b expresses the equality of two abt's a and b such that a nat and b nat is inductively defined by the following rules:

$$\frac{}{\texttt{zero is zero}} \tag{2.4a}$$

$$\frac{a\ \texttt{is}\ b}{\texttt{succ}(a)\ \texttt{is succ}(b)} \tag{2.4b}$$

In each of the preceding examples, we have made use of a notational convention for specifying an infinite family of rules by a finite number of patterns, or *rule schemes*. For example, rule (2.2b) is a rule scheme that determines one rule, called an *instance* of the rule scheme, for each choice of object a in the rule. We will rely on context to determine whether a rule is stated for a *specific* object a or is instead intended as a rule scheme specifying a rule for *each choice* of objects in the rule.

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

A collection of rules is considered to define the *strongest* judgment form that is *closed under*, or *respects*, those rules. To be closed under the rules simply means that the rules are *sufficient* to show the validity of a judgment: J holds *if* there is a way to obtain it using the given rules. To be the *strongest* judgment form closed under the rules means that the rules are also *necessary*: J holds *only if* there is a way to obtain it by applying the rules. The sufficiency of the rules means that we may show that J holds by *deriving* it by composing rules. Their necessity means that we may reason about it using *rule induction*.

2.3 Derivations

To show that an inductively defined judgment holds, it is enough to exhibit a *derivation* of it. A derivation of a judgment is a finite composition of rules, starting with axioms and ending with that judgment. It can be thought of as a tree in which each node is a rule whose children are derivations of its premises. We sometimes say that a derivation of J is evidence for the validity of an inductively defined judgment J.

We usually depict derivations as trees with the conclusion at the bottom, and with the children of a node corresponding to a rule appearing above it as evidence for the premises of that rule. Thus, if

$$\frac{J_1 \quad \dots \quad J_k}{J}$$

is an inference rule and $\nabla_1, \dots, \nabla_k$ are derivations of its premises, then

$$\frac{\nabla_1 \quad \dots \quad \nabla_k}{J}$$

is a derivation of its conclusion. In particular, if $k = 0$, then the node has no children.

For example, this is a derivation of $\texttt{succ(succ(succ(zero)))}\ \texttt{nat}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{zero nat}}}{\texttt{succ(zero) nat}}}{\texttt{succ(succ(zero)) nat}}}{\texttt{succ(succ(succ(zero))) nat}}\,. \tag{2.5}$$

Similarly, here is a derivation of $\texttt{node(node(empty;empty);empty)}\ \texttt{tree}$:

$$\dfrac{\dfrac{\dfrac{}{\texttt{empty tree}} \quad \dfrac{}{\texttt{empty tree}}}{\texttt{node(empty;empty) tree}} \quad \dfrac{}{\texttt{empty tree}}}{\texttt{node(node(empty;empty);empty) tree}}\,. \tag{2.6}$$

To show that an inductively defined judgment is derivable, we need only find a derivation for it. There are two main methods for finding derivations, called *forward chaining*, or *bottom-up construction*, and *backward chaining*, or *top-down construction*. Forward

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

chaining starts with the axioms and works forward towards the desired conclusion, whereas backward chaining starts with the desired conclusion and works backwards towards the axioms.

More precisely, forward chaining search maintains a set of derivable judgments and continually extends this set by adding to it the conclusion of any rule all of whose premises are in that set. Initially, the set is empty; the process terminates when the desired judgment occurs in the set. Assuming that all rules are considered at every stage, forward chaining will eventually find a derivation of any derivable judgment, but it is impossible (in general) to decide algorithmically when to stop extending the set and conclude that the desired judgment is not derivable. We may go on and on adding more judgments to the derivable set without ever achieving the intended goal. It is a matter of understanding the global properties of the rules to determine that a given judgment is not derivable.

Forward chaining is undirected in the sense that it does not take account of the end goal when deciding how to proceed at each step. In contrast, backward chaining is goal-directed. Backward chaining search maintains a queue of current goals, judgments whose derivations are to be sought. Initially, this set consists solely of the judgment we wish to derive. At each stage, we remove a judgment from the queue and consider all rules whose conclusion is that judgment. For each such rule, we add the premises of that rule to the back of the queue, and continue. If there is more than one such rule, this process must be repeated, with the same starting queue, for each candidate rule. The process terminates whenever the queue is empty, all goals having been achieved; any pending consideration of candidate rules along the way can be discarded. As with forward chaining, backward chaining will eventually find a derivation of any derivable judgment, but there is, in general, no algorithmic method for determining in general whether the current goal is derivable. If it is not, we may futilely add more and more judgments to the goal set, never reaching a point at which all goals have been satisfied.

2.4 Rule Induction

Because an inductive definition specifies the *strongest* judgment form closed under a collection of rules, we may reason about them by *rule induction*. The principle of rule induction states that to show that a property $a\ \mathcal{P}$ holds whenever $a\ \mathsf{J}$ is derivable, it is enough to show that $\mathcal{P}$ is *closed under*, or *respects*, the rules defining the judgment form J. More precisely, the property $\mathcal{P}$ respects the rule

$$\frac{a_1\ \mathsf{J} \quad \ldots \quad a_k\ \mathsf{J}}{a\ \mathsf{J}}$$

if $\mathcal{P}(a)$ holds whenever $\mathcal{P}(a_1), \ldots, \mathcal{P}(a_k)$ do. The assumptions $\mathcal{P}(a_1), \ldots, \mathcal{P}(a_k)$ are called the *inductive hypotheses*, and $\mathcal{P}(a)$ is called the *inductive conclusion* of the inference.

The principle of rule induction is simply the expression of the definition of an inductively defined judgment form as the *strongest* judgment form closed under the rules comprising the definition. Thus, the judgment form defined by a set of rules is both (a) closed under

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

those rules, and (b) sufficient for any other property also closed under those rules. The former means that a derivation is evidence for the validity of a judgment; the latter means that we may reason about an inductively defined judgment form by rule induction.

When specialized to rules (2.2), the principle of rule induction states that to show $\mathcal{P}(a)$ whenever a nat, it is enough to show:

1. $\mathcal{P}(\texttt{zero})$.
2. for every a, if $\mathcal{P}(a)$, then $\mathcal{P}(\texttt{succ}(a))$.

The sufficiency of these conditions is the familiar principle of *mathematical induction*.

Similarly, rule induction for rules (2.3) states that to show $\mathcal{P}(a)$ whenever a tree, it is enough to show

1. $\mathcal{P}(\texttt{empty})$.
2. for every a_1 and a_2, if $\mathcal{P}(a_1)$, and if $\mathcal{P}(a_2)$, then $\mathcal{P}(\texttt{node}(a_1;a_2))$.

The sufficiency of these conditions is called the principle of *tree induction*.

We may also show by rule induction that the predecessor of a natural number is also a natural number. Although this may seem self-evident, the point of the example is to show how to derive this from first principles.

Lemma 2.1. *If* $\texttt{succ}(a)$ nat*, then* a nat.

Proof It suffices to show that the property $\mathcal{P}(a)$ stating that a nat and that $a = \texttt{succ}(b)$ implies b nat is closed under rules (2.2).

Rule (2.2a) Clearly zero nat, and the second condition holds vacuously, because zero is not of the form $\texttt{succ}(-)$.

Rule (2.2b) Inductively, we know that a nat and that if a is of the form $\texttt{succ}(b)$, then b nat. We are to show that $\texttt{succ}(a)$ nat, which is immediate, and that if $\texttt{succ}(a)$ is of the form $\texttt{succ}(b)$, then b nat, and we have b nat by the inductive hypothesis. □

Using rule induction, we may show that equality, as defined by rules (2.4) is reflexive.

Lemma 2.2. *If* a nat*, then* a is a.

Proof By rule induction on rules (2.2):

Rule (2.2a) Applying rule (2.4a) we obtain zero is zero.

Rule (2.2b) Assume that a is a. It follows that $\texttt{succ}(a)$ is $\texttt{succ}(a)$ by an application of rule (2.4b). □

Similarly, we may show that the successor operation is injective.

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

Lemma 2.3. *If* $\mathtt{succ}(a_1)$ is $\mathtt{succ}(a_2)$*, then* a_1 is a_2.

Proof Similar to the proof of Lemma 2.1. □

2.5 Iterated and Simultaneous Inductive Definitions

Inductive definitions are often *iterated*, meaning that one inductive definition builds on top of another. In an iterated inductive definition, the premises of a rule

$$\frac{J_1 \quad \ldots \quad J_k}{J}$$

may be instances of either a previously defined judgment form, or the judgment form being defined. For example, the following rules define the judgment form $-$ list, which states that a is a list of natural numbers:

$$\frac{}{\mathtt{nil}\ \mathsf{list}} \tag{2.7a}$$

$$\frac{a\ \mathsf{nat} \quad b\ \mathsf{list}}{\mathtt{cons}(a;b)\ \mathsf{list}} \tag{2.7b}$$

The first premise of rule (2.7b) is an instance of the judgment form a nat, which was defined previously, whereas the premise b list is an instance of the judgment form being defined by these rules.

Frequently two or more judgments are defined at once by a *simultaneous inductive definition*. A simultaneous inductive definition consists of a set of rules for deriving instances of several different judgment forms, any of which may appear as the premise of any rule. Because the rules defining each judgment form may involve any of the others, none of the judgment forms can be taken to be defined prior to the others. Instead, we must understand that all of the judgment forms are being defined at once by the entire collection of rules. The judgment forms defined by these rules are, as before, the strongest judgment forms that are closed under the rules. Therefore, the principle of proof by rule induction continues to apply, albeit in a form that requires us to prove a property of each of the defined judgment forms simultaneously.

For example, consider the following rules, which constitute a simultaneous inductive definition of the judgments a even, stating that a is an even natural number, and a odd, stating that a is an odd natural number:

$$\frac{}{\mathtt{zero}\ \mathsf{even}} \tag{2.8a}$$

$$\frac{b\ \mathsf{odd}}{\mathtt{succ}(b)\ \mathsf{even}} \tag{2.8b}$$

$$\frac{a\ \mathsf{even}}{\mathtt{succ}(a)\ \mathsf{odd}} \tag{2.8c}$$

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

The principle of rule induction for these rules states that to show simultaneously that $\mathcal{P}(a)$ whenever a even and $\mathcal{Q}(b)$ whenever b odd, it is enough to show the following:

1. $\mathcal{P}(\mathtt{zero})$;
2. if $\mathcal{Q}(b)$, then $\mathcal{P}(\mathtt{succ}(b))$;
3. if $\mathcal{P}(a)$, then $\mathcal{Q}(\mathtt{succ}(a))$.

As an example, we may use simultaneous rule induction to prove that (1) if a even, then either a is zero or a is succ(b) with b odd, and (2) if a odd, then a is succ(b) with b even. We define $\mathcal{P}(a)$ to hold iff a is zero or a is succ(b) for some b with b odd, and define $\mathcal{Q}(b)$ to hold iff b is succ(a) for some a with a even. The desired result follows by rule induction, because we can prove the following facts:

1. $\mathcal{P}(\mathtt{zero})$, which holds because zero is zero.
2. If $\mathcal{Q}(b)$, then succ(b) is succ(b') for some b' with $\mathcal{Q}(b')$. Take b' to be b and apply the inductive assumption.
3. If $\mathcal{P}(a)$, then succ(a) is succ(a') for some a' with $\mathcal{P}(a')$. Take a' to be a and apply the inductive assumption.

2.6 Defining Functions by Rules

A common use of inductive definitions is to define a function by giving an inductive definition of its *graph* relating inputs to outputs, and then showing that the relation uniquely determines the outputs for given inputs. For example, we may define the addition function on natural numbers as the relation sum(a;b;c), with the intended meaning that c is the sum of a and b, as follows:

$$\frac{b\ \mathsf{nat}}{\mathsf{sum}(\mathtt{zero};b;b)} \tag{2.9a}$$

$$\frac{\mathsf{sum}(a;b;c)}{\mathsf{sum}(\mathtt{succ}(a);b;\mathtt{succ}(c))} \tag{2.9b}$$

The rules define a ternary (three-place) relation sum(a;b;c) among natural numbers a, b, and c. We may show that c is determined by a and b in this relation.

Theorem 2.4. *For every* a nat *and* b nat*, there exists a unique* c nat *such that* sum(a;b;c).

Proof The proof decomposes into two parts:

1. (Existence) If a nat and b nat, then there exists c nat such that sum(a;b;c).
2. (Uniqueness) If sum(a;b;c), and sum(a;b;c'), then c is c'.

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

For existence, let $\mathcal{P}(a)$ be the proposition *if* b nat *then there exists* c nat *such that* sum$(a;b;c)$. We prove that if a nat then $\mathcal{P}(a)$ by rule induction on rules (2.2). We have two cases to consider:

Rule (2.2a) We are to show $\mathcal{P}(\texttt{zero})$. Assuming b nat and taking c to be b, we obtain sum$(\texttt{zero};b;c)$ by rule (2.9a).

Rule (2.2b) Assuming $\mathcal{P}(a)$, we are to show $\mathcal{P}(\texttt{succ}(a))$. That is, we assume that if b nat then there exists c such that sum$(a;b;c)$ and are to show that if b' nat, then there exists c' such that sum$(\texttt{succ}(a);b';c')$. To this end, suppose that b' nat. Then by induction there exists c such that sum$(a;b';c)$. Taking c' to be $\texttt{succ}(c)$, and applying rule (2.9b), we obtain sum$(\texttt{succ}(a);b';c')$, as required.

For uniqueness, we prove that *if* sum$(a;b;c_1)$*, then if* sum$(a;b;c_2)$*, then* c_1 is c_2 by rule induction based on rules (2.9).

Rule (2.9a) We have a is $\texttt{zero}$ and c_1 is b. By an inner induction on the same rules, we may show that if sum$(\texttt{zero};b;c_2)$, then c_2 is b. By Lemma 2.2, we obtain b is b.

Rule (2.9b) We have that a is $\texttt{succ}(a')$ and c_1 is $\texttt{succ}(c_1')$, where sum$(a';b;c_1')$. By an inner induction on the same rules, we may show that if sum$(a;b;c_2)$, then c_2 is $\texttt{succ}(c_2')$ where sum$(a';b;c_2')$. By the outer inductive hypothesis, c_1' is c_2' and so c_1 is c_2. □

2.7 Notes

Aczel (1977) provides a thorough account of the theory of inductive definitions on which the present account is based. A significant difference is that we consider inductive definitions of judgments over abt's as defined in Chapter 1, rather than with natural numbers. The emphasis on judgments is inspired by Martin-Löf's logic of judgments (Martin-Löf, 1983, 1987).

Exercises

2.1. Give an inductive definition of the judgment max$(m;n;p)$, where m nat, n nat, and p nat, with the meaning that p is the larger of m and n. Prove that every m and n are related to a unique p by this judgment.

2.2. Consider the following rules, which define the judgment hgt$(t;n)$ stating that the binary tree t has *height* n.

$$\frac{}{\texttt{hgt}(\texttt{empty};\texttt{zero})} \tag{2.10a}$$

$$\frac{\texttt{hgt}(t_1;n_1) \quad \texttt{hgt}(t_2;n_2) \quad \texttt{max}(n_1;n_2;n)}{\texttt{hgt}(\texttt{node}(t_1;t_2);\texttt{succ}(n))} \tag{2.10b}$$

Prove that the judgment hgt defines a function from trees to natural numbers.

https://doi.org/10.1017/CBO9781316576892.004 Published online by Cambridge University Press

2.3. Given an inductive definition of *ordered variadic trees* whose nodes have a finite, but variable, number of children with a specified left-to-right ordering among them. Your solution should consist of a simultaneous definition of two judgments, t tree, stating that t is a variadic tree, and f forest, stating that f is a "forest" (finite sequence) of variadic trees.

2.4. Give an inductive definition of the height of a variadic tree of the kind defined in Exercise **2.3**. Your definition should make use of an auxiliary judgment defining the height of a forest of variadic trees and will be defined simultaneously with the height of a variadic tree. Show that the two judgments so defined each define a function.

2.5. Give an inductive definition of the *binary natural numbers*, which are either zero, twice a binary number, or one more than twice a binary number. The size of such a representation is logarithmic, rather than linear, in the natural number it represents.

2.6. Give an inductive definition of addition of binary natural numbers as defined in Exercise **2.5**. *Hint*: Proceed by analyzing both arguments to the addition, and make use of an auxiliary function to compute the successor of a binary number. *Hint*: Alternatively, define both the sum and the sum-plus-one of two binary numbers mutually recursively.