CHAPTER 11

Induction

11.1 Introduction

Induction is reasoning from the specific to the general. If various instances of a schema are true and there are no counterexamples, we are tempted to conclude a universally quantified version of the schema.

$$\begin{array}{l} p(a) \Rightarrow q(a) \\ p(b) \Rightarrow q(b) \\ p(c) \Rightarrow q(c) \end{array} \qquad \rightarrow \qquad \forall x.(p(x) \Rightarrow q(x))$$

Incomplete induction is induction where the set of instances is not exhaustive. From a reasonable collection of instances, we sometimes leap to the conclusion that a schema is always true even though we have not seen all instances. Consider, for example, the function f where $f(1)=1$, and $f(n+1)=f(n) + 2n + 1$. If we look at some values of this function, we notice a certain regularity - the value of f always seems to be the square of its input. From this sample, we are tempted to leap to the conclusion that $f(n)=n^2$. Lucky guess. In this case, the conclusion happens to be true; and we can prove it.

n	$f(n)$	n^2
1	1	1^2
2	4	2^2
3	9	3^2
4	16	4^2
5	25	5^2

Here is another example. This one is due to the mathematician Fermat (1601-1665). He looked at values of the expression $2^{2^n} + 1$ for various values of n and noticed that they were all prime. So, he concluded the value of the expression was prime number. Unfortunately, this was not a lucky guess. His conjecture was ultimately disproved, in fact with the very next number in the sequence. (Mercifully, the counterexample was found after his death.)

n	$2^{2^n}+1$	Prime?
1	5	Yes
2	17	Yes
3	257	Yes
4	65537	Yes

For us, this is not so good. In Logic, we are concerned with logical entailment. We want to derive only conclusions that are guaranteed to be true when the premises are true. Guesses like these are useful in suggesting possible conclusions, but they are not themselves proofs. In order to be sure of

universally quantified conclusions, we must be sure that *all* instances are true. This is called *complete induction*. When applied to numbers, it is usually called *mathematical induction*.

The technique used for complete induction varies with the structure of the language to which it is applied. We begin this chapter with a discussion of domain closure, a rule that applies when the Herbrand base of our language is finite. We then move on to Linear Induction, i.e. the special case in which the ground terms in the language form a linear sequence. We look at tree induction, i.e. the special case in which the ground terms of the language form a tree. And we look at Structural Induction, which applies to all languages. Finally, we look at two special cases that make inductive reasoning more complicated - Multidimensional Induction and Embedded Induction.

11.2 Domain Closure

Induction for finite languages is trivial. We simply use the Domain Closure rule of inference. For a language with object constants $\sigma_1, \ldots, \sigma_n$, the rule is written as shown below. If we believe a schema is true for every instance, then we can infer a universally quantified version of that schema.

$$\phi[\sigma_1]$$
$$\ldots$$
$$\phi[\sigma_n]$$
$$\forall \nu.\phi[\nu]$$

Recall that, in our formalization of the Sorority World, we have just four constants - *abby*, *bess*, *cody*, and *dana*. For this language, we would have the following Domain Closure rule.

Domain Closure (DC)

$\phi[abby]$
$\phi[bess]$
$\phi[cody]$
$\phi[dana]$
$\forall \nu.\phi[\nu]$

The following proof shows how we can use this rule to derive an inductive conclusion. Given the premises we considered earlier in this book, it is possible to infer that Abby likes someone, Bess likes someone, Cody likes someone, and Dana likes someone. Taking these conclusions as premises and using our Domain Closure rule, we can then derive the inductive conclusion $\forall x.\exists y.likes(x,y)$, i.e. everybody likes somebody.

1. $\exists y.likes(abby,y)$ Premise
2. $\exists y.likes(bess,y)$ Premise
3. $\exists y.likes(cody,y)$ Premise
4. $\exists y.likes(dana,y)$ Premise
5. $\forall x.\exists y.likes(x,y)$ Domain Closure: 1, 2, 3, 4

Unfortunately, this technique does not work when there are infinitely many ground terms. Suppose, for example, we have a language with ground terms $\sigma_1, \sigma_2, \ldots$ A direct generalization of the Domain Closure rule is shown below.

$$\phi[\sigma_1]$$
$$\phi[\sigma_2]$$
$$...$$
$$\forall \nu.\phi[\nu]$$

This rule is sound in the sense that the conclusion of the rule is logically entailed by the premises of the rule. However, it does not help us produce a proof of this conclusion. To use the rule, we would need to prove all of the rule's premises. Unfortunately, there are infinitely many premises. So, the rule cannot be used in generating a finite proof.

All is not lost. It is sometimes possible to write rules that cover all instances without enumerating them individually. The method depends on the structure of the language. The next sections describe how this can be done for languages with different structures.

11.3 Linear Induction

Imagine an infinite set of dominoes placed in a line so that, when one falls, the next domino in the line also falls. If the first domino falls, then the second domino falls. If the second domino falls, then the third domino falls. And so forth. By continuing this chain of reasoning, it is easy for us to convince ourselves that every domino eventually falls. This is the intuition behind a technique called Linear Induction.

Consider a language with a single object constant *a* and a single unary function constant *s*. There are infinitely many ground terms in this language, arranged in what resembles a straight line. See below. Starting from the object constant, we move to a term in which we apply the function constant to that constant, then to a term in which we apply the function constant to that term, and so forth.

$$a \rightarrow s(a) \rightarrow s(s(a)) \rightarrow s(s(s(a))) \rightarrow ...$$

In this section, we concentrate on languages that have linear structure of this sort. Hereafter, we call these *linear languages*. In all cases, there is a single object constant and a single unary function constant. In talking about a linear language, we call the object constant the *base element* of the language, and we call the unary function constant the *successor function*.

Although there are infinitely many ground terms in any linear language, we can still generate finite proofs that are guaranteed to be correct. The trick is to use the structure of the terms in the language in expressing the premises of an inductive rule of inference known as *Linear Induction*. See below for a statement of the induction rule for the language introduced above. In general, if we know that a schema holds of our base element and if we know that, whenever the schema holds of an element, it also holds of the successor of that element, then we can conclude that the schema holds of all elements.

Linear Induction

$$\phi[a]$$
$$\forall \mu.(\phi[\mu] \Rightarrow \phi[s(\mu)])$$
$$\forall \nu.\phi[\nu]$$

A bit of terminology before we go on. The first premise in this rule is called the *base case* of the induction, because it refers to the base element of the language. The second premise is called the *inductive case*. The antecedent of the inductive case is called the *inductive hypothesis*, and the consequent is called, not surprisingly, the *inductive conclusion*. The conclusion of the rule is sometimes called the *overall conclusion* to distinguish it from the inductive conclusion.

For the language introduced above, our rule of inference is sound. Suppose we know that a schema is true of a and suppose that we know that, whenever the schema is true of an arbitrary ground term τ, it is also true of the term $s(\tau)$. Then, the schema must be true of everything, since there are no other terms in the language.

The requirement that the signature consists of no other object constants or function constants is crucial. If this were not the case, say there were another object constant b, then we would have trouble. It would still be true that ϕ holds for every element in the set $\{a, s(a), s(s(a)),...\}$. However, because there are other elements in the Herbrand universe, e.g. b and $s(b)$, $\forall x.\phi(x)$ would no longer be guaranteed.

Here is an example of induction in action. Recall the formalization of Arithmetic introduced in Chapter 9. Using the object constant 0 and the unary function constant s, we represent each number n by applying the function constant to 0 exactly n times. For the purposes of this example, let's assume we have just one ternary relation constant, viz. *plus*, which we use to represent the addition table.

The following axioms describe *plus* in terms of 0 and s. The first sentence here says that adding 0 to any element produces that element. The second sentence states that adding the successor of a number to another number yields the successor of their sum. The third sentence is a functionality axiom for *plus*.

$$\forall y.plus(0,y,y)$$
$$\forall x.\forall y.\forall z.(plus(x,y,z) \Rightarrow plus(s(x),y,s(z)))$$
$$\forall x.\forall y.\forall z.\forall w.(plus(x,y,z) \wedge \neg same(z,w) \Rightarrow \neg plus(x,y,w))$$

It is easy to see that any table that satisfies these axioms includes all of the usual addition facts. The first axiom ensures that all cases with 0 as first argument are included. From this fact and the second axiom, we can see that all cases with $s(0)$ as first argument are included. And so forth.

The first axiom above tells us that 0 is a *left identity* for addition - 0 added to any number produces that number as result. As it turns out, given these definitions, 0 must also be a *right identity*, i.e it must be the case that $\forall x.plus(x,0,x)$.

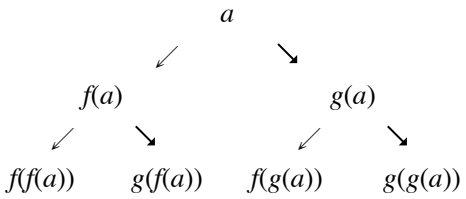
We can use induction to prove this result as shown below. We start with our premises. We use Universal Elimination on the first premise to derive the sentence on line 3. This takes care of the base case of our induction. We then start a subproof and assume the antecedent of the inductive case. We then use three applications of Universal Elimination on the second premise to get the sentence on line 5. We use Implication Elimination on this sentence and our assumption to derive the conclusion on line 6. We then discharge our assumption and form the implication shown on line 7 and then universalize this to get the result on line 8. Finally, we use Linear Induction to derive our overall conclusion.

1.	$\forall y.plus(0,y,y)$	Premise
2.	$\forall x.\forall y.\forall z.(plus(x,y,z) \Rightarrow plus(s(x),y,s(z)))$	Premise
3.	$plus(0,0,0)$	UE: 1
4.	$plus(x,0,x)$	Assumption
5.	$plus(x,0,x) \Rightarrow plus(s(x),0,s(x))$	3 x UE: 2
6.	$plus(s(x),0,s(x))$	IE: 5, 4
7.	$plus(x,0,x) \Rightarrow plus(s(x),0,s(x))$	II: 4, 6
8.	$\forall x.(plus(x,0,x) \Rightarrow plus(s(x),0,s(x)))$	UI: 7
9.	$\forall x.plus(x,0,x)$	Ind: 3, 8

Most inductive proofs have this simple structure. We prove the base case. We assume the inductive hypothesis; we prove the inductive conclusion; and, based on this proof, we have the inductive case. From the base case and the inductive case, we infer the overall conclusion.

11.4 Tree Induction

Tree languages are a superset of linear languages. While in linear languages the terms in the language form a linear sequence, in tree languages the structure is more tree-like. Consider a language with an object constant a and two unary function constants f and g. Some of the terms in this language are shown below.



As with linear languages, we can write an inductive rule of inference for tree languages. The tree induction rule of inference for the language just described is shown below. Suppose we know that a schema ϕ holds of a. Suppose we know that, whenever the schema holds of any element, it holds of the term formed by applying f to that element. And suppose we know that, whenever the schema holds of any element, it holds of the term formed by applying g to that element. Then, we can conclude that the schema holds of every element.

Tree Induction

$\phi[a]$

$\forall \mu.(\phi[\mu] \Rightarrow \phi[f(\mu)])$

$\forall \mu.(\phi[\mu] \Rightarrow \phi[g(\mu)])$

$\forall \nu.\phi[\nu]$

In order to see an example of tree induction in action, consider the ancestry tree for a particular dog. We use the object constant *rex* to refer to the dog; we use the unary function constant f to map an arbitrary dog to its father; and we use g map a dog to its mother. Finally, we have a single unary relation constant *purebred* that is true of a dog if and only if it is purebred.

Now, we write down the fundamental rule of dog breeding - we say that a dog is purebred if and only if both its father and its mother are purebred. See below. (This is a bit oversimplified on

several grounds. Properly, the father and mother should be of the same breed. Also, this formalization suggests that every dog has an ancestry tree that stretches back without end. However, let's ignore these imperfections for the purposes of our example.)

$$\forall x.(purebred(x) \Leftrightarrow purebred(f(x)) \wedge purebred(g(x)))$$

Suppose now that we discover the fact that our dog *rex* is purebred. Then, we know that every dog in his ancestry tree must be purebred. We can prove this by a simple application of tree induction.

A proof of our conclusion is shown below. We start with the premise that Rex is purebred. We also have our constraint on purebred animals as a premise. We use Universal Elimination to instantiate the second premise, and then we use Biconditional Elimination on the biconditional in line 3 to produce the implication on line 4. On line 5, we start a subproof with the assumption the x is purebred. We use Implication Elimination to derive the conjunction on line 6, and then we use And Elimination to pick out the first conjunct. We then use Implication Introduction to discharge our assumption, and we Universal Introduction to produce the inductive case for f. We then repeat this process to produce an analogous result for g on line 14. Finally, we use the tree induction rule on the sentences on lines 1, 9, and 14 and thereby derive the desired overall conclusion.

1.	$purebred(rex)$	Premise
2.	$\forall x.(purebred(x) \Leftrightarrow purebred(f(x)) \wedge purebred(g(x)))$	Premise
3.	$purebred(x) \Leftrightarrow purebred(f(x)) \wedge purebred(g(x))$	UE: 2
4.	$purebred(x) \Rightarrow purebred(f(x)) \wedge purebred(g(x))$	BE: 3
5.	$\vert$ $purebred(x)$	Assumption
6.	$\vert$ $purebred(f(x)) \wedge purebred(g(x))$	IE: 4, 5
7.	$\vert$ $purebred(f(x))$	AE: 6
8.	$purebred(x) \Rightarrow purebred(f(x))$	II: 5, 7
9.	$\forall x.(purebred(x) \Rightarrow purebred(f(x)))$	UI: 8
10.	$\vert$ $purebred(x)$	Assumption
11.	$\vert$ $purebred(f(x)) \wedge purebred(g(x))$	IE: 4, 10
12.	$\vert$ $purebred(g(x))$	AE: 11
13.	$purebred(x) \Rightarrow purebred(g(x))$	II: 10, 12
14.	$\forall x.(purebred(x) \Rightarrow purebred(g(x)))$	UI: 13
15.	$\forall x.purebred(x)$	Ind: 1, 9, 14

11.5 Structural Induction

Structural Induction is the most general form of induction. In Structural Induction, we can have multiple object constants, multiple function constants, and, unlike our other forms of induction, we can have function constants with multiple arguments. Consider a language with two object constants a and b and a single binary function constant c. See below for a list of some of the terms in the language. We do not provide a graphical rendering in this case, as the structure is more complicated than a line or a tree.

$$a, b, c(a,a), c(a,b), c(b,a), c(b,b), c(a,c(a,a)), c(c(a,a),a), c(c(a,a),c(a,a)), ...$$

The Structural Induction rule for this language is shown below. If we know that ϕ holds of our base elements a and b and if we know $\forall\mu.\forall\nu.((\phi[\mu] \wedge \phi[\nu]) \Rightarrow \phi[c(\mu,\nu)])$, then we can conclude $\forall\nu.\phi[\nu]$ in a single step.

Structural Induction

$\phi[a]$
$\phi[b]$
$\forall\lambda.\forall\mu.((\phi[\lambda] \wedge \phi[\mu]) \Rightarrow \phi[c(\lambda,\mu)])$
$\forall\nu.\phi[\nu]$

As an example of a domain where Structural Induction is appropriate, recall the world of lists and trees introduced in Chapter 9. Let's assume we have two object constants a and b, a binary function constant c, and two unary relation constants p and q. Relation p is true of a structured object if and only if every fringe node is an a. Relation q is true of a structured object if and only if at least one fringe node is an a. The positive and negative axioms defining the relations are shown below.

$p(a)$	$q(a)$
$\forall u.\forall v.(p(u) \wedge p(v) \Rightarrow p(c(u,v)))$	$\forall u.\forall v.(q(u) \Rightarrow q(c(u,v)))$
$\neg p(b)$	$\forall u.\forall v.(q(v) \Rightarrow q(c(u,v)))$
$\forall u.\forall v.(p(c(u,v)) \Rightarrow p(u))$	$\neg q(b)$
$\forall u.\forall v.(p(c(u,v)) \Rightarrow p(v))$	$\forall u.\forall v.(q(c(u,v)) \Rightarrow q(u) \vee q(v))$

Now, as an example of Structural Induction in action, let's prove that every object that satisfies p also satisfies q. In other words, we want to prove the conclusion $\forall x.(p(x) \Rightarrow q(x))$. As usual, we start with our premises.

1.	$p(a)$	Premise
2.	$\forall u.\forall v.(p(u) \wedge p(v) \Rightarrow p(c(u,v)))$	Premise
3.	$\neg p(b)$	Premise
4.	$\forall u.\forall v.(p(c(u,v)) \Rightarrow p(u))$	Premise
5.	$\forall u.\forall v.(p(c(u,v)) \Rightarrow p(v))$	Premise
6.	$q(a)$	Premise
7.	$\forall u.\forall v.(q(u) \Rightarrow q(c(u,v)))$	Premise
8.	$\forall u.\forall v.(q(v) \Rightarrow q(c(u,v)))$	Premise
9.	$\neg q(b)$	Premise
10.	$\forall u.\forall v.(q(c(u,v)) \Rightarrow q(u) \vee q(v))$	Premise

To start the induction, we first prove the base cases for the conclusion. In this world, with two object constants, we need to show the result twice - once for each object constant in the language.

Let's start with a. The derivation is simple in this case. We assume $p(a)$, reiterate $q(a)$ from line 6, then use Implication Introduction to prove $(p(a) \Rightarrow q(a))$.

11.	$p(a)$	Assumption
12.	$q(a)$	Reiteration: 6
13.	$p(a) \Rightarrow q(a)$	Implication Introduction

Now, let's do the case for b. As before, we assume $p(b)$, and our goal is to derive $q(b)$. This is a little strange. We know that $q(b)$ is false. Still, we should be able to derive it since we have assumed $p(b)$, which is also false. The trick here is to generate contradictory conclusions from the assumption $\neg q(b)$. To this end, we assume $\neg q(b)$ and first prove $p(b)$. Having done so, we use Implication Introduction to get one implication. Then, we assume $\neg q(b)$ again and this time derive $\neg p(b)$ and get an implication. At this point, we can use Negation Introduction to derive $\neg\neg q(b)$ and Negation Elimination to get $q(b)$. Finally, we use Implication Introduction to prove $(p(b) \Rightarrow q(b))$.

14.	$p(b)$	Assumption
15.	$\quad \neg q(b)$	Assumption
16.	$\quad p(b)$	Reiteration: 14
17.	$\neg q(b) \Rightarrow p(b)$	Implication Introduction: 14, 16
18.	$\quad \neg q(b)$	Assumption
19.	$\quad \neg p(b)$	Reiteration: 3
20.	$\neg q(b) \Rightarrow \neg p(b)$	Implication Introduction: 18, 19
21.	$\neg\neg q(b)$	Negation Introduction: 17, 20
22.	$q(b)$	Negation Elimination: 21
23.	$p(b) \Rightarrow q(b)$	Implication Introduction: 17, 19

Having dealt with the base cases, the next step is to prove the inductive case. We need to show that, if our conclusion holds of u and v, then it also holds of $c(u,v)$. To this end, we assume the conjunction of our assumptions and then use And Elimination to split that conjunction into its two conjuncts. Our inductive conclusion is also an implication; and, to prove it, we assume its antecedent $p(c(u,v))$. From this, we derive $p(u)$; from that, we derive $q(u)$; and, from that, we derive $q(c(u,v))$. We then use Implication Introduction to get the desired implication. A final use of Implication Introduction and a couple of applications of Universal Introduction gives us the inductive case for the induction.

24.	$(p(u) \Rightarrow q(u)) \wedge (p(v) \Rightarrow q(v))$	Assumption
25.	$p(u) \Rightarrow q(u)$	And Elimination: 24
26.	$p(v) \Rightarrow q(v)$	And Elimination: 24
27.	$p(c(u,v))$	Assumption
28.	$\forall v.(p(c(u,v)) \Rightarrow p(u))$	Universal Elimination: 4
29.	$p(c(u,v)) \Rightarrow p(u)$	Universal Elimination: 28
30.	$p(u)$	Implication Elimination: 29, 27
31.	$q(u)$	Implication Elimination: 25, 30
32.	$\forall v.(q(u) \Rightarrow q(c(u,v)))$	Universal Elimination: 7
33.	$q(u) \Rightarrow q(c(u,v))$	Universal Elimination: 32
34.	$q(c(u,v))$	Implication Elimination: 33, 31
35.	$p(c(u,v)) \Rightarrow q(c(u,v))$	Implication Introduction: 27, 34
36.	$(p(u) \Rightarrow q(u)) \wedge (p(v) \Rightarrow q(v)) \Rightarrow (p(c(u,v)) \Rightarrow q(c(u,v)))$	Implication Introduction: 24, 35
37.	$\forall v.((p(u) \Rightarrow q(u)) \wedge (p(v) \Rightarrow q(v)) \Rightarrow (p(c(u,v))) \Rightarrow q(c(u,v)))$	Universal Introduction: 36
38.	$\forall u.\forall v.((p(u) \Rightarrow q(u)) \wedge (p(v) \Rightarrow q(v)) \Rightarrow (p(c(u,v)) \Rightarrow q(c(u,v))))$	Universal Introduction: 37

Finally, using the base case on lines 13 and 23 and the inductive case on line 38, we use Structural Induction to give us the conclusion we set out to prove.

39.	$\forall x.(p(x) \Rightarrow q(x))$	Induction: 13, 23, 38

Although the proof in this case is longer than in the previous examples, the basic inductive structure is the same. Importantly, using induction, we can prove this result where otherwise it would not be possible.

11.6 Multidimensional Induction

In our look at induction thus far, we have been concentrating on examples in which the conclusion is a universally quantified sentence with just one variable. In many situations, we want to use induction to prove a result with more than one universally quantified variable. This is called *multidimensional induction* or, sometimes, *multivariate induction*.

In principle, multidimensional induction is straightforward. We simply use ordinary induction to prove the outermost universally quantified sentence. Of course, in the case of multidimensional induction the base case and the inductive conclusion are themselves universally quantified sentences; and, if necessary we use induction to prove these subsidiary results.

As an example, consider a language with a single object constant a, a unary function constant s, and a binary relation constant e. The axioms shown below define e.

$$e(a,a)$$
$$\forall x.\neg e(a,s(x))$$
$$\forall x.\neg e(s(x),a)$$
$$\forall x.\forall y.(e(x,y) \Rightarrow e(s(x),s(y)))$$
$$\forall x.\forall y.(e(s(x),s(y)) \Rightarrow e(x,y))$$

The relation e is an equivalence relation - it is reflexive, symmetric, and transitive. Proving reflexivity is easy. Proving transitivity is left as an exercise for the reader. Our goal here is to prove symmetry.

$$\text{Goal: } \forall x.\forall y.(e(x,y) \Rightarrow e(y,x)).$$

In what follows, we use induction to prove the outer quantified formula and then use induction on each of the inner conclusions as well. This means we have two immediate subgoals - the base case for the outer induction and the inductive case for the outer induction.

$$\text{Goal: } \forall y.(e(a,y) \Rightarrow e(y,a)).$$
$$\text{Goal: } \forall x.(\forall y.(e(x,y) => e(y,x)) \Rightarrow \forall y.(e(s(x),y) => e(y,s(x)))).$$

As usual, we start with our premises. We then prove the base case of the inner induction. This is easy. We assume $e(a,a)$ and then use Implication Introduction to prove the base case in one step.

1.	$e(a,a)$	Premise
2.	$\forall x.\neg e(a,s(x))$	Premise
3.	$\forall x.\neg e(s(x),a)$	Premise
4.	$\forall x.\forall y.(e(x,y) \Rightarrow e(s(x),s(y)))$	Premise
5.	$\forall x.\forall y.(e(s(x),s(y)) \Rightarrow e(x,y))$	Premise
6.	$\vert\ e(a,a)$	Assumption
7.	$e(a,a) \Rightarrow e(a,a)$	Implication Introduction: 6, 6

The next step is to prove the inductive case for the inner induction. To this end, we assume the inductive hypothesis and try to prove the inductive conclusion. Since the conclusion is itself an implication, we assume its antecedent and prove the consequent. As shown below, we do this by assuming the consequent is false and proving a sentence and its negation. We then use Negation Introduction and Negation Elimination to derives the consequent. We finish with two applications of Implication Introduction and an application of Universal Introduction.

8.	$e(a,y) \Rightarrow e(y,a))$	Assumption
9.	$\quad e(a,s(y))$	Assumption
10.	$\quad\quad \neg e(s(y),a)$	Assumption
11.	$\quad\quad e(a,s(y))$	Reiteration: 9
12.	$\quad \neg e(s(y),a,) \Rightarrow e(a,s(y))$	Implication Introduction: 10, 11
13.	$\quad\quad \neg e(s(y),a)$	Assumption
14.	$\quad\quad \neg e(a,s(y))$	Universal Elmination: 2
15.	$\quad \neg e(s(y),a) \Rightarrow \neg e(a,s(y))$	Implication Introduction: 13, 14
16.	$\quad \neg\neg e(s(y),a)$	Negation Introduction: 12, 15
17.	$\quad e(s(y),a)$	Negation Elimination: 16
18.	$e(a,s(y)) \Rightarrow e(s(y),a)$	Implication Introduction: 9, 17
19.	$(e(a,y) \Rightarrow e(y,a)) \Rightarrow (e(a,s(y)) \Rightarrow e(s(y),a))$	Implication Introduction: 8, 18
20.	$\forall y.((e(a,y) \Rightarrow e(y,a)) \Rightarrow (e(a,s(y)) \Rightarrow e(s(y),a)))$	Universal Introduction: 19
21.	$\forall y.(e(a,y) \Rightarrow e(y,a))$	Induction: 7, 20

That's a lot of work just to prove the base case of the outer induction. The inductive case of the outer induction is even more complex, and it is easy to make mistakes. The trick to avoiding these mistakes is to be methodical.

In order to prove the inductive case for the outer induction, we assume the inductive hypothesis $\forall y.(e(x,y) => e(y,x))$; and we then prove the inductive conclusion $\forall y.(e(s(x),y) => e(y,s(x)))$. We prove this by induction on the variable y.

We start by proving the base case for this inner induction. We start with the inductive hypothesis. We then assume the antecedent of the base case.

22.	$\forall y.(e(x,y) => e(y,x))$	Assumption
23.	$\quad e(s(x),a)$	Assumption
24.	$\quad\quad \neg e(a,s(x))$	Assumption
25.	$\quad\quad e(s(x),a)$	Reiteration: 23
26.	$\quad \neg e(a,s(x))) \Rightarrow e(s(x),a)$	Implication Introduction: 24, 25
27.	$\quad\quad \neg e(a,s(x))$	Assumption
28.	$\quad\quad \neg e(s(x),a)$	Universal Elimination: 3
29.	$\quad \neg e(a,s(x))) \Rightarrow \neg e(s(x),a)$	Implication Introduction: 27, 28
30.	$\quad \neg\neg e(a,s(x)))$	Negation Introduction: 26, 29
31.	$\quad e(a,s(x)))$	Negation Elimination: 30
32.	$e(s(x),a) \Rightarrow e(a,s(x)))$	Implication Introduction: 23, 31

Next, we work on the inductive case for the second inner induction. We start by assuming the inductive hypothesis. We then assume the antecedent of the inductive conclusion.

33.	$e(s(x),y) \Rightarrow e(y,s(x))$	Assumption
34.	$e(s(x),s(y))$	Assumption
35.	$\forall y.(e(s(x),s(y)) \Rightarrow e(x,y))$	Universal Elimination: 5
36.	$e(s(x),s(y)) \Rightarrow e(x,y)$	Universal Elimination: 35
37.	$e(x,y)$	Implication Elimination: 36, 34
38.	$e(x,y) => e(y,x)$	Universal Elimination: 22
39.	$e(y,x)$	Implication Elimination: 38, 37
40.	$\forall y.(e(y,x) \Rightarrow e(s(y),s(x)))$	Universal Elimination: 4
41.	$e(y,x) \Rightarrow e(s(y),s(x))$	Universal Elimination: 40
42.	$e(s(y),s(x))$	Implication Elimination: 41, 39
43.	$e(s(x),s(y)) \Rightarrow e(s(y),s(x))$	Implication Introduction: 34, 42
44.	$(e(s(x),y) \Rightarrow e(y,s(x))) \Rightarrow (e(s(x),s(y)) \Rightarrow e(s(y),s(x)))$	Implication Introduction: 33, 43
45.	$\forall y.((e(s(x),y) \Rightarrow e(y,s(x))) \Rightarrow (e(s(x),s(y)) \Rightarrow e(s(y),s(x))))$	Universal Introduction: 44

From the results on lines 32 and 45, we can conclude the inductive case for the outer induction.

46.	$\forall y.(e(s(x),y) \Rightarrow e(y,s(x)))$	Induction: 32, 45
47.	$\forall y.(e(x,y) => e(y,x)) \Rightarrow \forall y.(e(s(x),y) \Rightarrow e(y,s(x)))$	Implication Introduction: 22, 46
48.	$\forall x.(\forall y.(e(x,y) => e(y,x)) \Rightarrow \forall y.(e(s(x),y) \Rightarrow e(y,s(x))))$	Universal Introduction: 47

Finally, from the base case for the outer induction and this inductive case, we can conclude our overall result.

49.	$\forall x.\forall y.(e(x,y) \Rightarrow e(y,x))$	Induction: 21, 48

As this proof illustrates, the technique of using induction within induction works just fine. Unfortunately, it is tedious and error-prone. for this reason, many people prefer to use specialized forms of multidimensional induction.

11.7 Embedded Induction

In all of the examples in this chapter thus far, induction is used to prove the overall result. While this approach works nicely in many cases, it is not always successful. In some cases, it is easier to use induction on parts of a problem or to prove alternative conclusions and then use these intermediate results to derive the overall conclusions (using inductive or non-inductive methods).

As an example, consider a world characterized by a single object constant a, a single unary function constant s, and a single unary relation constant p. Assume we have the set of axioms shown below.

$$\forall x.(p(x) \Rightarrow p(s(s(x))))$$
$$p(a)$$
$$p(s(a))$$

A little thought reveals that these axioms logically entail the universal conclusion $\forall x.p(x)$. Unfortunately, we cannot derive this conclusion directly using Linear Induction. The base case is easy enough. And, from $p(x)$ we can easily derive $p(s(s(x)))$. However, it is not so easy to derive $p(s(x))$, which is what we need for the inductive case of Linear Induction.

The good news is that we can succeed in cases like this by proving a slightly more complicated intermediate conclusion and then using that conclusion to prove the result. One way to do this is shown below. In this case, we start by using Linear Induction to prove $\forall x.(p(x) \wedge p(s(x))$. The base case $p(a) \wedge p(s(a))$ is easy, since we are given the two conjuncts as axioms. The inductive case is straightforward. We assume $p(x) \wedge p(s(x))$. From this hypothesis, we use And Elimination to get $p(x)$ and $p(s(x))$. We then use Universal Elimination and Implication Elimination to derive $p(s(s(x)))$. We then conjoin these results, use Implication Introduction and Universal Introduction to get the inductive case for our induction. From the base case and the inductive case, we get our intermediate conclusion. Finally, starting with this conclusion, we use Universal Elimination, And Elimination, and Universal Introduction to get the overall result.

1.	$\forall x.(p(x) \Rightarrow p(s(s(x))))$	Premise
2.	$p(a)$	Premise
3.	$p(s(a))$	Premise
4.	$p(a) \wedge p(s(a))$	And Introduction
5.	$p(x) \wedge p(s(x))$	Assumption
6.	$p(x)$	And Elimination
7.	$p(s(x))$	And Elimination
8.	$p(x) \Rightarrow p(s(s(x)))$	Universal Elimination: 1
9.	$p(s(s(x)))$	Implication Elimination: 8, 6
10.	$p(s(x)) \wedge p(s(s(x)))$	And Introduction: 7, 9
11.	$p(x) \wedge p(s(x)) \Rightarrow p(s(x)) \wedge p(s(s(x)))$	Implication Introduction: 5, 10
12.	$\forall x.(p(x) \wedge p(s(x)) \Rightarrow p(s(x)) \wedge p(s(s(x))))$	Universal Introduction: 11
13.	$\forall x.(p(x) \wedge p(s(x)))$	Induction: 4, 12
14.	$p(x) \wedge p(s(x))$	Universal Elimination: 13
15.	$p(x)$	And Elimination: 14
16.	$p(s(x))$	And Elimination: 14
17.	$\forall x.p(x)$	Universal Introduction: 15

In this case, we are lucky that there is a useful conclusion that we can prove with standard Linear Induction. Things are not always so simple; and in some cases we need more complex forms of induction. Unfortunately, there is no finite collection of approaches to induction that covers all cases. If there were, we could build an algorithm for determining logical entailment for Herbrand Logic in all cases; and, as we discussed in Chapter 10, there is no such algorithm.

Recap

Induction is reasoning from the specific to the general. *Complete induction* is induction where the set of instances is exhaustive. *Incomplete induction* is induction where the set of instances is not exhaustive. *Linear Induction* is a type of complete induction for languages with a single object constants and a single unary function constant. *Tree Induction* is a type of complete induction for

languages with a single object constants and multiple unary function constants. *Structural Induction* is a generalization of both Linear Induction and Tree Induction that works even in the presence of multiple object constants and multiple n-ary function constants.

Exercises

Exercise 11.1: Assume a language with the object constants a and b and no function constants. Given $q(a)$ and $q(b)$, use the Fitch system with domain closure to prove $\forall x.(p(x) \Rightarrow q(x))$.

Exercise 11.2: Assume a language with the object constant a and the function constant s. Given $r(a)$, $\forall x.(p(x) \Rightarrow r(s(x)))$, $\forall x.(q(x) \Rightarrow r(s(x)))$, and $\forall x.(r(x) \Rightarrow p(x) \vee q(x))$, use the Fitch system with Linear Induction to prove $\forall x.r(x)$.

Exercise 11.3: Assume a language with object constant a and unary function constants f and g. Given $p(a)$, $\forall x.(p(x) \Rightarrow p(f(x)))$, and $\forall x.(p(f(x)) \Rightarrow p(g(x)))$, use the Fitch system with Tree Induction to prove $\forall x.p(x)$.

Exercise 11.4: Consider a language with object constants a and b, binary function constant c, and unary relation constants m and p and q. The definitions for the relations are shown below. Relation m is true of a and only a. Relation p is true of a structured object if and only if it is a linear list (as defined in Chapter 9) with a top-level element that satisfies m. Relation q is true of a structured object if and only if there is an element anywhere in the structure that satisfies m.

$$
\begin{array}{lll}
m(a) & \forall u.\forall v.(m(u) \Rightarrow p(c(u,v))) & \forall u.(m(u) \Rightarrow q(u))) \\
\neg m(b) & \forall u.\forall v.(p(v) \Rightarrow p(c(u,v))) & \forall u.\forall v.(q(u) \Rightarrow q(c(u,v))) \\
\forall u.\forall v.\neg m(c(u,v)) & \neg p(a) & \forall u.\forall v.(q(v) \Rightarrow q(c(u,v))) \\
 & \neg p(b) & \neg m(a) \Rightarrow \neg q(a) \\
 & \forall u.\forall v.(p(c(u,v)) \Rightarrow m(u) \vee p(v)) & \neg m(b) \Rightarrow \neg q(b) \\
 & & \forall u.\forall v.(q(c(u,v)) \Rightarrow q(u) \vee q(v))
\end{array}
$$

Your job is to show that any object that satisfies p also satisfies q. Starting with the preceding axioms, use Fitch with Structural Induction to prove $\forall x.(p(x) \Rightarrow q(x))$. Beware: The proof requires more than 50 steps (including the premises). The good news is that it is very similar to the proof in Section 11.5.

Exercise 11.5: Starting with the axioms for e given in Section 11.6, it is possible to prove that e is transitive, i.e. $\forall x.\forall y.\forall z.(e(x,y) \wedge e(y,z) \Rightarrow e(x,z))$. Doing this requires a three variable induction, and it is quite messy. Your job in this problem is to prove just the base case for the outermost induction, i.e. prove $\forall y.\forall z.(e(a,y) \wedge e(y,z) \Rightarrow e(a,z))$. Hint: Use the strategy illustrated in section 11.6. Extra credit: Do the full proof of transitivity.

Exercise 11.6: Consider a language with a single object constant a, a single unary function constant s, and two unary relation constants p and q. We start with the premises shown below. We know that p is true of $s(a)$ and only $s(a)$. We know that q is also true of $s(a)$, but we do not know whether it is true of anything else.

$$
\begin{array}{l}
\neg p(a) \\
p(s(a)) \\
\forall x.\neg p(s(s(x))) \\
q(s(a))
\end{array}
$$

Prove $\forall x.(p(x) \Rightarrow q(x))$. Hint: Break the problem into two parts - first prove the result for $s(x)$, and then use that intermediate conclusion to prove the overall result.