Mathematical Induction

In logic, we often want to prove that every member of an infinite set has some feature. E.g., we would like to show:

N^1: __ is a number
F^1: __ has the feature **Φ**

$(\forall x)(N^1x \supset F^1x)$

How could we do this?

One way: Show that $\sim(\forall x)(N^1x \supset F^1x)$ is inconsistent (obviously, you would need further premises).

Another way: Certain infinite sets, such as **N**, have internal structure that we can use. This use constitutes the method of *mathematical induction*.

The Principle of Mathematical Induction.
Let S be any set of natural numbers.

The principle says that

IF the following are true:

(i) $1 \in S$, AND

(ii) For any natural number *x*, if every number less than *x* is in S, then $x \in S$,

THEN it is also the case that:

(iii) Every natural number is in S; i.e.: $(\forall x)(N^1x \supset S^1x)$
i.e.: $\mathbf{N} \in S$

EXAMPLES:

The **Fundamental Theorem of Arithmetic** entails that every natural number is prime or is the product of some collection of prime numbers.

We can see that this is true at the beginning:

$1 = 1$
$2 = 2$
$3 = 3$
$4 = 2*2$
$5 = 5$
$6 = 2*3$
$7 = 7$
$8 = 2*2*2$
$9 = 3*3$
$10 = 2*5$
$11 = 11$
$12 = 2*2*3$

But how can we show this is *always* so?

Theorem: *Every natural number is prime or is the product of some collection of prime numbers.*

Proof. We will prove this claim by mathematical induction. Let

[1] P = {*x*: *x* is prime or *x* is the product of some collection of primes}.

(In other words, we are instantiating the S of the principle of induction to the set P. We want to show that every element in **N** is also an element of P.)

Base: $1 \in P$. We have seen that this is true. So we have established step (i).

Induction: We now want to show:

[2] for any natural number *x*, if all the numbers less than *x* are in P, then $x \in$ P.

We will show this by instantiating *x* to an *arbitrary* natural number *n*. (i.e., for all we know, *n* could be *any* natural number.) This means that we want to show:

[3] if every number less than *n* is in P, then $n \in$ P.

Obviously, if some numbers less than *n* are not in P, the conditional is true. So let's suppose that

[IH] every number less than *n* is in P.

(This is called the "Induction Hypothesis".) Now we must show that *n* is also in P.

We turn now to the details of the proof. There are two possibilities: *n* is prime or *n* is not prime.

Case one: *n* is prime. In this case, *n* is automatically a member of P.

Case two: *n* is not prime. In this case, there are numbers *y* and *z* such that $(y*z) = n$, where $1 < y$ & $z < n$. Note that both *y* and *z* must be less than *n* if they are to be numbers that "make" *n* composite (i.e. not prime). Since both *y* and *z* are less than *n*, by the induction hypothesis [IH], both *y* and *z* are in P. So each one of them is a prime of the product of a set of primes. So we have

$$y = a_1*a_2* \dots *a_j$$

$$z = b_1*b_2* \dots *b_k,$$

where all the *a*s and *b*s are prime numbers. (*j* and *k* are natural numbers, too; if *y* is prime, then $j = 1$ (and there is no a_2, etc.). Similarly, if *z* is prime, then $k = 1$.) But since $y * z = n$, it follows that

$$a_1*a_2* \dots *a_j*b_1*b_2* \dots *b_k, = n$$

So *n* is the product of some set of prime numbers. Thus, we have shown that if [IH] is true, then [3] is true. But *n* was arbitrary (it could have been any number), so the strategy we used would work for every number, so we have shown step (ii).

Since we have shown steps (i) and (ii) for our target set P, the principle of mathematical induction allows us to conclude that all every natural number is in P. In other words, we have just shown that every natural number is either prime or is the product of some collection of primes.

Theorem: *The sum of the first n odd numbers is* n^2.

Proof. We'll use induction again. This time our target set will be the set Q:

[4] $Q = \{x\text{: the sum of the first } x \text{ odd numbers is } x^2\}$

We want to show that every number is in Q, and we will do so by proving that (i) and (ii) of the induction principle are true for Q.

Base: $1 = 1^2$, so $1 \in Q$.

Induction: Now we want to show the induction step:

[5] for any natural number x, if every number less than x is in Q, then $x \in$ Q.

We do this by instantiating x to some arbitrary number n, and proving:

[6] if every number less than n is in Q, then $n \in$ Q.

As before, if the antecedent is false for n, the conditional is trivially true for n. So we assume the antecedent, our induction hypothesis:

[IH] every number less than n is in Q.

Now we must show that n is in Q.

Note that the nth odd number is $2n$-1. So the *(n-1)*th odd number is *2n-3.* Thus, by our induction hypothesis [IH] we may assume that:

$$(1 + 3 + 5 + . . . + (2n\text{-}5) + (2n\text{-}3)) = (n\text{-}1)^2.$$

We want to show that this number plus $2n$-1 (which is the nth odd number) is equal to n^2.

That is, we want to prove that:

$$(n\text{-}1)^2 + 2n\text{-}1 = n^2$$

i.e., that $2n\text{-}1 = n^2 - (n\text{-}1)^2$,

As $(n\text{-}1)^2 = n^2 - 2n + 1$,
We have: $2n\text{-}1 = n^2 - (n^2 - 2n + 1)$
$2n\text{-}1 = 2n\text{-}1$

So for our arbitrary n, [6] is true, and since n was arbitrary, it follows that [5] is true, and so (ii) is true. Hence by the induction principle, it follows that all the numbers are in Q, which is to say that every number n is such that n^2 is the sum of the first n odd numbers.

Mathematical Induction and PL.

Mathematical induction is a powerful device for studying the properties of logical systems.

We will practice using induction by proving a number of small theorems. We will then turn to a more interesting and slightly more involved theorem.

Consider the silly sequence T:

(T) P, (P & P), (P & (P & P)),
(P & (P & (P & P))), . . .

I.e, (T) is an infinite sequence that starts out with the statement P, and whenever α is the *n*th element in the sequence, (P & α) is the *n+1*th element in the sequence.

***Theorem**: Every statement in the sequence T is equivalent to P.*

Proof. Let

[7] E = {*x*: the *x*th element in the sequence T is equivalent to P}

We can now see that it will be enough to show that all numbers are in E.

Base: We want to show that 1 $\in$ E, and to do this, we only need to show that the first element in T is equivalent to P. But the first element in T is P, and P is trivially equivalent to itself.

Induction: As before, we want to show:

[8] for any natural number *x*, if every number less than *x* is in E, then *x* $\in$ E.

and so we pick an arbitrary number *n* and show

[9] if every number less than *n* is in E, then *n* $\in$ E.

And to show this, we get to assume the antecedent, our induction hypothesis:

[IH] every number less than *n* is in E.

Now we verify that *n* $\in$ E. For *n* to be in E, it must be that the *n*th element in T is equivalent to P. Suppose that α is the *(n-1)*th element in T. then we know that the *n*th element in T is (P & α). Moreover, by our induction hypothesis, we know that α is equivalent to P (i.e., α is true iff P is true). Clearly P is equivalent to P, and (P & α) is true iff P and α are true, which they are iff P is true, so (P & α) is true iff P is true, so it is equivalent to P. Hence, *n* $\in$ E, so [9] is true, and since *n* was arbitrary, [10] is true. So the base and induction steps are both true, so by the induction principle, we may conclude that every number is in E, which is just to say that every element in T is equivalent to P.

Exercises: Consider the following sequence:

(H) (P $\supset$ P), (P $\supset$ (P $\supset$ P)), (P $\supset$ (P $\supset$ (P $\supset$ P))) . . .

[i.e., the *n+1*th element in the sequence is $(P \supset \alpha)$, where α is the *n*th element.]
1. Show that every member of (H) is a sentence.
2. Show the every member of (H) is a tautology.

Induction on the Complexity of Sentences.

The set **N** has two important structural features that allow us to systematically prove things about all of its members:

It has a beginning: 0

There is a step-by-step method for going from this beginning to any element in the set (and this method only requires a finite number of steps): finite applications of the *successor* function---s(n) = n+1---will take you from zero to any other number.

It is no accident that formal languages have corresponding features, too. Let's look at PL:

It has a beginning: the simple statements: A, B, ... Z, A_1, ..., Z_1, ... A_n, ..., Z_n, ...

There is a finite step-by-step method for going from this beginning to any other sentence: you can get from the simple statements to any other sentence by a finite number of applications of the rules for constructing negations, conjunctions, disjunctions, and conditionals.

We can exploit this structure to use the induction principle to study PL, but first we need a couple definitions.

DEF: For any numbers *x* and *y*, **MAX[*x, y*]** = whichever of *x* and *y* is the largest number (or *x* if *x* = *y*).

DEF: The **complexity** of any given sentence α is given by the function **Cx(α)**, where this function is defined as follows:

$Cx(\alpha) = 1$, if α is a statement letter;

$Cx(\alpha) = x+1$, if $\alpha = {\sim}\beta$, for some β, and $Cx(\beta) = x$.

$Cx(\alpha) = x+1$, if there are β and γ such that α is of the form: $(\gamma \,\&\, \beta)$, $(\gamma \vee \beta)$, or $(\gamma \supset \beta)$, and $MAX[Cx(\beta), Cx(\gamma)] = x$.

Since every sentence of PL is finite in length, and is either a statement letter or is constructed from other sentences by one of the rules for forming negations, conjunctions, disjunctions, or conditionals, it follows that every sentence of PL has some number as its complexity. In quantificational terms:

P^1: __ is a sentence of PL
N^1: __ is a number

$(\forall\alpha)(P^1\alpha \supset (\exists x)(N^1x \,\&\, Cx(\alpha) = x))$[1]

Let's now show that for any sentence of PL, there is exactly one way to "build up" that sentence from the simple statements of the language, using the rules for constructing sentences of PL. (cf. pp. 94--95 of the book for the rules)

First let's get clear what we mean by "building up" a sentence

DEF: A **construction tree** for a sentence α is denoted by the function $T(\alpha)$, where this function is inductively defined as follows.

If α is a simple statement, then $T(\alpha) = \alpha$.

Now let β and γ be any sentences of PL.

If $\alpha = {\sim}\beta$, then $T(\alpha) =$

$$\begin{array}{c} {\sim}\beta \\ \\ T(\beta) \end{array}$$

If $\alpha = (\beta \,\&\, \gamma)$, then $T(\alpha) =$

$$\begin{array}{c} (\beta \,\&\, \gamma) \\ \\ T(\beta) \quad T(\gamma) \end{array}$$

If $\alpha = (\beta \supset \gamma)$, then $T(\alpha) =$

$$\begin{array}{c} (\beta \supset \gamma) \\ \\ T(\beta) \quad T(\gamma) \end{array}$$

[1] We could also express this statement in "pure" RPL. Let C^2: the complexity of __ is __. Then we have: $(\forall y)(P^1y \supset (\exists x)(N^1x \,\&\, C^2yx))$

[The case is similar for when $\alpha = (\beta \vee \gamma)$, except that we replace "$\supset$" with "$\vee$".] Nothing else is a construction tree.

Exercises: Create some construction trees of your own. For instance, what are the construction trees for:

1. $(A \supset (B \& \sim(C \& \sim D)))$
2. $(A \& (B \& (C \& D)))$
3. $((A \& B) \& (C \& D))$
4. $(((F \& \sim K) \vee \sim(B \& C)) \supset N)$

We can now show that there is only one way to build up a given sentence of PL.

Theorem: *Every sentence has exactly one construction tree.*
Proof. To show this fact, we will use the (extremely common) method of using mathematical induction on the complexity of sentences. So let

[10] C = {*x*: *every* sentence of complexity *x* has exactly one construction tree}

Base: We want to show that every sentence with complexity 1 has exactly one construction tree. To prove this universal claim, let α be any sentence such that $Cx(\alpha) = 1$. So α is a statement letter (with or without subscript). There are infinitely many such sentences, but regardless of this fact, we know that $T(\alpha) = \alpha$. So α has exactly one construction tree. So our Base step has been established.

Induction: We now want to show

[11] for any natural number *x*, if every number less than *x* is in C, then $x \in C$.

(By now this part should be very obvious!)
So we pick an arbitrary *n* and show

[12] if every number less than *n* is in C, then $n \in C$.

To show this, we assume that the antecedent is true, and this gives us our IH:

[IH] Every number less than *n* is in C.

Now we want to establish that $n \in C$. Showing that $n \in C$ is tantamount to showing that:

[13] every sentence with complexity *n* has exactly one construction tree.

To prove [13], a universal claim, we will show that it holds for an arbitrary instance. So let α be any sentence such that
Cx(α) = *n*. Now we only want to show:

[14] α has exactly one construction tree.

Obviously, if *n* = 1, then by our Base step, [14] is trivially true. So, without loss of generality, let's assume that *n* > 1. Since *n* > 1, we know, by our definition of complexity, that α must have one of the following forms, for some sentences β and γ: $\sim\beta$, $(\beta \,\&\, \gamma)$, $(\beta \vee \gamma)$, or $(\beta \supset \gamma)$. Let us consider each of these cases in turn.

Case i: $\alpha = \sim\beta$. Now, since Cx(α) = *n*, it follows that Cx(β) = *n-1*. But then by [IH], it follows that β has exactly one construction tree. By the definition of a construction tree, it follows that there is only one way to construct $\sim\beta$, so α has exactly one construction tree.

Case ii: $\alpha = (\beta \,\&\, \gamma)$. Since Cx($\alpha$) = *n*, it follows that MAX[β, γ] = *n-1*. So both β and γ have complexity less than *n*. This means, by [IH], that there is exactly one construction tree for β and exactly one for γ, too. Since there is only one way to construct β and γ respectively, there is only one way to construct $(\beta \,\&\, \gamma)$, and hence only one way to construct α.

Case iii: $\alpha = (\beta \supset \gamma)$. Since Cx($\alpha$) = *n*, it follows that MAX[β, γ] = *n-1*. So both β and γ have complexity less than *n*. This means, by [IH], that there is exactly one construction tree for β and exactly one for γ, too. Since there is only one way to construct β and γ respectively, there is only one way to construct $(\beta \supset \gamma)$, and hence only one way to construct α.

The case for $\alpha = (\beta \vee \gamma)$ is similar.

We've done all the work to prove this theorem, but let's take stock and note all that we did. Since these four cases exhaust the possible ways any sentence could be built up out of smaller sentences, regardless of what sentence of complexity *n* α is, it follows (given [IH], of course) that there is exactly one construction tree for α. In other words, we have shown [14].

α was of course an arbitrary sentence of complexity *n*, so it follows that every sentence of complexity *n* has exactly one construction tree, so, assuming that [IH] is true, we have shown [13].

We used [IH] in proving [13], so another way to say what we have shown is that if [IH] is true, [13] is true, and this is just to say that we have proven [12]. But our number *n* was arbitrary: we never made any special assumptions about it. So whatever we prove about *n* holds for every number, so our proof of [12] is really a proof of [11]. And [11] is our Induction step. We previously proved the Base step, so by the principle of mathematical induction, we can conclude that every number is in C, and this is to say no more and no less than for any complexity level, every sentence of that level has exactly one construction tree. Since every sentence has some complexity level (which follows immediately from the definitions of sentences and complexity), it follows that every sentence has exactly one construction tree.

Exercises:
1. Prove that every sentence of PL has an even number of parentheses.
2. Suppose that every sentence letter is assigned the value True. Prove that in this case, every sentence that contains no occurrences of the negation symbol is True. [*Hint:* Prove that every sentence either contains a negation symbol or is True.]

We will now raise a question about PL that is not obvious. After raising this question, we will make it mathematically precise. We will then supply a rigorous proof of the answer.

Suppose that instead of developing PL, we used a much weaker language, SL. SL is just like PL, except that SL does not contain the symbols ⊃ and ~. So the only sentences of SL are simple sentences, and conjunctions and disjunctions (and conjunctions of disjunctions and conjunctions, and disjunctions of conjunctions, etc.) Nevertheless, there are infinitely many sentences of SL of every complexity. (If you need another exercise, prove this last claim.) Intuitively speaking, is anything missing from SL? Yes, of course: SL cannot express negation. That is, we can't express what e.g., PL expresses by ~P. [The proof of this is an exercise; cf. below.]

So we see that PL is, intuitively speaking, *stronger*, or more expressive, than SL. But now we should ask: Could there be languages of propositional logic that are stronger than PL, in the way that PL is stronger than SL? PL is stronger than SL because there is no sentence α of SL (of *any* complexity) with the truth table:

P	α
T	F
F	T

Our question is: Is there some truth table for which there is no sentence α such that (the main connective of) α yields the correct values of the truth table for each row?

P_1 P_2 . . . P_n	α
T T T	X_1
F T T	X_2

F F F	X_{2^n}

If there is, then maybe we should expand PL to include, say, some new *n*-ary logical connective that has exactly the truth conditions given above, which no sentence of PL expresses.

Part of answering the intuitive question about whether PL is as strong as can be is turning the intutive question into a mathematical question. We have made some progress towards this, but we need to get a bit clearer.

Truth tables for individual sentences represent functions: Put in an *n*-ary sequence of Ts and Fs (i.e. a row of a truth table with *n* distinct types of sentence letters), and the truth table tells you what the output of the function is (i.e., the truth value under the main connective of the sentence).

The most expressive language of propositional logic would express all these functions. In other words, for every such function (i.e., the truth table schema above minus the Ps and α), there would be a sentence α of that language with exactly *n* distinct types of sentence letters that yielded the values of the function. Let's make these ideas "official" with some definitions.

DEF: For any number *n*, an **n-ary boolean function** is a collection of all the *n*-ary sequences of Ts and Fs (all 2^n of them), such that each such sequence is followed by either a T or an F. (These boldfaced Ts and Fs are the output of the function given the initial *n*-ary sequence of Ts and Fs.)

DEF: A sentence α **represents** a given *n*-ary boolean function *f* iff the truth table for α perfectly describes *f*. In other words, α represents *f* iff α contains exactly *n* distinct types of statement letters, and at any given row of the truth table for α, where the input is X_1, ..., X_n (where each X is either a T or an F), α is true iff $f(X_1, ..., X_n) = T$

Yet a third way to say that α represents *f* is to say that *f* is defined as some truth table where there are *n* distinct statement letters, and some column of Ts and Fs. In this case α represents *f* iff this is the truth table for α.

So for instance, can you find some α that has the following truth table (i.e., represents the following 3-ary boolean function?

P_1 P_2 P_3	α

T T T	T
F T T	T
T F T	F
F F T	T
T T F	T
F T F	T
T F F	F
F F F	T

DEF: A language L is **adequate** iff for every *n-ary* boolean function *f*, there is some sentence α of L which represents *f*.

We have seen that SL is not adequate, because there is a 1-ary boolean function that it does not represent, namely the one described in the first truth table above.

If every *n*-ary boolean function could be represented, for every *n*, then the propositional language is as expressive as can be. (Of course, we could change the game by adding further truth values, or including infinite sequences of Ts and Fs, but those are different stories!)

Let us now show that PL is indeed as strong as can be.

Theorem: *PL is adequate.*

Proof. As you might suspect, we will prove this claim by mathematical induction, but we will not do an induction on the complexity of sentences of PL. Rather, our induction will will be on the *n* of the *n*-ary boolean functions. So for starters let's define the set B:

B = {*x*: for every *x*-ary boolean function, there is some sentence of PL that represents it}

(Remember *x* and *n* are just place-holders for numbers, so switching them in these contexts is okay.)

Base: Now we want to show that every 1-ary boolean function is represented by some sentence of PL. What 1-ary functions are there? A bit of thought will show that there are four of them, described by the following truth tables:

P	α_1
T	T
F	T

P	α_2
T	T
F	F

P	α_3
T	F
F	T

P	α_4
T	F
F	F

[A fact, which you can prove, although it takes some thought, is that for every *n*, there are exactly $2^{(2^n)}$ *n*-ary boolean functions.]

Given that these are are all the 1-ary boolean functions, can we find sentences that represent them? Sure. Let:

$\alpha_1 = (P \vee {\sim}P)$
$\alpha_2 = P$ (or $(P \,\&\, P)$ if you like)
$\alpha_3 = {\sim}P$
$\alpha_4 = (P \,\&\, {\sim}P)$

You know by now that each of these sentences has the corresponding truth table given above. So every 1-ary boolean function is represented in PL. So $1 \in B$.

Induction: This should be familiar. We want to show [15] below, and we will do this by choosing an arbitrary instance of *x*, and showing [16]:

[15] for any natural number *x*, if all the numbers less than *x* are in B, then $x \in B$.

[16] if all the numbers less than *n* are in B, then $n \in B$.

Of course, to show [16], we assume the antecedent:

[IH] All the numbers less than *n* are in B.

Now we want to show that $n \in B$. In other words, we want to show that no matter what values (T or F) we assign to the various Xs in the truth table schema below, there is sure to be some sentence α that fits that truth table:

(Input side) $P_1\ P_2\ .\ .\ .\ P_n$	(Output side) α
T T ... T	X_1
F T ... T	X_2

F F ... F	$X_{2^\wedge n}$

[Remember also that at this point we are assuming some definite value for *n*; however, in order for *n* to be at the same time arbitrary, we can't know what that definite value is.]

At this point, pick some definite assignment of Ts and Fs to all the Xs.

The real trick to showing that we can find some α, regardless of what values the Xs are given, lies in observing the structure of the input side (the *n*-ary sequences of Ts and Fs) of the truth table. If $n = 1$, then by our Base step, we are done, so without loss of generality, let's assume $n > 1$. This means that we created the input side of the truth table by forming a copy of all the *(n-1)*-ary sequences, and then placing a T at the end of them. Then below that we form another copy of all the *(n-1)*-ary sequences, and then we place an F after each one of those.

Let's start thinking of the truth table as cut in half, the top half, where every *(n-1)*-ary sequence is followed by a T, and the bottom half, where every *(n-1)*-ary sequence is followed by a F.

Now let's think about the top half of the truth table. It has $2^{(n-1)}$ rows, and after each row, under the α, there is a T or an F. If we ignore the T that follows each of the *(n-1)*-ary sequences in the top half, then what we have is a description of an *(n-1)*-ary boolean function. (That is, ignore the last T at the end of every input sequence in the top half. If you do this, then the top half describes an (*n-1*)-ary boolean function.) By [IH], we know that (regardless of what values were chosen for the Xs), there is *some* sentence β of PL that represents that *(n-1)*-ary boolean function.

Notice, too, that the same thing holds over on the bottom half. If we ignore the F that follows each *(n-1)*-ary sequence on the input side, we get a description of some *(n-1)*-ary boolean function. Similarly, by [IH], we know that there is *some* sentence γ of PL that represents that *(n-1)*-ary boolean function.

Notice, too, that in representing their respective *(n-1)*-ary boolean functions, neither β nor γ makes any use of the final sentence letter, P_n. Rather it only uses $P_1, . . . P_{n-1}$.

So what we would like to do is construct a sentence that says:

"If P_n is true, then I will behave like β, and if P_n is false, then I will behave like γ."

But of course we can easily say that. Thus, our sentence α in this case will be:

[17] $((P_n \supset \beta) \& (\sim P_n \supset \gamma))$

Let us verify that this works. Suppose we pick at random any row of the *n*-ary truth table given above (once we have fixed values for the Xs). The last element in the *n*-ary sequence of Ts and Fs (i.e. the element corresponding to P_n) on the input side of the truth table will be either a T or an F. if it is a T, then the second conjunct of [17] will be trivially true, because the second conjunct is $(\sim P_n \supset \gamma)$, and so the antecedent of this conditional will be false, and so the conditional will be true. Similarly, at every row where P_n is false, the first conjunct of [17] will be true. For definiteness, let's suppose that we've picked some row where P_n is true. (a corresponding account will work in the case where P_n is false).

Assuming we've picked a row where P_n is T, the second conjunct of [17] is true, and [17] itself will be true if and only if $(P_n \supset \beta)$ is also true. Since we are assuming P_n is true, the sentence will be true if and only if β is true (you can convince yourself of these last two claims by consulting the truth table for "$\supset$"). But what we know about β is that at any given row in the top half, the truth value for β is identical to the truth value given at that row for that choice of value for the X at that row. That is, the truth values of β exactly mirror the truth values of the given *n*-ary function (i.e., the given choice of values of X). So in our present case, β will be true at that row iff the *n*-ary boolean function yields the value true for that input. This means that $(P_n \supset \beta)$ is true in the top half iff the *n*-ary boolean function yields the value true for that input. And this means that [17] is true iff the function yields true for that input.

The choice of rows was of course random, so we know that the result holds for every row in the top half. A very similar argument using the other conjunct $(\sim P_n \supset \gamma)$ will show that the corresponding result holds for every row in the bottom half.

So for every input, our sentence in [17] is true at that (corresponding) row iff the value of the function is T. So sentence [17] represents the given *n*-ary boolean function.

The choice of values for the Xs (which distinguishes each *n*-ary boolean function from every other *n*-ary boolean function) was obviously arbitrary. So for any *n*-ary boolean function, we can find some sentence α that represents it.

And thus we have shown that, given the [IH], it follows that $n \in B$.

Hence, have shown [16], and since our choice of *n* was arbitrary, we have shown [15].

[15] was the induction step, and we also proved the base step, so by the principle of mathematical induction, we can conclude that every number is in B. So for any number *n*, it follows that for any *n*-ary boolean function, there is some sentence α of PL that represents it. Hence, by definition, PL is adequate.

Exercises:

1. Prove that negation cannot be expressed in SL. I.e., prove that there is no sentence of SL that is equivalent to ~P.

2. Let SSL be just like SL except that the conjunction symbol is not a part of SSL. So the only complex statements of SSL are disjunctions. Prove that for any sentence α of SSL, if α contains *n* tokens of simple sentences, then there are exactly *n* open branches at the bottom of the tree for α. (You may assume there is exactly one tree for α.)

3. Prove that in SL, every truth tree is open.

4. Prove that every truth table for any sentence of PL has an even number of rows (you don't need to appeal to the complexity of sentences. Rather run the induction on the number of simple sentence types present in the matrix of the truth table).

Kent Johnson, 2000