Incomplete Induction

Definition

\[ f(1) = 1 \]
\[ f(x+1) = f(x) + 2x + 1 \]

Data

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
<th>( x^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1^2</td>
</tr>
<tr>
<td>1 + 3</td>
<td>4</td>
<td>2^2</td>
</tr>
<tr>
<td>1 + 3 + 5</td>
<td>9</td>
<td>3^2</td>
</tr>
<tr>
<td>1 + 3 + 5 + 7</td>
<td>16</td>
<td>4^2</td>
</tr>
<tr>
<td>1 + 3 + 5 + 7 + 9</td>
<td>25</td>
<td>5^2</td>
</tr>
</tbody>
</table>

Conjecture

\[ f(x) = x^2 \]

*In this case, the answer is correct. Lucky Guess.*
Not So Lucky Guess

Data:

\[ 2^{2^1} + 1 = 2^{2^1} = 5 \]
\[ 2^{2^2} + 1 = 2^{4+1} = 17 \]
\[ 2^{2^3} + 1 = 2^{8+1} = 257 \]
\[ 2^{2^4} + 1 = 2^{16+1} = 65537 \]

“Theorem” by Fermat (1601-1665):

\[ \text{prime}(2^{2^n} + 1) \]

Fact discovered (mercifully) after his death:

\[ 2^{2^5} + 1 = 4,294,967,297 = 641 \times 6,700,417 \]

Oops.

---

Mathematical Induction

Data

\[ 1 = 1 = 1^2 \]
\[ 1 + 3 = 4 = 2^2 \]
\[ 1 + 3 + 5 = 9 = 3^2 \]
\[ 1 + 3 + 5 + 7 = 16 = 4^2 \]

Base Case: Prove for \( n = 1 \)

\[ f(1) = 1^2 \]

Inductive Case: Assume true for \( x \); prove for \( x + 1 \)

\[ f(x + 1) = f(x) + 2 \times x + 1 \]
\[ f(x + 1) = x^2 + 2 \times x + 1 \]
\[ f(x + 1) = (x + 1)^2 \]

Jules Henri Poincare (1854-1912) credited with invention.
Outline

Linear Induction
Input: successor function on individuals
Output: universally quantified conclusions

Structural Induction
Input: constructor function for structures
Output: universally quantified conclusions

Linear Induction

Linearly Structured World:

\[
\bullet \rightarrow \bullet \rightarrow \bullet \\
e \quad f(e) \quad f(f(e))
\]

In other words, there is a distinguished base element and there is a successor function, which, starting at the base element, enumerates all elements in the universe of discourse.

Base element: \(e\)
Successor function: \(f\)
Linear Induction Schema

If a property holds of the base element and if it holds of a successor whenever it holds of an element, we would like to assert that it holds of all elements in the universe of discourse.

\[ \phi[e] \land \forall x. (\phi[x] \implies \phi[f(x)]) \implies \forall x. \phi[x] \]

Base case: \(\phi[e]\)
Inductive case: \(\forall x. (\phi[x] \implies \phi[f(x)])\)
  Inductive antecedent: \(\phi[x]\)
  Inductive consequent: \(\phi[f(x)]\)
Conclusion: \(\forall x. \phi[x]\)

Analogy to students in class passing on messages

Arithmetic Examples

Object constant: 0
Unary function constant: \(s\) \((+1)\)
Unary relation constants: \(p\), even, odd

Induction Schema:
\[ \phi[0] \land \forall x. (\phi[x] \implies \phi[s(x)]) \implies \forall x. \phi[x] \]

Instances of Induction Schema:
\[ p(0) \land \forall x. (p(x) \implies p(s(x))) \implies \forall x. p(x) \]
\[ even(0) \land \forall x. (even(x) \implies even(s(x))) \implies \forall x. even(x) \]
\[ (even(0) \lor odd(0)) \land \forall x. (even(x) \lor odd(x) \implies even(s(x)) \lor odd(s(x))) \implies \forall x. (even(x) \lor odd(x)) \]
Arithmetic Problem

Object constant: 0
Unary function constant: s (+1)
Unary relation constants: p

Axioms
\[ p(0) \]
\[ \forall x. (p(x) \Rightarrow p(s(x))) \]

Goal:
\[ \forall x. p(x) \]

Induction Schema:
\[ p(0) \land \forall x. (p(x) \Rightarrow p(s(x))) \Rightarrow \forall x. p(x) \]

Clausal Form
\[ p(0) \land \forall x. (p(x) \Rightarrow p(s(x))) \Rightarrow \forall x. p(x) \]
I: \[ \neg (p(0) \land \forall x. (\neg p(x) \lor p(s(x)))) \lor \forall x. p(x) \]
N: \[ \neg p(0) \lor \neg \forall x. (\neg p(x) \land p(s(x))) \lor \forall x. p(x) \]
\[ \neg (\neg p(0) \lor \exists x. (\neg p(x) \land p(s(x)))) \lor \forall x. p(x) \]
\[ \neg (\neg p(0) \lor \exists x. (p(x) \land \neg p(s(x)))) \lor \forall x. p(x) \]
S:
E: \[ \neg p(0) \lor (p(a) \land \neg p(s(a))) \lor \forall x. p(x) \]
A: \[ \neg p(0) \lor (p(a) \land \neg p(s(a))) \lor \forall x. p(x) \]
D: \[ \neg p(0) \lor p(a) \lor p(x) \]
\[ \neg p(0) \lor \neg p(s(a)) \lor \forall x. p(x) \]
O: \{ \neg p(0), p(a), p(x) \}
\{ \neg p(0), \neg p(s(a)), p(x) \}
Resolution Proof

1. \( \{ p(0) \} \)  
   Premise
2. \( \{ \neg p(x), p(s(x)) \} \)  
   Premise
3. \( \{ \neg p(0), p(a), p(x) \} \)  
   Induction
4. \( \{ \neg p(0), \neg p(s(a)), p(x) \} \)  
   Induction
5. \( \{ \neg p(c) \} \)  
   Goal
6. \( \{ p(a), p(x) \} \)  
   1,3
7. \( \{ \neg p(s(a)), p(x) \} \)  
   1,4
8. \( \{ p(s(a)), p(x) \} \)  
   2,6
9. \( \{ p(x) \} \)  
   7,8
10. \( \{ \} \)  
    5,9

Arithmetic Problem

Object constant: 0
Unary function constant: \( s \)
Binary relation constants: even, odd

Axioms

\[
\begin{align*}
even(0) \\
\forall x. (\even(x) \Rightarrow \odd(s(x))) \\
\forall x. (\odd(x) \Rightarrow \even(s(x)))
\end{align*}
\]

Goal:

\( \forall x. (\even(x) \lor \odd(x)) \)

Induction Schema:

\[
\begin{align*}
& (\even(0) \lor \odd(0)) \\
\land \forall x. (\even(x) \land \odd(x) \Rightarrow \even(s(x)) \lor \odd(s(x))) \\
\Rightarrow \forall x. (\even(x) \lor \odd(x))
\end{align*}
\]
Clausal Form

\[(\text{even}(0) \lor \text{odd}(0)) \land \forall x. (\text{even}(x) \lor \text{odd}(x) \Rightarrow \text{even}(s(x)) \lor \text{odd}(s(x))) \Rightarrow \forall x. (\text{even}(x) \lor \text{odd}(x))\]

\{-\text{even}(0), \text{even}(a), \text{odd}(a), \text{even}(x), \text{odd}(x)\}
\{-\text{odd}(0), \text{even}(a), \text{odd}(a), \text{even}(x), \text{odd}(x)\}
\{-\text{even}(0), \neg \text{even}(s(a)), \text{even}(x), \text{odd}(x)\}
\{-\text{even}(0), \neg \text{odd}(s(a)), \text{even}(x), \text{odd}(x)\}
\{-\text{odd}(0), \neg \text{even}(s(a)), \text{even}(x), \text{odd}(x)\}
\{-\text{odd}(0), \neg \text{odd}(s(a)), \text{even}(x), \text{odd}(x)\}

Induction and Resolution

Understandability
Okay for computers
Terrible for humans

Generality:
Induction is an axiom schema.
Resolution works on sentences.
Linear Induction Method

Using the Induction schema to prove a universally quantified formula.

(1) Base Case. Prove the base case.

(2) Inductive Case. Assume ground version of induction antecedent (induction hypothesis) and prove corresponding version of induction consequent.

If successful, the universally quantified conclusion holds. Why? Deduction Theorem.

Even and Odd

Axioms

\begin{align*}
even(0) \\
even(x) & \Rightarrow odd(s(x)) \\
odd(x) & \Rightarrow even(s(x))
\end{align*}

Desired Conclusion:

\[ \forall x. (even(x) \lor odd(x)) \]

Induction Axiom:

\begin{align*}
(even(0) \lor odd(0)) \\
\land \forall x. (even(x) \lor odd(x) \Rightarrow even(s(x)) \lor odd(s(x))) \\
\Rightarrow \forall x. (even(x) \lor odd(x))
\end{align*}
Even and Odd Problem

1. \{even(0)\}  Premise
2. \{\neg \text{even}(x), \text{odd}(s(x))\}  Premise
3. \{\neg \text{odd}(x), \text{even}(s(x))\}  Premise
4. \{\text{even}(a), \text{odd}(a)\}  Hypothesis
5. \{\neg \text{even}(s(a))\}  Goal
6. \{\neg \text{odd}(s(a))\}  Goal
7. \{\neg \text{odd}(a)\}  2,6
8. \{\neg \text{even}(a)\}  3,5
9. \{\text{even}(a)\}  4,7
10. \{}  8,9

Binary Addition

Object constant: 0
Unary function constant: \( s \)
Binary function constant: \( + \)

Chain Axioms:
\[ s(x) = s(y) \implies x = y \]

Addition Axioms:
\[ x + 0 = x \]
\[ x + s(y) = s(x + y) \]
\[ s(x) + y = s(x + y) \]
Binary Addition Problem I

Question: \[\forall x. 0 + x = x\]

Axioms:
- \[s(x) = s(y) \Rightarrow x = y\]
- \[x + 0 = x\]
- \[x + s(y) = s(x + y)\]
- \[s(x) + y = s(x + y)\]

Goal: \[\forall x. 0 + x = x\]

Induction Axiom:
\[0 + 0 = 0 \land \forall x. (0 + x = x \Rightarrow 0 + s(x) = s(x)) \Rightarrow \forall x. 0 + x = x\]
Binary Addition Problem II

Axioms:
\[ s(x) = s(y) \Rightarrow x = y \]
\[ x + 0 = x \]
\[ x + s(y) = s(x + y) \]
\[ s(x) + y = s(x + y) \]

Question:
\[ \forall x. x + y = y + x \]

Induction Axiom:
\[ 0 + y = y + 0 \land \forall x. (x + y = y + x) \Rightarrow s(x) + y = y + s(x) \Rightarrow \forall x. x + y = y + x \]

Binary Addition Problem II

1. \{\neg s(x) = s(y), x = y\} Axiom
2. \{x + 0 = x\} Axiom
3. \{x + s(y) = s(x + y)\} Axiom
4. \{s(x) + y = s(x + y)\} Axiom
5. \{a + y = y + a\} Induction
6. \{x = x\} Equality
7. \{s(a) + y \neq y + s(a)\} Goal
8. \{s(a + y) \neq y + s(a)\} 4, 7
9. \{s(y + a) \neq y + s(a)\} 5, 8
10. \{y + s(a) \neq y + s(a)\} 3, 9
11. \{\} 6, 10
Intuition for Structural Induction

Binary Tree Representation

Tree:

Representation as a term:
\[ \text{pair}(\text{pair}(a,b),\text{pair}(\text{pair}(c,d),\text{pair}(e,f))) \]

Infix version:
\[ (a*b)*((c*d)*(e*f)) \]
Induction and Resolution

Base elements: $a, b$

Successor function: $*$

Induction Schema:

$\phi[a] \land \phi[b] \land \forall x. \forall y. (\phi[x] \land \phi[y] \Rightarrow \phi[x*y]) \Rightarrow \forall x. \phi[x]$
Reverse Problem

Desired Conclusion:

\[ \text{rev}(\text{rev}(x)) = x \]

Axioms:

\[ \text{rev}(a) = a \]
\[ \text{rev}(b) = b \]
\[ \text{rev}(x * y) = \text{rev}(y) * \text{rev}(x) \]

Induction Axiom:

\[ \text{rev}(\text{rev}(a)) = a \land \text{rev}(\text{rev}(b)) = b \]
\[ \forall x. \forall y. (\text{rev}(\text{rev}(x)) = x \land \text{rev}(\text{rev}(y)) = y \Rightarrow \text{rev}(\text{rev}(x * y)) = x * y) \]
\[ \Rightarrow \forall x. \text{rev}(\text{rev}(x)) = x \]

---

Reverse Problem

1. \{ \text{rev}(a) = a \}  
   Axiom
2. \{ \text{rev}(b) = b \}  
   Axiom
3. \{ \text{rev}(x * y) = \text{rev}(y) * \text{rev}(x) \}  
   Axiom
4. \{ \text{rev}(\text{rev}(c)) = c \}  
   Induction
5. \{ \text{rev}(\text{rev}(d)) = d \}  
   Induction
6. \{ x = x \}  
   Equality
7. \{ \text{rev}(\text{rev}(c * d)) \neq c * d \}  
   Goal
8. \{ \text{rev}(\text{rev}(d) * \text{rev}(c)) \neq c * d \}  
   3, 7
9. \{ \text{rev}(\text{rev}(c)) * \text{rev}(\text{rev}(d)) \neq c * d \}  
   3, 8
10. \{ c * \text{rev}(\text{rev}(d)) \neq c * d \}  
    4, 9
11. \{ c * d \neq c * d \}  
    5, 10
12. \{ \}  
    6, 11
Metalevel Logic

Basic idea: represent expressions in Propositional Logic as terms in Relational Logic, write Relational sentences to define basic concepts of Propositional Logic, prove metatheorems.

NB: We can extend to Relational Logic as well. The formalization is messier, and some nasty problems need to be handled (notably paradoxes).

Syntactic Metavocabulary

Object Constants (propositions)

\[ p, q, r \]

Function constants

\[ \text{not}(p) \quad \text{implies}(p,q) \]
\[ \text{and}(p,q) \quad \text{impliedby}(p,q) \]
\[ \text{or}(p,q) \quad \text{iff}(p,q) \]

Relation Constants

\[ \text{proposition}(p) \]
\[ \text{sentence}(\text{and}(p,q)) \]
\[ \text{proves}(\text{and}(p,q), \text{or}(p,q)) \]
Syntactic Metadefinitions

\[
\begin{align*}
\text{negation}(\text{not}(x)) & \iff \text{sentence}(x) \\
\text{conjunction}(\text{and}(x,y)) & \iff \text{sentence}(x) \land \text{sentence}(y) \\
\text{disjunction}(\text{or}(x,y)) & \iff \text{sentence}(x) \lor \text{sentence}(y) \\
\text{implication}(\text{implies}(x,y)) & \iff \text{sentence}(x) \land \text{sentence}(y) \\
\text{reduction}(\text{impliedby}(x,y)) & \iff \text{sentence}(x) \land \text{sentence}(y) \\
\text{equivalence}(\text{iff}(x,y)) & \iff \text{sentence}(x) \land \text{sentence}(y)
\end{align*}
\]

\[
\begin{align*}
\text{sentence}(x) & \iff \\
& \quad \text{proposition}(x) \lor \text{negation}(x) \lor \\
& \quad \text{conjunction}(x) \lor \text{disjunction}(x) \lor \\
& \quad \text{implication}(x) \lor \text{reduction}(x) \lor \text{equivalence}(x)
\end{align*}
\]

\[
\begin{align*}
\text{proves}(x,y) & \iff \ldots
\end{align*}
\]

Semantic Metavocabulary

Object Constants (interpretations and expressions)
\[i, j, k, \ldots\]

Relation Constants
\[\begin{align*}
\text{interpretation}(i) \\
\text{satisfies}(i,p) \\
\text{valid}(\text{or}(p,\text{not}(p))) \\
\text{entails}(\text{and}(p,q), \text{or}(p,q))
\end{align*}\]
Semantic Metadefinitions

\[
\begin{align*}
satisfies(i, \text{not}(x)) & \iff \neg \text{satisfies}(i, x) \\
satisfies(i, \text{and}(x, y)) & \iff \text{satisfies}(i, x) \land \text{satisfies}(i, y) \\
satisfies(i, \text{or}(x, y)) & \iff \text{satisfies}(i, x) \lor \text{satisfies}(i, y) \\
satisfies(i, \text{implies}(x, y)) & \iff \neg \text{satisfies}(i, x) \lor \text{satisfies}(i, y) \\
satisfies(i, \text{impliesby}(x, y)) & \iff \text{satisfies}(i, x) \lor \neg \text{satisfies}(i, y) \\
satisfies(i, \text{iff}(x, y)) & \iff \text{satisfies}(i, x) \land \text{satisfies}(i, y) \lor \neg \text{satisfies}(i, x) \land \neg \text{satisfies}(i, y) \\
\end{align*}
\]

\[
\begin{align*}
\text{valid}(z) & \iff \forall x. (\text{interpretation}(x) \Rightarrow \text{satisfies}(x, z)) \\
\text{entails}(y, z) & \iff \forall x. (\text{interpretation}(x) \land \text{satisfies}(x, y) \Rightarrow \text{satisfies}(x, z))
\end{align*}
\]

Metatheorems

Validity of Axiom Scemata:

\[
\text{valid or}(x, \text{not}(x)) \iff \text{sentence}(x)
\]

Soundness:

\[
\text{proves}(x, y) \iff \text{entails}(x, y)
\]

Deduction Theorem:

\[
\text{proves} \langle \text{and}(x, y), z \rangle \iff \text{proves} \langle x, \text{implies}(y, z) \rangle
\]
Goldbach’s Conjecture

Data

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>$3 + 3$</td>
</tr>
<tr>
<td>8</td>
<td>$3 + 5$</td>
</tr>
<tr>
<td>10</td>
<td>$5 + 5$</td>
</tr>
<tr>
<td>12</td>
<td>$5 + 7$</td>
</tr>
<tr>
<td>14</td>
<td>$7 + 7$</td>
</tr>
<tr>
<td>16</td>
<td>$5 + 11$</td>
</tr>
</tbody>
</table>

Conjecture (1742):

$$\forall z.(\text{even}(z) \land z > 4 \Rightarrow \exists x.\exists y. (\text{prime}(x) \land \text{prime}(y) \land x + y = z))$$

As of late 60's, still not proved.

Single counterexample would disprove.

Summary

The key to induction is having some local way of working through space of all objects and using that to establish arbitrary formulas.

Reasoning by Induction:

- Instance of schema can be used by standard proof methods.
- Induction schema cannot be written as finite set of axioms.
- Induction method works when schema true, and it's simple.

Structural Induction useful in writing formal proofs of metatheorems.