Inductive Theorem Proving

Automated Reasoning

Petros Papapanagiotou
P.Papapanagiotou@sms.ed.ac.uk

11 October 2012
Proof Assistants:

- Formalise theories and prove properties.
- Ensure **soundness** and **correctness**.
- Interactive vs. Automated
- *Decision procedures, model elimination, rewriting, counterexamples,...*

eg.

- Interactive: Isabelle, Coq, HOL Light, HOL4, ...
- Automated: ACL2, IsaPlanner, SAT solvers, ...
Inductive datatypes are everywhere!

- Mathematics (eg. arithmetic)
- Hardware & software models
- ...

Petros Papapanagiotou
Inductive Theorem Proving
Definition (Natural Numbers)

0, Suc n
**Induction**

**Natural Numbers**

**Definition (Natural Numbers)**

\[ 0, \text{Suc } n \]

**Example**

- \[ \text{Suc } 0 = 1 \]
- \[ \text{Suc } (\text{Suc } 0) = 2 \]
- \[ \text{Suc } (\text{Suc } (\text{Suc } 0)) = 3 \]
Induction

Natural Numbers

Definition (Natural Numbers)

0, Suc n

Example

- Suc 0 = 1
- Suc (Suc 0) = 2
- Suc (Suc (Suc 0)) = 3

Induction principle

\[
P(0) \quad \forall n. P(n) \Rightarrow P(Suc \ n) \\
\forall n. P(n)
\]
Induction

Lists

Definition (Lists)

\[
[ ], h \# t
\]
Induction

Lists

Definition (Lists)

\[ [], h \neq t \]

Example

- \( 1 \# [] = [1] \)
- \( 1 \# (2 \# []) = [1, 2] \)
- \( 1 \# (2 \# (3 \# [])) = [1, 2, 3] \)
**Induction**

**Lists**

**Definition (Lists)**

$$[\ ], h \neq t$$

**Example**

- $$1 \neq [ ] = [1]$$
- $$1 \neq (2 \neq [ ]) = [1, 2]$$
- $$1 \neq (2 \neq (3 \neq [ ])) = [1, 2, 3]$$

**Induction principle**

$$\begin{align*}
P([ ]) & \\
\forall h. \forall l. \ P(l) & \Rightarrow \ P(h \neq l) \\
\forall l. \ P(l) &
\end{align*}$$
**Induction**

**Binary Partition Trees**

---

**Definition (Partition)**

Empty, Filled, Branch partition\(1\) partition\(2\)
**Definition (Partition)**

Empty, Filled, Branch partition1 partition2

**Example**

Branch Empty (Branch Filled Filled)
Induction
Binary Partition Trees

Definition (Partition)

Empty, Filled, Branch partition1 partition2

Example

Branch Empty (Branch Filled Filled)

Induction principle (partition.induct)

\[ P(\text{Empty}) \quad P(\text{Filled}) \quad \forall p1 \ p2. \ P(p1) \land P(p2) \Rightarrow P(\text{Branch } p1 \ p2) \quad \forall \text{partition. } P(\text{partition}) \]
Inductive Proofs

Generally

- Symbolic evaluation (rewriting).
  - Axioms - definitions
  - Rewrite rules
- Fertilization (use induction hypothesis).
Inductive Proofs
Simple Example: List Append

### Definition (List Append @)

1. \( \forall l. [ ] @ l = l \)
2. \( \forall h. \forall t. \forall l. (h \# t) @ l = h \# (t @ l) \)

### Example ([1; 2] @ [3] = [1; 2; 3])

\[
(1 \# (2 \# [ ])) @ (3 \# [ ]) = \\
1 \# ((2 \# [ ]) @ (3 \# [ ])) = \\
1 \# (2 \# ([ ] @ (3 \# [ ]))) = \\
1 \# (2 \# (3 \# [ ]))
\]
Inductive Proofs
Simple Example: List Append

Definition (List Append @)

1. \( \forall l. [] @ l = l \)
2. \( \forall h. \forall t. \forall l. (h \neq t) @ l = h \# (t @ l) \)

Theorem (Associativity of Append)

\( \forall k. \forall l. \forall m. k @ (l @ m) = (k @ l) @ m \)

Base Case.

\[ \begin{align*}
\vdash & \quad [ ] @ (l @ m) = ([ ] @ l) @ m \\
\hline
& \quad 1 \quad l @ m = ([ ] @ l) @ m \\
\hline
& \quad 1 \quad l @ m = l @ m \\
& \quad refl \quad true
\end{align*} \]
Inductive Proofs

Simple Example: List Append

Definition (List Append @)

1. \( \forall l. [ ] @ l = l \)

2. \( \forall h. \forall t. \forall l. (h \neq t) @ l = h \neq (t @ l) \)

Step Case.

\[ k @ (l @ m) = (k @ l) @ m \]

\[ \vdash (h \neq k) @ (l @ m) = ((h \neq k) @ l) @ m \]

\[ \iff h \neq (k @ (l @ m)) = (h \neq (k @ l)) @ m \]

\[ \iff h \neq (k @ (l @ m)) = h \neq ((k @ l) @ m) \]

\[ \iff repl \]

\[ h = h \land k @ (l @ m) = (k @ l) @ m \]

\[ \iff IH \]

\[ refl \]

\[ true \]
Inductive Proofs
Simple Example 2: Idempotence of Union

Definition (Partition Union @@)

3. Empty @@ q = q
4. Filled @@ q = Filled
5. p @@ Empty = p
6. p @@ Filled = Filled
7. (Branch l1 r1) @@ (Branch l2 r2) = Branch (l1 @@ l2) (r1 @@ r2)
Inductive Proofs
Simple Example 2: Idempotence of Union

Definition (Partition Union @@)

3. Empty @@ q = q
4. Filled @@ q = Filled
5. p @@ Empty = p
6. p @@ Filled = Filled
7. \((\text{Branch } l_1 r_1) @@ (\text{Branch } l_2 r_2) = \text{Branch } (l_1 @@ l_2) (r_1 @@ r_2)\)

Theorem (Idempotence of union)

\(\forall p. \ p @@ p = p\)
Inductive Proofs
Simple Example 2: Idempotence of Union

Definition (Partition Union @@)

3. $\text{Empty } @@ q = q$
4. $\text{Filled } @@ q = \text{Filled}$
7. $(\text{Branch } l_1 r_1) @@ (\text{Branch } l_2 r_2) = \\
\text{Branch } (l_1 @@ l_2) (r_1 @@ r_2)$

Base Case 1.

$\vdash \text{Empty } @@ \text{Empty} = \text{Empty}$
3. $\text{Empty} = \text{Empty}$
refl
$\iff true$
Inductive Proofs
Simple Example 2: Idempotence of Union

Definition (Partition Union ⊗⊗)

3. Empty ⊗⊗ q = q
4. Filled ⊗⊗ q = Filled
7. (Branch l1 r1) ⊗⊗ (Branch l2 r2) =
   Branch (l1 ⊗⊗ l2) (r1 ⊗⊗ r2)

Base Case 2.

\[
\begin{align*}
\vdash & \text{Filled } \bowtie \text{Filled } = \text{Filled} \\
& \iff \text{Filled } = \text{Filled} \\
& \iff \text{refl} \\
& \iff \text{true}
\end{align*}
\]
Inductive Proofs
Simple Example 2: Idempotence of union

Definition (Partition Union @@)

3. $\text{Empty} @@ q = q$
4. $\text{Filled} @@ q = \text{Filled}$
7. $(\text{Branch } l1 \ r1) @@ (\text{Branch } l2 \ r2) =$
   $\text{Branch} (l1 @@ l2) (r1 @@ r2)$

Step Case.

$p1 @@ p1 = p1 \land p2 @@ p2 = p2$
$\vdash (\text{Branch } p1 \ p2) @@ (\text{Branch } p1 \ p2) = \text{Branch} p1 \ p2$
$\iff \text{Branch} (p1 @@ p1) (p2 @@ p2) = \text{Branch} p1 \ p2$
$IH \iff \text{Branch} p1 \ p2 = \text{Branch} p1 \ p2$
$\iff \text{true}$
Is rewriting and fertilization enough?

No! Because:

- Incompleteness (Gödel)
- Undecidability of Halting Problem (Turing)
- Failure of Cut Elimination (Kreisel)
Inductive Proofs

Blocking Example

Definition (List Reverse \textit{rev})

8 \textit{rev} [ ] = [ ]

9 \forall h. \forall t. \textit{rev} (h \# t) = \textit{rev} t @ (h \# [ ])

Theorem (Reverse of reverse)

\forall l. \textit{rev} (\textit{rev} l) = l

Base Case.

\vdash \textit{rev} (\textit{rev} [ ]) = [ ]

\leftrightarrow \textit{rev} [ ] = [ ]

\leftrightarrow [ ] = [ ]

refl \leftrightarrow true
Inductive Proofs

Blocking Example

Definition (List Reverse \textit{rev})

8 \textit{rev}\ [[] = []

9 \forall h. \forall t. \textit{rev}\ (h \# t) = \textit{rev}\ t @ (h \# [])

Theorem (Reverse of reverse)

\forall l. \textit{rev}\ (\textit{rev}\ l) = l

Step Case.

\textit{rev}\ (\textit{rev}\ l) = l

\vdash \textit{rev}\ (\textit{rev}\ (h \# l)) = h \# l

9 \iff \textit{rev}\ (\textit{rev}\ l @ (h \# [])) = h \# l

Now what??
### Step Case.

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

\[
\vdash \text{rev (rev (h \# l))} = h \# l
\]

\[
\Leftarrow \Rightarrow \text{rev (rev } l@ (h \# [ ])) = h \# l
\]

**Now what??**

### Example (Possible Solutions)

- **Lemma:** \( \forall l. \forall m. \text{rev } (l@ m) = \text{rev } m@ \text{rev } l \)

- **Weak fertilization:**
  \[
  \Leftarrow \Rightarrow \text{rev } (\text{rev } l@ (h \# [ ])) = h \# (\text{rev } \text{rev } l)
  \]

- **Generalisation:** \( \text{rev } (l'@ (h \# [ ])) = h \# (\text{rev } l') \)
Demo
Automating Inductive Proofs

- Over 20 years of work by Boyer, Moore, Kaufmann
- The “Waterfall Model”
- Evolved into ACL2
- Used in industrial applications:
  - Hardware verification: AMD Processors
  - Software verification: Java bytecode
- Implemented for HOL88/90 by Boulton
- Reconstructed for HOL Light by Papapanagiotou
Waterfall of heuristics

1. Pour clauses recursively from the top.
2. Apply heuristics as the clauses trickle down.
   - Some get proven (evaporate).
   - Some get simplified or split $\Rightarrow$ Pour again from the top
   - Some reach the bottom.
3. Form a pool of unproven clauses.
4. Apply induction and pour base case and step case from the top.
The Waterfall Model

Waterfall of heuristics
Waterfall of heuristics
Heuristics (HOL Light version)

1. Tautology heuristic
2. Clausal form heuristic
3. Setify heuristic \((p \lor p \iff p)\)
4. Substitution heuristic (inequalities: \(x \neq a \lor P x \iff P a\))
5. Equality heuristic (fertilization)
6. Simplification heuristic (rewriting)
7. Generalization heuristic
8. Irrelevance heuristic
SUC m = m + SUC 0

Doing induction on 'm' for: SUC m = m + SUC 0

SUC 0 = 0 + SUC 0

-> HL Simplify Heuristic

Proven: |- SUC 0 = 0 + SUC 0

SUC n = n + SUC 0 --> SUC (SUC n) = SUC n + SUC 0

-> Clausal Form Heuristic

~(SUC n = n + SUC 0) \ SUC (SUC n) = SUC n + SUC 0

-> HL Simplify Heuristic

~(SUC n = n + SUC 0) \ SUC n = n + SUC 0

-> Tautology Heuristic

Proven: |- ~(SUC n = n + SUC 0) \ SUC n = n + SUC 0

val it : thm = |- SUC m = m + SUC 0

let rec waterfall_heuristics tmi =
  let rec flow_on_down rest_of_heuristics tmi =
    if (is F (fst tmi)) then (failwith "cannot prove")
    else if (rest_of_heuristics = []) then (Clause tmi)
    else try ((let (tms,f) = hd rest_of_heuristics tmi
      in if (tms = []) then (Clause_proved (f [])))
    else if ((tl tms) = []) then
      (Clause_split ([waterfall_heuristics (hd tms)]) f)
    else clause_split
      (dec_print_depth o
       map (waterfall_heuristics o proof_print_newline) o
       inc_print_depth) tms, f)
    )with Failure s -> (if (s = "cannot prove")
      then failwith s
    else (flow_on_down (tl rest_of_heuristics) tmi)
Petros Papapanagiotou

Inductive Theorem Proving

Conclusion

- **Inductive Proofs**
  - Appear very often in formal verification and automated reasoning tasks.
  - Are hard to automate.

- **So far**
  - Advanced automated provers (ACL2, IsaPlanner, etc)
  - Advanced techniques (Rippling, Decision Procedures, etc)
  - Still require fair amount of user interaction.

- **Still work on**
  - More advanced heuristics
    - Better generalization
    - Counterexample checking
    - Productive use of failure (IsaPlanner)
    - More decision procedures
    - ...
  - Termination heuristics
Questions?