Inductive Theorem Proving

Automated Reasoning

Petros Papapanagiotou

P.Papapanagiotou@sms.ed.ac.uk

11 October 2012

THE UNIVERSITY of EDINBURGH informatics

Theorem Proving

- 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, ...

Induction

- Inductive datatypes are everywhere!
  - Mathematics (eg. arithmetic)
  - Hardware & software models
  - ...

Induction

Natural Numbers

Definition (Natural Numbers)

0, *Suc n*

Induction
Natural Numbers

Definition (Natural Numbers)

0, $Suc\ n$

Example

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

Induction

Natural Numbers

Definition (Natural Numbers)

0, $Suc\ n$

Example

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

Induction principle

$$\frac{P(0) \qquad \forall n.\ P(n) \Rightarrow P(Suc\ n)}{\forall n.\ P(n)}$$

Induction
Lists

Definition (Lists)

$[\,],\ h \# t$

Induction

Lists

Definition (Lists)

$[\,]$, $h \# t$

Example

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

Induction

Lists

Definition (Lists)

$[\,],\ h \# t$

Example

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

Induction principle

$$\frac{P([\,]) \qquad \forall h. \forall l.\ P(l) \Rightarrow P(h \# l)}{\forall l.\ P(l)}$$

Induction

Binary Partition Trees

Definition (Partition)

$Empty$, $Filled$, $Branch\ partition1\ partition2$

Induction

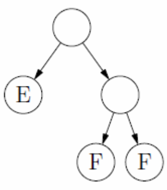
Binary Partition Trees

Definition (Partition)

Empty, *Filled*, *Branch partition*1 *partition*2

Example

Branch Empty (*Branch Filled Filled*)



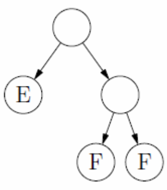
Induction
Binary Partition Trees

Definition (Partition)

$Empty, Filled, Branch\ partition1\ partition2$

Example

$Branch\ Empty\ (Branch\ Filled\ Filled)$



Induction principle (*partition.induct*)

$$\frac{P(Empty) \quad P(Filled) \quad \forall p1\ p2.\ P(p1) \wedge P(p2) \Rightarrow P(Branch\ p1\ p2)}{\forall partition.\ P(partition)}$$

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. $\forall h.\forall t.\forall l.\ (h \# t)\ @\ l \;=\; h \# (t\ @\ l)$

Theorem (Associativity of Append)

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

Base Case.

$$\begin{array}{rl} \vdash & [\,]\ @\ (l\ @\ m) \;=\; ([\,]\ @\ l)\ @\ m \\ \xleftrightarrow{1} & l\ @\ m \;=\; ([\,]\ @\ l)\ @\ m \\ \xleftrightarrow{1} & l\ @\ m \;=\; l\ @\ m \\ \xleftrightarrow{refl} & true \end{array}$$

$\square$

Inductive Proofs

Simple Example: List Append

Definition (List Append @)

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

Step Case.

$$
\begin{array}{l}
k\ @\ (l\ @\ m) \ = \ (k\ @\ l)\ @\ m \\
\quad \vdash \ (h \# k)\ @\ (l\ @\ m) \ = \ ((h \# k)\ @\ l)\ @\ m \\
\overset{2}{\Longleftrightarrow} h \# (k\ @\ (l\ @\ m)) \ = \ (h \# (k\ @\ l))\ @\ m \\
\overset{2}{\Longleftrightarrow} h \# (k\ @\ (l\ @\ m)) \ = \ h \# ((k\ @\ l)\ @\ m) \\
\overset{repl}{\Longleftrightarrow} h = h \ \wedge \ k\ @\ (l\ @\ m) \ = \ (k\ @\ l)\ @\ m \\
\overset{IH}{\Longleftrightarrow} h = h \\
\overset{refl}{\Longleftrightarrow} true
\end{array}
$$

$\square$

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. $\mathit{Empty}\ @@\ q = q$
4. $\mathit{Filled}\ @@\ q = \mathit{Filled}$
5. $p\ @@\ \mathit{Empty} = p$
6. $p\ @@\ \mathit{Filled} = \mathit{Filled}$
7. $(\mathit{Branch}\ l1\ r1)\ @@\ (\mathit{Branch}\ l2\ r2) =$
   $\mathit{Branch}\ (l1\ @@\ l2)\ (r1\ @@\ r2)$

Theorem (Idempotence of union)

$\forall p.\ p\ @@\ p = p$

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 1.

$$\vdash Empty\ @@\ Empty = Empty$$
$$\overset{3}{\iff} Empty = Empty$$
$$\overset{refl}{\iff} true \qquad \square$$

Inductive Proofs

Simple Example 2: Idempotence of Union

Definition (Partition Union @@)

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

Base Case 2.

$$\begin{array}{rl} \vdash & \mathit{Filled}\ @@\ \mathit{Filled} = \mathit{Filled} \\ \overset{4}{\Longleftrightarrow} & \mathit{Filled} = \mathit{Filled} \\ \overset{\mathit{refl}}{\Longleftrightarrow} & \mathit{true} \end{array}$$

$\square$

Inductive Proofs

Simple Example 2: Idempotence of union

Definition (Partition Union @@)

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

Step Case.

$$p1\ @@\ p1 = p1 \quad \wedge \quad p2\ @@\ p2 = p2$$
$$\vdash (Branch\ p1\ \ p2)\ @@\ (Branch\ p1\ \ p2) = Branch\ p1\ \ p2$$
$$\overset{7}{\Longleftrightarrow} Branch\ (p1\ @@\ p1)\ (p2\ @@\ p2) = Branch\ p1\ \ p2$$
$$\overset{IH}{\Longleftrightarrow} Branch\ p1\ \ p2 = Branch\ p1\ \ p2$$
$$\overset{refl}{\Longleftrightarrow} true \qquad \square$$

Automation

- Is rewriting and fertilization enough?
- No! Because:
  - Incompleteness (Gödel)
  - Undecidability of Halting Problem (Turing)
  - Failure of Cut Elimination (Kreisel)

Cut Rule

$$\frac{A, \Gamma \vdash \Delta \qquad \Gamma \vdash A}{\Gamma \vdash \Delta}$$

Inductive Proofs

Blocking Example

Definition (List Reverse *rev*)

8. $rev\ [\,] = [\,]$
9. $\forall h. \forall t. rev\ (h \# t) = rev\ t\ @\ (h \# [\,])$

Theorem (Reverse of reverse)

$\forall l. rev\ (rev\ l) = l$

Base Case.

$$\vdash rev\ (rev\ [\,]) = [\,]$$
$$\overset{8}{\Longleftrightarrow} rev\ [\,] = [\,]$$
$$\overset{8}{\Longleftrightarrow} [\,] = [\,]$$
$$\overset{refl}{\Longleftrightarrow} true$$
$\square$

Inductive Proofs

Blocking Example

Definition (List Reverse *rev*)

8. $rev\ [\,] \;=\; [\,]$
9. $\forall h. \forall t. rev\ (h \# t) \;=\; rev\ t\ @\ (h \# [\,])$

Theorem (Reverse of reverse)

$\forall l. rev\ (rev\ l) = l$

Step Case.

$$rev\ (rev\ l) = l$$
$$\vdash\ rev\ (rev\ (h \# l)) \;=\; h \# l$$
$$\stackrel{9}{\Longleftrightarrow}\ rev\ (rev\ l\ @ (h \# [\,])) \;=\; h \# l$$

Now what?? □

Inductive Proofs
Blocking Example

Step Case.

$rev\ (rev\ l) = l$

$\vdash\ rev\ (rev\ (h \# l)) \ = \ h \# l$

$\stackrel{9}{\Longleftrightarrow}\ rev\ (rev\ l\ @(h \# [\,]))\ =\ h \# l$

Now what?? $\square$

Example (Possible Solutions)

- Lemma: $\forall l. \forall m.\ rev\ (l\ @\ m)\ =\ rev\ m\ @\ rev\ l$
- Weak fertilization:
  $\stackrel{IH}{\Longleftrightarrow}\ rev\ (rev\ l\ @(h \# [\,]))\ =\ h \# (rev\ (rev\ l))$
- Generalisation: $rev\ (l'\ @\ (h \# [\,]))\ =\ h \# (rev\ l')$

Demo

Automating Inductive Proofs

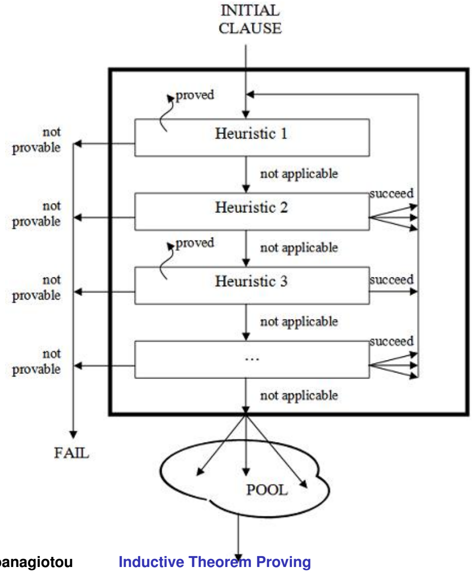
- Over 20 years of work by Boyer, Moore, Kaufmann
- The “Waterfall Model”
- Evolved into ACL2
- Used in industrial applications:
  - Hardware verification: AMD Processors
  - Software verification: Java bytecode
- Implemented for HOL88/90 by Boulton
- Reconstructed for HOL Light by Papapanagiotou

Waterfall of heuristics

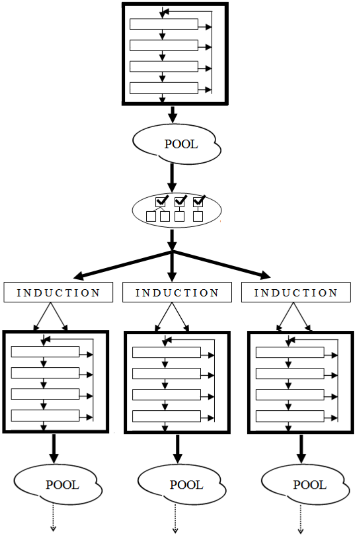
1. Pour clauses recursively from the top.
2. Apply heuristics as the clauses trickle down.
   - Some get proven (evaporate).
   - Some get simplified or split $\Rightarrow$ Pour again from the top
   - Some reach the bottom.
3. Form a pool of unproven clauses.
4. Apply induction and pour base case and 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 ($p \vee p \Leftrightarrow p$)
4. Substitution heuristic (inequalities: $x \neq a \vee P\ x \Leftrightarrow P\ a$)
5. Equality heuristic (fertilization)
6. Simplification heuristic (rewriting)
7. Generalization heuristic
8. Irrelevance heuristic

Demo

```
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)
           )
   in flow_on_down heuristics tmi;;
```

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?