An Implicational Logic for Conjecturing and Distributed Proof Attempts

Lucas Dixon

1 Nov 2007
The Issue

- **Asynchronous** and **distributed** contribution to a formalisation.

- A *common* situation:
  - **Proving a conjecture in parallel with using it:**
    e.g. Fermat’s Last theorem involves...
    * Lemma: “Elliptic Curves = Modular” can be converted to Galois Representation.
    * Theorem: Galois representation of “Elliptic Curves = Modular” proved by Iwasawa theory.
  - **Adding to existing theory libraries,**
    e.g. missing lemmas, new theorems...

- Problem: **lots of re-execution** of proof scripts.
The Meta-Logic of Theories

- A **theory** holds a set **theorems**
  (theorems are derivations of sequents: $\Gamma \vdash A$).

- There is a **meta-logic** to working with theorems, it says:
  - Theorems are given names so they can be referred to.
  - New theorems are derived using only the system’s axioms applied to old theorems.

- **How do we make a conjecture?**
  - Add a new theorem of the form: $A \vdash A$?
  - Add it as an (temporary) axiom? (Isabelle’s sorry)
  - Application of the cut rule?
Conjectures as cuts... ?

When you realise you need a conjecture $A$, use the cut rule:

\[
A, \Delta \vdash B \quad \Delta \vdash A \\
\hline
\Delta \vdash B
\]

- Conjecture never becomes a theorem in the theory.
- Can only use the conjecture on this branch of the proof.
Conjecture by dangling assumptions...?

- Leave the conjecture as dangling subgoals/assumptions wherever you plan to use it.

- To make these subgoals go away: prove the conjecture first and then apply it to every appropriate subgoals.

- Still prove the lemmas before using them:
  
  **Parallel Development**: conjecture can be proved in parallel with other proofs intend to use it (trail of FIXME comments in the file)
  
  **Script re-execution**: proving the conjecture requires re-checking all proofs after (and modifying them to use the conjecture appropriately).
Conjectures as axioms I promise to remove... ?

- What I actually do: conjectures are added as new axioms, an identical theorem can start to be proved in parallel with the use of the axiom.

- **Parallel Development:** but must remember to remove the axiom and replace it with the proved lemma.

- **Script re-execution:** once an conjecture is proved, need to re-execute everything afterwords.

- **Ugly** to have both axiom and proof attempt of conjecture, not to mention annoying to keep terms in sync.
A Logic of Conjecturing: Idea

Rephrase the rules for implication to support conjectures.

Theory: a set of results (theorems, assumptions, and conjectures) where each result as a unique name.

Result: \( x[A \vdash p : s] \)

- \( x \) = the unique name of the result.
- \( A \) = the set of result names of assumptions.
- \( p \) = the proof of this result; \( ? \) for unproved, \( \circ \) for assumed, and \( x\{g_0, \ldots, g_n\} \) for proved by \( x \) with subgoals \( g_0 \) to \( g_n \).
- \( s \) = the statement that this result makes, in some object language.
A Logic of Conjectures: Making a Theory

\[
\begin{align*}
\text{empty} & \quad \Delta & \quad \text{assume} & \quad \Delta \\
\emptyset & \quad \Delta & \quad \cup \{x[A \vdash \circ : s]\} & \quad \Delta & \quad \cup \{x[A \vdash \Box : s]\}
\end{align*}
\]

- where:
  - \(x\) is a unique name (fresh) in \(\Delta\), and
  - \(A\) is a set of assumption names that already exist in \(\Delta\).

- Uniqueness of names is an invariant of theories: no freshness conditions.
Example, part 1

**ND:**

\[ \begin{align*}
  & A \rightarrow B, B \rightarrow C \vdash A \rightarrow C \\
  & \vdash I
\end{align*} \]

**ILC:**

\[ \Delta \equiv \{^a \vdash \circ : A, ^{a_2} \vdash \circ : A, ^{ab} [^a_2 \vdash \circ : B], ^b \vdash \circ : B, ^{bc} [^b \vdash \circ : C] \} \]

\[ \Delta \cup \{^g_1 [^a, ^{ab}, ^{bc} \vdash \circ : C] \} \]

assume*  
conjecture

Lucas Dixon  
An Implicational Logic of Conjecturing (ILC)  
1 Nov 2007
A Logic of Conjecturing: Proving Things

To prove a conjecture \( x \) using a result \( y \):

\[
\Delta \cup \{^x[A \vdash ? : s]\} \quad y[B \vdash p : s] \in \Delta \quad \text{applicable}(y, x)
\]

\[
\Delta \cup \{^x[A \vdash y\{i' | i \in B - A\} : s]\} \cup \{^i'[A \cup \text{asms}(i) \vdash ? : \text{trm}(i)] \mid i \in B - A\}
\]

- where...
  - \( \text{asms}(i) \) = the assumptions of result \( i \) w.r.t. \( \Delta \).
  - \( \text{trm}(i) \) = conclusion term of result \( i \) w.r.t. \( \Delta \).
  - \( i' \) = a new name, w.r.t. \( \Delta \), generated from \( i \).
  - \( \text{applicable}(y, x) \) stops circular proofs; done efficiently by caching names.

Remark: tracking dependencies supports minimal rechecking when lemmas are modified/removed.
Example, part 2

\[ \Delta \equiv \{ a \vdash \circ : A \}, a_2 \vdash \circ : A, ab[a_2 \vdash \circ : B], b \vdash \circ : B, bc[b \vdash \circ : C] \} \]

\[ \Delta \cup \{ g_1[a, ab, bc \vdash \otimes : C] \} \]

\[ \Delta \cup \{ g_1[a, ab, bc \vdash bc \{ g_2 \} : C], g_2[a, ab, bc \vdash \otimes : B] \} \]

\[ \Delta \cup \{ g_1[a, ab, bc \vdash bc \{ g_2 \} : C], g_2[a, ab, bc \vdash ab \{ g_3 \} : B], g_3[a, ab, bc \vdash \otimes : A] \} \]

\[ \Delta \cup \{ g_1[a, ab, bc \vdash bc \{ g_2 \} : C], g_2[a, ab, bc \vdash ab \{ g_3 \} : B], g_3[a, ab, bc \vdash a : A] \} \]

assume*

conjecture

prove \( g_1 \) by \( bc \)

prove \( g_2 \) by \( ab \)

prove \( g_3 \) by \( a \)
Example, part 3

Mizar/Isar stylish:

\[
\{ \begin{array}{l}
a_2: A \vdash ab: B, \\
b: B \vdash bc: C, \\
a: A \\
\end{array} \}
\]

\[ \vdash \]

\[ \begin{array}{l}
g_1: C \text{ by } bc \text{ to } g_2 \\
g_2: B \text{ by } ab \text{ to } g_3 \\
g_3: A \text{ by } a \\
\end{array} \]
Remarks

• **ILC supports the process of conjecturing**: it does not describe the nature of conjecturing.

• **Parallel proof attempts**: conjectures can be used and proved in parallel.

• **no re-execution** is needed after proving a conjecture.

• Admissible rules can be useful: assumption $\leftrightarrow$ subgoal, theory merging.

• Implemented: ILC for propositions as 400 lines of SML. as 6000 lines in IsaPlanner for Isabelle’s intuitionistic meta-HOL.

• **Soundness/Completeness** working on proofs by translation to and from ND calculus.