

# Dependency in Refinement Logics

Jonathan Sterling

## Abstract

We propose a new generalization of LCF-style refinement logics called ***Dependent LCF***, which allows the definition of refinement rules which express a dependency between subgoals, without the use of unification variables.

Over the past four decades, there have been many incarnations of the *LCF* (Logic for Computable Functions) interface, but the one that will concern us here is its extension to *LCF with Validations* as found in Cambridge LCF, which outfits the proving activity with the synthesis of explicit evidence. The Cambridge LCF signature is as follows:

```
signature CAMBRIDGE_LCF =  
sig  
  type form  
  type thm  
  type proof = thm list  $\rightarrow$  thm  
  type goal = form list  $\otimes$  form  
  type tactic = goal  $\rightarrow$  goal list  $\otimes$  proof  
end
```

The goal type represents sequents, where *form* represents logical propositions. The LCF methodology, however, may be used to give a refinement treatment to many kinds of judgments, not just sequents; therefore, it is better to not include this in the core signature at all, and make the type of judgments abstract; for clarity, we will replace the name *thm* with *synthesis*.

```
signature LCF =  
sig  
  type judgment  
  type synthesis  
  type proof = synthesis list  $\rightarrow$  synthesis  
  type tactic = judgment  $\rightarrow$  judgment list  $\otimes$  proof  
end
```

Then, a tactic is something that refines a judgment to a list of judgments (its subgoals), and synthesizes its evidence (provided the syntheses of its subgoals). There are many different tactics which can be implemented generically over this signature; here are a few:

```

signature TACTICALS =
sig
  structure Lcf : LCF
  val ID : Lcf.tactic
  val FAIL : Lcf.tactic
  val THEN : Lcf.tactic  $\otimes$  Lcf.tactic  $\rightarrow$  Lcf.tactic
  val THENL : Lcf.tactic  $\otimes$  Lcf.tactic list  $\rightarrow$  Lcf.tactic
end

```

## 1 *Modernized LCF*: the logic of tactics

In order to study the design space for LCF refiners, we would like to give a judgmental characterization of tactic systems, which we will call ***Modernized LCF***. To begin with, note that the type of validations (**proof**) in the ML implementation is essentially a HOAS (higher-order abstract syntax) encoding of a hypothetical proof or synthesis of a judgment. With this insight in hand, we are in a position to unify the list of subgoals and the validation generated by a tactic into a single concept, namely that of a *hypothetical proof*  $E$  whose free variables are explained in a context  $\Psi$  of subgoals.

To make the preceding observations precise, we can characterize the behavior of a ***Modernized LCF*** refiner judgmentally via two forms of judgment,  $J \Vdash \tau \Rightarrow E \dashv \Psi$  and  $J \Vdash \tau \Uparrow$ , where  $J$  is a judgment of the logical theory,  $\tau$  is a tactic,  $\Psi$  is a context of judgments, representing the subgoals generated by the tactic  $\tau$ , and  $E$  is the synthesis of the judgment  $J$ , binding variables  $|\Psi|$  which represent the syntheses of the subgoals.

The meaning of  $J \Vdash \tau \Rightarrow E \dashv \Psi$  is that  $\tau$  is applicable to demonstrating the judgment  $J$ , producing synthesis  $e$  under the assumptions that the judgments in  $\Psi$  can be demonstrated. The divergence judgment  $J \Vdash \tau \Uparrow$  expresses the inapplicability of  $\tau$  to  $J$ . In practice, we will explain only the assertion conditions for one of  $J \Vdash \tau \Rightarrow E \dashv \Psi$  and  $J \Vdash \tau \Uparrow$ , and implicitly take the other to be its complement.

*Remark 1.1.* The refinement judgments  $J \Vdash \tau \Rightarrow E \dashv \Psi$  and  $J \Vdash \tau \Uparrow$  are not higher-order judgments, because the variable  $J$  ranges not over judgments of the refinement theory, but of the object theory.

Let  $\mathfrak{J}$  be the open-ended collection of judgments in our logical theory, and let  $\mathfrak{R}$  be a collection of rule names. Each rule  $R \in \mathfrak{R}$  must be interpretable as a tactic, i.e. the meaning of the assertions  $J \Vdash R \Rightarrow E \dashv \Psi$  and  $J \Vdash R \Uparrow$  must be explained for  $J \in \mathfrak{J}$ . We say  $\tau$  *tactic* in case for all object-judgments  $J \in \mathfrak{J}$ , the assertion conditions for  $J \Vdash \tau \Rightarrow E \dashv \Psi$  and  $J \Vdash \tau \Uparrow$  are disjoint, and moreover, if  $J \Vdash \tau \Rightarrow E \dashv \Psi$ , then  $FV(E) \subseteq |\Psi|$ .

Numerous general purpose tactics can be defined over a logical theory, including identity, failure, disjunction and sequencing:

$$\begin{array}{c}
\overline{J \Vdash \text{id} \Rightarrow \alpha \dashv \alpha : J} \quad \overline{J \Vdash \text{fail} \Uparrow} \\
\\
\frac{J \Vdash \tau_1 \Rightarrow E \dashv \Psi}{J \Vdash \tau_1 | \tau_2 \Rightarrow E \dashv \Psi} \quad \frac{J \Vdash \tau_1 \Uparrow \quad J \Vdash \tau_2 \Rightarrow E \dashv \Psi}{J \Vdash \tau_1 | \tau_2 \Rightarrow E \dashv \Psi} \\
\\
\frac{J \Vdash \tau_1 \Rightarrow E \dashv \Phi \quad |_{\alpha} \Phi(\alpha) \Vdash \tau_2 \Rightarrow F_{\alpha} \dashv \Psi_{\alpha} \quad (\alpha \in |\Phi|)}{J \Vdash \tau_1 ; \tau_2 \Rightarrow [F / |\Phi|] E \dashv \bigoplus_{|\Phi|} \Psi}
\end{array}$$

*Remark 1.2.* The nominal treatment that we have given here allows for a much more economical presentation of the standard tacticals, which are quite arduous to define in the HOAS treatment used in ML implementations.

**Theorem 1.3.** *The above rules all define valid tactics:.*

1. *id tactic.*
2. *fail tactic.*
3. *If  $\tau_1$  tactic and  $\tau_2$  tactic, then  $\tau_1 | \tau_2$  tactic.*
4. *If  $\tau_1$  tactic and  $\tau_2$  tactic, then  $\tau_1 ; \tau_2$  tactic.*

*Proof.* It suffices to verify that the synthesis  $E$  of each tactical is well-scoped in  $\Psi$ ; in each case, this follows by induction.

(1–2) Immediate.

(3) By induction on derivations of  $J \Vdash \tau_1 | \tau_2 \Rightarrow \Psi \dashv E$ .

**Case  $J \Vdash \tau_1 \Rightarrow E \dashv \Psi$ .** Validity follows from the inductive hypothesis  $\tau_1$  tactic.

**Case  $J \Vdash \tau_1 \Uparrow$ .** Validity follows from the inductive hypothesis  $\tau_1$  tactic.

(4) We need to show that  $FV([F / |\Phi|] E) \subseteq \left| \bigoplus_{|\Phi|} \Psi \right|$ . From our inductive hypotheses, it is evident that  $FV(E) \subseteq |\Phi|$  and that for each  $\alpha \in \Phi$ ,  $FV(F_{\alpha}) \subseteq |\Psi_{\alpha}|$ ; all of the free variables of the term got by substituting each  $F_{\alpha}$  for  $\alpha \in |\Phi|$  clearly reside in one of the fibres of the family of contexts  $\Psi$ , and so they must comprise a subset of the union  $\bigoplus_{|\Phi|} \Psi$  of  $\Psi$ 's fibres.

□

## 2 *Modernized LCF* and the constructible subgoals property

An LCF-style refiner has a property called *constructible subgoals*, which means that the subgoals incurred by a rule or tactic may be constructed independently, using only the statement of the main goal. One unfortunate consequence of this principle is that it is impossible to define a refinement rule which expresses a dependency between subgoals. The canonical example is the introduction rule for dependent pairs in Type Theory:

$$\frac{M \in A \quad N \in [M / x] B}{\langle M, N \rangle \in (x : A) \times B} \text{PairIntro}$$

However, consider what a refinement rule for this would look like; we would like to take the goal  $(x : A) \times B \text{ true}$  to two subgoals (one for each conjunct); the first subgoal  $A \text{ true}$  is clear enough, but it is not possible to even write down the second subgoal until we know the synthesis of the first one.

Because a tactic in *Modernized LCF* produces only a *context* of independent subgoals, we cannot give a refinement treatment to this rule, and must instead write a family of refinement rules  $\text{PairIntro}\{w\}$  fibred over witnesses of the left conjunct  $w$ :

$$(x : A) \times B \text{ true} \Vdash \text{PairIntro}\{w\} \Rightarrow \langle w, \beta \rangle \dashv \left\{ \begin{array}{l} \alpha : w \in A \\ \beta : [w / x] B \text{ true} \end{array} \right.$$

This is clearly unsatisfactory, since it breaks the natural flow of proof development, whereby multiple goals may be refined simultaneously without committing in advance to a particular solution. However, a more palatable rule that allows  $A \text{ true}$  to be demonstrated by refinement is simply not expressible in *Modernized LCF*, since the sense of the second subgoal cannot be expressed except by referring to the synthesis of the first subgoal.

## 3 *Dependent LCF* and generalized refinement rules

At the crux of our problem is the fact that a tactic produces a context of subgoals without any dependencies; if we were to construe the judgment  $J \Vdash \tau \Rightarrow E \dashv \Psi$  as synthesizing a *telescope*  $\Psi$  rather than a mere context, a proper refinement rule for *PairIntro* would be within reach. In fact, whilst the ML signature for LCF refiners rules out this interpretation, the notation we have used for *Modernized LCF* immediately suggests this generalization. Going forward, we will call the theory *Dependent LCF* when we take  $\Psi$  to be a telescope rather than a context.

We can now encode *PairIntro* with a single refinement rule,  $\text{PairIntro}$ :

$$(x : A) \times B \text{ true} \Vdash \text{PairIntro} \Rightarrow \langle \alpha, \beta \rangle \dashv \left\{ \begin{array}{l} \alpha : A \text{ true} \\ \beta : [\alpha / x] B \text{ true} \end{array} \right.$$

All that remains is to give a new definition of the sequencing tactical  $\tau_1; \tau_2$  which accounts for this dependency. In order to do this, we will need to implement an auxiliary judgment  $\boxed{E \multimap \Phi \Downarrow_\tau E' \multimap \Phi'}$  that allows us to gradually apply the tactic  $\tau$  to a proof state, propagating refinements rightward through substitution. This judgment is defined by recursion on the telescope  $\Phi$ , viewed as a *cons*-list:

$$\frac{\overline{E \multimap \cdot \Downarrow_\tau E \multimap \cdot}}{J \Vdash \tau \Rightarrow E_\alpha \multimap \Psi_\alpha \quad [E_\alpha / \alpha] E \multimap [E_\alpha / \alpha] \Phi \Downarrow_\tau E' \multimap \Phi' \over E \multimap \alpha : J, \Phi \Downarrow_\tau E' \multimap \Psi_\alpha \oplus \Phi'}$$

Now, the sequencing tactical is readily definable:

$$\frac{J \Vdash \tau_1 \Rightarrow E \multimap \Phi \quad E \multimap \Phi \Downarrow_{\tau_2} E' \multimap \Phi'}{J \Vdash \tau_1; \tau_2 \Rightarrow E' \multimap \Phi'}$$