

The Definition of **Red JonPRL**,
the people's refinement logic

The JonPRL Group

January 23, 2016

Contents

1	Signatures	2
1.1	Grammar	2
1.2	Static Semantics	2
2	Nominal LCF: a language for tactics	5
2.1	The LCF Tactic Language	5
2.1.1	Atomic tactics and their uniform continuity	5
2.2	Nominal LCF	6

Chapter 1

Signatures

*Decisively Smash The Formalist
Clique!*

Chairman Jon

A *signature* is a collection of definitions, including terms, tactics and theorems.

1.1 Grammar

The grammar of **Red JonPRL** signatures is presented in Figure 1.1. Note that an optional production of sort s is formatted $\langle s \rangle$ in the rules.

$sigexp$	$::=$	$\langle \cdot \rangle$ $sigexp\ sigdec.$	empty signature signature extension
$sigdec$	$::=$	$\text{Def } opid\langle [params] \rangle \langle (args) \rangle : sortid = [term]$ $\text{Tac } opid\langle [params] \rangle \langle (args) \rangle = [term]$ $\text{Thm } opid\langle [params] \rangle \langle (args) \rangle : [term] \text{ by } [term]$	operator definition tactic definition theorem declaration
$params$	$::=$	$\langle \cdot \rangle$ $params, symbind$	empty parameter list parameter list extension
$args$	$::=$	$\langle \cdot \rangle$ $args, metabind$	empty argument list argument list extension
$symbind$	$::=$	$symid : sortid$	symbol binding
$metabind$	$::=$	$metaid : valence$	metavariable binding
$valence$	$::=$	$\langle \{ [sortlist] \} \rangle \langle [sortlist] \rangle . sortid$	valence
$sortlist$	$::=$	$\langle \cdot \rangle$ $sortlist, sortid$	empty sort list sort list extension

Figure 1.1: Grammar of signature expressions. The identifier sorts $opid$, $sortid$, $symid$ and $metaid$ can be assumed to be arbitrary strings; the sort $term$ is left uninterpreted.

1.2 Static Semantics

The static semantics for **Red JonPRL** signatures begins with a specification of the class of *semantic* objects that will serve as the meanings for the *syntactic* objects defined in Section 1.1. We assume an ambient abstract binding tree signature such that at least the following facts hold:

$$\frac{\overline{\text{tac sort}} \quad \overline{\text{thm sort}} \quad \overline{\text{exp sort}} \quad \overline{\text{opid sort}}}{\Upsilon \Vdash \text{prove} : (\text{. exp}, \text{. tac}) \text{ thm}}$$

Then, our semantic objects are defined as in Figure 1.2.

$$\begin{array}{llll} a, b & \in & \text{Sym} \\ \mathbf{m}, \mathbf{n} & \in & \text{Metavar} \\ \sigma, \tau & \in & \text{Sort} & \triangleq \{ \tau \mid \tau \text{ sort} \} \\ v & \in & \text{ProdValence} & \triangleq \{ v \mid v \text{ valence} \} \\ \vartheta & \in & \text{Opid} & \triangleq \text{Sym} \\ \Upsilon & \in & \text{Params} & \triangleq \text{Sym} \rightarrow \text{Sort} \\ \Theta & \in & \text{Args} & \triangleq \text{Metavar} \rightarrow \text{ProdValence} \\ M, N & \in & \text{Tm}(\Theta, \Upsilon, \tau) & \triangleq \{ M \mid \Theta \triangleright \Upsilon \parallel \cdot \vdash M : \tau \} \\ D & \in & \text{Decl} & \triangleq \coprod_{\Upsilon, \Theta, \tau} \text{Tm}(\Theta, \Upsilon, \tau) \\ \Sigma & \in & \text{Sig} & \triangleq \text{Opid} \rightarrow \text{Decl} \end{array}$$

Figure 1.2: Specification of the semantic objects.

A *natural semantics* hinges on the elaboration judgment $E \vdash A \Rightarrow A'$, which means that the syntactic object A elaborates to the semantic object A' in the environment E . Let the $\Upsilon_\Sigma \in \text{Params}$ be defined as follows:

$$\Upsilon_\Sigma(\vartheta) \triangleq \begin{cases} \text{opid} & \text{if } \vartheta \in \text{dom}(\Sigma) \\ \perp & \text{otherwise} \end{cases}$$

Symbol Bindings

$$\boxed{\Sigma \vdash \text{sybind} \Rightarrow (a, \tau)}$$

$$\frac{\Sigma \vdash \text{symid} \Rightarrow a \quad \Sigma \vdash \text{sortid} \Rightarrow \tau}{\Sigma \vdash \text{symid} : \text{sortid} \Rightarrow (a, \tau)} \quad (1.1)$$

Metavariable Bindings

$$\boxed{\Sigma \vdash \text{metabind} \Rightarrow (\mathbf{m}, v)}$$

$$\frac{\Sigma \vdash \text{metaid} \Rightarrow \mathbf{m} \quad \Sigma \vdash \text{valence} \Rightarrow v}{\Sigma \vdash \text{metaid} : \text{valence} \Rightarrow (\mathbf{m}, v)} \quad (1.2)$$

Parameters

$$\boxed{\Sigma \vdash \text{params} \Rightarrow \Upsilon}$$

$$\overline{\Sigma \vdash \langle \cdot \rangle \Rightarrow \{ \}} \quad (1.3)$$

$$\frac{\Sigma \vdash \text{params} \Rightarrow \Upsilon \quad \Sigma \vdash \text{sybind} \Rightarrow (a, \tau)}{\Sigma \vdash \text{params}, \text{sybind} \Rightarrow \Upsilon \cup a \mapsto \tau} \quad (1.4)$$

Arguments

$$\boxed{\Sigma \vdash \text{args} \Rightarrow \Theta}$$

$$\overline{\Sigma \vdash \langle \cdot \rangle \Rightarrow \{ \}} \quad (1.5)$$

$$\frac{\Sigma \vdash \text{args} \Rightarrow \Theta \quad \Sigma \vdash \text{metabind} \Rightarrow (\mathbf{m}, v)}{\Sigma \vdash \text{args}, \text{metabind} \Rightarrow \Theta \cup \mathbf{m} \mapsto v} \quad (1.6)$$

Operator Identifiers

$$\boxed{\Sigma \vdash \textit{opid} \Longrightarrow \vartheta}$$

$$\frac{\vartheta \notin \mathbf{dom}(\Sigma)}{\Sigma \vdash \textit{opid} \Longrightarrow \vartheta} \quad (1.7)$$

Declarations

$$\boxed{\Sigma \vdash \textit{sigdec} \Longrightarrow (\vartheta, D)}$$

$$\frac{\begin{array}{lll} \Sigma \vdash \textit{params} \Longrightarrow \Upsilon & \Sigma \vdash \textit{sortid} \Longrightarrow \tau & \Sigma \vdash \textit{opid} \Longrightarrow \vartheta \\ \Sigma \vdash \textit{args} \Longrightarrow \Theta & \Sigma \vdash \textit{term} \Longrightarrow M & \Theta \triangleright \Upsilon_{\Sigma} \oplus \Upsilon \parallel \cdot \vdash M : \tau \end{array}}{\Sigma \vdash \mathbf{Def} \textit{ opid} \langle [\textit{params}] \rangle \langle (\textit{args}) \rangle : \textit{sortid} = [\textit{term}] \Longrightarrow (\vartheta, \langle \Upsilon, \Theta, \tau, M \rangle)} \quad (1.8)$$

$$\frac{\begin{array}{ll} \Sigma \vdash \textit{params} \Longrightarrow \Upsilon & \Sigma \vdash \textit{opid} \Longrightarrow \vartheta \\ \Sigma \vdash \textit{args} \Longrightarrow \Theta & \Theta \triangleright \Upsilon_{\Sigma} \oplus \Upsilon \parallel \cdot \vdash M : \mathbf{tac} \\ \Sigma \vdash \textit{term} \Longrightarrow M & \end{array}}{\Sigma \vdash \mathbf{Tac} \textit{ opid} \langle [\textit{params}] \rangle \langle (\textit{args}) \rangle = [\textit{term}] \Longrightarrow (\vartheta, \langle \Upsilon, \Theta, \mathbf{tac}, M \rangle)} \quad (1.9)$$

$$\frac{\begin{array}{llll} \Sigma \vdash \textit{params} \Longrightarrow \Upsilon & \Sigma \vdash \textit{term}_1 \Longrightarrow P & \Theta \triangleright \Upsilon_{\Sigma} \oplus \Upsilon \parallel \cdot \vdash P : \mathbf{exp} & \Sigma \vdash \textit{opid} \Longrightarrow \vartheta \\ \Sigma \vdash \textit{args} \Longrightarrow \Theta & \Sigma \vdash \textit{term}_2 \Longrightarrow M & \Theta \triangleright \Upsilon_{\Sigma} \oplus \Upsilon \parallel \cdot \vdash M : \mathbf{tac} & \end{array}}{\Sigma \vdash \mathbf{Thm} \textit{ opid} \langle [\textit{params}] \rangle \langle (\textit{args}) \rangle : [\textit{term}_1] \text{ by } [\textit{term}_2] \Longrightarrow (\vartheta, \langle \Upsilon, \Theta, \mathbf{thm}, \mathbf{prove}(P; M) \rangle)} \quad (1.10)$$

Signatures

$$\boxed{\vdash \textit{sigexp} \Longrightarrow \Sigma}$$

$$\overline{\vdash \langle \cdot \rangle \Longrightarrow \{ \}} \quad (1.11)$$

$$\frac{\vdash \textit{sigexp} \Longrightarrow \Sigma \quad \Sigma \vdash \textit{sigdec} \Longrightarrow (\vartheta, D)}{\vdash \textit{sigexp sigdec.} \Longrightarrow \Sigma \cup \vartheta \mapsto D} \quad (1.12)$$

Chapter 2

Nominal LCF: a language for tactics

In a sequent calculus, left rules add hypotheses to the context; for instance, consider the left rule for positive conjunctions:

$$\frac{H, x : A \otimes B, y : A, z : B \gg [\langle y, z \rangle / x] C}{H, x : A \otimes B \gg C} \otimes_L^{x, y, z}$$

From a proof refinement perspective (see [1]), such a rule is typically manifested as an ML tactic $\otimes_L[x, y, z]$ which takes three names as parameters: the target hypothesis x , and the names to use for the new hypotheses y, z . However, whilst the identity of the name x is essential to the meaning of the tactic, the names supplied for the generated hypotheses can be freshly renamed with impunity.

Indeed, in a proof term assignment for this sequent calculus, the corresponding elimination form would *bind* variables x, y rather than take them, as parameters. However, in the standard LCF tactic paradigm, it is not possible to reproduce this structure, because the sequencing of rules is mediated by the general purpose THEN tactical, which has no knowledge of names or binding.

We will design a language for tactics called **Nominal LCF** which supports a distinction between names bound and names taken as parameters, and then show how it can be elaborated into standard LCF.

2.1 The LCF Tactic Language

The essence of the LCF tactic system is captured in the following (idealized) ML signature:

```
type judgment
type evidence
type tactic      = judgment → judgment list ⊗ (evidence list → evidence)
```

In other words, a tactic is a partial function that takes a goal to its subgoals, and specifies how to transform the evidence of its subgoals into the evidence for the main goal. In the case of the sequent calculus we were considering, **judgment** would be a type of sequents. Now, in general a tactic may need to consume names from a *name store*, which is an infinite stream of *atoms* or *symbols*:

```
type A
type atactic = Aℕ → tactic
```

2.1.1 Atomic tactics and their uniform continuity

Left sequent rules can be coded as so-called *atomic tactics*, tactics which consume a stream of names. In fact, every such tactic is *uniformly continuous* in a specific sense. For an atomic sequence $\alpha \in A^\mathbb{N}$ and a natural number $n \in \mathbb{N}$, let $\bar{\alpha}(n)$ be the initial segment of α of length n . Let $M \approx N$ be *observational equivalence*: M evaluate to the same value, or they diverge.

Then, for any atomic tactic t and judgment \mathcal{J} , we can calculate a uniform modulus of continuity:

$$\exists n \in \mathbb{N}. \forall \alpha, \beta \in \mathbb{A}^{\mathbb{N}}. \bar{\alpha}(n) = \bar{\beta}(n) \implies t(\alpha, \mathcal{J}) \approx t(\beta, \mathcal{J}) \quad (\text{uniform continuity})$$

This calculation can be realized computationally in our metalanguage in a number of ways, but for our purposes it suffices to remark that it is a consequence of Brouwer's Fan Theorem, which we hold to be evident. Let $\text{umod}(t, \mathcal{J}) \in \mathbb{N}$ be the uniform modulus of continuity for a tactic t at goal \mathcal{J} .

2.2 Nominal LCF

We will define the **Nominal LCF** language by specifying an abt signature for it; at its heart is the refactoring of the various sequencing tacticals THEN, THENL, etc. of LCF into a single sequencing tactical combined a separate notion of *multi-tactic*.

First, we define sorts for tactics, atomic tactics, binding tactics and multi-tactics respectively:

$$\overline{\text{tac sort}} \quad \overline{\text{atac sort}} \quad \overline{\text{btac sort}} \quad \overline{\text{mtac sort}}$$

We also provide a sort to classify hypotheses:

$$\overline{\text{hyp sort}}$$

Now, we'll define the operators of **Nominal LCF**; note that the operators of sort **atac** are arbitrary and are provided only for the sake of illustration.

$$\begin{array}{c} \overline{\Upsilon \Vdash \text{id} : () \text{ atac}} \quad \overline{\Upsilon \Vdash \text{fail} : () \text{ atac}} \quad \overline{\Upsilon, a : \text{hyp} \Vdash \text{elim}[a] : () \text{ atac}} \\[10pt] \frac{n \in \mathbb{N}}{\Upsilon \Vdash \text{seq}_n : (. \text{btac}, \{\text{hyp}^n\}. \text{mtac}) \text{ tac}} \\[10pt] \frac{n \in \mathbb{N}}{\Upsilon \Vdash \text{smash}_n : (. \text{atac}^n) \text{ btac}} \\[10pt] \overline{\Upsilon \Vdash \text{all} : (. \text{tac}) \text{ mtac}} \quad \frac{n \in \mathbb{N}}{\Upsilon \Vdash \text{each}_n : (. \text{tac}^{n+1}) \text{ mtac}} \quad \frac{i \in \mathbb{N}}{\Upsilon \Vdash \text{some}_i : (. \text{tac}) \text{ mtac}} \end{array}$$

For the sake of clarity, we introduce the following notational abbreviations for tactic expressions:

$$\begin{aligned} t_1; t_2 &\triangleq \text{seq}_0(. t_1; . t_2) \\ a_0, \dots, a_n \leftarrow t_1; t_2 &\triangleq \text{seq}_n(. t_1; \{a_0, \dots, a_n\}. t_2) \\ \langle t_1, \dots, t_n \rangle &\triangleq \text{smash}_n(. t_1; \dots; . t_n) \\ \Box t &\triangleq \text{all}(. t) \\ \langle t_1, \dots, t_n \rangle &\triangleq \text{each}_n(. t) \\ \Diamond_i t &\triangleq \text{some}_i(. t) \end{aligned}$$

Bibliography

- [1] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.