Type Refinements for the Working Class

Jon Sterling and Darin Morrison

There are two conflicting views which bedevil any discussion of the nature of type theory. First, there is the notion of type theory as an extension or generalization of universal algebra to support interdependency of sorts and operations, possibly subject to an arbitrary equational theory [3, 5]; we will call this *formal type theory*. Typing, in such a setting, is a mere matter of grammar and is nearly always decidable. In hindsight, we may observe that this is the sort of type theory which Martin-Löf first proposed in 1972 [17], even if we will admit that this was not the intention at the time. A model for such a type theory is usually given by interpreting the types or sorts as presheaves or sheaves over contexts of hypotheses, and as such, a proof theoretic interpretation of the hypothetical judgment is possible.

Secondly, there is the view of type theory as semi-formal theory of constructions for the Brouwer-Heyting-Kolmogorov interpretation of intuitionistic mathematical language, which we will call *behavioral* or *semantic type theory*. The most widely known development of this program is Martin-Löf's 1979 "extensional" type theory [18, 19], but we must give priority to Dana Scott for inventing this line of research in 1970 with his prophetic report, *Constructive Validity* [20]. Since the 1980s, behavioral type theory has been developed much further in the Nuprl family [4] of proof assistants, including MetaPRL [13] and JonPRL [21].

Martin-Löf's key innovation was the commitment to pervasive functionality (extensionality) as part of the *definitions* of the judgments and the types, in contrast to the state of affairs in formal type theory where functionality is a metatheorem which must be shown to obtain, based on the (somewhat arbitrary) equational theory which has been imposed. Furthermore, models for behavioral type theory interpret the types as partial equivalence relations (PERs) on only closed terms, and the meaning of the hypothetical judgment is defined separately and uniformly in the logical relations style.

Our position is that these views of type theory are not in conflict, but rather describe two distinct layers in a single, harmonious system. From this perspective, formal type theories can do little more than negotiate matters of grammar, and therefore may serve as a linguistic framework for mathematical expression, being responsible for the management of variable binding and substitution. On the other hand, mathematical objects find *meaning* in their extension within the behavioral type theory, and are subject to an extensional equality.

The types of the semantic theory can then be said to *refine* the types of the syntactic theory [10, 11, 9], both by placing restrictions on membership and by coarsening equivalence.

Thus far, most developments of behavioral type theory have been built on a *unityped* syntactic framework, and so the relation to type refinements has been difficult to see.

In this paper, we contribute a full theory of behavioral refinements over multi-sorted nominal abstract binding trees with symbolic parameters, a simple but interesting formal type theory [9, 22]; this hybrid system allows the deployment of a Nuprl-style type theory over any signature of sorts and operators.

Owing to the *nominal* aspect of our abstract binding tree (abt) framework, this development constitutes a possible alternative to Allen's semantics for unguessable atoms in Nuprl [1]; Allen proposed a "supervaluation" semantics, that explained sequents involving atoms (symbols) by quantifying over the possible ways to interpret the class of atoms. Our semantics seem to be a more direct and concrete way to approach the problem, which is licensed in part by the nominal character of the abt framework.

Colors In this paper, we hint at the *modes* of judgments and assertions [9] using colors, marking inputs with `blue` and outputs with `red`. As a rule of thumb, inputs are things which are supplied when checking the correctness of a judgment, and outputs are things which are synthesized in the process.

1 Abstract Binding Trees and Symbols

We will briefly reproduce the syntax of abts here; see [22] for a full development of the syntax and semantics of multi-sorted nominal abstract binding trees, however. The point of nominal abts is that they extend the standard treatments of second order universal algebra [7] with a notion of *symbolic parameter* and name abstraction; this is absolutely essential for correctly treating the syntax of assignables, open exceptions, fluid bindings, and processes, among other things. See Harper's *Practical Foundations for Programming Languages* [9] for numerous examples of the use of symbolic parameters.

Fix a set of *sorts* $\mathcal{S}$; we'll say τ *sort* when $\tau \in \mathcal{S}$. Let $\mathbb{I}$ be the category of finite cardinals and injective maps between them; using the comma construction, we can define the category $\mathbb{I} \downarrow \mathcal{S}$ of *contexts* of symbols, whose objects are finite sets of symbols $\mathcal{U}$ and sort-assignments $\mathcal{U} \xrightarrow{\mathfrak{s}} \mathcal{S}$, and whose morphisms are sort-preserving renamings; we will write Υ for a symbol context $(\mathcal{U}, \mathfrak{s})$. We also write Υ *sctx* when $\Upsilon \in \mathbb{I} \downarrow \mathcal{S}$.

We will say that an $\mathcal{S}$-valence is a form $\{\vec{\sigma}\}[\vec{\tau}].\,\tau$, where $\vec{\sigma} \cup \vec{\tau} \subseteq \mathcal{S}$ and τ *sort*; valences represent the *binding structure* of an abstraction of sort τ, where $\vec{\sigma}$ are the sorts of the bound symbols and $\vec{\tau}$ are the sorts of the bound variables. When ν is a valence, we will say ν *valence*; let $\mathcal{V}_{\mathcal{S}}$ be the set of $\mathcal{S}$-valences.

An $\mathcal{S}$-arity is of the form $(\vec{\nu})\,\tau$, where $\vec{\nu}$ are $\mathcal{S}$-valences and τ *sort*; arities classify *operators* (also called "function symbols" in older treatments of algebra). When α is an arity, we will say α *arity*; let $\mathcal{A}_{\mathcal{S}}$ be the set of $\mathcal{S}$-arities.

An $\mathcal{S}$-*operator signature* is an $\mathcal{S}$-arity-indexed family of copresheaves on $\mathbb{I} \downarrow \mathcal{S}$, namely $\mathcal{O}_a : \mathbf{Set}^{\mathbb{I}\downarrow\mathcal{S}}$ (a *arity*); this is an intensional generalization of the state of affairs for standard universal algebra, where operators are arity-indexed families of *sets*. The point of fibering over symbol contexts is to induce an open-ended indexing mechanism for operators; the functorial liftings of $\mathbb{I} \downarrow \mathcal{S}$-morphisms $\Upsilon \xhookrightarrow{\rho} \Upsilon'$ equip operators $\vartheta \in \mathcal{O}_a(\Upsilon)$ with a suitable renaming action $\vartheta \cdot \rho \in \mathcal{O}_a(\Upsilon')$.[1] We call the pair $\Sigma \equiv \langle \mathcal{S}, \mathcal{O} \rangle$ an *abt signature*.

Just as we have defined symbol contexts as $\mathbb{I} \downarrow \mathcal{S}$, we will use the category $\mathbb{F}$ of finite cardinals and *all* maps between them to define variable contexts $\mathbb{F} \downarrow \mathcal{S}$ and metavariable contexts $\mathbb{F} \downarrow \mathcal{V}_\mathcal{S}$, writing Γ *vctx* when $\Gamma \in \mathbb{F} \downarrow \mathcal{S}$ and Θ *mctx* when $\Theta \in \mathbb{F} \downarrow \mathcal{V}_\mathcal{S}$. The difference between $\mathbb{I} \downarrow \mathcal{S}$ and $\mathbb{F} \downarrow \mathcal{S}$ is that the latter allows renamings that may identify distinct names, whereas renamings in $\mathbb{I} \downarrow \mathcal{S}$ are always injective.

For a given abt signature, the syntax of abts is given by the judgments $\Theta \triangleright \Upsilon \parallel \Gamma \vdash M : \tau$ (presupposing Θ *mctx*, Υ *sctx*, Γ *vctx* and τ *sort*) and $\Theta \triangleright \Upsilon \parallel \Gamma \vdash E : \nu$ (presupposing Θ *mctx*, Υ *sctx*, Γ *vctx* and ν *valence*), defined in mutual induction as follows:

$$\frac{\Gamma \ni x : \tau}{\Theta \triangleright \Upsilon \parallel \Gamma \vdash x : \tau}\ \vdash_{var}$$

$$\frac{\begin{array}{l}\Theta \ni m : \{\sigma_0, \ldots, \sigma_m\}[\tau_0, \ldots, \tau_n].\,\tau \\ \Upsilon \ni u_i : \sigma_i \quad (i \leqslant m) \\ \Theta \triangleright \Upsilon \parallel \Gamma \vdash M_i : \tau_i \quad (i \leqslant n)\end{array}}{\Theta \triangleright \Upsilon \parallel \Gamma \vdash m\{u_0, \ldots, u_m\}(M_0, \ldots, M_n) : \tau}\ \vdash_{mvar}$$

$$\frac{\begin{array}{l}\Upsilon \Vdash \vartheta : (\nu_1, \ldots, \nu_n)\,\tau \\ \Theta \triangleright \Upsilon \parallel \Gamma \vdash E_i : \nu_i \quad (i \leqslant n)\end{array}}{\Theta \triangleright \Upsilon \parallel \Gamma \vdash \vartheta(E_0, \ldots, E_n) : \tau}\ \vdash_{app}$$

$$\frac{\Theta \triangleright \Upsilon, \vec{u} : \vec{\sigma} \parallel \Gamma, \vec{x} : \vec{\tau} \vdash M : \tau}{\Theta \triangleright \Upsilon \parallel \Gamma \vdash \mathbb{\lambda}\{\vec{u}\}[\vec{x}].\,M : \{\vec{\sigma}\}[\vec{\tau}].\,\tau}\ \vdash_{abs}$$

We have several meta-operations on abts, including the calculation of free symbols $\mathbf{FS}(M)$, the calculation of free variables $\mathbf{FV}(M)$, the transport $M \cdot \rho$ across a renaming ρ, the substitution $[N / x]\,M$ of a term N for the variable x in M, and the substitution $[E / m]\,M$ of an abstraction E for the metavariable m in M. Please see [22] for the explicit definition of these operations.

[1] In [22], the authors defined operator signatures as *sheaves* on $(\mathbb{I} \downarrow \mathcal{S})^{\mathrm{op}}$ under the atomic topology (alternatively, pullback-preserving presheaves), where all non-empty sieves cover; the sheaf condition induces the unique existence of *strengthenings* for operators. However, because we do not use strengthening in the present development, we have omitted this part of the apparatus for the sake of simplicity.

2 Generalized Predicative Foundations

Before we continue, it makes sense to remark briefly on the strength of the mathematical foundations that we are assuming for the development of our framework. As in Martin-Löf's meaning-theoretic judgmental method, our apparatus is meant to be understood in a constructive and predicative metatheory; therefore, by **Set**, we do not mean the classical category of sets, but rather the intuitionistic one, which lacks the powerset operation.

Therefore, in our setting, **Set** is not an elementary topos, and nor is any category of **Set**-valued (pre)sheaves. However, for a category of classes **Class**, we can express a power class operation $\wp(X) \triangleq \{Y \mid Y \subseteq X, Y \in \mathbf{Set}\}$ which gives the proper class of sub*sets* of a class X [8].

A category of **Class**-valued presheaves then is a ΠWM-pretopos (a predicative topos with dependent function objects, wellfounded trees, and non-wellfounded trees); such a pretopos also admits a large subobject classifier $\Omega(c)$, defined as the class of small sieves on c,[2] whence we can form the large power object $\wp(X) \triangleq \Omega^X$ for any object X.

In practice, the shift to a predicative topos with a large power object was a mere technical nuisance, as all the constructions we needed to perform carried over perfectly to this setting. For the remainder of this paper, we assume the semantic universe $\mathbf{J} \triangleq \mathbf{Class}^{\mathbb{I}\downarrow\mathcal{S}}$, the category of copresheaves on symbol contexts. Every judgment $\mathcal{J}$ which we define in this paper should be understood as such a copresheaf, which is at each stage Υ the class of derivations of $\mathcal{J}$ using symbols in Υ; we will also make use of *internal* forms of judgment, which are sections of $\wp(X)$ for some object X.

Our purpose in fixing a semantic universe is emphatically *not* to give an interpretation or model of type theory inside set theory, though it is possible to read our construction in this metamathematical light—however, the vocabulary of the pretopos **J** gives us a perspicuous notation in which to propound what we consider to be an informal (pre-mathematical) meaning explanation in the sense of Heyting [12] and Martin-Löf [18]. [3]

3 Behavioral Refinements

Fixing a signature $\Sigma \equiv \langle \mathcal{S}, \mathcal{O} \rangle$ in the abt framework, we will define the notion of behavioral refinement by propounding several judgments and their semantical explanations. Let us first define the copresheaf of τ-sorted expressions, $\mathbf{E}_\tau$ as follows

$$\mathbf{E}_\tau(\Upsilon \parallel \Gamma) \triangleq \{M \mid \cdot \triangleright \Upsilon \parallel \Gamma \vdash M : \tau\}$$

We will also write $\mathbf{E}_\tau$ for the copresheaf $\mathbf{E}_\tau(- \parallel \cdot)$ on $\mathbb{I} \downarrow \mathcal{S}$.

[2] A sieve S on c is a subobject of the Yoneda embedding $\mathbf{y}(c)$, and S is *small* if $S(d)$ is a set for every stage d.

[3] Both Gambino [8] and Dybjer [6] give lucid explanations of the distinction between *meaning theory* and *model theory*.

3.1 Parametric Refinement

The first judgment that will concern us is called *parametric refinement*, $\Upsilon \parallel \phi \sqsubset \tau$, which means that ϕ refines the sort τ under the symbolic parameters Υ. We define this judgment through a meaning explanation in the style of Martin-Löf as follows:

Definition 3.1. To know $\Upsilon \parallel \phi \sqsubset \tau$ (presupposing Υ *sctx* and τ *sort*) is to know, for any $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, what it means for any $M, N \in \mathbf{E}_\tau(\Upsilon')$ to be equated by ϕ (written $\phi\{\rho\}(M, N)$), such that $\phi\{\rho\}(-,-)$ is a partial equivalence relation (PER) on $\mathbf{E}_\tau(\Upsilon')$. Morever, that for any $\Upsilon' \xhookrightarrow{\rho'} \Upsilon''$, from $\phi\{\rho\}(M, N)$ we may conclude $\phi\{\rho' \circ \rho\}(M \cdot \rho', N \cdot \rho')$.

Then, we say that ϕ *globally refines* τ (written $\phi \sqsubset \tau$) when we have $\cdot \parallel \phi \sqsubset \tau$. Furthermore, when $\Upsilon \parallel \phi \sqsubset \tau$, for any renaming $\Upsilon \xhookrightarrow{\rho} \Upsilon'$ we can clearly define a new refinement $\Upsilon' \parallel \phi \cdot \rho \sqsubset \tau$.

Remark 3.2. At this point, it should be noted that this is very similar to the semantical explanation of typehood given in [18], except that we have generalized it to a multi-sorted setting, and that we have fibred the entire apparatus over collections Υ of symbols; furthermore, whereas Martin-Löf defines types in terms of evaluation to canonical form, we have remained agnostic on this point as far as refinements are concerned. We will come back to the notions of *type* and *computation* in section 5.

In fact, the complexity of the above meaning explanation is an artifact of the pointwise style in which we have expressed it. Considered internally to the semantic universe $\mathbf{J}$, a refinement is merely a section of the object of $\mathbf{E}_\tau$-PERs $\mathfrak{per}(\mathbf{E}_\tau)$, defined internally as a subobject of $\wp(\mathbf{E}_\tau \times \mathbf{E}_\tau)$:

$$\mathfrak{per}(X) \triangleq \{\phi : \wp(X \times X) \mid \mathtt{symmetric}(\phi) \wedge \mathtt{transitive}(\phi)\}$$

So, $\Upsilon \parallel \phi \sqsubset \tau$ obtains just when we have $\phi \in \mathfrak{per}(\mathbf{E}_\tau)(\Upsilon)$. The benefit of explaining such judgments as copresheaves is that we do not need to deal with the complexities of quantifying over renamings, since this is implicit in the definition of the exponential of presheaves, where $\mathfrak{y}(\Upsilon) \triangleq \Upsilon \hookrightarrow -$ is the co-Yoneda embedding:

$$\begin{aligned} B^A(\Upsilon) &\triangleq \mathbf{J}[\mathfrak{y}(\Upsilon) \times A, B] \\ &\cong \int_{\Upsilon'} (\Upsilon \hookrightarrow \Upsilon') \implies B(\Upsilon')^{A(\Upsilon')} \end{aligned}$$

It should be noted that PERs can be understood as 0-semigroupoids. In other words, these are 1-groupoids without identity morphisms and without coherence laws regarding composition and inverse operations. This distinction is especially relevant in formal type theory where groupoids "one dimension down" are not sets but setoids (i.e., groupoids sans coherence). Thus, a PER can formally be thought of as a kind of partial setoid that facilitates constructions on subsets in addition to quotients.

Say something about internal (semi)-groupoids?

3.1.1 Order and Equality of Parametric Refinements

We will write $\Upsilon \parallel \phi \subseteq \psi \sqsubset \tau$ to mean that ϕ is a *subrefinement* of ψ.

Definition 3.3. To know $\Upsilon \parallel \phi \subseteq \psi \sqsubset \tau$ (presupposing $\Upsilon \parallel \phi \sqsubset \tau$ and $\Upsilon \parallel \psi \sqsubset \tau$), is to know, for any renamings $\Upsilon \xhookrightarrow{\rho} \Upsilon' \xhookrightarrow{\rho'} \Upsilon''$, that from $\phi\{\rho\}(M, N)$ you can conclude $\psi\{\rho' \circ \rho\}(M \cdot \rho', N \cdot \rho')$ for any $M, N \in \mathbf{E}_\tau(\Upsilon')$.

Two refinements are equal when they denote the same PER.That is, we have $\Upsilon \parallel \phi = \psi \sqsubset \tau$ just when both $\Upsilon \parallel \phi \subseteq \psi \sqsubset \tau$ and $\Upsilon \parallel \psi \subseteq \phi \sqsubset \tau$.

3.2 Parametric Equality

The primary judgment concerning refinements is the parametric equality, $\Upsilon \parallel M = N \in \phi$, which presupposes $\Upsilon' \parallel \phi \sqsubset \tau$ for some Υ' such that we have $\Upsilon' \xhookrightarrow{\rho} \Upsilon$, and $M, N \in \mathbf{E}_\tau(\Upsilon)$. The meaning of this judgment is that M and N are identified by ϕ at Υ:

$$\frac{\phi\{\rho\}(M, N)}{\Upsilon \parallel M = N \in \phi}$$

3.3 Functional Refinement

Next, we define functional refinement $\Upsilon \parallel \Psi \vDash \phi = \psi \sqsubset \tau$ in terms of parametric refinement, simultaneously with parametric context refinement $\Upsilon \parallel \Psi \sqsubset^\star \Gamma$ and functional equality $\Upsilon \parallel \Psi \vDash M = N \in \phi$. We will write $\Upsilon \parallel \Psi \vDash \phi \sqsubset \tau$ as a shorthand for $\Upsilon \parallel \Psi \vDash \phi = \phi \sqsubset \tau$.

Parametric context refinement $\Upsilon \parallel \Psi \sqsubset^\star \Gamma$ shall be defined as follows, presupposing Υ *sctx* and Γ *vctx*:

$$\frac{}{\Upsilon \parallel \cdot \sqsubset^\star \cdot}\ \sqsubset^\star_{\mathtt{nil}} \qquad \frac{\Upsilon \parallel \Psi \sqsubset^\star \Gamma \quad \Upsilon \parallel \Psi \vDash \phi \sqsubset \tau}{\Upsilon \parallel \Psi, x : \phi \sqsubset^\star \Gamma, x : \tau}\ \sqsubset^\star_{\mathtt{snoc}}$$

For a context refinement $\Upsilon \parallel \Psi \sqsubset^\star \Gamma$ and a renaming $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, we can define a new context refinement $\Upsilon' \parallel \Psi \cdot \rho \sqsubset^\star \Gamma$ by applying ρ pointwise at each of the sort refinements in the context.

Functional refinement $\Upsilon \parallel \Psi \vDash \phi = \psi \sqsubset \tau$ presupposes $\Upsilon \parallel \Psi \sqsubset^\star \Gamma$ and τ *sort*, and is explained by induction on the evidence for its first presupposition.

Case $\dfrac{}{\Upsilon \parallel \cdot \sqsubset^\star \cdot}\ \sqsubset^\star_{\mathtt{nil}}$.

$$\frac{\Upsilon \parallel \phi = \psi \sqsubset \tau}{\Upsilon \parallel \cdot \vDash \phi = \psi \sqsubset \tau}\ \vDash^{\sqsubset}_{\mathtt{nil}}$$

Case $\dfrac{\Upsilon \parallel \Psi \sqsubset^\star \Gamma \quad \Upsilon \parallel \Psi \vDash \chi \sqsubset \sigma}{\Upsilon \parallel \Psi, x : \chi \sqsubset^\star \Gamma, x : \sigma}\ \sqsubset^\star_{\mathtt{snoc}}$.

To know $\Upsilon \parallel \Psi, x : \chi \vDash \phi = \psi \sqsubset \tau$ is to know, for any renaming $\Upsilon \xhookrightarrow{\rho} \Upsilon'$ and closed terms $M_0, M_1 \in \mathbf{E}_\sigma(\Upsilon')$, that from $\Upsilon' \parallel \Psi \cdot \rho \vDash M_0 = M_1 \in \chi \cdot \rho$, you can conclude $\Upsilon' \parallel \Psi \cdot \rho \vDash [M_0 / x]\,\phi \cdot \rho = [M_1 / x]\,\psi \cdot \rho \sqsubset \tau$. In other words:

$$\dfrac{\begin{array}{l}\forall \Upsilon \xhookrightarrow{\rho} \Upsilon'.\, \forall M_0, M_1 \in \mathbf{E}_\sigma(\Upsilon').\\ (\Upsilon' \parallel \Psi \cdot \rho \vDash M_0 = M_1 \in \chi \cdot \rho)\\ \quad \Longrightarrow\ (\Upsilon' \parallel \Psi \cdot \rho \vDash [M_0 / x]\,\phi \cdot \rho = [M_1 / x]\,\psi \cdot \rho \sqsubset \tau)\end{array}}{\Upsilon \parallel \Psi, x : \chi \vDash \phi = \psi \sqsubset \tau}\ \vDash^\sqsubset_{\mathtt{snoc}}$$

3.4 Functional Equality

Functional equality $\Upsilon \parallel \Psi \vDash N_0 = N_1 \in \phi$ presupposes $\Upsilon \parallel \Psi \sqsubset^\star \Gamma$, $\Upsilon \parallel \Psi \vDash \phi \sqsubset \tau$, and $N_0, N_1 \in \mathbf{E}_\tau(\Upsilon \parallel \Gamma)$; this judgment is defined by induction on the evidence for the first presupposition $\Upsilon \parallel \Psi \sqsubset^\star \Gamma$:

Case $\dfrac{}{\Upsilon \parallel \cdot \sqsubset^\star \cdot}\ \sqsubset^\star_{\mathtt{nil}}$.

$$\dfrac{\Upsilon \parallel N_0 = N_1 \in \phi}{\Upsilon \parallel \cdot \vDash N_0 = N_1 \in \phi}\ \vDash^=_{\mathtt{nil}}$$

Case $\dfrac{\Upsilon \parallel \Psi \sqsubset^\star \Gamma \quad \Upsilon \parallel \Psi \vDash \chi \sqsubset \sigma}{\Upsilon \parallel \Psi, x : \chi \sqsubset^\star \Gamma, x : \sigma}\ \sqsubset^\star_{\mathtt{snoc}}$.

To know $\Upsilon \parallel \Psi, x : \chi \vDash N_0 = N_1 \in \phi$ is to know, for any renaming $\Upsilon \xhookrightarrow{\rho} \Upsilon'$ and closed terms $M_0, M_1 \in \mathbf{E}_\sigma(\Upsilon')$, that from $\Upsilon' \parallel \Psi \cdot \rho \vDash M_0 = M_1 \in \chi \cdot \rho$, you can conclude $\Upsilon' \parallel \Psi \cdot \rho \vDash [M_0 / x]\,N_0 \cdot \rho = [M_1 / x]\,N_1 \cdot \rho \in [M_0 / x]\,\phi \cdot \rho$. In other words:

$$\dfrac{\begin{array}{l}\forall \Upsilon \xhookrightarrow{\rho} \Upsilon'.\, \forall M_0, M_1 \in \mathbf{E}_\sigma(\Upsilon').\\ (\Upsilon' \parallel \Psi \cdot \rho \vDash M_0 = M_1 \in \chi \cdot \rho)\\ \quad \Longrightarrow\ (\Upsilon' \parallel \Psi \cdot \rho \vDash [M_0 / x]\,N_0 \cdot \rho = [M_1 / x]\,N_1 \cdot \rho \in [M_0 / x]\,\phi \cdot \rho)\end{array}}{\Upsilon \parallel \Psi, x : \psi \vDash N_0 = N_1 \in \phi}\ \vDash^=_{\mathtt{snoc}}$$

4 Constructions on Refinements

Refinements can be arranged in a partial order via the $\Upsilon \parallel \phi \subseteq \psi \sqsubset \tau$ judgment, and admit a lattice structure at any sort τ.

We have the bottom refinement $\bot_\tau \sqsubset \tau$, which simply contains no elements. Next, the top refinement $\top_\tau \sqsubset \tau$ identifies all terms $M, N \in \mathbf{E}_\tau(\Upsilon)$. The join (union) $\Upsilon \parallel \phi \cup \psi \sqsubset \tau$ of two refinements $\Upsilon \parallel \phi \sqsubset \tau$ and $\Upsilon \parallel \psi \sqsubset \tau$ is defined as follows, for any $\Upsilon \xhookrightarrow{\rho} \Upsilon'$:

$$\frac{\Upsilon' \parallel M = N \in \phi}{\Upsilon' \parallel M = N \in \phi \cup \psi} \qquad \frac{\Upsilon' \parallel M = N \in \psi}{\Upsilon' \parallel M = N \in \phi \cup \psi}$$

Finally the meet (intersection) $\Upsilon \parallel \phi \cap \psi \sqsubset \tau$ of two refinements $\Upsilon \parallel \phi \sqsubset \tau$ and $\Upsilon \parallel \psi \sqsubset \tau$ is defined dually:

$$\frac{\Upsilon' \parallel M = N \in \phi \quad \Upsilon' \parallel M = N \in \psi}{\Upsilon' \parallel M = N \in \phi \cap \psi}$$

It is trivial to verify that these constructions on refinements give rise to a lattice.

Remark 4.1. Observe that *definition* of a refinement $\Upsilon \parallel \phi \sqsubset \tau$ is given in terms of an arbitrary Υ' that is related to Υ by a renaming. This is because Υ is the stage at which the refinement has come into being, but the definition of the refinement must be applicable at all subsequent stages.

4.1 Families of Refinements

Given a family of refinements (i.e. a functional refinement) $\Upsilon \parallel x : \phi \vDash \psi \sqsubset \tau$ such that $\Upsilon \parallel \phi \sqsubset \sigma$, including the family intersection, the dependent intersection, the family union, and the set comprehension.

Family intersection The intersection $\Upsilon \parallel \bigcap_{(x:\phi)} \psi \sqsubset \tau$ of a family of refinements is defined by the following rule, for any $\Upsilon \xhookrightarrow{\rho} \Upsilon'$:

$$\frac{\Upsilon' \parallel x : \phi \vDash M = N \in [M/x]\,\psi}{\Upsilon' \parallel M = N \in \bigcap_{(x:\phi)} \psi}$$

The intersection of a family of refinements is a kind of uniform universal quantification, in the sense that the terms themselves bear no trace of the quantification, but are guaranteed in the metalogic to be related in all fibers of ψ.

In practice, the family intersection is perhaps the most ubiquitous type constructor in Nuprl developments, as it facilitates the simultaneous specification and implementation of programs in type theory, without causing details of the program's verification to leak into the extracted algorithm. Additionally, the family intersection can be used to implement coinductive types.

Dependent intersection The dependent intersection $\Upsilon \parallel (x : \phi) \cap \psi \sqsubset \sigma$ contains the members M of ϕ which are also members of $[M\,/\,x]\,\psi$:

$$\frac{\Upsilon' \parallel M = N \in \phi \quad \Upsilon' \parallel M = N \in [M\,/\,x]\,\psi}{\Upsilon' \parallel M = N \in (x : \phi) \cap \psi}$$

The dependent intersection, first introduced by Kopylov in [15], is a generalization of the binary intersection to the dependent case, not to be confused with the intersection of a family of sets. The dependent intersection has many applications, including the compositional representation of *dependent records* or modules in Type Theory.

Family union The union $\Upsilon \parallel \bigcup_{(x:\phi)} \psi \sqsubset \tau$ of a family of refinements is defined by the following rule:

$$\frac{\Upsilon' \parallel L_0 = L_1 \in \phi \quad \Upsilon' \parallel M = N \in [L_0\,/\,x]\,\psi}{\Upsilon' \parallel M = N \in \bigcup_{(x:\phi)} \psi}$$

The union of a family of refinements can be used to implement inductive types, among other things.

Set comprehension We can also use a family of refinements to define something analogous to set comprehension, $\Upsilon' \parallel \{\, x : \phi \mid \psi \,\} \sqsubset \sigma$, defined as follows:

$$\frac{\Upsilon' \parallel M = N \in \phi \quad \Upsilon' \parallel L_0 = L_1 \in [M\,/\,x]\,\psi}{\Upsilon' \parallel M = N \in \{\, x : \phi \mid \psi \,\}}$$

The set comprehension $\{\, x : \phi \mid \psi \,\}$ contains just those elements of ϕ such that their fibers of ψ are inhabited; this can be seen as an "implicit" version of existential quantification, and can be used to simultaneously specify and implement program without the evidence for ψ intruding into the extracted algorithm.

5 Multi-Sorted Computational Type Theory

Martin-Löf's type theory is a theory of *value types*: to define a type, you specify how to form its equal canonical inhabitants, or values. Then, the inhabitants of the types are explained uniformly by extending this relation over an operational semantics such that two terms are equal members of a type just when they compute to equal values of that type.

More generally, one may wish to consider types which include computations in addition to values; an example of such a type would be $\psi_\perp$ when ψ is a type, which would include all the members of ψ as well as a divergent computation. Another example is the type $\top_\tau$ which identifies all closed terms of sort τ.

In order to ensure that such types are sensible, we must formulate a notion of computational equivalence, and then say that a refinement is a *type* just when it respects computational equivalence; then, we are able to define types as PERs on *programs* in addition to values.

5.1 Lazy Computation Systems

In this section we generalize Howe's notion of lazy computation system [14] to the multi-sorted, symbol-parameterized setting. An *lazy computation language* is an abt signature $\Sigma \equiv \langle \mathcal{S}, \mathcal{O} \rangle$ along with a distinguished arity-indexed family of copresheaves $\mathcal{K}_a : \mathbf{Set}^{\mathbb{I}\downarrow\mathcal{S}}$ $(a \in \mathcal{A}_{\mathcal{S}})$ of *canonical* operators such that $\mathcal{K}_a \subseteq \mathcal{O}_a$ for each arity a.

For a lazy computation language $L \equiv \langle \mathcal{S}, \mathcal{O}, \mathcal{K} \rangle$, let us define sort-indexed (resp. valence-indexed) copresheaves on $\mathbb{I} \downarrow \mathcal{S}$ of closed values (resp. bound terms) as follows:

$$\mathbf{V}_\tau(\Upsilon) \triangleq \left\{ M \equiv \vartheta(\vec{E}) \mid M \in \mathbf{E}_\tau(\Upsilon) \wedge \exists a. \mathcal{K}_a(\Upsilon) \ni \vartheta \right\}$$
$$\mathbf{B}_\nu(\Upsilon) \triangleq \{ E \mid \cdot \triangleright \Upsilon \parallel \cdot \vdash E : \nu \}$$

Then, a *lazy computation system* (lcs) is a lazy computation language $L \equiv \langle \Sigma, \mathcal{K} \rangle$ along with a partial, $\mathbb{I} \downarrow \mathcal{S}$-indexed $\equiv_\alpha$-functional evaluation relation $\Upsilon \parallel M \Downarrow_\tau^n N$ presupposing $n \in \mathbb{N}$, $M \in \mathbf{E}_\tau(\Upsilon)$ and $N \in \mathbf{V}_\tau(\Upsilon)$, expressing that M evaluates to N in n steps. We will write $\Upsilon \parallel M \Downarrow_\tau N$ to mean that there exists an n such that $\Upsilon \parallel M \Downarrow_\tau^n N$.

Notation. *For a family of copresheaves $X_i : \mathbf{J}$ $(i \in I)$, we will use $X^{[\vec{i}]}$ as a shorthand for the following iterated product, as in [22]:*

$$X^{[\vec{i}]} \triangleq \prod_{i \in \vec{i}} X_i$$

Fix a sort-indexed family of binary relations on closed expressions $R_\tau : \mathbf{J}\left[\mathbb{1}, \wp\left(\mathbf{E}_\tau^2\right)\right]$; we will also write the relation as a judgment scheme $\Upsilon \parallel M \; R_\tau \; N$. First of all, we can lift such a relation to operate on open terms by quantifying over substitutions, as follows:

$$\frac{\forall \Upsilon \xhookrightarrow{\rho} \Upsilon'. \quad \forall \vec{X} : \mathbf{E}^{[\vec{\sigma}]}(\Upsilon'). \quad \Upsilon' \parallel \left[\vec{X}/\vec{x}\right] M \cdot \rho \; R_\tau \; \left[\vec{X}/\vec{x}\right] M \cdot \rho}{\Upsilon \parallel \vec{x} : \vec{\sigma} \vDash M \; R_\tau \; N}$$

Next, we can always extend R_τ to a new family of valenced-indexed relations R_ν : $\mathbf{J}\left[\mathbb{1}, \wp\left(\mathbf{B}_\nu^2\right)\right]$ for $\nu \equiv \{\vec{\sigma}\}[\vec{\tau}].\,\tau$, defined pointwise as follows:

$$\frac{\begin{array}{l}\forall \vec{w} \# |\Upsilon| \cup \vec{u} \cup \vec{v}. \quad \forall \vec{z} \# \vec{x} \cup \vec{y}. \\ \Upsilon, \vec{w} : \vec{\sigma} \parallel \vec{z} : \vec{\tau} \vDash [\vec{z}/\vec{x}] M \cdot (\vec{u} \mapsto \vec{w}) \; R_\tau \; [\vec{z}/\vec{y}] M \cdot (\vec{v} \mapsto \vec{w})\end{array}}{\Upsilon \parallel \mathbb{\lambda}\{\vec{u}\}[\vec{x}].\, M \; R_{\{\vec{\sigma}\}[\vec{\tau}].\,\tau} \; \mathbb{\lambda}\{\vec{v}\}[\vec{y}].\, N}$$

As a matter of convenience, we'll also define this judgment over vectors of bound terms and valences:

$$\frac{\forall (E, F, \nu) \in (\vec{E}, \vec{F}, \vec{\nu}).\ \Upsilon \parallel E\ R_\nu\ F}{\Upsilon \parallel \vec{E}\ R_{\vec{\nu}}\ \vec{F}}$$

Now, we can extend R_τ to a new relation $[R_\tau]$ on closed expressions which respects a single "layer" of computation. We will say $\Upsilon \parallel M\ [R_\tau]\ N$ when, supposing $\Upsilon \parallel M \Downarrow_\tau \vartheta(\vec{E})$ such that $\Upsilon \Vdash_{\mathcal{K}} \vartheta : (\vec{\nu})\,\tau$, for any $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, we have $\Upsilon' \parallel N \cdot \rho \Downarrow_\tau \vartheta \cdot \rho(\vec{F} \cdot \rho)$ and $\Upsilon' \parallel \vec{E} \cdot \rho\ R_{\vec{\nu}}\ \vec{F} \cdot \rho$.

Definition 5.1 (Computational Approximation)**.** Because $[-_\tau]$ is monotonic, it has a greatest fixed point, which we will call *computational approximation*, $\preccurlyeq_\tau\ :\ \mathbf{J}\left[\mathbb{1}, \wp\left(\mathbf{E}^2_\tau\right)\right]$, written pointwise as $\Upsilon \parallel M \preccurlyeq_\tau N$.

Theorem 5.2 (Reflexivity)**.** *For all* $M \in \mathbf{E}_\tau(\Upsilon)$*, we have* $\Upsilon \parallel M \preccurlyeq_\tau M$*.*

Proof. The proof proceeds by coinduction, using the relation $M, N \mapsto M = N$ as our motive; then, it suffices to show that for all M, $\Upsilon \parallel M\ [=]\ M$. Expanding the definition of $[-]$, this means that we have to show that when $\Upsilon \parallel M \Downarrow_\tau \vartheta(\vec{E})$, then $\vec{E} = \vec{E}$; but this is always the case, because of the functionality/determinacy of the evaluation relation. □

Definition 5.3 (Computational Equivalence)**.** Two terms are said to be *computationally equivalent* when they approximate each other:

$$\frac{\Upsilon \parallel M \preccurlyeq_\tau N \quad \Upsilon \parallel N \preccurlyeq_\tau M}{\Upsilon \parallel M \sim_\tau N}$$

5.2 Extensionality and Observational Congruence

A sort-indexed family of binary relations $R_\tau : \mathbf{J}\left[\mathbb{1}, \wp\left(\mathbf{E}^2_\tau\right)\right]\ \ (\tau \in \mathcal{S})$ is a *congruence* when for all operators $\Upsilon \Vdash \vartheta : (\vec{\nu})\,\tau$, the following rule obtains:

$$\frac{\Upsilon \parallel \vec{E}\ R_{\vec{\nu}}\ \vec{F}}{\Upsilon \parallel \vartheta(\vec{E})\ R_\tau\ \vartheta(\vec{F})} \qquad \text{(Congruence)}$$

In general, computational equivalence is not a congruence unless we impose a certain extensionality condition on operators. First, let us define a new version of approximation $\preccurlyeq^\star_\tau$ over open terms:

$$\frac{\Upsilon \parallel \Gamma \vDash x \preccurlyeq_\tau y}{\Upsilon \parallel \Gamma \vDash x \preccurlyeq^\star_\tau y}\ \mathit{var} \qquad \frac{\Upsilon \Vdash \vartheta : (\vec{\nu})\,\tau \quad \exists \vec{F}.\quad \Upsilon \parallel \Gamma \vDash \vec{E} \preccurlyeq^\star_{\vec{\nu}} \vec{F} \quad \Upsilon \parallel \Gamma \vDash \vartheta(\vec{F}) \preccurlyeq_\tau N}{\Upsilon \parallel \Gamma \vDash \vartheta(\vec{E}) \preccurlyeq^\star_\tau N}\ \mathit{app}$$

We will write $\Upsilon \parallel M \preccurlyeq^\star_\tau N$ for $\Upsilon \parallel \cdot \vDash M \preccurlyeq^\star_\tau N$. By definition, $\preccurlyeq^\star_\tau$ is a congruence; to show that $\sim_\tau$ is a congruence, it suffices to show that the following inference is valid:

$$\frac{\Upsilon \parallel \Gamma \vDash M \preccurlyeq^\star_\tau N}{\Upsilon \parallel \Gamma \vDash M \preccurlyeq_\tau N}\ ?$$

We will not reproduce the full proof of the above rule here (see [14]), but will simply remark that it holds only for what is called an *extensional lcs*, an lcs in which each operator is extensional in the following sense:

Definition 5.4 (Extensionality)**.** An operator $\Upsilon \Vdash \vartheta : (\vec{\nu})\,\tau$ is *extensional* when, for any $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, for all $\vec{E}, \vec{F} \in \mathbf{B}^{[\vec{\nu}]}(\Upsilon')$ and $M \in \mathbf{E}_\tau(\Upsilon')$ and $k \geqslant 0$, if we can conclude $\Upsilon' \parallel M \preccurlyeq^\star_\tau \vartheta \cdot \rho(\vec{F})$ from the following conditions:

1. We have $\Upsilon' \parallel \vartheta \cdot \rho(\vec{E}) \Downarrow^k_\tau M$.
2. We have $\Upsilon' \parallel \vec{E} \preccurlyeq^\star_{\vec{\nu}} \vec{F}$.
3. For any $\Upsilon' \xhookrightarrow{\rho'} \Upsilon''$, for all $\sigma \in \mathcal{S}$ and $X, X', Y \in \mathbf{E}_\sigma(\Upsilon'')$, if from $\Upsilon'' \parallel X \Downarrow^{k'}_\sigma X'$ for $k' < k$ and $\Upsilon'' \parallel X \preccurlyeq^\star_\sigma Y$, we can conclude $\Upsilon'' \parallel X' \preccurlyeq^\star_\sigma Y$.

Theorem 5.5. *Every canonical operator $\Upsilon \Vdash_{\mathcal{K}} \vartheta : (\vec{\nu})\,\tau$ is extensional.*

Proof. Fix $\Upsilon \xhookrightarrow{\rho} \Upsilon'$and $\vec{E}, \vec{F} \in \mathbf{B}^{[\vec{\nu}]}(\Upsilon')$ and $M \in \mathbf{E}_\tau(\Upsilon')$ and $k \geqslant 0$. By hypothesis (1) from Definition 5.4 above, we have $\Upsilon' \parallel \vartheta \cdot \rho(\vec{E}) \Downarrow^k_\tau M$; because M is a closed term, by the idempotency of evaluation we have $M \equiv \vartheta \cdot \rho(\vec{E})$.

It suffices, then, to show that $\Upsilon' \parallel \vartheta \cdot \rho(\vec{E}) \preccurlyeq^\star_\tau \vartheta \cdot \rho(\vec{F})$. Backward chaining through the *app*-rule of $\preccurlyeq^\star$, we must come up with a certain $\vec{G}$ such that the following subgoals obtain:

1. $\Upsilon' \parallel \vec{E} \preccurlyeq^\star_{\vec{\nu}} \vec{G}$
2. $\Upsilon' \parallel \vartheta \cdot \rho(\vec{G}) \preccurlyeq_\tau \vartheta \cdot \rho(\vec{F})$

Setting $\vec{G} \triangleq \vec{F}$, we discharge the first subgoal through hypothesis (2); the second hypothesis holds because of the reflexivity of approximation (Theorem 5.2).

□

5.3 When is a refinement a type?

As we remarked in section 3.1, the notion of *refinement* that we described is very much akin to a multi-sorted generalization of Martin-Löf's explanation of types in [18], except that we have not required an operational semantics to be present prior to the definition of refinements.

Following Martin-Löf, a type is a refinement which behaves well with respect to computation, in some specified sense; contrary to Martin-Löf, however, we wish to consider a full range of *computational types*, including those which are not necessarily defined by their values; a general apparatus for computation-respecting refinements will allow us to define, for instance, a type for partial functions.

Definition 5.6 (Typehood)**.** For any refinement $\Upsilon \parallel \phi \sqsubset \tau$, we say $\Upsilon \parallel \phi \ \mathit{type}_\tau$ just when for any $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, and for all $L, M, N \in \mathbf{E}_\tau(\Upsilon')$, from $\Upsilon' \parallel L \sim_\tau M$ and $\Upsilon' \parallel M = N \in \phi$ we can conclude $\Upsilon' \parallel L = N \in \phi$.

Stated internally as an object in $\mathbf{J}$, the meaning of parametric typehood for ϕ is perhaps a bit more clear:

$$- \parallel \phi \ \mathit{type}_\tau \triangleq \prod_{L,M,N \in \mathbf{E}_\tau} \phi(L,N)^{L \sim_\tau M \times \phi(M,N)}$$

Let us write $\mathfrak{type}_\tau$ for the object of types,

$$\mathfrak{type}_\tau \triangleq \left\{ \phi \in \mathfrak{per}(\mathbf{E}_\tau) \mid - \parallel \phi \ \mathit{type}_\tau \right\}$$

Now, let us return briefly to Martin-Löf's notion of type, which we call a *value type*. The minimal data of such a type's definition consists in a partial equivalence relation on values, $\phi : \mathbf{J}\left[\mathfrak{K}(\Upsilon), \mathfrak{per}(\mathbf{V}_\tau)\right]$. It is easy to turn such a relation into a computational type $\Upsilon \parallel \mathfrak{c}(\phi) \ \mathit{type}_\tau$, as follows, where $\lambda(\chi)$ is the exponential transpose of χ:

$$\begin{array}{ccc} \mathfrak{K}(\Upsilon) \times \mathbf{V}_\tau^2 & \xrightarrow{\lambda(\phi)} & \Omega \\ {\scriptstyle \langle 1, i \rangle} \downarrow & \nearrow_{\lambda(\mathfrak{c}(\phi))} & \\ \mathfrak{K}(\Upsilon) \times \mathbf{E}_\tau^2 & & \end{array}$$

where $\mathfrak{c}(\phi)$ is the computational extension of ϕ in the following sense:

$$\mathfrak{c}(\phi)(M,N) \triangleq \forall M', N' \in \mathbf{V}_\tau.\ M \sim_\tau M' \wedge N \sim_\tau N' \implies \phi(M',N')$$

Theorem 5.7. *When viewed as an internal functor* $\mathfrak{per}(\mathbf{V}_\tau) \to \mathfrak{type}_\tau$, $\mathfrak{c}$ *is the left adjoint to the canonical inclusion* $\mathfrak{i} : \mathfrak{type}_\tau \hookrightarrow \mathfrak{per}(\mathbf{V}_\tau)$.

Proof. We will exhibit a counit $\epsilon : \mathfrak{c} \circ \mathfrak{i} \to 1_{\mathfrak{per}(\mathbf{V}_\tau)}$ and a unit $\eta : 1_{\mathfrak{type}_\tau} \to \mathfrak{i} \circ \mathfrak{c}$ for the adjunction $\mathfrak{c} \dashv \mathfrak{i}$.

The counit witnesses the fact that for any refinement $\phi : \mathfrak{type}_\tau$ and terms $M, N \in \mathbf{E}_\tau$, if $\mathfrak{c}(\mathfrak{i}(\phi))(M,N)$, then $\phi(M,N)$; this is evidently the case, because the premise only obtains if M, N are computationally equal to values, and ϕ respects computational equivalence by virtue of being a type. The unit witnesses the fact that for any value PER $\psi : \mathfrak{per}(\mathbf{V}_\tau)$

and values $M, N \in \mathbf{V}_\tau$, if $\psi(M, N)$, then $\iota(\mathfrak{c}(\psi))(M, N)$; this is clearly true, since $\mathfrak{c}(\psi)$ can do nothing but coarsen ψ's relation on values.

□

Definition 5.8 (Value Closure)**.** For any computational type $\Upsilon \parallel \phi\ \mathit{type}_\tau$, let $\underline{\phi} \triangleq \mathfrak{c}(\iota(\phi))$ be called ϕ's *value closure*, which we will use shortly in order to give a formal definition to the notion of value types.

Definition 5.9 (Value Types)**.** We can give a simple characterization of Martin-Löf's *value types* in terms of our more general computational types. For any type $\Upsilon \parallel \phi\ \mathit{type}_\tau$, we will say $\Upsilon \parallel \phi\ \mathit{vtype}_\tau$ in case ϕ is equal to its value closure, i.e. $\Upsilon \parallel \phi = \underline{\phi} \sqsubset \tau$.

Functional Typehood Let us define functional typehood $\Upsilon \parallel \Psi \vDash \phi = \psi\ \mathit{type}_\tau$, presupposing $\Upsilon \parallel \Psi \vDash \phi = \psi \sqsubset \tau$. As usual, we give our meaning explanation by induction on the evidence for the validity of the refined context Ψ.

Case $\dfrac{}{\Upsilon \parallel \cdot \sqsubset^\star \cdot}\ \sqsubset^\star_{\mathtt{nil}}$.

$$\frac{\Upsilon \parallel \phi\ \mathit{type}_\tau}{\Upsilon \parallel \cdot \vDash \phi = \psi\ \mathit{type}_\tau}\ \vDash^{\mathit{type}}_{\mathtt{nil}}$$

Note that this case is not trivial, because it is only defined in case the presupposition $\Upsilon \parallel \Psi \vDash \phi = \psi \sqsubset \tau$ obtains. It suffices to require only that ϕ be a type, since because typehood respects equality of refinement.

Case $\dfrac{\Upsilon \parallel \Psi \sqsubset^\star \Gamma \quad \Upsilon \parallel \Psi \vDash \chi \sqsubset \sigma}{\Upsilon \parallel \Psi, x : \chi \sqsubset^\star \Gamma, x : \sigma}\ \sqsubset^\star_{\mathtt{snoc}}$.

To know $\Upsilon \parallel \Psi, x : \chi \vDash \phi = \psi\ \mathit{type}_\tau$ is to know, for any renaming $\Upsilon \xhookrightarrow{\rho} \Upsilon'$ and closed terms $M_0, M_1 \in \mathbf{E}_\sigma(\Upsilon')$, that from $\Upsilon' \parallel \Psi \cdot \rho \vDash M_0 = M_1 \in \chi \cdot \rho$, you can conclude $\Upsilon' \parallel \Psi \cdot \rho \vDash [M_0 / x]\, \phi \cdot \rho = [M_1 / x]\, \psi \cdot \rho\ \mathit{type}_\tau$. In other words:

$$\frac{\begin{array}{l}\forall \Upsilon \xhookrightarrow{\rho} \Upsilon'.\ \forall M_0, M_1 \in \mathbf{E}_\sigma(\Upsilon').\\ (\Upsilon' \parallel \Psi \cdot \rho \vDash M_0 = M_1 \in \chi \cdot \rho)\\ \quad \Longrightarrow \left(\Upsilon' \parallel \Psi \cdot \rho \vDash [M_0 / x]\, \phi \cdot \rho = [M_1 / x]\, \psi \cdot \rho\ \mathit{type}_\tau\right)\end{array}}{\Upsilon \parallel \Psi, x : \chi \vDash \phi = \psi\ \mathit{type}_\tau}\ \vDash^{\mathit{type}}_{\mathtt{snoc}}$$

As with refinements, we will write $\Upsilon \parallel \Psi \vDash \phi\ \mathit{type}_\tau$ to mean $\Upsilon \parallel \Psi \vDash \phi = \phi\ \mathit{type}_\tau$.

5.4 Examples of refinements and their typehood

We will demonstrate how to define certain refinements, and prove that they are types.

5.4.1 The type of closed expressions

The *discrete* refinement on a sort (which equates each closed term only with itself, up to alpha equivalence) is not a type in general, because it may not respect computational equivalence. However, we can form the next best thing, which is called the *base* refinement, which identifies closed expressions of a sort up to computational equivalence.

Let $\mathtt{Base}_\tau \sqsubset \tau$ for any sort τ be defined as follows:

$$\frac{\Upsilon \parallel M \sim_\tau N}{\Upsilon \parallel M = N \in \mathtt{Base}_\tau}$$

Then, we clearly have $\Upsilon \parallel \mathtt{Base}_\tau \; \mathit{type}_\tau$, because it respects computational equivalence by definition.

5.4.2 The type of atoms

Fix an extensional lcs whose signature contains the following:

$$\frac{}{\mathtt{atom}\ \mathit{sort}} \qquad \frac{}{\Upsilon, a : \mathtt{atom} \Vdash_{\mathcal{K}} \mathtt{tok}[a] : ()\,\mathtt{atom}}$$

Then, we can define a refinement $\mathbb{A}_0 \sqsubset \mathtt{atom}$ as follows:

$$\frac{\Upsilon \ni a : \mathtt{atom}}{\Upsilon \parallel \mathtt{tok}[a]() = \mathtt{tok}[a]() \in \mathbb{A}_0}$$

However, it is not the case that $\mathbb{A}_0$ is a type, because it doesn't respect computational equivalence. This is easily resolved by defining the type of atoms as the computational extension of $\mathbb{A}_0$,

$$\mathbb{A} \triangleq \mathfrak{c}(\mathbb{A}_0)$$

Theorem 5.10. *For any Υ, we have $\Upsilon \parallel \mathbb{A}\ \mathit{type}_{\mathtt{atom}}$.*

Proof. Trivial, because the computational extension of any value PER is a type. □

5.4.3 Dependent function types

Fix an extensional lcs such that its signature contains the following:

$$\frac{\sigma\ \mathit{sort} \quad \tau\ \mathit{sort}}{\sigma \rightharpoonup \tau\ \mathit{sort}} \qquad \frac{\sigma\ \mathit{sort} \quad \tau\ \mathit{sort}}{\Upsilon \Vdash \mathtt{ap}^\tau_\sigma : (.\,\sigma \rightharpoonup \tau; .\,\sigma)\,\tau}$$

The sort $\sigma \rightharpoonup \tau$ is intended to classify the (syntax of) partial operations, and $\mathtt{ap}^\tau_\sigma$ is the application operator. Now, we can define a refinement on this sort for dependent functions,

and then prove that it is a type. We intend to define a refinement $\prod_{(x:\phi)} \psi$ such that the following rule holds:

$$\frac{\Upsilon \parallel \phi \sqsubset \sigma \quad \Upsilon \parallel x : \phi \vDash \psi \sqsubset \tau}{\Upsilon \parallel \prod_{(x:\phi)} \psi \sqsubset \sigma \rightharpoonup \tau}$$

For any renaming $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, the meaning of $\prod_{(x:\phi)} \psi$ at $M, N \in \mathbf{E}_{\sigma \rightharpoonup \tau}(\Upsilon')$ is given by the following rule:

$$\frac{\Upsilon' \parallel x : \phi \vDash \mathsf{ap}^{\tau}_{\sigma}(M; x) = \mathsf{ap}^{\tau}_{\sigma}(N; x) \in \psi}{\Upsilon' \parallel M = N \in \prod_{(x:\phi)} \psi}$$

Note that we have not required an introduction form such as $\mathtt{lam}^{\tau}_{\sigma}$, and have instead opted for a negative definition of the dependent function refinement. This is strictly more open-ended than the positive definition, because it does not fix in advance a syntax for the canonical formation of functions.

Theorem 5.11. *The dependent function refinement* $\Upsilon \parallel \prod_{(x:\phi)} \psi \sqsubset \sigma \rightharpoonup \tau$ *is a type assuming* $\Upsilon \parallel \phi$ *type*$_{\sigma}$ *and* $\Upsilon \parallel x : \phi \vDash \psi$ *type*$_{\tau}$.

Proof. Fix $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, and $L, M, N \in \mathbf{E}_{\sigma \rightharpoonup \tau}(\Upsilon')$ such that

1. $\Upsilon' \parallel L \sim_{\sigma \rightharpoonup \tau} M$
2. $\Upsilon' \parallel x : \phi \cdot \rho \vDash \mathsf{ap}^{\tau}_{\sigma}(M; x) = \mathsf{ap}^{\tau}_{\sigma}(N; x) \in \psi \cdot \rho$

It suffices to show that $\Upsilon' \parallel x : \phi \cdot \rho \vDash \mathsf{ap}^{\tau}_{\sigma}(L; x) = \mathsf{ap}^{\tau}_{\sigma}(N; x) \in \psi \cdot \rho$. Now fix $\Upsilon' \xhookrightarrow{\rho'} \Upsilon''$, and $X, Y \in \mathbf{E}_{\sigma}(\Upsilon'')$ such that $\Upsilon'' \parallel X = Y \in \phi \cdot (\rho' \circ \rho)$. Then, we have to show that $\Upsilon'' \parallel \mathsf{ap}^{\tau}_{\sigma}(L \cdot \rho'; X) = \mathsf{ap}^{\tau}_{\sigma}(N \cdot \rho'; Y) \in [X / x] \psi \cdot (\rho' \circ \rho)$.

Because ψ is a functional family of types, $[X / x] \psi \cdot (\rho' \circ \rho)$ must respect computational equivalence, so by drawing hypothesis (1) along ρ', we can rewrite hypothesis (2) into the goal. □

5.5 Refinement Modalities

In Section 5.4.2 we introduced our first *intensional* type, that is, a type whose meaning varies over worlds Υ. It is also possible to introduce *refinement modalities* which change the meaning of a refinement in an intensional manner.

5.5.1 Nominal Independence

Mark Bickford introduced in [2] a type in Nuprl that expresses the independence of a term M of type A from a symbol a, written $M : A \parallel a$, which is used in order to express the notion that some agent is not aware of a particular token/atom. In our setting, nominal

independence is best expressed through a comonadic refinement modality, $\phi \setminus a$, whose inhabitants are the inhabitants of ϕ which do not make use of the symbol a. To be precise, we intend to explain the following refinement formation rule:

$$\frac{\Upsilon, a : \sigma \parallel \phi \sqsubset \tau}{\Upsilon, a : \sigma \parallel \phi \setminus a \sqsubset \tau}$$

The definition of nominal independence is as follows, for any renaming $\Upsilon, a : \sigma \xhookrightarrow{\rho} \Upsilon'$:

$$\frac{\exists L \in \mathbf{E}_\tau(\Upsilon' \setminus \rho(a)). \quad \begin{array}{c}\Upsilon' \parallel L = M \in \phi \\ \Upsilon' \parallel L = N \in \phi\end{array}}{\Upsilon' \parallel M = N \in \phi \setminus \rho(a)}$$

In other words, a construction is independent of a symbol when it is equal to one which was constructed without knowledge of that symbol.

Remark 5.12. Note that we could not have simply used $\Upsilon' \setminus \rho(a) \parallel M = N \in \phi$ as the premise in our definition, since that would be outside the range of significance of this form of judgment (i.e. the presuppositions fail to be satisfied); in particular, it is not guaranteed that the refinement ϕ is defined at $\Upsilon' \setminus \rho(a)$, and nor do we have $M, N \in \mathbf{E}_\tau(\Upsilon' \setminus \rho(a))$.

In fact, far from being a technical nuisance, the definition that is forced by the nature of our presuppositions is completely desirable, since it expresses a *behavioral* notion of independence rather than a *structural* one. Moreover, under the structural definition, even if ϕ is a type, $\phi \setminus a$ might fail to be a type in case there is a member of ϕ that uses a which is merely computationally equivalent to another member that is independent of a.

Theorem 5.13. *The refinement operator $- \setminus a$ is an idempotent comonad on refinements in the sense that the following rules are valid:*

$$\frac{\Upsilon \parallel M = N \in \phi \setminus a}{\Upsilon \parallel M = N \in \phi}\ \textit{extract} \qquad \frac{\Upsilon \parallel M = N \in \phi \setminus a}{\Upsilon \parallel M = N \in \phi \setminus a \setminus a}\ \textit{duplicate}$$

$$\frac{\forall M, N \in \mathbf{E}_\tau(\Upsilon).\ \Upsilon \parallel M = N \in \phi \implies \Upsilon \parallel M = N \in \psi}{\forall M, N \in \mathbf{E}_\tau(\Upsilon).\ \Upsilon \parallel M = N \in \phi \setminus a \implies \Upsilon \parallel M = N \in \psi \setminus a}\ \textit{lift}$$

Proof. The first two inferences are obviously evident. To validate *lift*, suppose that $\Upsilon \parallel M = N \in \phi \setminus a$; by inversion, we have an $L \in \mathbf{E}_\tau(\Upsilon \setminus a)$ such that $\Upsilon \parallel L = M \in \phi$ and $\Upsilon \parallel L = N \in \phi$. Now, it suffices to exhibit an $L' \in \mathbf{E}_\tau(\Upsilon \setminus a)$ such that $\Upsilon \parallel L' = M \in \psi$ and $\Upsilon \parallel L' = N \in \psi$; but we may simply choose $L' \equiv L$ and discharge the remaining subgoals by appealing to the premise of the rule. □

5.5.2 Nominal Necessity

Another comonadic refinement modality is *nominal necessity* $\Upsilon \parallel \Box\phi \sqsubset \tau$ $(\Upsilon \parallel \phi \sqsubset \tau)$, which expresses the independence of an object from *any* atoms at all. For any renaming $\Upsilon \xhookrightarrow{\rho} \Upsilon'$, we define nominal necessity as follows:

$$\frac{\exists L \in \mathbf{E}_\tau(\cdot).\ \begin{array}{l}\Upsilon' \parallel L = M \in \phi \\ \Upsilon' \parallel L = N \in \phi\end{array}}{\Upsilon' \parallel M = N \in \Box\phi}$$

Theorem 5.14. *Nominal independence is an idempotent comonadic modality on refinements.*

Proof. The demonstration is essentially analogous to the one for independence in Theorem 5.13. □

6 Related Work

In [1], Stuart Allen gave a *supervaluation* semantics for Nuprl sequents involving atoms (symbols) by quantifying over plenty of choices for the interpretation of atoms—possible interpretations including strings, finite sets, etc. One benefit of Allen's indirect treatment is that the logical theory remains profoundly open-ended, in the sense that it is neither the case that the naturals may be injected into the atoms, nor is it the case that they cannot.

Unlike in our treatment, where the judgments of type theory are intensional with respect to symbol contexts, Allen takes a more model-theoretic approach and parameterizes the explanation of type theory over a model $\mathfrak{A}$ for the abstract atoms.

A Nuprl sequent $\mathfrak{s} \triangleq H \gg P$ means "P is true under hypotheses H", and it may contain (abstract) atoms in its syntax; we write $\mathbf{FS}(\mathfrak{s})$ for the set of free atoms of the sequent $\mathfrak{s}$. Then, a model $\mathfrak{A}$ consists in a carrier set $|\mathfrak{A}|$ which shall interpret the atoms, and a valuation $\mathfrak{A} \models \mathfrak{s}$ for any Nuprl sequent $\mathfrak{s}$ which takes its atoms from $|\mathfrak{A}|$. A model is called *discrete* in case its carrier set has decidable equality. For a Nuprl sequent $\mathfrak{s}$ with atoms in $\mathfrak{A}$ and an injective function $\rho : |\mathfrak{A}| \hookrightarrow |\mathfrak{A}'|$, let $\mathfrak{s} \cdot \rho$ be the sequent with atoms in $|\mathfrak{A}'|$ got by replacing each atom a with $\rho(a)$.

Then, a supervaluation is given for Nuprl sequents which take atoms in **String**, written $* \models \mathfrak{s}$, which holds if there exists a $k \in \mathbb{N}$ such that for every discrete model $\mathfrak{A}$ having at least k members, for every injection $\rho : \mathbf{FS}(\mathfrak{s}) \hookrightarrow \mathfrak{A}$, we have $\mathfrak{A} \models \mathfrak{s} \cdot \rho$. Under the supervaluation, we can rule out certain observations about atoms which might hold in particular models.

For instance, assuming a type $\mathbb{N}$ of natural numbers and $\mathbb{A}$ of atoms, with $\mathbb{N} \hookrightarrow \mathbb{A}$ the type of embeddings of the naturals into the atoms, we do have $\mathbf{String} \models \cdot \gg \mathbb{N} \hookrightarrow \mathbb{A}$, because there are numerous ways to embed the naturals into the strings; however, because the embedding fails in every finite model $\mathfrak{A}$, we do not have $* \models \cdot \gg \mathbb{N} \hookrightarrow \mathbb{A}$. Furthermore, we also do not have $* \models \iota : \mathbb{N} \hookrightarrow \mathbb{A} \gg \bot$, because this would fail in any infinite model.

Comparison with the present work Our explanation of atoms is not equivalent to the supervaluation semantics. First, let us introduce a notion analogous to the Nuprl sequent: when $\cdot \parallel \Phi \sqsubset^\star \Gamma$ and $\cdot \parallel \Phi \vDash \phi \sqsubset \tau$, we can form the sequent $\Phi \gg \phi$, meaning that there exists an $\Upsilon \in \mathbb{I} \downarrow \mathcal{S}$ and $M \in \mathbf{E}_\tau(\Upsilon \parallel \Gamma)$ such that $\Upsilon \parallel \Phi \vDash M = M \in \phi$. We will write $\models \Phi \gg \phi$ when $\Phi \gg \phi$ obtains.

Theorem 6.1. *As in the supervaluation semantics, we have* $\not\models \cdot \gg \mathbb{N} \hookrightarrow \mathbb{A}$.

Proof. Suppose $\models \cdot \gg \mathbb{N} \hookrightarrow \mathbb{A}$. Then, there exists an $\Upsilon \in \mathbb{I} \downarrow \mathcal{S}$ and $M \in \mathbf{E}_\tau(\Upsilon)$ such that $\Upsilon \parallel M = M \in \mathbb{N} \hookrightarrow \mathbb{A}$; but this is impossible because it would require a means of embedding the naturals into finite $|\Upsilon|$. □

Theorem 6.2. *Contrary to the supervaluation semantics, we do have* $\models \iota : \mathbb{N} \hookrightarrow \mathbb{A} \gg \bot$.

Proof. It suffices to show that for all Υ, there is no ι such that $\Upsilon \parallel \iota = \iota \in \mathbb{N} \hookrightarrow \mathbb{A}$. However, by the definition of $\mathbb{I} \downarrow \mathcal{S}$, all $\Upsilon \in \mathbb{I} \downarrow \mathcal{S}$ are finite, so there can be no injection of the naturals into $|\Upsilon|$. □

Another seemingly paradoxical observation is possible:

Theorem 6.3. *The atoms may not be embedded into the naturals, i.e.* $\models \iota : \mathbb{A} \hookrightarrow \mathbb{N} \gg \bot$.

Proof. Fix a symbol context Υ; suppose we have an $\iota \in \mathbf{E}(\Upsilon)$ such that $\Upsilon \parallel \iota = \iota \in \mathbb{A} \hookrightarrow \mathbb{N}$. Then, by inversion we have $\Upsilon \parallel x : \mathbb{A} \vDash \mathtt{ap}(\iota; x) = \mathtt{ap}(\iota; x) \in \mathbb{N}$; in other words, for all $\Upsilon \xhookrightarrow{\rho} \Upsilon'$ and $a \in |\Upsilon|$, we have $\Upsilon' \parallel \mathtt{ap}(\iota; a) = \mathtt{ap}(\iota; a) \in \mathbb{N}$ such that ι is injective. But the only way for ι to assign a different index to each atom is for it to test each atom in $|\Upsilon'|$ separately—but ι may only mention the atoms in $|\Upsilon|$. □

Should the above be worrisome? At the very least, the fact that neither the naturals nor the atoms may be embedded into the other commits us to a truly intuitionistic ontology, as opposed to a merely constructive one. In particular, our framework is anti-classical in at least a naïve sense, because the Axiom of Choice puts each set in bijection with a cardinal, whence it is impossible for neither of two sets to embed in the other. However, it is important to note that as far as the propositions-as-types principle is concerned, Computational Type Theory is already anti-classical, as the undecidability of the halting problem can be observed internally.

In fact, we view the simultaneous validity of Theorems 6.2 and 6.3 as totally desirable, because we adhere to a strictly Brouwerian conception of data. The proper way to view the type of atoms is as an ongoing act of free generation—and it is a basic fact concerning the activity of an idealized mathematician that at each stage, only a finite number of constructions shall have been effected so far.

On the other hand, (the analogues of) these theorems are *not* valid under Allen's supervaluation semantics, because one would have to account for worlds at which an

infinite number of atoms had already been generated; in this way, the open-endedness of Allen's semantics relies crucially on a condition that is simply impossible under a Brouwerian conception of mathematical activity.

Acknowledgements

The first author wishes to thank Robert Harper for the inspiration to pursue a generalization of the Nuprl type theory from the refinements perspective. We also thank Mark Bickford for clarifying the status of atoms and generative phenomena in Nuprl; finally, we owe a debt of gratitude to Vincent Rahli and Abhishek Anand for their tireless work on the *Nuprl in Coq* project, which was helpful in our efforts to understand Howe's computational equivalence and its generalization to the nominal setting.

References

[1] S. Allen. An abstract semantics for atoms in Nuprl. Technical report, 2006.

[2] M. Bickford. Unguessable atoms: A logical foundation for security. In N. Shankar and J. Woodcock, editors, *Verified Software: Theories, Tools, Experiments*, volume 5295 of *Lecture Notes in Computer Science*, pages 30–53. Springer Berlin Heidelberg, 2008.

[3] J. Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209 – 243, 1986.

[4] 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.

[5] P. Dybjer. Internal type theory. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer Berlin Heidelberg, 1996.

[6] P. Dybjer. Program testing and the meaning explanations of intuitionistic type theory. In P. Dybjer, S. Lindström, E. Palmgren, and G. Sundholm, editors, *Epistemology versus Ontology*, volume 27 of *Logic, Epistemology, and the Unity of Science*, pages 215–241. Springer, 2012.

[7] M. Fiore and C.-K. Hur. Second-order equational logic (extended abstract). In A. Dawar and H. Veith, editors, *Computer Science Logic*, volume 6247 of *Lecture Notes in Computer Science*, pages 320–335. Springer Berlin Heidelberg, 2010.

[8] N. Gambino. *Sheaf interpretations for generalized predicative systems*. PhD thesis, University of Manchester, 2002.

[9] R. Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, New York, NY, USA, 2016.

[10] R. Harper and R. Davies. Refining Objects (Preliminary Summary). `https://www.cs.cmu.edu/~rwh/papers/lc60/lc60.pdf`, 2014.

[11] R. Harper and W. Duff. Type Refinements in an Open World. `https://www.cs.cmu.edu/~rwh/papers/trow/summary.pdf`, 2015.

[12] A. Heyting. *Intuitionism, an Introduction*. Studies in Logic and the Foundations of Mathematics. North-Holland, 1956. Revised edition, 1966.

[13] J. Hickey, A. Nogin, R. L. Constable, B. E. Aydemir, E. Barzilay, Y. Bryukhov, R. Eaton, A. Granicz, A. Kopylov, C. Kreitz, V. N. Krupski, L. Lorigo, S. Schmitt, C. Witty, and X. Yu. MetaPRL – a modular logical environment. In D. Basin and B. Wolff, editors, *Theorem Proving in Higher Order Logics*, volume 2758 of *Lecture Notes in Computer Science*, pages 287–303. Springer Berlin Heidelberg, 2003.

[14] D. J. Howe. Equality in lazy computation systems. In *Proceedings of Fourth IEEE Symposium on Logic in Computer Science*, pages 198–203.

[15] A. Kopylov. Dependent intersection: A new way of defining records in type theory. Technical report, Ithaca, NY, 2000.

[16] S. Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer-Verlag, 1971.

[17] P. Martin-Löf. An intuitionistic theory of types, 1972.

[18] P. Martin-Löf. Constructive mathematics and computer programming. In L. J. Cohen, J. Łoś, H. Pfeiffer, and K.-P. Podewski, editors, *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science, Hannover 1979*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 153–175. North-Holland, 1982.

[19] P. Martin-Löf and G. Sambin. *Intuitionistic type theory*. Studies in proof theory. Bibliopolis, Napoli, 1984.

[20] D. Scott. Constructive validity. In M. Laudet, D. Lacombe, L. Nolin, and M. Schtzenberger, editors, *Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 237–275. Springer Berlin Heidelberg, 1970.

[21] J. Sterling, D. Gratzer, and V. Rahli. JonPRL. `http://www.jonprl.org/`, 2015.

[22] J. Sterling and D. Morrison. Syntax and Semantics of Abstract Binding Trees. 2015.