Staged Compilation With Two-Level Type Theory (Appendix)

ANDRÁS KOVÁCS, Eötvös Loránd University, Hungary

In this appendix, we give two complete specifications of models of 2LTT, one in the style of derivation rules, and one as a second-order algebraic signature.

1 A RULE-BASED PRESENTATION OF 2LTT

In this section, we give a rule-based description of the notion of model of 2LTT. This is the same as what is described in Section 3.3 of the paper, but presented in a way which is closer to the conventional rule-based specification of lambda calculi.

Note that we do not only specify the notion of *syntax* for the theory, but more generally the notion of *model*, because the rules constitute a signature for an algebraic theory. We do a small warmup example first.

1.1 Warmup: Signature for Monoids

The theory of monoids is a simple example of an algebraic theory. We can write the following signature. We implicitly universally quantify over the x, y, z variables.

$$\begin{aligned} C &: \mathrm{Set} \\ - + - &: C \to C \to C \\ e &: C \\ \mathsf{id}_\mathsf{l} &: e + x = x \\ \mathsf{id}_\mathsf{r} &: x + e = x \\ \mathsf{assoc} &: x + (y + z) = (x + y) + z \end{aligned}$$

A model of the theory of monoids is just a monoid: a set C together with an associative operation and an identity element. The initial monoid is the trivial monoid which has only e in the underlying set. We can rephrase the above signature using judgments and rules:

$$\boxed{x \vdash} \qquad \textit{means } x : C$$

$$\boxed{x = y \vdash} \qquad \textit{means } x = y \textit{ assuming } x : C \textit{ and } y : C$$

$$\textsc{op}\ \frac{x \vdash \quad y \vdash}{x + y \vdash} \qquad \textsc{identity}\ \frac{}{e \vdash} \qquad \textsc{left-identity}\ \frac{x \vdash}{e + x = x \vdash} \qquad \textsc{right-identity}\ \frac{x \vdash}{x + e = x \vdash} \qquad \textsc{associativity}\ \frac{x \vdash \quad y \vdash \quad z \vdash}{x + (y + z) = (x + y) + z \vdash}$$

Additionally, we assume that $- = - \vdash$ behaves as metatheoretic equality: it is an equivalence, it is respected by all rules and constructions, and we can freely replace x with y whenever $x = y \vdash$ is derivable.

Here, the derivation rule notation is a bit peculiar and verbose. But note that the same phenomenon happens in the specification of type theories, where the "signature" notation is much more compact than the derivation rule notation. As far as the author of this work sees, this is a strong

Author's address: András Kovács, kovacsandras@inf.elte.hu, Eötvös Loránd University, Hungary, Budapest.

motivation for generally moving away from derivation rule notations in the metatheory of type theories. This is regardless whether we specify conversion using metatheoretic equality or using explicit relations; recall Section 3.3.3 in the paper where the latter style is sketched.

Nevertheless, the mental switch from rule-based presentation to algebraic signatures is not a trivial task, so in the following we present the signature for 2LTT using derivation rules.

1.2 Judgments of 2LTT

First we look at the judgments. Below, and in all following rules in this section, we assume that $i \in \{0, 1\}$ and $j \in \mathbb{N}$. We use Γ, Δ to refer to contexts, σ, δ for substitutions, A, B, C for types and t, u, v for terms.

$\boxed{\Gamma \vdash}$	*context formation*
$\boxed{\Gamma \vdash \sigma : \Delta}$	*explicit substitution formation, assuming* $\Gamma \vdash$ *and* $\Delta \vdash$
$\boxed{\Gamma \vdash \sigma = \delta : \Delta}$	*substitution equality, assuming* $\Gamma \vdash \sigma : \Delta$ *and* $\Gamma \vdash \delta : \Delta$
$\boxed{\Gamma \vdash_{i,j} A}$	*type formation, assuming* $\Gamma \vdash$
$\boxed{\Gamma \vdash_{i,j} t : A}$	*term formation, assuming* $\Gamma \vdash_{i,j} A$
$\boxed{\Gamma \vdash_{i,j} A = B}$	*type equality, assuming* $\Gamma \vdash_{i,j} A$ *and* $\Gamma \vdash_{i,j} B$
$\boxed{\Gamma \vdash_{i,j} t = u : A}$	*term equality, assuming* $\Gamma \vdash_{i,j} t : A$ *and* $\Gamma \vdash_{i,j} u : A$

The judgments correspond to the prefix of the signature which specifies the underlying sets ("sorts"). If equality judgments are identified with metatheoretic equality, that is automatically available, so we do not have to introduce them in the signature.

$$
\begin{aligned}
&\mathsf{Con} &&: \mathsf{Set}\\
&\mathsf{Sub} &&: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}\\
&\mathsf{Ty}_{i,j} &&: \mathsf{Con} \to \mathsf{Set}\\
&\mathsf{Tm}_{i,j} &&: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}_{i,j}\,\Gamma \to \mathsf{Set}
\end{aligned}
$$

In the following we only present the rule-based specification. We will often omit assumptions from rules which are implied from other judgments. For example, if $\Gamma \vdash_{i,j} A$ is assumed, we can omit $\Gamma \vdash$.

1.3 Core Substitution Calculus

$$\textsc{empty-cxt}\ \frac{}{\bullet \vdash} \qquad \textsc{cxt-extension}\ \frac{\Gamma \vdash \qquad \Gamma \vdash_{i,j} A}{\Gamma \triangleright A \vdash} \qquad \textsc{identity-sub}\ \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{id} : \Gamma} \qquad \textsc{sub-composition}\ \frac{\Delta \vdash \sigma : \Theta \qquad \Gamma \vdash \delta : \Delta}{\Gamma \vdash \sigma \circ \delta : \Theta}$$

$$\textsc{empty-sub}\ \frac{\Gamma \vdash}{\Gamma \vdash \epsilon : \bullet} \qquad \textsc{empty-sub-}\eta\ \frac{\Gamma \vdash \sigma : \bullet}{\Gamma \vdash \sigma = \epsilon : \bullet} \qquad \textsc{id-left}\ \frac{\Gamma \vdash \sigma : \Delta}{\Gamma : \mathsf{id} \circ \sigma = \sigma : \Delta} \qquad \textsc{id-right}\ \frac{\Gamma \vdash \sigma : \Delta}{\Gamma : \sigma \circ \mathsf{id} = \sigma : \Delta}$$

COMP-ASSOC
$$\frac{\Theta \vdash \sigma : \Xi \qquad \Delta \vdash \delta : \Theta \qquad \Gamma \vdash \sigma : \Delta}{\Gamma : \sigma \circ (\delta \circ \nu) = (\sigma \circ \delta) \circ \nu : \Xi}$$

TYPE-SUB
$$\frac{\Delta \vdash_{i,j} A \qquad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,j} A[\sigma]}$$

TERM-SUB
$$\frac{\Delta \vdash_{i,j} t : A \qquad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,j} t[\sigma] : A[\sigma]}$$

TYPE-ID-SUB
$$\frac{\Gamma \vdash_{i,j} A}{\Gamma \vdash_{i,j} A[\mathsf{id}] = A}$$

TERM-ID-SUB
$$\frac{\Gamma \vdash_{i,j} t : A}{\Gamma \vdash_{i,j} t[\mathsf{id}] = t : A}$$

TYPE-COMP-SUB
$$\frac{\Theta \vdash_{i,j} A \qquad \Delta \vdash \delta : \Theta \qquad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,j} A[\sigma \circ \delta] = A[\sigma][\delta]}$$

TERM-COMP-SUB
$$\frac{\Theta \vdash_{i,j} t : A \qquad \Delta \vdash \delta : \Theta \qquad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,j} t[\sigma \circ \delta] = t[\sigma][\delta] : A[\sigma][\delta]}$$

SUB-EXTENSION
$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Gamma \vdash t : A[\sigma]}{\Gamma \vdash (\sigma,\, t) : \Delta \triangleright A}$$

SUB-FIRST-PROJ
$$\frac{\Gamma \vdash_{i,j} A}{\Gamma \triangleright A \vdash \mathsf{p} : \Gamma}$$

SUB-SECOND-PROJ
$$\frac{\Gamma \vdash_{i,j} A}{\Gamma \triangleright A \vdash \mathsf{q} : A[\mathsf{p}]}$$

p-β
$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Gamma \vdash_{i,j} t : A[\sigma]}{\Gamma \vdash \mathsf{p} \circ (\sigma,\, t) = \sigma : \Delta}$$

q-β
$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Gamma \vdash_{i,j} t : A[\sigma]}{\Gamma \vdash_{i,j} \mathsf{q}[\sigma,\, t] = t : A[\sigma]}$$

SUB-EXTENSION-η
$$\frac{}{\Gamma \vdash (\mathsf{p},\, \mathsf{q}) = \mathsf{id} : \Gamma}$$

SUB-EXTENSION-$\circ$
$$\frac{\Delta \vdash \sigma : \Theta \qquad \Delta \vdash_{i,j} t : A[\sigma] \qquad \Gamma \vdash \delta : \Delta}{\Gamma \vdash (\sigma,\, t) \circ \delta = (\sigma \circ \delta,\, t[\delta]) : \Theta \triangleright A}$$

As we noted in Definition 3.2 in the paper, we recover De Bruijn indices in the following way: the n-th index is $\mathsf{q}[\mathsf{p}^n]$, where p^n is the n-fold composition of p (0-fold composition is id).

1.4 Lifting Structure

LIFT
$$\frac{\Gamma \vdash_{0,j} A}{\Gamma \vdash_{1,j} \Uparrow A}$$

LIFT-SUB
$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash_{0,j} A}{\Gamma \vdash_{1,j} (\Uparrow A)[\sigma] = \Uparrow (A[\sigma])}$$

QUOTE
$$\frac{\Gamma \vdash_{0,j} t : A}{\Gamma \vdash_{1,j} \langle t \rangle : \Uparrow A}$$

QUOTE-SUB
$$\frac{\Delta \vdash_{0,j} t : A \qquad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{1,j} \langle t \rangle[\sigma] = \langle t[\sigma] \rangle : \Uparrow (A[\sigma])}$$

SPLICE
$$\frac{\Gamma \vdash_{1,j} t : \Uparrow A}{\Gamma \vdash_{0,j} {\sim} t : A}$$

QUOTE-SPLICE
$$\frac{\Gamma \vdash_{1,j} t : \Uparrow A}{\Gamma \vdash_{1,j} \langle {\sim} t \rangle = t : \Uparrow A}$$

SPLICE-QUOTE
$$\frac{\Gamma \vdash_{0,j} t : A}{\Gamma \vdash_{0,j} {\sim}\langle t \rangle = t : A}$$

Here, note that the substitution rule for splicing is omitted, because it can be derived from the fact that splicing is a bijection; in other words, if one component map of an isomorphism is natural, so is the other map.

1.5 Universes

UNIVERSE
$$\frac{}{\Gamma \vdash_{i,j+1} \mathsf{U}_j}$$

UNIVERSE-SUB
$$\frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,j+1} \mathsf{U}_j[\sigma] = \mathsf{U}_j}$$

UNIVERSE-DECODING
$$\frac{\Gamma \vdash_{i,j+1} t : \mathsf{U}_j}{\Gamma \vdash_{i,j} \mathsf{El}\, t}$$

DECODING-SUB
$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash_{i,j+1} t : \mathsf{U}_j}{\Gamma \vdash_{i,j} (\mathsf{El}\, t)[\sigma] = \mathsf{El}(t[\sigma])}$$

UNIVERSE-ENCODING
$$\frac{\Gamma \vdash_{i,j} A}{\Gamma \vdash_{i,j+1} \mathsf{Code}\, A : \mathsf{U}_j}$$

DECODING-ENCODING
$$\frac{\Gamma \vdash_{i,j} A}{\Gamma \vdash_{i,j} \mathsf{El}\,(\mathsf{Code}\, A) = A}$$

ENCODING-DECODING
$$\frac{\Gamma \vdash_{i,j+1} t : \mathsf{U}_j}{\Gamma \vdash_{i,j+1} \mathsf{Code}\,(\mathsf{El}\, t) = t : \mathsf{U}_j}$$

1.6 Σ-types

$$\frac{\Gamma \vdash_{i,j} A \qquad \Gamma \triangleright A \vdash_{i,j} B}{\Gamma \vdash_{i,j} \Sigma\, A\, B}\ \Sigma\text{-formation} \qquad \frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash_{i,j} A \qquad \Delta \triangleright A \vdash_{i,j} B}{\Gamma \vdash_{i,j} (\Sigma\, A\, B)[\sigma] = \Sigma\, (A[\sigma])\, (B[\sigma \circ \mathsf{p},\, \mathsf{q}])}\ \Sigma\text{-sub}$$

$$\frac{\Gamma \vdash_{i,j} t : A \qquad \Gamma \vdash_{i,j} u : B[\mathsf{id},\, t]}{\Gamma \vdash_{i,j} (t,\, u) : \Sigma\, A\, B}\ \text{pairing} \qquad \frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash_{i,j} t : A \qquad \Delta \vdash_{i,j} u : B[\mathsf{id},\, t]}{\Gamma \vdash_{i,j} (t,\, u)[\sigma] = (t[\sigma],\, u[\sigma]) : \Sigma\, (A[\sigma])\, (B[\sigma \circ \mathsf{p},\, \mathsf{q}])}\ \text{pairing-sub}$$

$$\frac{\Gamma \vdash_{i,j} t : \Sigma\, A\, B}{\Gamma \vdash_{i,j} \mathsf{fst}\, t : A}\ \text{fst} \qquad \frac{\Gamma \vdash_{i,j} t : \Sigma\, A\, B}{\Gamma \vdash_{i,j} \mathsf{snd}\, t : B[\mathsf{id},\, \mathsf{fst}\, t]}\ \text{snd} \qquad \frac{\Gamma \vdash_{i,j} t : A \qquad \Gamma \vdash_{i,j} u : B[\mathsf{id},\, t]}{\Gamma \vdash_{i,j} \mathsf{fst}\, (t,\, u) = t : A}\ \text{fst-}\beta$$

$$\frac{\Gamma \vdash_{i,j} t : A \qquad \Gamma \vdash_{i,j} u : B[\mathsf{id},\, t]}{\Gamma \vdash_{i,j} \mathsf{snd}\, (t,\, u) = u : B[\mathsf{id},\, t]}\ \text{snd-}\beta \qquad \frac{\Gamma \vdash_{i,j} t : \Sigma\, A\, B}{\Gamma \vdash_{i,j} (\mathsf{fst}\, t,\, \mathsf{snd}\, t) = t : \Sigma\, A\, B}\ \Sigma\text{-}\eta$$

1.7 Π-types

$$\frac{\Gamma \vdash_{i,j} A \qquad \Gamma \triangleright A \vdash_{i,j} B}{\Gamma \vdash_{i,j} \Pi\, A\, B}\ \Pi\text{-formation} \qquad \frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash_{i,j} A \qquad \Delta \triangleright A \vdash_{i,j} B}{\Gamma \vdash_{i,j} (\Pi\, A\, B)[\sigma] = \Pi\, (A[\sigma])\, (B[\sigma \circ \mathsf{p},\, \mathsf{q}])}\ \Pi\text{-sub}$$

$$\frac{\Gamma \triangleright A \vdash_{i,j} t : B}{\Gamma \vdash_{i,j} \mathsf{lam}\, t : \Pi\, A\, B}\ \text{lam} \qquad \frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \triangleright A \vdash_{i,j} t : B}{\Gamma \vdash_{i,j} (\mathsf{lam}\, t)[\sigma] = \mathsf{lam}\, (t[\sigma \circ \mathsf{p},\, \mathsf{q}]) : \Pi\, (A[\sigma])\, (B[\sigma \circ \mathsf{p},\, \mathsf{q}])}\ \text{lam-sub}$$

$$\frac{\Gamma \triangleright A \vdash_{i,j} t : B}{\Gamma \triangleright A \vdash_{i,j} \mathsf{app}\, (\mathsf{lam}\, t) = t : B}\ \text{app-lam} \qquad \frac{\Gamma \vdash_{i,j} t : \Pi\, A\, B}{\Gamma \vdash_{i,j} \mathsf{lam}\, (\mathsf{app}\, t) = t : \Pi\, A\, B}\ \text{lam-app}$$

Like before, for universes, Σ-types and Π-types, it is enough to specify substitution rules in one direction of isomorphisms.

1.8 Natural Numbers

$$\frac{}{\Gamma \vdash_{i,j} \mathsf{Nat}}\ \text{nat-formation} \qquad \frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,j} \mathsf{Nat}[\sigma] = \mathsf{Nat}}\ \text{nat-sub} \qquad \frac{}{\Gamma \vdash_{i,j} \mathsf{zero} : \mathsf{Nat}}\ \text{zero}$$

$$\frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,j} \mathsf{zero}[\sigma] = \mathsf{zero} : \mathsf{Nat}}\ \text{zero-sub} \qquad \frac{\Gamma \vdash_{i,j} t : \mathsf{Nat}}{\Gamma \vdash_{i,j} \mathsf{suc}\, t : \mathsf{Nat}}\ \text{suc} \qquad \frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash_{i,j} t : \mathsf{Nat}}{\Gamma \vdash_{i,j} (\mathsf{suc}\, t)[\sigma] = \mathsf{suc}\, (t[\sigma]) : \mathsf{Nat}}\ \text{suc-sub}$$

$$\textsc{nat-elim}\quad \frac{\Gamma \triangleright \mathsf{Nat} \vdash_{i,k} P \quad \Gamma \vdash_{i,k} z : P[\mathsf{id},\ \mathsf{zero}] \quad \Gamma \triangleright \mathsf{Nat} \triangleright P \vdash_{i,k} s : P[\mathsf{p} \circ \mathsf{p},\ \mathsf{suc}\,(\mathsf{q}[\mathsf{p}])] \quad \Gamma \vdash_{i,j} t : \mathsf{Nat}}{\Gamma \vdash_{i,k} \mathsf{NatElim}\, P\, z\, s\, t : P[\mathsf{id},\ t]}$$

$$\textsc{nat-elim-sub}\quad \frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash_{i,k} (\mathsf{NatElim}\, P\, z\, s\, t)[\sigma] = \mathsf{NatElim}\,(P[\sigma \circ \mathsf{p},\ q])\,(z[\sigma])\,(s[\sigma \circ \mathsf{p} \circ \mathsf{p},\ \mathsf{q}[\mathsf{p}],\ \mathsf{q}])\,(t[\sigma]) : P[\sigma,\ t[\sigma]]}$$

$$\textsc{zero-}\beta\quad \frac{}{\Gamma \vdash_{i,j} \mathsf{NatElim}\, P\, z\, s\, \mathsf{zero} = z : P[\mathsf{id},\ \mathsf{zero}]}$$

$$\textsc{suc-}\beta\quad \frac{}{\Gamma \vdash_{i,j} \mathsf{NatElim}\, P\, z\, s\,(\mathsf{suc}\, t) = s[\mathsf{id},\ t,\ \mathsf{NatElim}\, P\, z\, s\, t] : P[\mathsf{id},\ \mathsf{suc}\, t]}$$

2 A SECOND-ORDER ALGEBRAIC SIGNATURE FOR 2LTT

The presentation in the previous section was still quite verbose; although note that it is a *complete* set of rules. Using explicit conversion relations the number of rules would increase greatly, because we would need to specify the congruence closure rules and the coercion/coherence rules everywhere.

A signature-style presentation is more concise the above rule-based one, but we can do even better. There is a style of specification which dramatically reduces the boilerplate, based on the following observations:

- Everything must be stable under substitution.
- Contexts are threaded through rules in a mechanical way.

This might be familiar from the semantics of 2LTT itself: by working in the internal language of presheaves over a category of contexts, we can entirely skip talking about contexts and substitution rules. This is a *higher-order abstract syntax* presentation, alternatively called a *logical framework* [1] presentation. In this specific case, 2LTT has a *second-order* signature, which means that rules can have assumptions which themselves universally quantify over terms.

In the following we present such a signature. We will not detail the syntax and the semantics; the syntax of the signature below is meant to be a variation on Uemura's second-order signatures for representable map categories [2]. As additional notation, we write isomorphisms as $(f,\ g) : A \simeq B$, where $f : A \to B$ with inverse g.

$$\begin{array}{ll}
\mathsf{Ty}_{i,j} & : \mathsf{Set} \\
\mathsf{Tm}_{i,j} & : \mathsf{Ty}_{i,j} \to \mathsf{Set}^{\mathsf{rep}} \\
\Uparrow & : \mathsf{Ty}_{0,j} \to \mathsf{Ty}_{1,j} \\
(\langle - \rangle,\ \sim -) & : \mathsf{Tm}_{0,j}\, A \simeq \mathsf{Tm}_{1,j}\,(\Uparrow A) \\
\mathsf{U}_j & : \mathsf{Ty}_{i,j+1} \\
(\mathsf{El},\ \mathsf{Code}) & : \mathsf{Tm}_{i,j+1}\, \mathsf{U}_j \simeq \mathsf{Ty}_{i,j}
\end{array}$$

$$
\begin{array}{ll}
\Sigma & : (A : \mathsf{Ty}_{i,j}) \to (\mathsf{Tm}_{i,j}\, A \to \mathsf{Ty}_{i,j}) \to \mathsf{Ty}_{i,j} \\
(\mathsf{proj},\ (-,-)) & : \mathsf{Tm}_{i,j}\, (\Sigma\, A\, B) \simeq ((t : \mathsf{Tm}_{i,j}\, A) \times \mathsf{Tm}_{i,j}\, (B\, t)) \\
\Pi & : (A : \mathsf{Ty}_{i,j}) \to (\mathsf{Tm}_{i,j}\, A \to \mathsf{Ty}_{i,j}) \to \mathsf{Ty}_{i,j} \\
(\mathsf{app},\ \mathsf{lam}) & : \mathsf{Tm}_{i,j}\, (\Pi\, A\, B) \simeq ((t : \mathsf{Tm}_{i,j}\, A) \to \mathsf{Tm}_{i,j}\, (B\, t)) \\
\mathsf{Nat} & : \mathsf{Ty}_{i,j} \\
\mathsf{zero} & : \mathsf{Tm}_{i,j}\, \mathsf{Nat} \\
\mathsf{suc} & : \mathsf{Tm}_{i,j}\, \mathsf{Nat} \to \mathsf{Tm}_{i,j}\, \mathsf{Nat} \\
\mathsf{NatElim} & : (P : \mathsf{Tm}_{i,j}\, \mathsf{Nat} \to \mathsf{Ty}_{i,k}) \to \mathsf{Tm}_{i,k}\, (P\, \mathsf{zero}) \\
& \quad \to ((n : \mathsf{Tm}_{i,j}\, \mathsf{Nat}) \to \mathsf{Tm}_{i,k}\, (P\, n) \to \mathsf{Tm}_{i,j}\, (P\, (\mathsf{suc}\, n))) \\
& \quad \to (n : \mathsf{Tm}_{i,j}\, \mathsf{Nat}) \to \mathsf{Tm}_{i,k}\, (P\, n) \\
\mathsf{zero}\beta & : \mathsf{NatElim}\, P\, z\, s\, \mathsf{zero} = z \\
\mathsf{suc}\beta & : \mathsf{NatElim}\, P\, z\, s\, (\mathsf{suc}\, n) = s\, n\, (\mathsf{NatElim}\, P\, z\, s\, n)
\end{array}
$$

Note the $\mathsf{Set}^{\mathsf{rep}}$ in the return type of $\mathsf{Tm}_{i,j}$ above. This marks the sort of terms as being *locally representable*: this means that we can use second-order quantification over terms, for instance in the specification of Π.

Remarkably, the above signature can be mechanically translated to the verbose rule set. Informally, the signature is interpreted in presheaves over an implicit underlying category of contexts, so that we recover a naturality condition for each rule, corresponding to the substitution rule. Abstraction over locally representable sorts is interpreted as context extension. For instance, the second-order function space $\mathsf{Tm}_{i,j}\, A \to \mathsf{Ty}_{i,j}$ in the type of Π is translated to a set of types in a context extended with A.

REFERENCES

[1] Robert Harper, Furio Honsell, and Gordon D. Plotkin. 1993. A Framework for Defining Logics. *J. ACM* 40, 1 (1993), 143–184. https://doi.org/10.1145/138027.138060

[2] Taichi Uemura. 2019. A General Framework for the Semantics of Type Theory. *CoRR* abs/1904.04097 (2019). arXiv:1904.04097 http://arxiv.org/abs/1904.04097