JFP **25**, e19, 30 pages, 2015.
doi:10.1017/S0956796815000210

Pure type systems with explicit substitutions

DANIEL FRIDLENDER and MIGUEL PAGANO
FaMAF, Universidad Nacional de Córdoba, Córdoba, Argentina
(*e-mail:* fridlend@famaf.unc.edu.ar,pagano@famaf.unc.edu.ar)

Abstract

We introduce a new formulation of pure type systems (PTSs) with explicit substitution and de Bruijn indices and formally prove some of its meta-theory. Using techniques based on Normalisation by Evaluation, we prove that untyped conversion can be typed for predicative PTSs. Although this equivalence was settled by Siles and Herbelin for the conventional presentation of PTSs, we strongly conjecture that our proof method can also be applied to PTSs with η.

1 Introduction

In this article, we introduce a new formulation of pure type systems (PTSs) with explicit substitutions and de Bruijn indices. We consider two formulations: a *semantical* variant with typed equality and a *syntactical* version with untyped conversion. For both of them, we formally develop some of its meta-theory including the substitution lemma.

Whereas the first formulation is convenient for theoretical considerations (Streicher, 1989; Miquel & Werner, 2002), the latter is often used in implementations like Coq (2004) and Matita (Asperti *et al.*, 2011). The kernels of these systems are implemented with terms using de Bruijn indices instead of named variables (Barras, 1998) and substitutions are handled explicitly rather than as meta-level operations.

The equivalence between these two formulations bridges the gap between the systems usually considered in theoretical considerations and the ones actually implemented in proof assistants. This result has been established by Siles and Herbelin (2012) for the conventional presentation of PTSs using syntactic methods.

Our main contribution is the proof of that equivalence for a class of PTSs which we call predicative, by semantical means, using techniques based on Normalisation by Evaluation (NbE). We have a strong confidence that this method can be used to prove the key result for other systems. In particular, we are starting to work out the details for PTSs with η and plan to report them in a forthcoming paper.

1.1 Related work

As far as we know, our formulation of PTSs is new, although the variations that we add were already considered separately. The three aspects in which we depart from usual PTSs are: explicit substitutions, de Bruijn indices, and abstractions without

domain annotations. PTSs with explicit substitutions were considered by Bloo (2001); Muñoz (2001) studied one particular PTSs with explicit substitutions and de Bruijn indices. Barthe and Sørensen (2000) introduced domain-free PTSs and studied their meta-theory, which was further extended by Barthe and Coquand (2006).

Geuvers (1993) conjectured that the presentation of PTSs with external equality is equivalent to the PTSs with judgemental equality. Soon after, Geuvers and Werner (1994) identified sufficient conditions for the equivalence in the presence of η. For PTSs without η, Adams (2006) proved the equivalence for functional PTSs. Later, Siles and Herbelin (2010) extended and formalised Adams' result to semi-full systems; finally the equivalence for every PTS was settled by Siles (2010), and reported also in Siles & Herbelin (2012).

We construct a model and use it to show a weak version of subject reduction for *predicative* PTSs; this result leads to a proof of equivalence between the variants with typed equality and untyped conversion for predicative PTSs. The semantical method we use in this paper is based on previous works (Abel *et al.*, 2007; Abel *et al.*, 2011) that define NbE for Martin–Löf type theory.

1.2 Outline

In Section 2, we introduce both families of PTSs with explicit substitutions and present the results formally proved in Agda, in the article we briefly explain the proof method for each result.[1] In Section 3, we state the equivalence between both families and prove that every judgement in a PTS with typed equality can be derived in its untyped counterpart. In Section 4, we define a normalisation function for predicative PTSs with judgemental equality. In Section 5, we prove the correctness of the normalisation function; finally in Section 6, we prove that in predicative PTSs every untyped conversion can be typed.

2 Formal systems

We assume some familiarity with traditional PTSs as presented in Barendregt (1992). In this section, we introduce and develop the meta-theory of two new formulations of PTSs. The first family, λ^{σ}, corresponds to PTSs where the equality between types in the conversion rule is generated by the reduction relation over pre-terms (and pre-substitutions); this variant is also called *syntactical*. The second formulation, $\lambda^{\sigma=}$, has a typed notion of equality, and is referred as *semantical*.

As in traditional PTSs, both families of type systems are parameterised by *signatures*; each signature determines a concrete calculus and stipulates the kind of dependency allowed.

A signature $\mathscr{S} = (S, A, R)$ is given by a set S of *sorts*, a binary relation A over S, and a ternary relation R over S. Elements in A are called *axioms* and elements in R, *rules*.

[1] The formal proofs accompanying this paper can be found at `http://cs.famaf.unc.edu.ar/~mpagano/pts/`.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

Let $\mathscr{S} = (S, A, R)$ be a signature and let s ranges over S, the syntax of pre-terms and pre-substitutions are defined by

$$\begin{aligned} \mathit{Term} \ni A, B, t, r &::= \mathsf{q} \mid s \mid \mathsf{App}\ t\ r \mid \lambda t \mid \mathsf{Fun}\, A\, B \mid t\,\sigma \\ \mathit{Sub} \ni \sigma, \delta &::= \mathsf{p} \mid \langle \sigma, t \rangle \mid \delta\,\sigma \mid \mathsf{id} \\ \mathit{Ctx} \ni \Gamma &::= \diamond \mid \Gamma, A \end{aligned}$$

2.1 Notation

We use some conventions for meta-variables ranging over pre-terms, pre-substitutions, and pre-contexts: capital Greek letters are used for pre-contexts, lower case Greek letters range over *Sub*, and Latin letters are for arbitrary pre-terms. Whenever we write c or s, sometimes primed, we mean a sort, i.e. an element of the set S. The length of a pre-context Γ is denoted by $|\Gamma|$. We use $[r]$ to denote $\langle \mathsf{id}, r \rangle$.

2.2 About pre-terms and pre-substitutions

The constructors for pre-terms and pre-substitutions are a variant of the calculus $\lambda\sigma$ of Abadi *et al.* (1990): terms are extended with sorts s and $\mathsf{Fun}\, A\, B$, the constructor for dependent function spaces. We use juxtaposition both for composition of substitutions and for the formal application of a substitution to a term; application of t to r is written $\mathsf{App}\ t\ r$. Variables are de Bruijn indices made up from q and the shifting operator p, so for example the index 3 is written $\mathsf{q}(\mathsf{p}\,\mathsf{p})$; here q and p are the respective names for 1 and $\uparrow$ in $\lambda\sigma$. In the following, we use p^i to denote the ith-fold self-composition of p, with $\mathsf{p}^0 = \mathsf{id}$.

The constructors of *Sub* formalise the various steps involved when performing substitutions: p is the shifting operator which is used to increment indices when traversing a binder; $\sigma\,\delta$ is the composition of substitutions σ and δ; finally, $\langle \sigma, t \rangle$ permits to extend the substitution σ with t, in particular $\langle \sigma, t \rangle$ maps q to t and other variables to the value assigned by σ. The identity substitution id maps each variable to itself. In the formalism of Categories with Families of Dybjer (1996), there is an operator for the empty substitution which is equal to the identity substitution only under the empty context; we avoid the empty substitution, because it is unclear how to orient that equation in an untyped reduction rule.

2.3 Normal forms

We define inductively predicates characterising neutral and normal forms; these notions will be used to define a normalisation function.

$$\begin{aligned} \mathrm{Ne} \ni k &::= \mathsf{q} \mid \mathsf{q}\,\mathsf{p}^{i+1} \mid \mathsf{App}\ k\ v \\ \mathrm{Nf} \ni v, V, W &::= s \mid \mathsf{Fun}\ V\ W \mid \lambda v \mid k \end{aligned}$$

Usually both s and $\mathsf{Fun}\ V\ W$ would be considered neutral terms because when they appear in head position in an application they do not produce a redex; but such an application would not be typable, so we treat both s and $\mathsf{Fun}\ V\ W$ as normal forms.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

Beta-reduction

$$\mathsf{App}\ (\lambda t)\ r \to_\beta t\,\langle \mathsf{id}, r\rangle$$

Substitutions

$$(\sigma\,\delta)\,\gamma \to_{\mathsf{x}} \sigma\,(\delta\,\gamma)$$
$$\sigma\,\mathsf{id} \to_{\mathsf{x}} \sigma$$
$$\mathsf{id}\,\sigma \to_{\mathsf{x}} \sigma$$
$$\mathsf{p}\,\langle \sigma, t\rangle \to_{\mathsf{x}} \sigma$$
$$\langle \sigma, t\rangle\,\delta \to_{\mathsf{x}} \langle \sigma\,\delta, t\,\delta\rangle$$

Resolution of substitutions

$$(\mathsf{Fun}\,A\,B)\,\sigma \to_{\mathsf{x}} \mathsf{Fun}\,(A\,\sigma)\,(B\,\langle \sigma\,\mathsf{p}, \mathsf{q}\rangle)$$
$$(t\,\sigma)\,\delta \to_{\mathsf{x}} t\,(\sigma\,\delta)$$
$$t\,\mathsf{id} \to_{\mathsf{x}} t$$
$$\mathsf{q}\,\langle \sigma, t\rangle \to_{\mathsf{x}} t$$
$$(\lambda t)\,\sigma \to_{\mathsf{x}} \lambda\,(t\,\langle \sigma\,\mathsf{p}, \mathsf{q}\rangle)$$
$$(\mathsf{App}\ t\ r)\,\sigma \to_{\mathsf{x}} \mathsf{App}\ (t\,\sigma)\ (r\,\sigma)$$
$$s\,\sigma \to_{\mathsf{x}} s$$

Fig. 1. Reduction rules.

2.4 PTSs with untyped equality (λ^σ)

The first family of type systems is the more usual presentation of PTSs, where equality between types is the congruence relation generated by an untyped reduction relation between pre-terms.

Definition 1 (*Untyped equivalence*)
The relation $\equiv_{\beta\mathsf{x}} \subseteq (\mathit{Term} \cup \mathit{Sub}) \times (\mathit{Term} \cup \mathit{Sub})$ is the congruence closure of the reduction $\to_{\beta\mathsf{x}} = \to_\beta \cup \to_{\mathsf{x}}$, shown in Figure 1. The equivalence $\equiv_{\beta\mathsf{x}}$ induces an equivalence relation between pre-contexts, generated by the following rules:

$$\frac{}{\diamond \equiv_{\beta\mathsf{x}} \diamond} \qquad \frac{\Gamma \equiv_{\beta\mathsf{x}} \Delta \qquad A \equiv_{\beta\mathsf{x}} B}{\Gamma, A \equiv_{\beta\mathsf{x}} \Delta, B}$$

The confluence of the reduction system $\to_{\beta\mathsf{x}}$ can be proved by the same strategy used in (Abadi *et al.*, 1990; Curien *et al.*, 1996) to prove the confluence of the system $\lambda\sigma$, without meta-variables: first one proves the termination of $\to_{\mathsf{x}}$, see Curien *et al.* (1992), and then one applies Hardin's interpretation technique (Hardin, 1989). Kesner (2000) presents an alternative proof which also takes into account η expansion. The variations – two new constructors for terms and the more liberal rules for reducing compositions with id in the right – in our reduction rules with respect to those of $\lambda\sigma$ do not introduce any difficulty.

2.5 The type systems

There are three forms of judgements in λ^σ: (i) well-formedness of pre-contexts $\Gamma \vdash$, (ii) typing of terms under a context $\Gamma \vdash t : A$, and (iii) typing of substitutions $\Gamma \vdash \sigma : \Delta$. In traditional presentations of PTSs where substitution is an operation at the meta-level, only the second form of judgement is present; we need the first form of judgement to ensure that the context of each derivable judgement is well formed. The third form of judgement states that a pre-substitution is well typed, its informal meaning is that if one has derived $\Delta \vdash t : A$ and the substitution σ can be typed $\Gamma \vdash \sigma : \Delta$, then $t\,\sigma$ can be typed under Γ with type $A\,\sigma$.

The typing rules of λ^σ are presented in Figures 2 and 3. The rules are basically the same as those in Muñoz (2001), but without meta-variables. In particular, we also

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

$$\text{(EMPTY-CTX)}\quad \frac{}{\diamond \vdash} \qquad \text{(EXT-CTX)}\quad \frac{\Gamma \vdash A : s}{\Gamma, A \vdash}$$

$$\text{(AXIOM)}\quad \frac{\Gamma \vdash}{\Gamma \vdash c : s}\ (c,s) \in A \qquad \text{(FUN-F)}\quad \frac{\Gamma \vdash A : s \qquad \Gamma, A \vdash B : s'}{\Gamma \vdash \mathsf{Fun} A B : s''}\ (s,s',s'') \in R$$

$$\text{(HYP)}\quad \frac{\Gamma \vdash A : s}{\Gamma, A \vdash \mathsf{q} : A\,\mathsf{p}} \qquad \text{(FUN-I)}\quad \frac{\Gamma \vdash A : s \qquad \Gamma, A \vdash B : s' \qquad \Gamma, A \vdash t : B}{\Gamma \vdash \lambda t : \mathsf{Fun} A B}\ (s,s',s'') \in R$$

$$\text{(FUN-EL)}\quad \frac{\Gamma \vdash t : \mathsf{Fun} A B \qquad \Gamma \vdash r : A}{\Gamma \vdash \mathsf{App}\ t\ r : B\langle \mathsf{id}, r \rangle} \qquad \text{(CNV-TY)}\quad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash B : s \qquad A \equiv_{\beta \mathsf{x}} B}{\Gamma \vdash t : B}$$

$$\text{(SUB-SORT)}\quad \frac{\Delta \vdash A : s \qquad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash A\sigma : s} \qquad \text{(SUB-TY)}\quad \frac{\Delta \vdash A : s \qquad \Delta \vdash t : A \qquad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash t\sigma : A\sigma}$$

Fig. 2. Rules for contexts and terms λ^σ.

$$\text{(ID-SUB)}\quad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{id} : \Gamma} \qquad \text{(FST-SUB)}\quad \frac{\Gamma \vdash A : s}{\Gamma, A \vdash \mathsf{p} : \Gamma} \qquad \text{(EXT-SUB)}\quad \frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash A : s \qquad \Gamma \vdash t : A\sigma}{\Gamma \vdash \langle \sigma, t \rangle : \Delta, A}$$

$$\text{(COMP-SUB)}\quad \frac{\Gamma \vdash \delta : \Theta \qquad \Theta \vdash \sigma : \Delta}{\Gamma \vdash \sigma\delta : \Delta} \qquad \text{(CNV-SUB)}\quad \frac{\Gamma \vdash \sigma : \Delta \qquad \Delta' \vdash \qquad \Delta \equiv_{\beta \mathsf{x}} \Delta'}{\Gamma \vdash \sigma : \Delta'}$$

Fig. 3. Rules for substitutions of λ^σ.

have two rules for substitutions: (SUB-TY) and (SUB-SORT). The latter rule is needed because otherwise we cannot resolve substitutions in the type when it is a top-sort; this duplication of rules for substitutions requires to explicitly state by separate the same result for top-sorts and types. This distinction is avoided in Bloo (2001), where substitution is explicit on the subject, but a meta-operation in the type.

2.6 Properties of λ^σ

One of the most important results for a type system is subject reduction, saying that typing is preserved by untyped reductions. In this section, we prove step-by-step all the technical lemmas needed for subject reduction.

The first lemma we prove is that every context in a judgement is well formed.

Lemma 2 (*Well-formedness of contexts*)

1. If $\Gamma \vdash t : C$, then $\Gamma \vdash$.
2. If $\Gamma \vdash \sigma : \Delta$, then $\Gamma \vdash$ and $\Delta \vdash$.

Proof

By simultaneous induction on $\Gamma \vdash t : C$ and $\Gamma \vdash \sigma : \Delta$. □

Well formedness of contexts is enough to prove that typing is preserved under equal contexts, provided the second context is also well formed. We point out that this result is similar to subject reduction of contexts for traditional PTSs.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

Lemma 3 (*Context Conversion*)
Let $\Gamma \equiv_{\beta\mathsf{x}} \Gamma'$ and $\Gamma' \vdash$.

1. If $\Gamma \vdash t : A$, then $\Gamma' \vdash t : A$.
2. If $\Gamma \vdash \sigma : \Delta$, then $\Gamma' \vdash \sigma : \Delta$.

Proof
By simultaneous induction on $\Gamma \vdash t : A$ and $\Gamma \vdash \sigma : \Delta$. □

For a well-typed term $\Gamma \vdash t : C$, the *generation lemma* gives information about the type C depending on the shape of the term t. In traditional PTSs, this lemma comes after the so called *thinning lemma*; in λ^σ thinning is internalised by the use of explicit substitutions.

Lemma 4 (*Inversion for terms*)

1. If $\Gamma \vdash s : C$, then there exists $(s, s') \in A$ and $C \equiv_{\beta\mathsf{x}} s'$.
2. If $\Gamma \vdash \mathsf{q} : C$, then there exists $s \in S$, and $\Gamma' \vdash A : s$, $\Gamma = \Gamma', A$, and $C \equiv_{\beta\mathsf{x}} A\,\mathsf{p}$.
3. If $\Gamma \vdash \mathsf{Fun}\,A\,B : C$, then there exist $(s_0, s_1, s_2) \in R$, and $\Gamma \vdash A : s_0$, $\Gamma, A \vdash B : s_1$, and $C \equiv_{\beta\mathsf{x}} s_2$.
4. If $\Gamma \vdash \lambda t : C$, then there exist $A, B \in \mathit{Term}$, $(s_0, s_1, s_2) \in R$, $\Gamma \vdash A : s_0$, $\Gamma, A \vdash B : s_1$, $\Gamma, A \vdash t : B$, and $C \equiv_{\beta\mathsf{x}} \mathsf{Fun}\,A\,B$.
5. If $\Gamma \vdash \mathsf{App}\ t\ r : C$, then there exist $A, B \in \mathit{Term}$, $\Gamma \vdash t : \mathsf{Fun}\,A\,B$, $\Gamma \vdash r : A$, and $C \equiv_{\beta\mathsf{x}} B\,\langle \mathsf{id}, r\rangle$.
6. If $\Gamma \vdash t\,\sigma : C$, then there exist Σ and s such that $\Gamma \vdash \sigma : \Sigma$ and either
   a. $\Sigma \vdash t : s$ and $C \equiv_{\beta\mathsf{x}} s$; or
   b. there exists $A \in \mathit{Term}$, such that $\Sigma \vdash A : s$, $\Sigma \vdash t : A$, and $C \equiv_{\beta\mathsf{x}} A\,\sigma$.

Proof
The proof of each statement is by straightforward induction on derivations: for each point, there are two possible cases for the last rule used, either the introductory rule or the conversion rule. □

The analogous lemma for substitutions is proved by the same reasoning; here we use the fact that $\equiv_{\beta\mathsf{x}}$ is an equivalence relation for contexts.

Lemma 5 (*Inversion for substitutions*)

1. If $\Gamma \vdash \mathsf{id} : \Delta$, then $\Gamma \equiv_{\beta\mathsf{x}} \Delta$.
2. If $\Gamma \vdash \mathsf{p} : \Delta$, then there exist Σ, s, and A such that $\Sigma \vdash A : s$, $\Gamma = \Sigma, A$, and $\Sigma \equiv_{\beta\mathsf{x}} \Delta$.
3. If $\Gamma \vdash \sigma\,\delta : \Delta$, then there exist Θ and Σ such that $\Gamma \vdash \delta : \Theta$, $\Theta \vdash \sigma : \Sigma$, and $\Sigma \equiv_{\beta\mathsf{x}} \Delta$.
4. If $\Gamma \vdash \langle \sigma, t\rangle : \Delta$, then there exist Σ, s, and A such that $\Gamma \vdash \sigma : \Sigma$, $\Sigma \vdash A : s$, $\Gamma \vdash t : A\,\sigma$, and $\Sigma, A \equiv_{\beta\mathsf{x}} \Delta$.

The inversion lemmas also give information about the types of the immediate sub-terms of a well-typed term (or substitution); this information is most useful for proving subject reduction: if $\Gamma \vdash_{\overline{\mathsf{e}}} t : C$ and $t \to_{\beta\mathsf{x}} t'$, we use the typing of the sub-terms of t to reconstruct a typing derivation for t'.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

The substitution lemma states that one can replace a hypothesis $x : A$ with a fact $r : A$, no matter how many other assumptions depended on x. In less picturesque terms, substitution is usually stated as $\Gamma, x : A, \Delta \vdash t : C$ and $\Gamma \vdash r : A$, then $\Gamma, \Delta\{x := r\} \vdash t\{x := r\} : C\{x := r\}$. To formalise the proof of the substitution lemma, we introduce a new judgement which corresponds to the typing of a list of pre-terms in a context.

We write $[]$ for the empty list, which can be typed in any well-formed context, and $A \triangleright \Delta$ for the list with head A and tail Δ. By abuse of notation, we write $|\Delta|$ for the length of the list Δ.

Definition 6 (*Well-typed contexts*)
Let Γ be a context and Δ be a list of terms. We define inductively the predicate $\Gamma \vdash \Delta$:

$$\frac{\Gamma \vdash}{\Gamma \vdash []} \qquad \frac{\Gamma \vdash A : s \qquad \Gamma, A \vdash \Delta}{\Gamma \vdash A \triangleright \Delta}$$

If $\Gamma \vdash \Delta$, then one can concatenate Γ with Δ to obtain a well-formed context:

$$\Gamma +\!\!+ [] = \Gamma \qquad \Gamma +\!\!+ (A \triangleright \Delta) = (\Gamma, A) +\!\!+ \Delta$$

Lemma 7
$\Gamma \vdash \Delta$ if and only if $\Gamma +\!\!+ \Delta \vdash$.

This notion of concatenation aptly captures the informal notation Γ, A, Δ; we still have to formalise the meta-operation of applying a substitution over some suffix of a context. To treat this operation properly, we define the application of a pre-substitution σ over a list of pre-terms Δ; notice that this does not correspond to mapping the application of the substitution to each pre-term of the list (as in traditional PTSs). We also define an operation to repeatedly lift a substitution; this lifting corresponds to the updating manipulation needed when variables are de Bruijn indices.

Definition 8 (*Substitutions over lists and lifting of substitutions*)

$$[]\,\sigma = [] \qquad \sigma^0 = \sigma$$
$$(A \triangleright \Delta')\,\sigma = (A\,\sigma) \triangleright \Delta'\,\langle \sigma\,\mathsf{p}, \mathsf{q}\rangle \qquad \sigma^{n+1} = \langle \sigma\,\mathsf{p}, \mathsf{q}\rangle^n$$

As is to be expected, well typedness of contexts is preserved under substitutions; and lifting of well-typed substitutions is also well typed.

Lemma 9
Let Δ be a list of terms with $n = |\Delta|$.

1. If $\Gamma \vdash \Delta$ and $\Theta \vdash \sigma : \Gamma$, then $\Theta \vdash \Delta\,\sigma$ and $\Theta +\!\!+ \Delta\,\sigma \vdash \sigma^n : \Gamma +\!\!+ \Delta$.
2. If $\Gamma \vdash A \triangleright \Delta$ and $\Gamma \vdash r : A$, then $\Gamma \vdash \Delta\,[r]$ and $\Gamma +\!\!+ \Delta\,[r] \vdash [r]^n : \Gamma +\!\!+ A \triangleright \Delta$.

Proof
The first point is by induction on $\Gamma \vdash \Delta$; the second point follows from the first one. □

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

With all this formal paraphernalia, it is easy to prove substitution. Consider $\Gamma +\!\!+ A \triangleright \Delta \vdash$ and $\Gamma \vdash r : A$; thanks to the previous lemma, we can type the substitution $[r]^{|\Delta|}$. This lemma is the first one where we need to distinguish between top-sorts and types.

Lemma 10 (*Substitution lemma*)
Let $\Gamma +\!\!+ A \triangleright \Delta \vdash$, $\Gamma \vdash r : A$, and $n = |\Delta|$.

1. If $\Gamma +\!\!+ A \triangleright \Delta \vdash t : s$, then $\Gamma +\!\!+ \Delta\,[r] \vdash t\,[r]^n : s$.
2. If $\Gamma +\!\!+ A \triangleright \Delta \vdash B : s$ and $\Gamma +\!\!+ A \triangleright \Delta \vdash t : B$, then $\Gamma +\!\!+ \Delta\,[r] \vdash t\,[r]^n : B\,[r]^n$.
3. If $\Gamma +\!\!+ A \triangleright \Delta \vdash \sigma : \Sigma$, then $\Gamma +\!\!+ \Delta\,[r] \vdash \sigma\,[r]^n : \Sigma$.

Definition 11 (*Types*)
A pre-term A is a *type under* Γ, denoted by $\Gamma \vdash A$, if either A is a sort or there exists $s \in S$ such that $\Gamma \vdash A : s$.

Lemma 12 (*Type validity*)
If $\Gamma \vdash t : A$, then $\Gamma \vdash A$.

Proof
The proof is by induction on derivations, the only non-trivial case is (FUN-EL): the premises are $\Gamma \vdash \lambda t : \mathsf{Fun}\,A\,B$ and $\Gamma \vdash r : A$; we need to prove $\Gamma \vdash B\,\langle \mathsf{id}, r\rangle$. By i.h. on the first premise we know $\Gamma \vdash \mathsf{Fun}\,A\,B : s$. We can apply Lemma 4 to conclude $\Gamma, A \vdash B : s'$, from which we conclude $\Gamma \vdash B\,\langle \mathsf{id}, r\rangle : s'$ using Lemma 10. □

All the results up to here are formalised without any postulate; the proof of subject reduction depends indirectly on the Church–Rosser property of the reduction $\to_{\beta\mathsf{x}}$. The main result we need, which can be proved assuming Church–Rosser, is *Injectivity of Products*.

Theorem 13 (*Subject reduction for* λ^{σ})

1. If $\Gamma \vdash t : A$ and $t \to_{\beta\mathsf{x}} t'$, then $\Gamma \vdash t' : A$.
2. If $\Gamma \vdash \sigma : \Delta$ and $\sigma \to_{\beta\mathsf{x}} \sigma'$, then $\Gamma \vdash \sigma' : \Delta$.

Proof
The proof proceeds by induction on derivations; in each case, we analyse the reduction and use inversion for getting the types for the sub-terms involved in the redex. Let us consider the case when the last rule used is (FUN-EL) and the redex is at head position: $\Gamma \vdash \mathsf{App}\ (\lambda t)\ r : B\,[r]$, the hypotheses are $\Gamma \vdash \lambda t : \mathsf{Fun}\,A\,B$ and $\Gamma \vdash r : A$. From inversion (Lemma 4), we know there exist $C, D \in \mathit{Term}$ such that $\Gamma, C \vdash t : D$ and $\mathsf{Fun}\,A\,B \equiv_{\beta\mathsf{x}} \mathsf{Fun}\,C\,D$. By injectivity of products, we also know $A \equiv_{\beta\mathsf{x}} C$ and $B \equiv_{\beta\mathsf{x}} D$; thus we can conclude $\Gamma, A \vdash t : B$ by context conversion, Lemma 3. By the substitution lemma, Lemma 10, we conclude $\Gamma \vdash t\,[r] : B\,[r]$. □

2.7 PTSs with typed equality ($\lambda^{\sigma=}$)

In $\lambda^{\sigma=}$, equality is axiomatised as in Martin–Löf type theory, i.e. by a system of axioms typable under some context. In $\lambda^{\sigma=}$, there are six forms of judgements (we use $\vDash_{\mathsf{e}}$ for judgements in this family); besides those forms already present in λ^{σ},

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

$$\text{(EMPTY-CTX)}\ \frac{}{\diamond \vdash_{\mathsf{e}}} \qquad \text{(EMP-EQ-CTX)}\ \frac{}{\vdash_{\mathsf{e}} \diamond = \diamond} \qquad \text{(EXT-CTX)}\ \frac{\Gamma \vdash_{\mathsf{e}} A : s}{\Gamma, A \vdash_{\mathsf{e}}} \qquad \text{(EXT-EQ-CTX)}\ \frac{\vdash_{\mathsf{e}} \Gamma = \Gamma' \qquad \Gamma \vdash_{\mathsf{e}} A = B : s}{\vdash_{\mathsf{e}} \Gamma, A = \Gamma', B}$$

Fig. 4. Rules for well-formed contexts and their equality.

$$\text{(AXIOM)}\ \frac{\Gamma \vdash_{\mathsf{e}}}{\Gamma \vdash_{\mathsf{e}} c : s}\ (c,s) \in A \qquad \text{(FUN-F)}\ \frac{\Gamma \vdash_{\mathsf{e}} A : s \qquad \Gamma, A \vdash_{\mathsf{e}} B : s'}{\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, A\, B : s''}\ (s,s',s'') \in R$$

$$\text{(HYP)}\ \frac{\Gamma \vdash_{\mathsf{e}} A : s}{\Gamma, A \vdash_{\mathsf{e}} \mathsf{q} : A\,\mathsf{p}} \qquad \text{(FUN-I)}\ \frac{\Gamma \vdash_{\mathsf{e}} A : s \qquad \Gamma, A \vdash_{\mathsf{e}} B : s' \qquad \Gamma, A \vdash_{\mathsf{e}} t : B}{\Gamma \vdash_{\mathsf{e}} \lambda t : \mathsf{Fun}\, A\, B}\ (s,s',s'') \in R$$

$$\text{(FUN-EL)}\ \frac{\Gamma \vdash_{\mathsf{e}} t : \mathsf{Fun}\, A\, B \qquad \Gamma \vdash_{\mathsf{e}} r : A}{\Gamma \vdash_{\mathsf{e}} \mathsf{App}\, t\, r : B\langle \mathsf{id}, r \rangle} \qquad \text{(CNV-TY)}\ \frac{\Gamma \vdash_{\mathsf{e}} t : A \qquad \Gamma \vdash_{\mathsf{e}} A = B : s}{\Gamma \vdash_{\mathsf{e}} t : B}$$

$$\text{(SUB-SORT)}\ \frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s}{\Gamma \vdash_{\mathsf{e}} A\,\sigma : s} \qquad \text{(SUB-TY)}\ \frac{\Delta \vdash_{\mathsf{e}} A : s \qquad \Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} t : A}{\Gamma \vdash_{\mathsf{e}} t\,\sigma : A\,\sigma}$$

Fig. 5. Rules for terms of $\lambda^{\sigma=}$.

there are three more corresponding to typed equalities for each of the syntactical categories. Summarising we have the following forms of judgements: (i) well-formed pre-contexts $\Gamma \vdash_{\mathsf{e}}$, shown in Figure 4, (ii) typing of terms $\Gamma \vdash_{\mathsf{e}} t : A$, presented in Figure 5, and (iii) typing of substitutions $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$, in Figure 6; (iv) equality of contexts $\vdash_{\mathsf{e}} \Gamma = \Delta$, also in Figure 4, (v) equality of terms $\Gamma \vdash_{\mathsf{e}} t = t' : A$, see Figures 7 and 8, and (vi) equality of substitutions $\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta$, in Figures 9 and 10.

Remark 14
Notice that if $\Delta \vdash_{\mathsf{e}} \mathsf{p} : \Gamma_1, \Gamma_1 \vdash_{\mathsf{e}} \mathsf{p} : \Gamma_2, \ldots, \Gamma_n \vdash_{\mathsf{e}} \mathsf{p} : \Gamma$, then $\Delta \vdash_{\mathsf{e}} \mathsf{p}^{n+1} : \Gamma$, where p^i is the ith self-composition of p. We write $\Delta \leqslant^n \Gamma$ if $\Delta \vdash_{\mathsf{e}} \mathsf{p}^n : \Gamma$.

The following fairly technical lemma will be freely used in Section 5. The proof goes by induction on j.

Lemma 15
If $\Delta \leqslant^i \Gamma$ and $\Gamma \leqslant^j \Sigma$, then $\Delta \leqslant^{i+j} \Sigma$ and $\Delta \vdash_{\mathsf{e}} \mathsf{p}^{i+j} = \mathsf{p}^i \circ \mathsf{p}^j : \Sigma$.

2.8 Properties of $\lambda^{\sigma=}$

Some of the meta-theoretical results of $\lambda^{\sigma=}$require more work than those of λ^{σ}, because of the mutual dependency between the various forms of judgements. For

$$\text{(ID-SUB)}\ \frac{\Gamma \vdash_{\mathsf{e}}}{\Gamma \vdash_{\mathsf{e}} \mathsf{id} : \Gamma} \qquad \text{(FST-SUB)}\ \frac{\Gamma \vdash_{\mathsf{e}} A : s}{\Gamma, A \vdash_{\mathsf{e}} \mathsf{p} : \Gamma} \qquad \text{(EXT-SUBS)}\ \frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s \qquad \Gamma \vdash_{\mathsf{e}} t : A\,\sigma}{\Gamma \vdash_{\mathsf{e}} \langle \sigma, t \rangle : \Delta, A}$$

$$\text{(COMP-SUBS)}\ \frac{\Gamma \vdash_{\mathsf{e}} \delta : \Theta \qquad \Theta \vdash_{\mathsf{e}} \sigma : \Delta}{\Gamma \vdash_{\mathsf{e}} \sigma\,\delta : \Delta} \qquad \text{(CNV-SUB)}\ \frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \vdash_{\mathsf{e}} \Delta = \Delta'}{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta'}$$

Fig. 6. Rules for substitutions of $\lambda^{\sigma=}$.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

(BETA)
$$\frac{\Gamma \vdash_{\mathsf{e}} A : s \qquad \Gamma, A \vdash_{\mathsf{e}} B : s' \qquad \Gamma, A \vdash_{\mathsf{e}} t : B \qquad \Gamma \vdash_{\mathsf{e}} r : A}{\Gamma \vdash_{\mathsf{e}} \mathsf{App}\ (\lambda t)\ r = t\langle \mathsf{id}, r\rangle : B\langle \mathsf{id}, r\rangle}\ (s,s',s'') \in R$$

(SUB-FUN)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s \qquad \Delta, A \vdash_{\mathsf{e}} B : s'}{\Gamma \vdash_{\mathsf{e}} (\mathsf{Fun} A B)\sigma = \mathsf{Fun}\,(A\sigma)\,(B\langle \sigma \mathsf{p}, \mathsf{q}\rangle) : s''}\ (s,s',s'') \in R$$

(SUB-SORT)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta}{\Gamma \vdash_{\mathsf{e}} c\sigma = c : s}\ (c,s) \in A$$

(SUB-ABS)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s \qquad \Delta, A \vdash_{\mathsf{e}} B : s' \qquad \Delta, A \vdash_{\mathsf{e}} t : B}{\Gamma \vdash_{\mathsf{e}} (\lambda t)\sigma = \lambda(t\langle \sigma \mathsf{p}, \mathsf{q}\rangle) : (\mathsf{Fun} A B)\sigma}\ (s,s',s'') \in R$$

(SUB-APP)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} t : \mathsf{Fun} A B \qquad \Delta \vdash_{\mathsf{e}} r : A}{\Gamma \vdash_{\mathsf{e}} (\mathsf{App}\ t\ r)\sigma = \mathsf{App}\ (t\sigma)\ (r\sigma) : B\langle \sigma, r\sigma\rangle}$$

(SUB-SND)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s \qquad \Gamma \vdash_{\mathsf{e}} t : A\sigma}{\Gamma \vdash_{\mathsf{e}} \mathsf{q}\langle \sigma, t\rangle = t : A\sigma}$$

(SUB-COMP-T)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Sigma \qquad \Sigma \vdash_{\mathsf{e}} \delta : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s \qquad \Delta \vdash_{\mathsf{e}} t : A}{\Gamma \vdash_{\mathsf{e}} t\,(\delta\sigma) = (t\,\delta)\sigma : A\,(\delta\sigma)}$$

(SUB-ID-T)
$$\frac{\Gamma \vdash_{\mathsf{e}} A : s \qquad \Gamma \vdash_{\mathsf{e}} t : A}{\Gamma \vdash_{\mathsf{e}} t\,\mathsf{id} = t : A}$$

(SUB-COMP-S)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Sigma \qquad \Sigma \vdash_{\mathsf{e}} \delta : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s}{\Gamma \vdash_{\mathsf{e}} A\,(\delta\sigma) = (A\,\delta)\sigma : s}$$

(SUB-ID-S)
$$\frac{\Gamma \vdash_{\mathsf{e}} A : s}{\Gamma \vdash_{\mathsf{e}} A\,\mathsf{id} = A : s}$$

Fig. 7. Axioms for terms.

(REFL)
$$\frac{\Gamma \vdash_{\mathsf{e}} t : A}{\Gamma \vdash_{\mathsf{e}} t = t : A}$$

(SYM)
$$\frac{\Gamma \vdash_{\mathsf{e}} t = t' : A}{\Gamma \vdash_{\mathsf{e}} t' = t : A}$$

(TRANS)
$$\frac{\Gamma \vdash_{\mathsf{e}} t = t' : A \qquad \Gamma \vdash_{\mathsf{e}} t' = t'' : A}{\Gamma \vdash_{\mathsf{e}} t = t'' : A}$$

(FUN-CNG)
$$\frac{\Gamma \vdash_{\mathsf{e}} A = B : s \qquad \Gamma, A \vdash_{\mathsf{e}} C = D : s'}{\Gamma \vdash_{\mathsf{e}} \mathsf{Fun} A C = \mathsf{Fun} B D : s''}\ (s,s',s'') \in R$$

(ABS-CNG)
$$\frac{\Gamma \vdash_{\mathsf{e}} A : s \qquad \Gamma, A \vdash_{\mathsf{e}} B : s' \qquad \Gamma, A \vdash_{\mathsf{e}} t = t' : B}{\Gamma \vdash_{\mathsf{e}} \lambda t = \lambda t' : \mathsf{Fun} A B}\ (s,s',s'') \in R$$

(APP-CNG)
$$\frac{\Gamma \vdash_{\mathsf{e}} t = t' : \mathsf{Fun} A B \qquad \Gamma \vdash_{\mathsf{e}} r = r' : A}{\Gamma \vdash_{\mathsf{e}} \mathsf{App}\ t\ r = \mathsf{App}\ t'\ r' : B\langle \mathsf{id}, r\rangle}$$

(CONV-EQ)
$$\frac{\Gamma \vdash_{\mathsf{e}} A = B : s \qquad \Gamma \vdash_{\mathsf{e}} t = t' : A}{\Gamma \vdash_{\mathsf{e}} t = t' : B}$$

(SUB-EQ-TY)
$$\frac{\Delta \vdash_{\mathsf{e}} A : s \qquad \Delta \vdash_{\mathsf{e}} t = t' : A \qquad \Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta}{\Gamma \vdash_{\mathsf{e}} t\sigma = t'\sigma' : A\sigma}$$

(SUB-EQ-SORT)
$$\frac{\Gamma \vdash_{\mathsf{e}} t = t' : s \qquad \Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta}{\Gamma \vdash_{\mathsf{e}} t\sigma = t'\sigma' : s}$$

Fig. 8. Equality is an equivalence and a congruence for terms.

(ASS)
$$\frac{\Gamma \vdash_{\mathsf{e}} \delta : \Theta \qquad \Theta \vdash_{\mathsf{e}} \sigma : \Sigma \qquad \Sigma \vdash_{\mathsf{e}} \gamma : \Delta}{\Gamma \vdash_{\mathsf{e}} (\sigma\,\delta)\,\gamma = \sigma\,(\delta\,\gamma) : \Delta}$$

(IDR)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta}{\Gamma \vdash_{\mathsf{e}} \sigma\,\mathsf{id} = \sigma : \Delta}$$

(IDL)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta}{\Gamma \vdash_{\mathsf{e}} \mathsf{id}\,\sigma = \sigma : \Delta}$$

(FST)
$$\frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s \qquad \Gamma \vdash_{\mathsf{e}} t : A\sigma}{\Gamma \vdash_{\mathsf{e}} \mathsf{p}\langle \sigma, t\rangle = \sigma : \Delta}$$

(DIST)
$$\frac{\Gamma \vdash_{\mathsf{e}} \delta : \Sigma \qquad \Sigma \vdash_{\mathsf{e}} \sigma : \Delta \qquad \Delta \vdash_{\mathsf{e}} A : s \qquad \Sigma \vdash_{\mathsf{e}} t : A\sigma}{\Gamma \vdash_{\mathsf{e}} \langle \sigma, t\rangle\,\delta = \langle \sigma\,\delta, t\,\delta\rangle : \Delta, A}$$

Fig. 9. Axioms for substitutions.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

$$\text{(S-REFL)}\ \frac{\Gamma \vdash_{\mathsf{e}} \sigma : \Delta}{\Gamma \vdash_{\mathsf{e}} \sigma = \sigma : \Delta} \qquad \text{(S-SYM)}\ \frac{\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta}{\Gamma \vdash_{\mathsf{e}} \sigma' = \sigma : \Delta} \qquad \text{(S-TRANS)}\ \frac{\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta \qquad \Gamma \vdash_{\mathsf{e}} \sigma' = \sigma'' : \Delta}{\Gamma \vdash_{\mathsf{e}} \sigma = \sigma'' : \Delta}$$

$$\text{(EXT-EQ)}\ \frac{\Delta \vdash_{\mathsf{e}} A : s \qquad \Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta \qquad \Gamma \vdash_{\mathsf{e}} t = t' : A\sigma}{\Gamma \vdash_{\mathsf{e}} \langle \sigma, t \rangle = \langle \sigma', t' \rangle : \Delta, A} \qquad \text{(COMP-EQ)}\ \frac{\Gamma \vdash_{\mathsf{e}} \delta = \delta' : \Theta \qquad \Theta \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta}{\Gamma \vdash_{\mathsf{e}} \sigma\,\delta = \sigma'\delta' : \Delta}$$

$$\text{(CONV-S-EQ)}\ \frac{\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta \qquad \vdash_{\mathsf{e}} \Delta = \Delta'}{\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta'}$$

Fig. 10. Equality is an equivalence and a congruence for substitutions.

example, to prove that $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$ implies $\Delta \vdash_{\mathsf{e}}$, one needs to prove $\vdash_{\mathsf{e}} \Gamma = \Delta$ implies $\Delta \vdash_{\mathsf{e}}$, which in turn depends on validity of equality and context conversion.

Lemma 16 (*Well formedness of contexts*)

1. If $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$, then $\Gamma \vdash_{\mathsf{e}}$. Moreover if $\Gamma \vdash_{\mathsf{e}} t : A$, then $\Gamma \vdash_{\mathsf{e}}$.
2. If $\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta$, then $\Gamma \vdash_{\mathsf{e}}$; and if $\Gamma \vdash_{\mathsf{e}} t = t' : A$, then $\Gamma \vdash_{\mathsf{e}}$.

Proof

First one proves the case for substitutions by induction on derivations; then using that result, one proceeds to the case for terms, again by induction on derivations. □

Notice that successive application of the rule (CNV-TY) can use equalities under different sorts, so we introduce a notion of heterogeneous equality of types in order to prove an inversion lemma for $\lambda^{\sigma=}$. We write $\Gamma \vdash_{\mathsf{e}} t \approx t'$ if there is a sequence of equivalences $\Gamma \vdash_{\mathsf{e}} t = t_1 : s_1$, $\Gamma \vdash_{\mathsf{e}} t_1 = t_2 : s_2$, …, $\Gamma \vdash_{\mathsf{e}} t_n = t' : s_n$; when $n = 0$, we have that t and t' are the same term.

Definition 17 (*Heterogeneous equalities*)

$$\frac{}{\Gamma \vdash_{\mathsf{e}} t \approx t} \qquad \frac{\Gamma \vdash_{\mathsf{e}} t = t' : s}{\Gamma \vdash_{\mathsf{e}} t \approx t'} \qquad \frac{\Gamma \vdash_{\mathsf{e}} t \approx t' \qquad \Gamma \vdash_{\mathsf{e}} t' = t'' : s}{\Gamma \vdash_{\mathsf{e}} t \approx t''}$$

$$\frac{}{\vdash_{\mathsf{e}} \Gamma \approx \Gamma} \qquad \frac{\vdash_{\mathsf{e}} \Gamma = \Delta}{\vdash_{\mathsf{e}} \Gamma \approx \Delta} \qquad \frac{\vdash_{\mathsf{e}} \Gamma \approx \Delta \qquad \vdash_{\mathsf{e}} \Delta \approx \Sigma}{\vdash_{\mathsf{e}} \Gamma \approx \Sigma}$$

It is immediate that typing judgements are preserved under these equalities, i.e. if $\Gamma \vdash_{\mathsf{e}} t : A$ and $\Gamma \vdash_{\mathsf{e}} A \approx B$, then $\Gamma \vdash_{\mathsf{e}} t : B$; the analogous statement for preservation under heterogeneous equality of contexts should wait until we have proved more meta-theoretical results.

Lemma 18

If $\Gamma \vdash_{\mathsf{e}} t : A$ and $\Gamma \vdash_{\mathsf{e}} A \approx B$, then $\Gamma \vdash_{\mathsf{e}} t : B$.

Lemma 19 (*Inversion of typing*)

1. If $\Gamma \vdash_{\mathsf{e}} s : C$, then there exists $(s, s') \in A$ and $\Gamma \vdash_{\mathsf{e}} C \approx s'$.
2. If $\Gamma \vdash_{\mathsf{e}} \mathsf{q} : C$, then there exists $s \in S$, $\Gamma' \vdash_{\mathsf{e}}$ and $A \in \mathit{Term}$, such that $\Gamma' \vdash_{\mathsf{e}} A : s$, $\Gamma = \Gamma', A$, and $\Gamma \vdash_{\mathsf{e}} C \approx A\,\mathsf{p}$.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

3. If $\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, A\, B : C$, then there exists $(s_0, s_1, s_2) \in R$, such that $\Gamma \vdash_{\mathsf{e}} A : s_0$, $\Gamma, A \vdash_{\mathsf{e}} B : s_1$, and $\Gamma \vdash_{\mathsf{e}} C \approx s_2$.
4. If $\Gamma \vdash_{\mathsf{e}} \lambda t : C$, then there exist $A, B \in \mathit{Term}$ and $(s_0, s_1, s_2) \in R$, such that $\Gamma \vdash_{\mathsf{e}} A : s_0$, $\Gamma, A \vdash_{\mathsf{e}} B : s_1$, $\Gamma, A \vdash_{\mathsf{e}} t : B$, and $\Gamma \vdash_{\mathsf{e}} C \approx \mathsf{Fun}\, A\, B$.
5. If $\Gamma \vdash_{\mathsf{e}} \mathsf{App}\ t\ r : C$, then there exist $A, B \in \mathit{Term}$, such that $\Gamma \vdash_{\mathsf{e}} t : \mathsf{Fun}\, A\, B$, $\Gamma \vdash_{\mathsf{e}} r : A$, and $\Gamma \vdash_{\mathsf{e}} C \approx B\, \langle id, r \rangle$,

Proof
Each point is proved independently of the others. All the cases follow by induction on derivations. □

Lemma 20 (*Inversion of typing for substitutions*)

1. If $\Gamma \vdash_{\mathsf{e}} \mathsf{id} : \Delta$, then $\vdash_{\mathsf{e}} \Gamma \approx \Delta$.
2. If $\Gamma \vdash_{\mathsf{e}} \mathsf{p} : \Delta$, then there exist a pre-context Σ, a sort s, and a term A, such that $\Sigma \vdash_{\mathsf{e}} A : s$, $\Gamma = \Sigma, A$, and $\vdash_{\mathsf{e}} \Sigma \approx \Delta$.
3. If $\Gamma \vdash_{\mathsf{e}} \sigma\, \delta : \Delta$, then there exist pre-contexts Θ and Σ, such that $\Gamma \vdash_{\mathsf{e}} \delta : \Theta$, $\Theta \vdash_{\mathsf{e}} \sigma : \Sigma$, and $\vdash_{\mathsf{e}} \Sigma \approx \Delta$.
4. If $\Gamma \vdash_{\mathsf{e}} \langle \sigma, r \rangle : \Delta$, then there exist a pre-context Σ, a sort s, and a pre-term A, such that $\Gamma \vdash_{\mathsf{e}} \sigma : \Sigma$, $\Sigma \vdash_{\mathsf{e}} A : s$, $\Gamma \vdash_{\mathsf{e}} t : A\, \sigma$, and $\vdash_{\mathsf{e}} \Sigma, A \approx \Delta$.

Using the same notions defined in the previous section to prove Lemma 10 for λ^{σ}, we can also prove the substitution lemma for $\lambda^{\sigma=}$. We avoid the repetition of those definitions and state directly the lemma, which is also valid for equalities. Remember the need to distinguish the cases for terms typed with a sort and terms typed with another typable term.

Lemma 21 (*Substitution lemma*)
Let $\Gamma +\!\!+ A \triangleright \Delta \vdash_{\mathsf{e}}$, $\Gamma \vdash_{\mathsf{e}} r : A$, $n = |\Delta|$, and $\Gamma' = \Gamma +\!\!+ \Delta\, [r]$.

1. If $\Gamma, A, \Delta \vdash_{\mathsf{e}} t : s$, then $\Gamma' \vdash_{\mathsf{e}} t\, [r]^n : s$.
2. If $\Gamma, A, \Delta \vdash_{\mathsf{e}} B : s$ and $\Gamma, A, \Delta \vdash_{\mathsf{e}} t : B$, then $\Gamma' \vdash_{\mathsf{e}} t\, [r]^n : B\, [r]^n$.
3. If $\Gamma, A, \Delta \vdash_{\mathsf{e}} \sigma : \Sigma$, then $\Gamma' \vdash_{\mathsf{e}} \sigma\, [r]^n : \Sigma$.
4. If $\Gamma, A, \Delta \vdash_{\mathsf{e}} t = t' : s$, then $\Gamma' \vdash_{\mathsf{e}} t\, [r]^n = t'\, [r]^n : s$.
5. If $\Gamma, A, \Delta \vdash_{\mathsf{e}} B : s$ and $\Gamma, A, \Delta \vdash_{\mathsf{e}} t = t' : B$, then $\Gamma' \vdash_{\mathsf{e}} t\, [r]^n = t'\, [r]^n : B\, [r]^n$.
6. If $\Gamma, A, \Delta \vdash_{\mathsf{e}} \sigma = \sigma' : \Sigma$, then $\Gamma' \vdash_{\mathsf{e}} \sigma\, [r]^n = \sigma'\, [r]^n : \Sigma$. □

Reflexivity of well-formed contexts can be proved without any other result, but it is necessary to generalise its statement.

Lemma 22 (*Reflexivity of contexts*)

1. If $\Gamma \vdash_{\mathsf{e}}$, then $\vdash_{\mathsf{e}} \Gamma = \Gamma$.
2. If $\Gamma \vdash_{\mathsf{e}} t : A$, then $\vdash_{\mathsf{e}} \Gamma = \Gamma$.
3. If $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$, then $\vdash_{\mathsf{e}} \Gamma = \Gamma$.

Proof
We prove simultaneously the three points by induction on derivations. □

Remember that we say a term A is a *type* under Γ, written $\Gamma \vdash_{\mathsf{e}} A$, if A is a sort or $\Gamma \vdash_{\mathsf{e}} A : s$ for some sort s. As we mentioned, the mutual dependency between the different forms of judgement complicates the proof of validity of types for $\lambda^{\sigma=}$. We

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

will explain the mutual relationship between the lemmas needed to prove validity; this intricacy results in a huge lemma that can be proved by mutual induction in all the forms of judgements.

Lemma 23 (*Validity of types and contexts*)

1. If $\Gamma \vdash_{\mathsf{e}} t : A$, then $\Gamma \vdash_{\mathsf{e}} A$.
2. If $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$, then $\Delta \vdash_{\mathsf{e}}$.

Because of (CNV-TY), and (CNV-SUB), to prove validity of types we need to prove validity of equality.

Lemma 24 (*Validity of equality*)

1. If $\vdash_{\mathsf{e}} \Gamma = \Delta$, then $\Gamma \vdash_{\mathsf{e}}$ and $\Delta \vdash_{\mathsf{e}}$.
2. If $\Gamma \vdash_{\mathsf{e}} t = t' : A$, then $\Gamma \vdash_{\mathsf{e}} t : A$ and $\Gamma \vdash_{\mathsf{e}} t' : A$.
3. If $\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta$, then $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$ and $\Gamma \vdash_{\mathsf{e}} \sigma' : \Delta$.

In turn to prove validity of equality, one needs to know context conversion, as can be seen when analysing the following rule:

$$\frac{\Gamma \vdash_{\mathsf{e}} A = B : s \qquad \Gamma, A \vdash_{\mathsf{e}} C = D : s'}{\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, A\, C = \mathsf{Fun}\, B\, D : s''}\ (\text{FUN-CNG})\ (s, s', s'') \in R$$

By i.h. on the first premise, we have $\Gamma \vdash_{\mathsf{e}} A : s$ and $\Gamma \vdash_{\mathsf{e}} B : s$; also by i.h. on the second one, we obtain $\Gamma, A \vdash_{\mathsf{e}} C : s'$ and $\Gamma, A \vdash_{\mathsf{e}} D : s'$. So we can conclude $\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, A\, C : s''$, but not $\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, B\, D : s''$. Notice, however, that we can prove $\vdash_{\mathsf{e}} \Gamma, A = \Gamma, B$; which, had we context conversion at hand, would suffice to conclude $\Gamma, B \vdash_{\mathsf{e}} D : s'$ and so $\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, B\, D : s''$.

Lemma 25 (*Context conversion*)
If $\Gamma \vdash_{\mathsf{e}} J$ and $\vdash_{\mathsf{e}} \Gamma = \Delta$, then $\Delta \vdash_{\mathsf{e}} J$.

To prove context conversion, we need in turn Lemma 24. Indeed, consider the case (AXIOM), we know $\vdash_{\mathsf{e}} \Gamma = \Delta$ and $\Gamma \vdash_{\mathsf{e}}$, but we should have $\Delta \vdash_{\mathsf{e}}$. To deal with this circularity we prove all these points simultaneously.

Lemma 26 (*Validity of judgements for* $\lambda^{\sigma=}$)

1. If $\vdash_{\mathsf{e}} \Gamma = \Gamma'$, then (a) symmetry of equality: $\vdash_{\mathsf{e}} \Gamma' = \Gamma$, and (b) validity of contexts: $\Gamma \vdash_{\mathsf{e}}$ and $\Gamma' \vdash_{\mathsf{e}}$.
2. If $\Gamma \vdash_{\mathsf{e}} t : A$ and $\vdash_{\mathsf{e}} \Gamma = \Gamma'$, then (a) type validity: $\Gamma \vdash_{\mathsf{e}} A$ and $\Gamma' \vdash_{\mathsf{e}} A$, and (b) context conversion: $\Gamma' \vdash_{\mathsf{e}} t : A$.
3. If $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$ and $\vdash_{\mathsf{e}} \Gamma = \Gamma'$, then (a) validity of context: $\Delta \vdash_{\mathsf{e}}$, and (b) context conversion: $\Gamma' \vdash_{\mathsf{e}} \sigma : \Delta$.
4. If $\Gamma \vdash_{\mathsf{e}} t = t' : A$ and $\vdash_{\mathsf{e}} \Gamma = \Gamma'$, then (a) type validity: $\Gamma \vdash_{\mathsf{e}} A$ and $\Gamma' \vdash_{\mathsf{e}} A$; (b) equality validity: $\Gamma \vdash_{\mathsf{e}} t : A$, $\Gamma \vdash_{\mathsf{e}} t' : A$, $\Gamma' \vdash_{\mathsf{e}} t : A$, and $\Gamma' \vdash_{\mathsf{e}} t' : A$; and (c) context conversion: $\Gamma' \vdash_{\mathsf{e}} t = t' : A$.
5. If $\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta$ and $\vdash_{\mathsf{e}} \Gamma = \Gamma'$, then (a) validity of context: $\Delta \vdash_{\mathsf{e}}$; (b) type validity: $\Delta \vdash_{\mathsf{e}}$; (c) equality validity: $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$ and $\Gamma \vdash_{\mathsf{e}} \sigma' : \Delta$, $\Gamma' \vdash_{\mathsf{e}} \sigma : \Delta$, and $\Gamma' \vdash_{\mathsf{e}} \sigma' : \Delta$; and (d) context conversion: $\Gamma' \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta$.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

Proof
All the points are proved by simultaneous induction on derivations. Let us analyse the following rule:

$$\frac{\vDash_{\mathsf{e}} \Gamma = \Gamma' \qquad \Gamma \vDash_{\mathsf{e}} A = B : s}{\vDash_{\mathsf{e}} \Gamma, A = \Gamma', B} \text{(EXT-EQ-CTX)}$$

By i.h. on the first premise, we know (a) $\Gamma \vDash_{\mathsf{e}}$, (b) $\Gamma' \vDash_{\mathsf{e}}$, and (c) $\vDash_{\mathsf{e}} \Gamma' = \Gamma$. By i.h. on the second premise, using the first premise as the equality of contexts, we obtain (d) $\Gamma \vDash_{\mathsf{e}} A : s$, (e) $\Gamma \vDash_{\mathsf{e}} B : s$, (f) $\Gamma' \vDash_{\mathsf{e}} A : s$, (g) $\Gamma' \vDash_{\mathsf{e}} B : s$, and (h) $\Gamma' \vDash_{\mathsf{e}} A = B : s$. From (c) and (h), we conclude $\vDash_{\mathsf{e}} \Gamma', B = \Gamma, A$; validity of of Γ, A follows from (a) and (d). □

An immediate corollary of context conversion is context conversion with respect to the heterogeneous equality of contexts.

Corollary 27
If $\Gamma \vDash_{\mathsf{e}} t : A$ and $\vDash_{\mathsf{e}} \Gamma \approx \Delta$, then $\Delta \vDash_{\mathsf{e}} t : A$.

3 Equivalence between λ^{σ} and $\lambda^{\sigma=}$

Now that we have formally introduced the two families of PTSs, we can precisely state the equivalence between the syntactical, λ^{σ}, and semantical, $\lambda^{\sigma=}$, versions.

Theorem 28 (*Equivalence between λ^{σ} and $\lambda^{\sigma=}$ (Siles, 2010)*)
1. $\Gamma \vDash_{\mathsf{e}} t : A$ iff $\Gamma \vdash t : A$.
2. $\Gamma \vDash_{\mathsf{e}} t = t' : A$ iff $\Gamma \vdash t : A, \Gamma \vdash t' : A,$ and $t \equiv_{\beta \mathsf{x}} t'$.

In the Introduction, we have commented on the previous works on this equivalence. Let us remark that the direction from $\lambda^{\sigma=}$to λ^{σ} is straightforward; a fact formally proved by Adams and recognised in Geuvers' PhD thesis (Geuvers, 1993). The key result to prove the other implication depends on *subject reduction* for $\lambda^{\sigma=}$: *if* $\Gamma \vDash_{\mathsf{e}} t : A$ *and* $t \equiv_{\beta \mathsf{x}} t'$, *then* $\Gamma \vDash_{\mathsf{e}} t' : A$. In Sections 4 and 5, we use the machinery of NbE to prove a slightly weaker, but strong enough, version of that lemma for a class of PTSs. In contrast with our semantical approach, Adams' and Siles' results are based on a typed parallel reduction for PTSs with typed equality. It is not clear whether their method is applicable to domain-free PTSs.

The proof that any derivation in $\lambda^{\sigma=}$ has a corresponding derivation in λ^{σ} is relatively straightforward and can be proved for every PTS. We have formally proved it in Agda.

Theorem 29 (*From $\lambda^{\sigma=}$to λ^{σ}*)
If $\Gamma \vDash_{\mathsf{e}} J$, then $\Gamma \vdash J$.

Proof
The proof is by mutual induction on all form of judgements; if the last rule used is an introductory rule, then one can use the same rule on the untyped system. Typed equalities are transformed into untyped conversions by removing the type information. □

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

The proof in the other direction will occupy the rest of the paper and is quite different from the others. Let us sketch here the strategy, in the hope of it being a guide through the technicalities. If we try to prove the statement "$\Gamma \vdash J$ implies $\Gamma \vdash_{\mathsf{e}} J$" by induction on derivations, we encounter a problem in the conversion rule: the premises of the rule are $\Gamma \vdash t : A$, $\Gamma \vdash B : s$, and $A \equiv_{\beta \mathsf{x}} B$. We get $\Gamma \vdash_{\mathsf{e}} t : A$ and $\Gamma \vdash_{\mathsf{e}} B : s$ as the inductive hypotheses; however nothing tell us that there is a typed equivalence $\Gamma \vdash_{\mathsf{e}} A = B : s'$ for any sort s'. In the light of Lemma 18, it is enought to prove $\Gamma \vdash_{\mathsf{e}} A \approx B$, in words: one has to find some term C and sorts s_1, s_2 such that $\Gamma \vdash_{\mathsf{e}} A = C : s_1$ and $\Gamma \vdash_{\mathsf{e}} B = C : s_2$. In the next section, we define a function $\mathbf{nbe} : \mathit{Term} \to \mathit{Term}$ with the following properties:

1. If $\Gamma \vdash_{\mathsf{e}} t : A$, then $\Gamma \vdash_{\mathsf{e}} t = \mathbf{nbe}(t) : A$;
2. if $t \equiv_{\beta \mathsf{x}} t'$, then $\mathbf{nbe}(t) \equiv \mathbf{nbe}(t')$.

With such a function at hand, we can complete the proof for conversion: let $C = \mathbf{nbe}(A)$; from both properties we get $\Gamma \vdash_{\mathsf{e}} B = C : s$. On the other hand, from inversion on $\Gamma \vdash_{\mathsf{e}} t : A$ there is a sort s_1 such that $\Gamma \vdash_{\mathsf{e}} A : s_1$; therefore by the first property we obtain $\Gamma \vdash_{\mathsf{e}} A = C : s_1$.

Let us remark that any function **nbe** with those properties will be enough to finish the proof. For example, one could get such a function from a proof of normalisation which can be carried out in Agda (Danielsson, 2007; Altenkirch & Chapman, 2009). The rest of the paper is not formalised in Agda, as we use fairly complicated mathematics (mainly the existence of solutions for equations of domain equations).

4 Semantics

In this section, we construct a model suitable to define a NbE function for a restricted class of PTSs. This model validates the untyped conversion $\to_{\beta \mathsf{x}}$; later we exploit this property to prove a weak version of subject reduction. We refer the reader to Abel *et al.* (2011) for an explanation of the model and the NbE function. The model construction is based on the existence of a well-founded relation on sorts; we discuss briefly the issue of modelling impredicative type systems.

4.1 Domain model and reification

Given a signature $\mathscr{S} = (S, A, R)$, our model will be based on the least solution for the following recursive domain equation, which can be solved in the category $\mathbf{DOM}_\perp$ of pointed ω-*cpo* and continuous functions (Abramsky & Jung, 1994):

$$D_{\mathscr{S}} = \{\top\}_\perp \oplus D_{\mathscr{S}} \times D_{\mathscr{S}} \oplus \mathbb{N}_\perp \oplus [D_{\mathscr{S}} \to D_{\mathscr{S}}] \oplus D_{\mathscr{S}} \times [D_{\mathscr{S}} \to D_{\mathscr{S}}] \oplus D_{\mathscr{S}} \times D_{\mathscr{S}} \oplus \{S\}_\perp$$

We use the sub-index $\mathscr{S}$ to indicate that the domain is parameterised by the signature; we skip the sub-index in the domain $D_{\mathscr{S}}$ when no confusion arises. Let us remark that the first two components of the domain are used to internalise environments, which are the interpretation of substitutions. We use the following

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

conventions for denoting non-bottom elements of $D_{\mathscr{S}}$:

$\top$	(d, d')	for $d, d' \in D_{\mathscr{S}}$
$\mathsf{Var}\, i$	s	for $i \in \mathbb{N}$ and $s \in S$
$\mathsf{lam}\, f$	$\mathsf{Fun}\, d\, f$	for $d \in D_{\mathscr{S}}$ and $f \in [D_{\mathscr{S}} \to D_{\mathscr{S}}]$
$\mathsf{App}\, d\, d'$		for $d, d' \in D_{\mathscr{S}}$

We internalise application, denoted by $\_ \cdot \_$, as a binary operation on D and projections, π_1 and π_2, as unary functions over D.

$$d \cdot e = \begin{cases} f\ e & \text{if } d = \mathsf{lam}\, f \\ \mathsf{App}\, d\, e & \text{if } d = \mathsf{Var}\, i \text{ or } d = \mathsf{App}\, d'\, d'' \\ \bot & \text{otherwise} \end{cases} \qquad \pi_i\ d = \begin{cases} d_i & \text{if } d = (d_1, d_2) \\ \bot & \text{otherwise} \end{cases}$$

At this point, we can introduce a *reification function* which maps some elements of D to pre-terms; later, the normalisation function consists of the composition of reification with the interpretation function. Notice that the lack of (η) allows us to simplify the definition of NbE with respect to Abel *et al.* (2011).

Definition 30 (*Reification function*)

$$\mathsf{R}_j\,(s) = s \qquad \mathsf{R}_j\,(\mathsf{Fun}\, X\, F) = \mathsf{Fun}\,(\mathsf{R}_j\, X)\,(\mathsf{R}_{j+1}\,(F\,(\mathsf{Var}\, j)))$$

$$\mathsf{R}_j\,(\mathsf{App}\, d\, d') = \mathsf{App}\ (\mathsf{R}_j\, d)\ (\mathsf{R}_j\, d') \qquad \mathsf{R}_j\,(\mathsf{lam}\, f) = \lambda(\mathsf{R}_{j+1}\,(f\,(\mathsf{Var}\, j)))$$

$$\mathsf{R}_j\,(\mathsf{Var}\, i) = \begin{cases} \mathsf{q} & \text{if } j \leqslant i+1 \\ \mathsf{q}\,\mathsf{p}^{j-(i+1)} & \text{if } j > i+1 \end{cases}$$

We can identify normal forms and neutral values in the domain by taking the inverse image of the reification function over the sets of normal forms and neutral terms, respectively:

$$\mathsf{Ne} = \bigcap_{i \in \mathbb{N}} \{d \in D \mid \mathsf{R}_i\, d \in \mathrm{Ne}\} \quad \text{and} \quad \mathsf{Nf} = \bigcap_{i \in \mathbb{N}} \{d \in D \mid \mathsf{R}_i\, d \in \mathrm{Nf}\}$$

4.2 Denoting types and sorts

In a previous work (Abel *et al.*, 2011), we have used Dybjer's (2000) schema of induction–recursion to define the denotation of types; small types were interpreted as subsets of D and the interpretation of universes was defined together with a function $[\_]$ mapping elements of the subset of (semantical) types to subsets of D. For PTSs, it is not possible to use the same approach, because there are signatures which do not have a well-founded order between sorts.

Given a signature $\mathscr{S} = (S, A, R)$, let us analyse informally which elements of $D_{\mathscr{S}}$ should be in the subset denoting types; remember that a *type* is either a sort or a term that can be typed with a sort. By rule (AXIOM), we know that sorts are among the possible *types*; so, S should be included in the universe *types*. Note also that terms typed with a sort can also be types; which suggests to add any element of $[s]$

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

to *types*, for each $s \in S$. The definition of $[s] \subseteq D_{\mathscr{S}}$ depends both on axioms and rules of the system: axioms tell us that if $(c, s) \in A$, then $c \in [s]$. The set R of rules indicates when one element of the form $\mathsf{Fun}\, d\, f$ should be a member of $[s]$. Since we add $\mathsf{Fun}\, d\, f$ to *types* as soon as it is in some $[s]$, we have to define $[\mathsf{Fun}\, d\, f]$.

If we try to apply that reasoning for the type system *type-in-type* given by the signature $\lambda^\star = (\{\star\}, \{(\star, \star)\}, \{(\star, \star, \star)\})$, we would have the following clause:

$$\mathsf{Fun}\, d\, f \in [\star], \text{ if } d \in [\star] \text{ and } f\, e \in [\star] \text{ for all } e \in [d]$$

This is a typical issue of impredicative definitions: to decide if some element belongs to $[\star]$ we should test a condition over all the elements in that very same set.

We can safely use the inductive–recursive schema by restricting our attention to predicative PTSs as captured by the following definition.

Definition 31 (*Predicative PTSs*)
A specification $\mathscr{S} = (S, A, R)$ is called *predicative* if there is a well-founded order over the sorts, $\preccurlyeq \subseteq S^2$, such that

1. if $(s_1, s_2) \in A$, then $s_1 \prec s_2$;
2. if $(s_1, s_2, s_3) \in R$, then $s_1 \preccurlyeq s_3$ and $s_2 \preccurlyeq s_3$.

The next definition introduces subsets of D for predicative PTSs. In order to understand it, one should think that first one defines $[s]$ inductively for every minimal element $s \in S$ (minimal with respect to the underlying order making the signature predicative).

Definition 32
Let $\mathscr{S} = (S, A, R)$ be a predicative specification. The following rules define simultaneously $T \subseteq D_{\mathscr{S}}$ and $[d] \subseteq D_{\mathscr{S}}$, for every $d \in T$.

$$\frac{}{\mathsf{Ne} \subseteq T} \qquad \frac{}{S \subseteq T} \qquad \frac{d \in T \qquad f\, e \in T, \text{ for all } e \in [d]}{\mathsf{Fun}\, d\, f \in T}$$

$$\frac{(s, s') \in A}{s \in [s']} \qquad \frac{(s, s', s'') \in R \qquad d \in [s] \qquad f\, e \in [s'], \text{ for all } e \in [d]}{\mathsf{Fun}\, d\, f \in [s'']}$$

$$\frac{d \in T}{\mathsf{Ne} \subseteq [d]} \qquad \frac{g\, e \in [f\, e], \text{for all } e \in [d]}{\mathsf{lam}\, g \in [\mathsf{Fun}\, d\, f]}$$

The well-founded order over S extends to a well-founded order $\sqsubseteq$ over $[s]$, for every $s \in S$, and over T. We can characterise the order $\sqsubseteq$ as follows: an element d is minimal with respect to $\sqsubseteq$ if $d \in \mathsf{Ne}$, or $d = s \in S$ and s is minimal with respect to $\preccurlyeq$; $s \sqsubseteq s'$ if $s \preccurlyeq s'$, and if $\mathsf{Fun}\, d\, f \in T$, then $d \sqsubseteq \mathsf{Fun}\, d\, f$ and $f\, e \sqsubseteq \mathsf{Fun}\, d\, f$, for all $e \in [d]$.

Remark 33
Notice that if $d \in [\mathsf{Fun}\, X\, F]$ and $e \in [X]$, then $d \cdot e \in [F\, e]$.

By abusing the terminology, we say that $X \subseteq D$ is *saturated* if $\mathsf{Ne} \subseteq X \subseteq \mathsf{Nf}$. The first result we prove using well-founded induction over T is that every subset denoting types, that is $[d]$ for every element $d \in T$, is saturated.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

Lemma 34
For all $d \in T$, $[d]$ is saturated. An immediate corollary is that T itself is saturated.

4.3 Interpretation and soundness

The missing pieces of the semantics are the interpretation of terms and the satisfaction of judgements in the model.

$$\begin{aligned}
[\![_]\!]^t_ &: \mathit{Term} \times D \to D & [\![_]\!]^s_ &: \mathit{Sub} \times D \to D \\
[\![s]\!]^t d &= s & [\![\mathsf{id}]\!]^s d &= d \\
[\![\mathsf{q}]\!]^t d &= \pi_2\ d & [\![\langle \sigma, t\rangle]\!]^s d &= ([\![\sigma]\!]^s d, [\![t]\!]^t d) \\
[\![\mathsf{App}\ t\ r]\!]^t d &= [\![t]\!]^t d \cdot [\![r]\!]^t d & [\![\mathsf{p}]\!]^s d &= \pi_1\ d \\
[\![\lambda t]\!]^t d &= \mathsf{lam}\,(e \mapsto [\![t]\!]^t (d, e)) & [\![\sigma\ \delta]\!]^s d &= [\![\sigma]\!]^s ([\![\delta]\!]^s d) \\
[\![t\ \sigma]\!]^t d &= [\![t]\!]^t ([\![\sigma]\!]^s d) \\
[\![\mathsf{Fun}\ A\ B]\!]^t d &= \mathsf{Fun}\,([\![A]\!]^t d)\,(e \mapsto [\![B]\!]^t (d, e))
\end{aligned}$$

Since the interpretation is given for pre-terms and pre-substitutions, we can prove that the interpretation models the untyped equality.

Lemma 35
If $t \equiv_{\beta\mathsf{x}} t'$ and $d \in D$, then $[\![t]\!]d = [\![t']\!]d$.

The semantics of judgements uses the set T and the mapping $[\_] : T \to \mathscr{P}(D)$. In the following definition, we state formally the validity of judgements in the model and introduce, at the same time, the semantics of well-formed contexts.

Definition 36 (*Validity*)

1. Well formedness of contexts:
   a. $\diamond \vDash$;
   b. $\Gamma, A \vDash$ iff $\Gamma \vDash A : s$ for some sort s;
2. Semantics of valid contexts:
   a. $\top \in [\![\diamond]\!]$;
   b. If $d \in [\![\Gamma]\!]$ and $d' \in [[\![A]\!]d]$, then $(d, d') \in [\![\Gamma, A]\!]$.
3. $\vDash \Gamma = \Gamma'$ iff $[\![\Gamma]\!] = [\![\Gamma]\!]'$.
4. $\Gamma \vDash t : A$ iff $\Gamma \vDash$ and $[\![t]\!]d \in [[\![A]\!]d]$, for all $d \in [\![\Gamma]\!]$.
5. $\Gamma \vDash \delta : \Delta$ iff $\Gamma \vDash$, $\Delta \vDash$ and $[\![\delta]\!]d \in [\![\Delta]\!]$, for all $d \in [\![\Gamma]\!]$.
6. $\Gamma \vDash t = t' : A$ iff $\Gamma \vDash t : A$ and $[\![t]\!]d = [\![t']\!]d$, for all $d \in [\![\Gamma]\!]$.
7. $\Gamma \vDash \delta = \delta' : \Delta$ iff $\Gamma \vDash \delta : \Delta$ and $[\![\delta]\!]d = [\![\delta']\!]d$, for all $d \in [\![\Gamma]\!]$.

Theorem 37 (*Soundness of the model*)
If $\Gamma \vdash_{\mathsf{e}} J$, then $\Gamma \vDash J$.

By soundness, we know that each typed term is interpreted as an element of the denotation of the type; moreover, we know that judgmentally equal elements have the same denotation. As a corollary, we know that they have the same normal form.

Corollary 38 (*Completeness of normalisation*)
If $\Gamma \vdash_{\mathsf{e}} t = t' : A$ and $d \in [\![\Gamma]\!]$, then $\mathsf{R}_j\,([\![t]\!]d) \equiv \mathsf{R}_j\,([\![t']\!]d)$, for any $j \in \mathbb{N}$.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

5 Correctness of NbE

The proof of the equivalence between λ^{σ} and $\lambda^{\sigma=}$can be completed once subject reduction is proved for the latter:

Definition 39 (*Subject reduction*)
Let $\mathscr{S} = (S, A, R)$ be a signature, we say that the corresponding system $\lambda^{\sigma=}$satisfies *Subject reduction* if $\Gamma \vdash_{\mathsf{e}} t : A$ and $t \to_{\beta\mathsf{x}} t'$ imply $\Gamma \vdash_{\mathsf{e}} t = t' : A$.

Subject reduction in turn depends on having injectivity of Fun:

Definition 40 (*Injectivity of* Fun)
We say that a system satisfies Injectivity of Fun, denoted by $Inj(\mathsf{Fun})$, if $\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, A\, B = \mathsf{Fun}\, A'\, B' : C$ implies $\Gamma \vdash_{\mathsf{e}} A = A' : s_0$ and $\Gamma, A \vdash_{\mathsf{e}} B = B' : s_1$, for some sorts s_0 and s_1.

Adams devised a new type system, with more annotations, satisfying several properties, which lead to a proof of $Inj(\mathsf{Fun})$ for $\lambda^{\sigma=}$. His proofs depends on uniqueness of types, which in turn depends on the signature being injective. Siles and Herbelin get rid of that condition by modifying the annotations; they manage to prove subject reduction for the annotated system which can later be used to transfer that result for $\lambda^{\sigma=}$. Their results are purely syntactical and do not rely on the normalisation of the calculus.

As we advanced at the end of Section 3, our approach is different. Following previous works (Abel *et al.*, 2007, 2008, 2011), we use Kripke logical relation to prove the correctness of the normalisation function **nbe**: if $\Gamma \vdash_{\mathsf{e}} t : A$, then $\Gamma \vdash_{\mathsf{e}} t = \mathbf{nbe}(t) : A$. From correctness and Lemma 35, we get *weak subject reduction* for $\lambda^{\sigma=}$: if $\Gamma \vdash_{\mathsf{e}} t : A$ and $t \to^{*}_{\beta\mathsf{x}} t'$, then there exists t'' such that $\Gamma \vdash_{\mathsf{e}} t = t'' : A$, and $t' \to^{*}_{\beta\mathsf{x}} t''$.

The logical relations are slightly different than those for Martin–Löf type theory: for PTSs, there is only one kind of logical relations relating terms typable with A under context Γ with elements of $[X]$ with $X \in T$:

$$\Gamma \vdash_{\mathsf{e}} \_ : A \bowtie \_ \in [X] \subseteq \{t \mid \Gamma \vdash_{\mathsf{e}} t : A\} \times [X]$$

Definition 41 (*Logical Relations*)
Let $X \in T$, $\Gamma \vdash_{\mathsf{e}} A$, $d \in [X]$, and $\Gamma \vdash_{\mathsf{e}} t : A$, then $\Gamma \vdash_{\mathsf{e}} t : A \bowtie d \in [X]$ if and only if

1. for $X \in \mathsf{Ne}$
   a. there exists $s \in S$, such that $\Gamma \vdash_{\mathsf{e}} A : s$ and for all $\Delta \leqslant^{i} \Gamma$, $\Delta \vdash_{\mathsf{e}} A\, \mathsf{p}^{i} = \mathsf{R}_{|\Delta|}\, X : s$; and
   b. for all $\Delta \leqslant^{i} \Gamma$, $\Delta \vdash_{\mathsf{e}} t\, \mathsf{p}^{i} = \mathsf{R}_{|\Delta|}\, d : A\, \mathsf{p}^{i}$.
2. for $X = s \in S$ and $d \in \mathsf{Ne} \cup S$:
   a. either $A = s$ or there exists $s' \in S$ such that, for all $\Delta \leqslant^{i} \Gamma$, $\Delta \vdash_{\mathsf{e}} A\, \mathsf{p}^{i} = s : s'$; and
   b. for all $\Delta \leqslant^{i} \Gamma$, $\Delta \vdash_{\mathsf{e}} t\, \mathsf{p}^{i} = \mathsf{R}_{|\Delta|}\, d : s$.
3. for $X = s \in S$ and $d = \mathsf{Fun}\, X'\, F$
   a. either $A = s$ or there exists $s' \in S$ such that, for all $\Delta \leqslant^{i} \Gamma$, $\Delta \vdash_{\mathsf{e}} A\, \mathsf{p}^{i} = s : s'$; and

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

   b. there exist $s_0, s_1 \in S$, such that $(s_0, s_1, s) \in R$; and
   c. there exist $A', B \in \mathit{Term}$ such that $\Gamma \vDash_{\mathsf{e}} t = \mathsf{Fun}\, A'\, B : s$
   d. $\Gamma \vDash_{\mathsf{e}} A' : s_0 \bowtie X' \in [s_0]$; and
   e. for all $\Delta \leqslant^i \Gamma$ and $\Delta \vDash_{\mathsf{e}} r : A'\, \mathsf{p}^i \bowtie e \in [X']$, $\Delta \vDash_{\mathsf{e}} B\, \langle \mathsf{p}^i, r\rangle : s_1 \bowtie F\, e \in [s_1]$.
4. for $X = \mathsf{Fun}\, X'\, F$
   a. there exist $s_0, s_1, s \in S$, such that $(s_0, s_1, s) \in R$; and
   b. there exist $A', B \in \mathit{Term}$, such that $\Gamma \vDash_{\mathsf{e}} A = \mathsf{Fun}\, A'\, B : s$ and $\Gamma \vDash_{\mathsf{e}} A' : s_0 \bowtie X' \in [s_0]$; and
   c. for all $\Delta \leqslant^i \Gamma$ and $\Delta \vDash_{\mathsf{e}} r : A'\, \mathsf{p}^i \bowtie e \in [X']$, $\Delta \vDash_{\mathsf{e}} \mathsf{App}\, (t\, \mathsf{p}^i)\, r : B\, \langle \mathsf{p}^i, r\rangle \bowtie d \cdot e \in [F\, e]$; and
   d. if $d = \mathsf{lam}\, f$, then there exists $t' \in \mathit{Term}$, such that $\Gamma \vDash_{\mathsf{e}} t = \lambda\, t' : A$; and
   e. if $d \in \mathsf{Ne}$, then for all $\Delta \leqslant^i \Gamma$, $\Delta \vDash_{\mathsf{e}} t\, \mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : A\, \mathsf{p}^i$.

The following two technical lemmas are necessary to prove the third one, sometimes called *in-out* lemma. This lemma allows us to deduce that if a term t is related with some element d, then t is judgementally equal to the reification of d. The proofs of these three lemmas are presented in the Appendix.

Lemma 42 (*Preservation of the logical relation by judgemental equality*)
Let $\Gamma \vDash_{\mathsf{e}} t : A \bowtie d \in [X]$.

1. If $\Gamma \vDash_{\mathsf{e}} t = t' : A$, then $\Gamma \vDash_{\mathsf{e}} t' : A \bowtie d \in [X]$.
2. If $\Gamma \vDash_{\mathsf{e}} A : s \bowtie X \in [s]$ and $\Gamma \vDash_{\mathsf{e}} A = B : s$, then $\Gamma \vDash_{\mathsf{e}} t : B \bowtie d \in [X]$.

Lemma 43 (*Monotonicity of logical relations*)

1. If $\Gamma \vDash_{\mathsf{e}} t : s \bowtie d \in [s]$ and $\Delta \leqslant^i \Gamma$, then $\Delta \vDash_{\mathsf{e}} t\, \mathsf{p}^i : s \bowtie d \in [s]$.
2. If $\Gamma \vDash_{\mathsf{e}} A : s$, $\Gamma \vDash_{\mathsf{e}} t : A \bowtie d \in [X]$ and $\Delta \leqslant^i \Gamma$, then $\Delta \vDash_{\mathsf{e}} t\, \mathsf{p}^i : A\, \mathsf{p}^i \bowtie d \in [X]$.

Lemma 44 (*Derivability of equality of reified elements*)

1. Let $\Gamma \vDash_{\mathsf{e}} t : A \bowtie d \in [X]$ and $\Delta \leqslant^i \Gamma$, then $\Delta \vDash_{\mathsf{e}} t\, \mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : \mathsf{R}_{|\Delta|}\, X$.
2. Conversely, if $\Gamma \vDash_{\mathsf{e}} A : s \bowtie X \in [s]$, $k \in \mathsf{Ne}$ and $\Delta \vDash_{\mathsf{e}} t\, \mathsf{p}^i = \mathsf{R}_{|\Delta|}\, k : A\, \mathsf{p}^i$, for all $\Delta \leqslant^i \Gamma$, then $\Gamma \vDash_{\mathsf{e}} t : A \bowtie k \in [X]$.

To deduce correctness of NbE, we just have to show that every well-typed term t is related with its denotation under a suitable environment. The *fundamental theorem* is a more general version of that result; its statement requires to introduce logical relations between substitutions and environments modelling contexts. These relations are defined by recursion on the codomain of substitutions.

Definition 45 (*Logical relation for substitutions*)
If $\Gamma \vDash_{\mathsf{e}}$ and $\Delta \vDash_{\mathsf{e}}$, then $\Gamma \vDash_{\mathsf{e}} \_ : \Delta \bowtie \_ \in [\![\Delta]\!] \subseteq \{\sigma \mid \Gamma \vDash_{\mathsf{e}} \sigma : \Delta\} \times \{d \mid d \in [\![\Delta]\!]\}$.

1. $\Gamma \vDash_{\mathsf{e}} \sigma : \diamond \bowtie d \in [\![\diamond]\!]$.
2. $\Gamma \vDash_{\mathsf{e}} \sigma : \Delta, A \bowtie (d, d') \in [\![\Delta, A]\!]$ iff $\Gamma \vDash_{\mathsf{e}} \mathsf{p}\, \sigma : \Delta \bowtie d \in [\![\Delta]\!]$ and $\Gamma \vDash_{\mathsf{e}} \mathsf{q}\, \sigma : A\, (\mathsf{p}\, \sigma) \bowtie d' \in [[\![A]\!] d]$.

By induction on the codomain of substitutions, we can easily prove preservation of the relations by judgemental equality of substitutions and weakening. Notice the lack of a counterpart for lemma 44, which is justified because we do not reify environments.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

Lemma 46
Let $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta \bowtie d \in [\![\Delta]\!]$.

1. If $\Gamma \vdash_{\mathsf{e}} \sigma = \sigma' : \Delta$, then $\Gamma \vdash_{\mathsf{e}} \sigma' : \Delta \bowtie d \in [\![\Delta]\!]$.
2. If $\Theta \leqslant^i \Gamma$, then $\Theta \vdash_{\mathsf{e}} \sigma\, \mathsf{p}^i : \Delta \bowtie d \in [\![\Delta]\!]$.

Theorem 47 (*Fundamental theorem of logical relations*)
1. If $s \in S$, s is a top-sort, $\Gamma \vdash_{\mathsf{e}} t : s$, and $\Delta \vdash_{\mathsf{e}} \delta : \Gamma \bowtie d \in [\![\Gamma]\!]$, then $\Delta \vdash_{\mathsf{e}} t\,\delta : s \bowtie [\![t]\!]d \in [s]$.
2. If $\Gamma \vdash_{\mathsf{e}} t : A$ and $\Delta \vdash_{\mathsf{e}} \delta : \Gamma \bowtie d \in [\![\Gamma]\!]$, then $\Delta \vdash_{\mathsf{e}} t\,\delta : A\,\delta \bowtie [\![t]\!]d \in [[\![A]\!]d]$, providing A is not a top-sort.
3. If $\Gamma \vdash_{\mathsf{e}} \gamma : \Theta$ and $\Delta \vdash_{\mathsf{e}} \delta : \Gamma \bowtie d \in [\![\Gamma]\!]$, then $\Delta \vdash_{\mathsf{e}} \gamma\,\delta : \Theta \bowtie [\![\gamma]\!]d \in [\![\Theta]\!]$.

Proof
By induction on typing derivations.

1. The first two points are shown by the following cases.
   a. $\Gamma \vdash_{\mathsf{e}} s' : s$ by (AXIOM): we have $[\![s]\!]d = s$, since $(s, s') \in A$ we also have $s \in [s']$, and the reification of s is s itself. On the other hand, we can use (SUB-SORT) to conclude $\Delta' \vdash_{\mathsf{e}} (s\,\delta)\,\mathsf{p}^i = s : s'$, for any $\Delta' \leqslant^i \Delta$.
   b. $\Gamma \vdash_{\mathsf{e}} \mathsf{Fun}\, A\, B : s_3$ by (FUN-F): first we need to prove that there exists $A', B' \in \mathit{Term}$ such that $\Delta \vdash_{\mathsf{e}} (\mathsf{Fun}\, A\, B)\,\delta = \mathsf{Fun}\, A'\, B' : s_3$; we get this by (FUN-SUB): $A' = A\,\delta$ and $B' = B\,\langle \delta\, \mathsf{p}, \mathsf{q}\rangle$. The second point to prove is $\Delta \vdash_{\mathsf{e}} A\,\delta : s_1 \bowtie [\![A]\!]d \in [s_1]$; this follows from the i.h. on $\Gamma \vdash_{\mathsf{e}} A : s_1$. Finally, let $\Delta' \vdash_{\mathsf{e}} r : (A\,\delta)\,\mathsf{p}^i \bowtie e \in [[\![A]\!]d]$; then, by definition of logical relations for substitutions, $\Delta' \vdash_{\mathsf{e}} \langle \delta\, \mathsf{p}^i, r\rangle : \Gamma, A \bowtie (d, e) \in [\![\Gamma, A]\!]$. By i.h. on $\Gamma, A \vdash_{\mathsf{e}} B : s_2$, we have $\Delta' \vdash_{\mathsf{e}} B\,\langle \delta\, \mathsf{p}^i, r\rangle : s_2 \bowtie [\![B]\!](d, e) \in [s_2]$; by Lemma 42 we conclude $\Delta' \vdash_{\mathsf{e}} B\,\langle \delta\, \mathsf{p}, q\rangle\,\langle \mathsf{p}^i, r\rangle : s_2 \bowtie [\![B]\!](d, e) \in [s_2]$.
   c. $\Gamma \vdash_{\mathsf{e}} \lambda t : \mathsf{Fun}\, A\, B$ by (FUN-I): It is easily proved, using (ABS-SUB), that there exists t', such that $\Gamma \vdash_{\mathsf{e}} (\lambda t)\,\delta = \lambda t' : (\mathsf{Fun}\, A\, B)\,\delta$. The main point is proved in a similar way as we proved the second condition in the previous rule. First, we note $\mathsf{App}\ \lambda(t\,\langle \delta\, \mathsf{p}, \mathsf{q}\rangle)\,\mathsf{p}^i\ r = t\,\langle \delta\, \mathsf{p}^i, r\rangle$ by applying successively the following rules (APP-CONG), (COMP-SUBS), (BETA), (ASS-SUBS), (DIST-SUBS), (FST-SUBS), (SND-SUBS). Now let $\Delta' \leqslant^i \Delta$ and $\Delta' \vdash_{\mathsf{e}} r : (A\,\delta)\,\mathsf{p}^i \bowtie e \in [[\![A]\!]d]$; so by i.h. on $\Gamma, A \vdash_{\mathsf{e}} t : B$ we know $\Delta' \vdash_{\mathsf{e}} t\,\langle \delta\, \mathsf{p}^i, r\rangle : B\,\langle \mathsf{p}^i, r\rangle \bowtie [\![t]\!](d, e) \in [([\![B]\!]d)\, e]$.
   d. $\Gamma, A \vdash_{\mathsf{e}} \mathsf{q} : A\,\mathsf{p}$ by (HYP): by inversion on the hypothesis, $\Delta \vdash_{\mathsf{e}} \delta : \Gamma \bowtie d \in [\![\Gamma]\!]$, we know $\delta = \langle \delta', t\rangle$ and $d = (d', e)$; since $\mathsf{q}\,\langle \delta', t\rangle = t$ and $[\![\mathsf{q}]\!](d', e) = e$; the only point to prove $\Delta \vdash_{\mathsf{e}} t : A\,\delta' \bowtie e \in [[\![A]\!]d']$ comes again from the hypothesis.
   e. $\Gamma \vdash_{\mathsf{e}} t\,\sigma : A\,\sigma$ by (SUBS-TM): this is easily seen to hold by i.h. on both premises.
   f. $\Gamma \vdash_{\mathsf{e}} A\,\sigma : s$ by (SUBS-S): as in the previous point by i.h.
   g. $\Gamma \vdash_{\mathsf{e}} t : B$ by (CONV): we use the i.h. and Lemma 42.
2. For substitutions:
   a. $\Gamma \vdash_{\mathsf{e}} \mathsf{id} : \Gamma$ by (ID-SUBS): by Lemma 46 on the main hypothesis and $\Gamma \vdash_{\mathsf{e}} \mathsf{id}\,\delta = \delta : \Delta$.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

b. $\Gamma \vDash_{\mathsf{e}} \langle \sigma, t \rangle : \Theta, A$ by (EXT-SUBS): we apply the i.h. on both premises, $\Gamma \vDash_{\mathsf{e}} \sigma : \Theta$ and $\Gamma \vDash_{\mathsf{e}} t : A\,\sigma$, to get, respectively, $\Delta \vDash_{\mathsf{e}} \sigma\,\delta : \Theta \bowtie [\![\sigma]\!]d \in [\![\Theta]\!]$ and $\Delta \vDash_{\mathsf{e}} t\,\delta : (A\,\sigma)\,\delta \bowtie [\![t]\!]d \in [[\![A\,\sigma]\!]d]$. By Lemma 46 and Lemma 42 we conclude $\Delta \vDash_{\mathsf{e}} \mathsf{p}\,\langle \sigma\,\delta, t\,\sigma \rangle : \Theta \bowtie [\![\sigma]\!]d \in [\![\Theta]\!]$ and $\Delta \vDash_{\mathsf{e}} \mathsf{q}\,\langle \sigma\,\delta, t\,\sigma \rangle : A\,(\sigma\,\delta) \bowtie [\![t]\!]d \in [[\![A\,\sigma]\!]d]$.

c. $\Gamma \vDash_{\mathsf{e}} \sigma\,\gamma : \Theta$ by (COMP-SUBS): by applying the i.h. twice: first on $\Gamma \vDash_{\mathsf{e}} \gamma : \Sigma$ with the main hypothesis to get $\Delta \vDash_{\mathsf{e}} \delta\,\gamma : \Sigma \bowtie [\![\gamma]\!]d \in [\![\Sigma]\!]$; and then with that substitution we can apply the i.h. on $\Sigma \vDash_{\mathsf{e}} \sigma : \Theta$ to get $\Delta \vDash_{\mathsf{e}} \sigma\,(\gamma\,\delta) : \Theta \bowtie [\![\sigma]\!]([\![\gamma]\!]d) \in [\![\Theta]\!]$.

d. $\Gamma, A \vDash_{\mathsf{e}} \mathsf{p} : \Gamma$ by (FST-SUBS): by definition of the logical relation for substitutions $\Delta \vDash_{\mathsf{e}} \mathsf{p}\,\delta : \Gamma \bowtie d \in [\![\Gamma]\!]$.

□

By induction on $\Gamma \vDash_{\mathsf{e}}$, we define an environment ρ_Γ which models Γ and is logically related with the identity substitution id. The normalisation function is the composition of the reification with the evaluation under the environment ρ_Γ.

Definition 48 (*Canonical environment and normalisation function*)
$\mathbf{nbe}^\Gamma(t) = \mathsf{R}_{|\Gamma|}\,([\![t]\!]\rho_\Gamma)$, where $\rho_\diamond = \top$

$$\rho_{\Gamma,A} = (\rho_\Gamma, \mathsf{Var}\,|\Gamma|)$$

Lemma 49
If $\Gamma \vDash_{\mathsf{e}}$, then $\Gamma \vDash_{\mathsf{e}} \mathsf{id} : \Gamma \bowtie \rho_\Gamma \in [\![\Gamma]\!]$.

An immediate corollary of the previous results in this section is correctness of NbE.

Theorem 50 (*Correctness of NbE*)
If $\Gamma \vDash_{\mathsf{e}} t : A$, then $\Gamma \vDash_{\mathsf{e}} t : A \bowtie [\![t]\!]\rho_\Gamma \in [[\![A]\!]\rho_\Gamma]$ and $\Gamma \vDash_{\mathsf{e}} t = \mathbf{nbe}^\Gamma(t) : A$.

Proof
We can instantiate Theorem 47 with the identity substitution and the canonical environment deducing $\Gamma \vDash_{\mathsf{e}} t\,\mathsf{id} : A\,\mathsf{id} \bowtie [\![t]\!]\rho_\Gamma \in [[\![A]\!]\rho_\Gamma]$, and by Lemma 42, using (SUB-ID-T), $\Gamma \vDash_{\mathsf{e}} t : A \bowtie [\![t]\!]\rho_\Gamma \in [[\![A]\!]\rho_\Gamma]$; by Lemma 44 we obtain $\Gamma \vDash_{\mathsf{e}} t = \mathsf{R}_{|\Gamma|}\,([\![t]\!]\rho_\Gamma) : A$. □

Although it is not needed in our proof method, we can prove a weaker version of *Inj*(Fun) by using the fundamental theorem of logical relations and completeness of the normalisation function.

Lemma 51 (*Weak injectivity of* Fun)
If $\Gamma \vDash_{\mathsf{e}} \mathsf{Fun}\,A\,B = \mathsf{Fun}\,A'\,B' : C$, then $\Gamma \vDash_{\mathsf{e}} A \approx A'$ and $\Gamma, A \vDash_{\mathsf{e}} B \approx B'$.

Proof
From $\Gamma \vDash_{\mathsf{e}} \mathsf{Fun}\,A\,B = \mathsf{Fun}\,A'\,B' : C$, we get by validity of equality (Lemma 26,4b)

$$\Gamma \vDash_{\mathsf{e}} \mathsf{Fun}\,A\,B : C \quad (a) \qquad \text{and} \qquad \Gamma \vDash_{\mathsf{e}} \mathsf{Fun}\,A'\,B' : C \quad (b)$$

By inversion (Lemma 19) on (a), we know $\Gamma \vDash_{\mathsf{e}} A : s_0$ and $\Gamma, A \vDash_{\mathsf{e}} B : s_1$ for some $s_0, s_1 \in S$; by Theorem 50 we conclude $\Gamma \vDash_{\mathsf{e}} A = \mathbf{nbe}^\Gamma(A) : s_0$ and $\Gamma, A \vDash_{\mathsf{e}} B = \mathbf{nbe}^{\Gamma,A}(B) : s_1$. By the same token, on (b) we get $\Gamma \vDash_{\mathsf{e}} A' = \mathbf{nbe}^\Gamma(A') : t_0$ and $\Gamma, A' \vDash_{\mathsf{e}} B = \mathbf{nbe}^{\Gamma,A'}(B') : t_1$.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

To conclude, notice that by Corollary 38 we know $\mathbf{nbe}^{\Gamma}(\mathsf{Fun}\, A\, B) \equiv \mathbf{nbe}^{\Gamma}(\mathsf{Fun}\, A'\, B')$, which implies $\mathbf{nbe}^{\Gamma}(A) \equiv \mathbf{nbe}^{\Gamma}(A')$, therefore we can derive $\Gamma \vDash_{\mathsf{e}} A = \mathbf{nbe}^{\Gamma}(A') : s_0$. The proof of the heterogeneous equality for the codomains is slightly more involved: first notice that $\vDash_{\mathsf{e}} \Gamma, A' \approx \Gamma, A$, therefore by Corollary 27 we get $\Gamma, A \vDash_{\mathsf{e}} B' = \mathbf{nbe}^{\Gamma, A'}(B') : t_1$. Since $\mathbf{nbe}^{\Gamma, A}(B) \equiv \mathbf{nbe}^{\Gamma, A'}(B')$, we can also derive $\Gamma, A \vDash_{\mathsf{e}} B = \mathbf{nbe}^{\Gamma, A'}(B') : s_1$, so we have by transitivity $\Gamma, A \vDash_{\mathsf{e}} B \approx B'$, □

The counterexample for injectivity of products given by Siles and Herbelin (2012) uses the type of abstractions. Although not directly translatable, their example can be adapted.

Lemma 52 (*Failure of injectivity of* Fun)
There is a predicative signature $\mathscr{S} = (S, Ax, Rel)$, terms A, B, A', C, $s \in S$, and a context Γ, such that $\Gamma \vDash_{\mathsf{e}} \mathsf{Fun}\, A\, B = \mathsf{Fun}\, A'\, B : C$, but $\Gamma \nvDash_{\mathsf{e}} A = A' : s$.

Proof
The signature is given by $S = \{u, v_1, v_2, w, w_1, w_2, z\}$, $Ax = \{(u, v_1), (u, v_2), (v_1, w_1), (v_2, w_2), (v_1, w), (v_2, w), (w, z)\}$, and $Rel = \{(v_1, w, w), (v_2, w, w), (z, w_1, z), (z, w_2, z)\}$, with $s \prec s'$ if $(s, s') \in Ax$.

We can derive $\diamond \vDash_{\mathsf{e}} \lambda u : \mathsf{Fun}\, v_1\, v_1$ and also $\diamond \vDash_{\mathsf{e}} \lambda u : \mathsf{Fun}\, v_2\, v_2$. Notice that the absence of a rule (w_1, w_2, s) impedes us to derive $\diamond \vDash_{\mathsf{e}} \lambda u : \mathsf{Fun}\, v_1\, v_2$. Let $A_i = \mathsf{App}\ (\lambda u)\ v_i$, then $\diamond \vDash_{\mathsf{e}} A_i : v_i$; moreover if $\diamond \vDash_{\mathsf{e}} A_i : D_i$, then $\diamond \vDash_{\mathsf{e}} D_i \approx v_i$. We can easily prove $\diamond \vDash_{\mathsf{e}} A_i = u : v_i$. Leading to derivations of $\diamond \vDash_{\mathsf{e}} \mathsf{Fun}\, A_i\, v_1 = \mathsf{Fun}\, u\, v_1 : w$. From those premises is easy to conclude $\diamond \vDash_{\mathsf{e}} \mathsf{Fun}\, A_1\, v_1 = \mathsf{Fun}\, A_2\, v_1 : w$.

Now suppose $\diamond \vDash_{\mathsf{e}} A_1 = A_2 : s$, then by validity of equality $\diamond \vDash_{\mathsf{e}} A_i : s$; then $\diamond \vDash_{\mathsf{e}} s \approx v_1$ and $\diamond \vDash_{\mathsf{e}} s \approx v_2$, which is absurd by consistency. □

6 From λ^{σ} to $\lambda^{\sigma=}$

In the last section, we prove subject reduction for $\lambda^{\sigma=}$using correctness and completeness of NbE. Subject reduction is enough to complete the proof of the equivalence between λ^{σ} and $\lambda^{\sigma=}$for predicative PTSs. Since our model construction is only suitable for predicative PTSs, the results in this section are restricted to that class of PTSs.

Lemma 53 (*Logical relations are preserved by untyped conversion*)

1. If $\Gamma \vDash_{\mathsf{e}} t : A$ and $t \equiv_{\beta \mathsf{x}} t'$, then $\Gamma \vDash_{\mathsf{e}} t : A \bowtie [\![t']\!]\rho_{\Gamma} \in [[\![A]\!]\rho_{\Gamma}]$.
2. If $\Gamma \vDash_{\mathsf{e}} \sigma : \Delta$ and $\sigma \equiv_{\beta \mathsf{x}} \sigma'$, then $\Gamma \vDash_{\mathsf{e}} \sigma : \Delta \bowtie [\![\sigma']\!]\rho_{\Gamma} \in [\![\Delta]\!]$.

Proof
By Theorem 50 on the first hypothesis $\Gamma \vDash_{\mathsf{e}} t : A \bowtie [\![t]\!]\rho_{\Gamma} \in [[\![A]\!]\rho_{\Gamma}]$ and by Lemma 35, $[\![t]\!]\rho_{\Gamma} = [\![t']\!]\rho_{\Gamma}$, therefore $\Gamma \vDash_{\mathsf{e}} t : A \bowtie [\![t']\!]\rho_{\Gamma} \in [[\![A]\!]\rho_{\Gamma}]$. □

Corollary 54 (*Weak subject reduction for* $\lambda^{\sigma=}$)

1. If $\Gamma \vDash_{\mathsf{e}} t : A$ and $t \equiv_{\beta \mathsf{x}} t'$, then $\Gamma \vDash_{\mathsf{e}} t = \mathbf{nbe}_{\Gamma}(t') : A$.
2. If $\Gamma \vDash_{\mathsf{e}}$, $\Delta \vDash_{\mathsf{e}}$, and $\Gamma \equiv_{\beta \mathsf{x}} \Delta$, then there exists Σ such that $\vDash_{\mathsf{e}} \Gamma = \Sigma$ and $\vDash_{\mathsf{e}} \Sigma = \Delta$.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

Proof

1. Immediate from the previous lemma.
2. By induction on $\Gamma \equiv_{\beta\mathsf{x}} \Delta$: if both are the empty context, then Σ is also the empty context. If $\Gamma \equiv \Gamma', A$ and $\Delta \equiv \Delta', B$, then by i.h. there exists Σ' such that $\vdash_{\mathsf{e}} \Gamma' = \Sigma'$ and $\vdash_{\mathsf{e}} \Sigma' = \Delta'$; by inversion and by Corollary 54 we have $\Gamma' \vdash_{\mathsf{e}} A = \mathbf{nbe}_{\Gamma'}(B) : s$, so we conclude $\vdash_{\mathsf{e}} \Gamma', A = \Sigma', \mathbf{nbe}_{\Gamma'}(B)$. On the other hand, by Theorem 50, and (SYM-EQ), we also have $\Delta' \vdash_{\mathsf{e}} \mathbf{nbe}_{\Gamma'}(B) = B : s'$, so we can conclude $\vdash_{\mathsf{e}} \Delta', B = \Sigma', \mathbf{nbe}_{\Gamma'}(B)$.

□

Finally, we can prove the second part of Theorem 28; the formal proof of the next theorem assumes as a postulate the last corollary. Since that corollary depends on the results of the previous two sections, the next theorem only applies to predicative PTSs.

Theorem 55 (*From λ^{σ} to $\lambda^{\sigma=}$*)
Let $\mathscr{S} = (S, A, R)$ be a predicative signature.

1. If $\Gamma \vdash$, then $\Gamma \vdash_{\mathsf{e}}$.
2. If $\Gamma \vdash t : A$, then $\Gamma \vdash_{\mathsf{e}} t : A$.
3. If $\Gamma \vdash \sigma : \Delta$, then $\Gamma \vdash_{\mathsf{e}} \sigma : \Delta$.

Proof
By induction on derivations. We explain the case for (CONV): the premises are $\Gamma \vdash t : A$, $\Gamma \vdash B : s$ and $A \equiv_{\beta\mathsf{x}} B$. By i.h. on the first premise, we know $\Gamma \vdash_{\mathsf{e}} t : A$, and by i.h. on the second premise, $\Gamma \vdash_{\mathsf{e}} B : s$. By type validity on $\Gamma \vdash_{\mathsf{e}} t : A$, there is a sort s', such that $\Gamma \vdash_{\mathsf{e}} A : s'$. By Theorem 50 $\Gamma \vdash_{\mathsf{e}} A = \mathbf{nbe}(A) : s'$; therefore by (CONV), $\Gamma \vdash_{\mathsf{e}} t : \mathbf{nbe}(A)$. From $\Gamma \vdash_{\mathsf{e}} B : s$ and $A \equiv_{\beta\mathsf{x}} B$, by Corollary 54 we know $\Gamma \vdash_{\mathsf{e}} B = \mathbf{nbe}_{\Gamma}(A) : s$. Now we can apply (CONV) again to get $\Gamma \vdash_{\mathsf{e}} t : B$. □

Corollary 56
If $\Gamma \vdash t : A$, $\Gamma \vdash t' : A$, and $t \equiv_{\beta\mathsf{x}} t'$, then $\Gamma \vdash_{\mathsf{e}} t = t' : A$.

Proof
By Theorem 55, we have $\Gamma \vdash_{\mathsf{e}} t : A$ and $\Gamma \vdash_{\mathsf{e}} t' : A$. By Corollary 54, we know $\Gamma \vdash_{\mathsf{e}} t = \mathbf{nbe}_{\Gamma}(t') : A$; and, by Theorem 50 $\Gamma \vdash_{\mathsf{e}} t' = \mathbf{nbe}_{\Gamma}(t') : A$. We conclude by (SYM) and (TRANS). □

7 Conclusion and future work

We introduced a new formulation of PTSs with explicit substitutions and formalised in Agda some meta-theoretic results about them. By exploiting proof methods of previous works on NbE for dependent-type theory, we have been able to show the equivalence between predicative PTSs with typed equality and untyped conversion. Let us remark that it is the proof method, and not the result in itself, which should be considered as novel.

Our next goal is to prove the equivalence between λ^{σ} and $\lambda^{\sigma=}$extended with η. We are confident that our method can be adapted, again only for predicative PTSs,

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

and we are now working on that. There are, however, several details to be sorted out, beginning with the formulation of the η rule in the untyped setting. As far as we know, the equivalence is known only for functional, normalising PTSs (Geuvers & Werner, 1994) and for a restricted η-reduction in the setting of domain-free PTSs (Barthe & Coquand, 2006).

Another interesting direction is to extend NbE to impredicative PTSs, Abel (2010) has made some progress towards defining NbE for the core calculus of constructions. Since NbE can also cope with systems featuring subtyping (Fridlender & Pagano, 2013), the method should also be useful for PTSs with a hierarchy of sorts as in ECC (Luo, 1994).

We plan also to study more deeply the meta-theory of PTSs with explicit substitutions and complete its formalisation in Agda, for instance we would like to formalise the equivalence between λ^σ and the domain-free PTSs of Barthe & Sørensen (2000); this result would allow to transfer the results in this article to PTSs presented without explicit substitutions and named variables. It would be also interesting to extend our presentation with meta-variables, which will require to change the calculus of explicit substitutions since σ is not confluent in open terms.

Acknowledgements

We are grateful to the anonymous referees for their careful reading and suggestions. In particular, to one of them who discovered an embarrassing mistake.

References

Abadi, M., Cardelli, L., Curien, P.-L., & Lévy, J. J. (1990) Explicit substitutions. In Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages POPL '90: San Francisco, California, USA, January 1990. ACM, pp. 31–46.

Abel, A. (2010) Towards normalization by evaluation for the $\beta\eta$-calculus of constructions. In *Functional and Logic Programming*, Blume, M., Kobayashi, N. & Vidal, G. (eds), Lectures Notes in Computer Science, vol. 6009. Sendai, Japan, April 19–21, 2010, Berlin Heidelberg: Springer, pp. 224–239.

Abel, A., Coquand, T. & Dybjer, P. (2007) Normalization by evaluation for Martin-Löf type theory with typed equality judgements. In Lics. Wrocław, Poland, July 10–14 2007. IEEE Computer Society, pp. 3–12.

Abel, A., Coquand, T. & Dybjer, P. (2008) Verifying a semantic $\beta\eta$-conversion test for Martin-Löf type theory. In *Proceedings of the 9th International Conference on Mathematics of Program Construction, MPC 2008*, Marseille (Luminy), France, 15–18 July 2008, Audebaud, P. & Paulin-Mohring, C. (eds), Lectures Notes in Computer Science, vol. 5133. Springer, pp. 29–56.

Abel, A., Coquand, T. & Pagano, M. (2011) A modular type-checking algorithm for type theory with singleton types and proof irrelevance. *Log. Methods Comput. Sci.* **7**(2:4), 1–57.

Abramsky, S. & Jung, A. (1994) *Domain Theory*. Oxford University Press. pp. 1–168.

Adams, R. (2006) Pure type systems with judgemental equality. *J. Funct. Program.* **16**(2), 219–246.

Altenkirch, T. & Chapman, J. (2009) Big-step normalisation. *J. Funct. Program.* **19**(3-4), 311–333.

Asperti, A., Ricciotti, W., Coen, C. S. & Tassi, E. (2011) The Matita interactive theorem prover. In *Cade*, Wrocław, Poland, 31 July–5 August 2011, Bjørner, N. & Sofronie-Stokkermans, V. (eds), Lecture Notes in Computer Science, vol. 6803. Springer, pp. 64–69.

Barendregt, H. (1992) Lambda calculi with types. In *Handbook of Logic in Computer Science*, Abramsky, S., Gabbay, D. M., & Maibaum, T. S. E. (eds), Oxford University Press, pp. 117–309.

Barras, B. (1998) Verification of the interface of a small proof system in coq. In *Types for Proofs and Programs*, Aussois, France, December 15–19, 1996, Giménez, E. & Paulin-Mohring, C. (eds), Lectures Notes in Computer Science, vol. 1512. Berlin Heidelberg: Springer, pp. 28–45.

Barthe, G. & Coquand, T. (2006) Remarks on the equational theory of non-normalizing pure type systems. *J. Funct. Program.* **16**, 137–155.

Barthe, G., & Sørensen, M. H. (2000) Domain-free pure type systems. *J. Funct. Program.* **10**(5), 417–452.

Bloo, R. (2001) Pure type systems with explicit substitution. *Math. Struct. Comput. Sci.* **11**(1), 3–19.

Coq, Development Team. (2004) *The Coq Proof Assistant Reference Manual*. Version 8.0.

Curien, P.-L., Hardin, T. & Lévy, J.-J. (1996) Confluence properties of weak and strong calculi of explicit substitutions. *J. Assoc. Comput. Mach.* **43**, 362–397.

Curien, P.-L., Hardin, T. & Ríos, A. (1992) Strong normalization of substitutions. In *Mfcs*, Prague, Czechoslovakia, August 24–28, 1992, Havel, I. M., & Koubek, V. (eds), Lectures Notes in Computer Science, vol. 629. Springer, pp. 209–217.

Danielsson, N. A. (2007) A formalisation of a dependently typed language as an inductive-recursive family. In *Types for Proofs and Programs*, Cividale del Friuli, Italy, May 2–5, 2007, Altenkirch, T. & McBride, C. (eds), Lecture Notes in Computer Science, vol. 4502. Berlin Heidelberg: Springer, pp. 93–109.

Dybjer, P. (1996) Internal type theory. In *Types for Proofs and Programs, International Workshop, TYPES'95*, Torino, Italy, June 5–8, 1995, Berardi, S. & Coppo, M. (eds), Lectures Notes in Computer Science, vol. 1158. Springer, pp. 120–134.

Dybjer, P. (2000) A general formulation of simultaneous inductive-recursive definitions in type theory. *J. Symb. Log.* **65**(2), 525–549.

Fridlender, D. & Pagano, M. (2013) A type-checking algorithm for Martin-Löf type theory with subtyping based on normalisation by evaluation. In *Typed Lambda Calculi and Applications*, Eindhoven, The Netherlands, 26–28 June 2013, Hasegawa, M. (ed), Lecture Notes in Computer Science, vol. 7941. Berlin Heidelberg: Springer, pp. 140–155.

Geuvers, H. (1993) *Logics and Type Systems*. Ph.D. thesis, Katholieke Universiteit Nijmegen.

Geuvers, H., & Werner, B. (1994) On the Church-Rosser property for expressive type systems and its consequences for their metatheoretic study. In Proceedings Symposium on Logic in Computer Science, 1994. LICS '94, Paris, France, July 4–7, 1994. pp. 320–329.

Hardin, T. (1989) Confluence results for the pure strong categorical logic CCL: Lambda-Calculi as subsystems of CCL. *Theor. Comput. Sci.* **65**(3), 291–342.

Kesner, D. (2000) Confluence of extensional and non-extensional λ-calculi with explicit substitutions. *Theor. Comput. Sci.* **238**(1–2), 183–220.

Luo, Z. (1994) *Computation and Reasoning: A Type Theory for Computer Science*. International Series of Monographs on Computer Science. Clarendon Press.

Miquel, A. & Werner, B. (2002) The not so simple proof-irrelevant model of CC. In *Types*, Berg en Dal, The Netherlands, April 24–28, 2002, Geuvers, H. & Wiedijk, F. (eds), Lecture Notes in Computer Science, vol. 2646. Springer, pp. 240–258.

Muñoz, C. (2001) Dependent types and explicit substitutions: A meta-theoretical development. *Math. Struct. Comput. Sci.* **11**, 91–129.

Siles, V. (2010) *Investigation on the Typing of Equality in Type Systems*. Ph.D. thesis, École Polytechnique.

Siles, V. & Herbelin, H. (2010) Equality is typable in semi-full pure type systems. In Lics. Edinburgh, Scotland, UK, July 11–14, 2010, IEEE Computer Soc. Press, pp. 21–30.

Siles, V. & Herbelin, H. (2012) Pure type System conversion is always typable. *J. Funct. Program.* **22**(2), 153–180.

Streicher, T. (1989) *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions*. Ph.D. thesis, Universität Passau, Passau, West Germany.

A Proofs

Proof of Lemma 34

By well-founded induction over $\sqsubseteq$. If d is minimal, then $[d] = \mathsf{Ne}$. If d is not minimal, then either $d = s$ or $d = \mathsf{Fun}\, d'\, f$:

1. for $d = s$: since $S \subseteq \mathsf{Nf}$, we only consider $\mathsf{Fun}\, d'\, f \in [s]$ arising from some rule $(s_0, s_1, s) \in R$. By i.h. on s_0, $d' \in \mathsf{Nf}$; and by i.h. on d', $\mathsf{Ne} \subseteq [d']$; therefore $f\,(\mathsf{Var}\, i) \in [s_1]$ for every $\mathsf{Var}\, i$. By i.h. on s_1, $f\,(\mathsf{Var}\, i) \in \mathsf{Nf}$. Therefore, $\mathsf{Fun}\, d'\, f \in \mathsf{Nf}$.
2. for $d = \mathsf{Fun}\, d'\, f$: if $e \in [\mathsf{Fun}\, d'\, f]$, then either $e \in \mathsf{Ne}$ or $e = \mathsf{lam}\, g$. We consider only this last case: by i.h. on d' we know $\mathsf{Ne} \subseteq [d']$; therefore, for every $\mathsf{Var}\, i$, $g\,(\mathsf{Var}\, i) \in [f\,(\mathsf{Var}\, i)]$. By i.h. on $f\,(\mathsf{Var}\, i)$ we know $g\,(\mathsf{Var}\, i) \in \mathsf{Nf}$.

□

Proof of Lemma 35

By induction on $t \equiv_{\beta\mathsf{x}} t'$; note that it is enough to prove that the interpretation models the reduction relations. We show some cases.

1. (BETA):

$$\begin{aligned}[\![\mathsf{App}\ (\lambda t)\ t']\!]d &= [\![\lambda t]\!]d \cdot [\![t']\!]d = \mathsf{lam}\,(e \mapsto [\![t]\!](d, e)) \cdot [\![t']\!]d \\ &= [\![t]\!](d, [\![t']\!]d) = [\![t]\!]([\![\langle \mathsf{id}, t'\rangle]\!]d) = [\![t\ \langle \mathsf{id}, t'\rangle]\!]d\end{aligned}$$

2. (FUN-SUBS):

$$\begin{aligned}[\![(\mathsf{Fun}\, A\, B)\,\sigma]\!]d &= [\![\mathsf{Fun}\, A\, B]\!]([\![\sigma]\!]d) \\ &= \mathsf{Fun}\,([\![A]\!]([\![\sigma]\!]d))\,(e \mapsto [\![B]\!]([\![\sigma]\!]d, e)) \\ &= \mathsf{Fun}\,([\![A]\!]([\![\sigma]\!]d))\,(e \mapsto [\![B]\!]([\![\sigma]\!]([\![\mathsf{p}]\!](d, e)), e)) \\ &= \mathsf{Fun}\,([\![A]\!]([\![\sigma]\!]d))\,(e \mapsto [\![B]\!]([\![\sigma\ \mathsf{p}]\!](d, e), [\![\mathsf{q}]\!](d, e))) \\ &= \mathsf{Fun}\,([\![A]\!]([\![\sigma]\!]d))\,(e \mapsto [\![B]\!]([\![\langle \sigma\ \mathsf{p}, \mathsf{q}\rangle]\!](d, e)))\end{aligned}$$

□

Proof of Theorem 37

By induction on $\Gamma \vdash_{\mathsf{e}} J$; note that the last two items are proved by the inductive hypothesis and using Lemma 35.

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

1. (EMPTY-CTX): trivial.
2. (EXT-CTX) $\Gamma, A \vDash_{\mathsf{e}}$: by i.h. on $\Gamma \vDash_{\mathsf{e}} A : s$.
3. (AXIOM) $\Gamma \vDash_{\mathsf{e}} s' : s$: by i.h. $\Gamma \vDash$ and by definition of the model $s \in [\![s']\!]$.
4. (FUN-F) $\Gamma \vDash_{\mathsf{e}} \mathsf{Fun}\, A\, B : s_2$: by i.h. $\Gamma \vDash$, $\Gamma \vDash A : s_0$; also by i.h. $\Gamma, A \vDash B : s_1$. From the last two conditions we see that $\mathsf{Fun}\,([\![A]\!]d)\,(e \mapsto [\![B]\!](d,e)) \in [\![s_2]\!]$ for any $d \in [\![\Gamma]\!]$.
5. (FUN-I) $\Gamma \vDash_{\mathsf{e}} \lambda t : \mathsf{Fun}\, A\, B$: by i.h. $\Gamma \vDash$, $\Gamma \vDash A : s_0$, $\Gamma, A \vDash B : s_1$, and $\Gamma, A \vDash t : B$. The last condition implies $[\![t]\!](d,e) \in [\![B]\!](d,e)]$ for any $d \in [\![\Gamma]\!]$ and $e \in [\![A]\!]d]$; therefore $[\![\lambda t]\!]d \in [\![\mathsf{Fun}\, A\, B]\!]d]$ for any $d \in [\![\Gamma]\!]$.
6. (FUN-EL) $\Gamma \vDash_{\mathsf{e}} \mathsf{App}\ t\ r : B\,\langle \mathsf{id}, r\rangle$: by i.h. $\Gamma \vDash t : \mathsf{Fun}\, A\, B$ and $\Gamma \vDash r : A$; therefore for any $d \in [\![\Gamma]\!]$, $[\![r]\!]d \in [\![A]\!]d]$. By Rem. 33, $[\![t]\!]d \cdot [\![r]\!]d \in [\![B]\!](d, [\![r]\!]d)]$.
7. (SUB-TY) $\Gamma \vDash_{\mathsf{e}} t\,\sigma : A\,\sigma$: by i.h. $\Gamma \vDash \sigma : \Delta$. For $d \in [\![\Gamma]\!]$, then $[\![\sigma]\!]d \in [\![\Delta]\!]$; therefore by i.h. on $\Delta \vDash_{\mathsf{e}} t : A$, we have $[\![t]\!]([\![\sigma]\!]d) \in [\![A]\!]([\![\sigma]\!]d)]$.
8. (SUB-SORT) $\Gamma \vDash_{\mathsf{e}} A\,\sigma : s$: as in the previous item.
9. (HYP) $\Gamma, A \vDash_{\mathsf{e}} \mathsf{q} : A\,\mathsf{p}$: $(d,e) \in [\![\Gamma, A]\!]$ if $d \in [\![\Gamma]\!]$ and $e \in [\![A]\!]d]$; that is enough to conclude, because $[\![A\,\mathsf{p}]\!](d,e) = [\![A]\!]d$ and $[\![\mathsf{q}]\!](d,e) = e$.
10. (CONV) $\Gamma \vDash_{\mathsf{e}} t : B$: by i.h. $[\![t]\!]d \in [\![A]\!]d]$ for any $d \in [\![\Gamma]\!]$; by i.h. $[\![A]\!]d = [\![B]\!]d$, therefore $[\![t]\!]d \in [\![B]\!]d]$.
11. (EXT-SUBS) $\Gamma \vDash_{\mathsf{e}} \langle \sigma, t\rangle : \Delta, A$: for $d \in [\![\Gamma]\!]$, by i.h. $[\![\sigma]\!]d \in [\![\Gamma]\!]$ and $[\![t]\!]d \in [\![A\,\sigma]\!]d]$.
12. (COMP-SUBS) $\Gamma \vDash_{\mathsf{e}} \sigma\,\delta : \Delta$: using the same reasoning as for (SUB-TY).
13. (FST-SUBS) $\Gamma, A \vDash_{\mathsf{e}} \mathsf{p} : \Gamma$: using the same reasoning as for (HYP).

□

Proof of Lemma 42
By induction on $X \in T$.

1. $X \in \mathsf{Ne}$: the first condition is obtained directly from the main hypothesis it depends on the type. For the second condition, let $\Delta \leqslant^i \Gamma$:
$$\text{(TRANS)}\ \frac{\text{(SYM)}\ \dfrac{\text{(SUB-EQ-TY)}\ \dfrac{\Delta \vDash_{\mathsf{e}} \mathsf{p}^i = \mathsf{p}^i : \Gamma \qquad \Gamma \vDash_{\mathsf{e}} A : s \qquad \Gamma \vDash_{\mathsf{e}} t = t' : A}{\Delta \vDash_{\mathsf{e}} t\,\mathsf{p}^i = t'\,\mathsf{p}^i : A\,\mathsf{p}^i}}{\Delta \vDash_{\mathsf{e}} t'\,\mathsf{p}^i = t\,\mathsf{p}^i : A\,\mathsf{p}^i} \qquad \Delta \vDash_{\mathsf{e}} t\,\mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : A\,\mathsf{p}^i}{\Delta \vDash_{\mathsf{e}} t'\,\mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : A\,\mathsf{p}^i}$$
2. $X = s \in S$ and $d \in \mathsf{Ne}$: if $A = s$, then
$$\text{(TRANS)}\ \frac{\text{(SYM)}\ \dfrac{\text{(SUB-EQ-SORT)}\ \dfrac{\Delta \vDash_{\mathsf{e}} \mathsf{p}^i = \mathsf{p}^i : \Gamma \qquad \Gamma \vDash_{\mathsf{e}} t = t' : s}{\Delta \vDash_{\mathsf{e}} t\,\mathsf{p}^i = t'\,\mathsf{p}^i : s}}{\Delta \vDash_{\mathsf{e}} t'\,\mathsf{p}^i = t\,\mathsf{p}^i : s} \qquad \Delta \vDash_{\mathsf{e}} t\,\mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : s}{\Delta \vDash_{\mathsf{e}} t'\,\mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : s}$$
If there exists $s' \in S$ and for all $\Delta \leqslant^i \Gamma$, $\Delta \vDash_{\mathsf{e}} A\,\mathsf{p}^i = s : s'$, then we know, by validity of equality, $\Gamma \vDash_{\mathsf{e}} A : s'$, therefore
$$\frac{\dfrac{\dfrac{\dfrac{\Delta \vDash_{\mathsf{e}} \mathsf{p}^i = \mathsf{p}^i : \Gamma \qquad \Gamma \vDash_{\mathsf{e}} A : s' \qquad \Gamma \vDash_{\mathsf{e}} t = t' : A}{\Delta \vDash_{\mathsf{e}} t\,\mathsf{p}^i = t'\,\mathsf{p}^i : A\,\mathsf{p}^i}}{\Delta \vDash_{\mathsf{e}} t'\,\mathsf{p}^i = t\,\mathsf{p}^i : A\,\mathsf{p}^i} \qquad \Delta \vDash_{\mathsf{e}} A\,\mathsf{p}^i = s : s'}{\Delta \vDash_{\mathsf{e}} t'\,\mathsf{p}^i = t\,\mathsf{p}^i : s} \qquad \Delta \vDash_{\mathsf{e}} t\,\mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : s}{\Delta \vDash_{\mathsf{e}} t'\,\mathsf{p}^i = \mathsf{R}_{|\Delta|}\, d : s}$$

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

3. $X = s \in S$ and $d = \mathsf{Fun}\, X'\, F$: from the main hypothesis, we get $s_0, s_1 \in S$ and $(s_0, s_1, s) \in R$; moreover we know there are $A', B \in \mathit{Term}$ such that $\Gamma \vdash_{\mathsf{e}} t = \mathsf{Fun}\, A'\, B : A$; hence our witnesses are also A' and B:
$$\dfrac{\dfrac{\Gamma \vdash_{\mathsf{e}} t = t' : A}{\Gamma \vdash_{\mathsf{e}} t' = t : A} \qquad \Gamma \vdash_{\mathsf{e}} t = \mathsf{Fun}\, A'\, B : A}{\Gamma \vdash_{\mathsf{e}} t' = \mathsf{Fun}\, A'\, B : A}$$
Since A' and B satisfy the third and four items we are done.
4. $X = \mathsf{Fun}\, X'\, F$: from the main hypothesis we get $s_0, s_1, s \in S$ such that $(s_0, s_1, s) \in R$ and $A', B \in \mathit{Term}$, such that $\Gamma \vdash_{\mathsf{e}} A = \mathsf{Fun}\, A'\, B : s$. For the third point, we use the inductive hypothesis with $\Delta \vdash_{\mathsf{e}} \mathsf{App}\ (t\, \mathsf{p}^i)\ r : B\, \langle \mathsf{p}^i, r\rangle \bowtie d \cdot e \in [F\, e]$ and $\Delta \vdash_{\mathsf{e}} \mathsf{App}\ (t\, \mathsf{p}^i)\ r = \mathsf{App}\ (t'\, \mathsf{p}^i)\ r : B\, \langle \mathsf{p}^i, r\rangle$.
The last two items are immediate using (SYM) and (TRANS).

□

Proof of Lemma 43
By induction on $X \in T$.

1. $X \in \mathsf{Ne}$: from the main hypothesis, we know there exists $s \in S$ such that $\Gamma \vdash_{\mathsf{e}} A : s$ and for all $\Theta \leqslant^k \Gamma$, $\Theta \vdash_{\mathsf{e}} A\, \mathsf{p}^k = \mathsf{R}_{|\Theta|}\, X : s$. Let $\Sigma \leqslant^j \Delta$, then by Lemma 15 $\Sigma \leqslant^{j+i} \Gamma$, thus $\Sigma \vdash_{\mathsf{e}} A\, \mathsf{p}^{j+i} = \mathsf{R}_{|\Sigma|}\, X : s$, from which we can conclude $\Sigma \vdash_{\mathsf{e}} (A\, \mathsf{p}^i)\, \mathsf{p}^j = \mathsf{R}_{|\Sigma|}\, X : s$. The second part follows the same reasoning.
2. $X = s$ and $d \in \mathsf{Ne} \cup S$: As the previous point.
3. $X = S$ and $d = \mathsf{Fun}\, X'\, F$:
4. $X = \mathsf{Fun}\, X'\, F$: From the definition of the logical relations, we know that there exist $(s_1, s_2, s_3) \in R$ and $C, D \in \mathit{Term}$ such that $\Gamma \vdash_{\mathsf{e}} A = \mathsf{Fun}\, C\, D : s_3$. We get by congruence $\Delta \vdash_{\mathsf{e}} A\, \mathsf{p}^i = \mathsf{Fun}\, (B\, \mathsf{p}^i)\, (C\, \langle \mathsf{p}^{i+1}, \mathsf{q}\rangle) : s_3$, and by i.h. $\Delta \vdash_{\mathsf{e}} B\, \mathsf{p}^i : s_1 \bowtie X' \in [s_1]$. Let $\Theta \leqslant^j \Delta$ and $\Theta \vdash_{\mathsf{e}} r : (B\, \mathsf{p}^i)\, \mathsf{p}^j \bowtie e \in [X']$; by Lemma 42 we have $\Theta \vdash_{\mathsf{e}} r : B\, \mathsf{p}^{i+j} \bowtie e \in [X']$; thus by def. $\Theta \vdash_{\mathsf{e}} \mathsf{App}\ (t\, \mathsf{p}^{i+j})\ r : C\, \langle \mathsf{p}^{i+j}, r\rangle \bowtie d \cdot e \in [F\, e]$; and using Lemma 42 again we conclude $\Theta \vdash_{\mathsf{e}} \mathsf{App}\ ((t\, \mathsf{p}^j)\, \mathsf{p}^j)\ r : C\, \langle \mathsf{p}^i\, \mathsf{p}, \mathsf{q}\rangle\, \langle \mathsf{p}^j, r\rangle \bowtie d \cdot e \in [F\, e]$. Finally, we do case analysis on $d \in [\mathsf{Fun}\, X'\, F]$. If $d = \mathsf{lam} f$, then $\Gamma \vdash_{\mathsf{e}} t = \lambda t' : A$, hence by congruence $\Theta \vdash_{\mathsf{e}} t = \lambda(t'\, \langle \mathsf{p}^i\, \mathsf{p}, \mathsf{q}\rangle) : A$. If $d \in \mathsf{Ne}$, then we have $\Theta \vdash_{\mathsf{e}} (t\, \mathsf{p}^i)\, \mathsf{p}^j = t\, \mathsf{p}^{i+j} : (A\, \mathsf{p}^i)\, \mathsf{p}^j$, thus $\Theta \vdash_{\mathsf{e}} (t\, \mathsf{p}^i)\, \mathsf{p}^j = \mathsf{R}_{|\Theta|}\, d : (A\, \mathsf{p}^i)\, \mathsf{p}^j$, from def. of logical relations.

□

Proof of Lemma 44
By induction on $X \in T$. The case for $X \in \mathsf{Ne}$ is trivial.

1. $X = s \in S$: by induction on $d \in [s]$: The first part for $d \in \mathsf{Ne}$ and $d \in S$ are obtained directly by definition. Moreover, the second part is trivial, because the hypothesis is the main condition in the definition of the logical relation.
We show only the first part for $d = \mathsf{Fun}\, X'\, F$: by i.h. on $\Gamma \vdash_{\mathsf{e}} A : s_0 \bowtie X' \in [s_0]$ we have $\Delta \vdash_{\mathsf{e}} A\, \mathsf{p}^i = \mathsf{R}_{|\Delta|}\, X' : s_0$. By i.h. on $\Delta \vdash_{\mathsf{e}} B\, \langle \mathsf{p}^i, r\rangle : s_1 \bowtie F\, e \in [s_1]$. On the other hand, we have $\Delta' \vdash_{\mathsf{e}} \mathsf{q}\, \mathsf{p}^j = \mathsf{R}_{|\Delta|}\, (\mathsf{Var}\, |\Gamma|) : A\, \mathsf{p}^j$, for all $\Delta' \leqslant^j \Gamma, A$, so by i.h. on the second part, we know $\Gamma, A \vdash_{\mathsf{e}} \mathsf{q} : A\, \mathsf{p} \bowtie \mathsf{Var}\, |\Gamma| \in [X']$. From which

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press

we conclude $\Gamma, A \vDash_{\mathsf{e}} B\ \langle \mathsf{p}, \mathsf{q}\rangle = \mathsf{R}_{|\Delta|}\ (F\ (\mathsf{Var}\ |\Gamma|)) : s_1$. Using (PROD-EQ) we conclude the equality.

2. $X = \mathsf{Fun}\ X'\ F$: For the first part, we only show the case when $d = \mathsf{lam}\ f$.
   (a) If $d = \mathsf{lam}\ f$: by definition of the logical relation $\Gamma \vDash_{\mathsf{e}} t = \lambda t' : A$. As was shown in the previous point, we have $\Gamma, A' \vDash_{\mathsf{e}} \mathsf{q} : A'\ \mathsf{p} \bowtie \mathsf{Var}\ |\Gamma| \in [X']$; by definition of the logical relation and Lemma 42 we know $\Gamma, A' \vDash_{\mathsf{e}} : B\ \langle \mathsf{p}, \mathsf{q}\rangle \bowtie \mathsf{App}\ ((\lambda t')\ \mathsf{p})\ \mathsf{q} \in [\mathsf{lam}\ f \cdot (\mathsf{Var}\ |\Gamma|)]F\ (\mathsf{Var}\ |\Gamma|)$. By Lemma 42 and by definition of application $\Gamma, A' \vDash_{\mathsf{e}} t' : B \bowtie f\ (\mathsf{Var}\ |\Gamma|) \in [F\ (\mathsf{Var}\ |\Gamma|)]$. Now we can apply the i.h. to obtain $\Gamma, A' \vDash_{\mathsf{e}} t' = \mathsf{R}_{|\Gamma|+1}\ (f\ (\mathsf{Var}\ |\Gamma|)) : B$. Since $\mathsf{R}_{|\Gamma|}\ (\mathsf{lam}\ f) = \lambda(\mathsf{R}_{|\Gamma|+1}\ (f\ (\mathsf{Var}\ |\Gamma|)))$, by (CONG-ABS) we deduce $\Gamma \vDash_{\mathsf{e}} \lambda t' = \mathsf{R}_{|\Gamma|}\ (\mathsf{lam}\ f) : \mathsf{Fun}\ A'\ B$.
   (b) For the second point, we only need to prove $\Delta \vDash_{\mathsf{e}} \mathsf{App}\ \ t\ \ r : B\ \langle \mathsf{p}^i, r\rangle \bowtie k \cdot d' \in [F\ d']$, for any $\Delta \leqslant^i \Gamma$ and $\Delta \vDash_{\mathsf{e}} r : A'\ \mathsf{p}^i \bowtie d' \in [X']$. By Lemma 34, $d' \in \mathsf{Nf}$, therefore $k \cdot d' = \mathsf{App}\ k\ d' \in \mathsf{Ne}$. We prove that by i.h. on the second part. By i.h. we know $\Delta' \vDash_{\mathsf{e}} r\ \mathsf{p}^j = \mathsf{R}_{|\Delta'|}\ d' : A'\ \mathsf{p}^{i+j}$, for any $\Delta' \leqslant^j \Delta$ and also, from the main hypothesis and (CONV-EQ), $\Delta' \vDash_{\mathsf{e}} t\ \mathsf{p}^{i+j} = \mathsf{R}_{|\Delta'|}\ k : \mathsf{Fun}\ (A'\ \mathsf{p}^{i+j})\ (B\ \langle \mathsf{p}^{(i+j)+1}, \mathsf{q}\rangle)$. By definition of reification we have $\mathsf{R}_{|\Delta'|}\ (\mathsf{App}\ k\ d') = \mathsf{App}\ (\mathsf{R}_{|\Delta'|}\ k)\ (\mathsf{R}_{|\Delta'|}\ d')$, therefore we use (APP-CONG) to deduce $\Delta' \vDash_{\mathsf{e}} \mathsf{App}\ \ (t\ \mathsf{p}^{i+j})\ \ (r\ \mathsf{p}^j) = \mathsf{R}_{|\Delta'|}\ (\mathsf{App}\ k\ d') : B\ \langle \mathsf{p}^{i+j}, r\ \mathsf{p}^j\rangle$.

□

Proof of Lemma 46

By induction on $\Delta \vDash_{\mathsf{e}}$. Both points are trivial for $\diamond$.

1. Preservation of logical relations by judgemental equality for Δ, A is proved by applying the i.h. on $\Gamma \vDash_{\mathsf{e}} \mathsf{p}\ \sigma = \mathsf{p}\ \sigma' : \Delta$ and $\Gamma \vDash_{\mathsf{e}} \mathsf{p}\ \sigma : \Delta \bowtie d \in [\![\Delta]\!]$. The second part is obtained by using Lemma 42 on $\Gamma \vDash_{\mathsf{e}} \mathsf{q}\ \sigma = \mathsf{q}\ \sigma' : A\ (\mathsf{p}\ \delta)$ and $\Gamma \vDash_{\mathsf{e}} \mathsf{q}\ \delta : A\ (\mathsf{p}\ \delta) \bowtie d' \in [[\![A]\!]d]$.
2. Monotonicity of logical relations for Δ, A is obtained using the i.h. and Lemma 43.

□

Proof of Lemma 49

By induction on $\Gamma \vDash_{\mathsf{e}}$. The base case is trivial. For the inductive case, $\Gamma, A \vDash_{\mathsf{e}}$, we have by i.h. $\Gamma \vDash_{\mathsf{e}} \mathsf{id} : \Gamma \bowtie \rho_\Gamma \in [\![\Gamma]\!]$. By both parts of Lemma 46 $\Gamma, A \vDash_{\mathsf{e}} \mathsf{p}\ \mathsf{id} : \Gamma \bowtie \rho_\Gamma \in [\![\Gamma]\!]$. Let $n = |\Gamma|$, then $\Delta \vDash_{\mathsf{e}} \mathsf{q}\ \mathsf{p}^i = \mathsf{R}_{|\Delta|}\ (\mathsf{Var}\ n) : A\ \mathsf{p}^i$, for all $\Delta \leqslant^i \Gamma, A$; so, by Lemma 44, we have $\Gamma, A \vDash_{\mathsf{e}} \mathsf{q} : A\ \mathsf{p} \bowtie \mathsf{Var}\ n \in [[\![A]\!]\rho_\Gamma]$ and by Lemma 42 $\Gamma, A \vDash_{\mathsf{e}} \mathsf{q}\ \mathsf{id} : A\ (\mathsf{p}\ \mathsf{id}) \bowtie \mathsf{Var}\ n \in [[\![A]\!]\rho_\Gamma]$. So, we conclude $\Gamma, A \vDash_{\mathsf{e}} \mathsf{id} : \Gamma, A \bowtie (\rho_\Gamma, \mathsf{Var}\ n) \in [\![\Gamma, A]\!]$. □

https://doi.org/10.1017/S0956796815000210 Published online by Cambridge University Press