JFP **16** (3): 327–372, 2006. © 2006 Cambridge University Press
doi:10.1017/S0956796806005892 Printed in the United Kingdom

Consistency of the theory of contexts

ANNA BUCALO, FURIO HONSELL, MARINO MICULAN, IVAN SCAGNETTO

Department of Mathematics and Computer Science, University of Udine, Italy
email: miculan@dimi.uniud.it

MARTIN HOFFMAN

Institut für Informatik, Ludwig-Maximilians-Universität, München, Germany

Abstract

The Theory of Contexts is a type-theoretic axiomatization aiming to give a metalogical account of the fundamental notions of *variable* and *context* as they appear in Higher Order Abstract Syntax. In this paper, we prove that this theory is consistent by building a model based on *functor categories*. By means of a suitable notion of *forcing*, we prove that this model validates Classical Higher Order Logic, the Theory of Contexts, and also (parametrised) structural *induction* and *recursion principles* over contexts. Our approach, which we present in full detail, should also be useful for reasoning on other models based on functor categories. Moreover, the construction could also be adopted, and possibly generalized, for validating other theories of names and binders.

1 Introduction

In recent years there has been growing interest in developing systems for defining, programming with and reasoning about, languages with *variable binding* constructors (*binders*). Dealing with these syntactical structures requires context-sensitive notions such as scope, free and bound variables, α-conversion, generation of fresh names, capture-avoiding susbstitution, and so on. It is well-known that traditional representations of languages with binders, such as those based on first-order abstract syntax or de Bruijn indexes, are not satisfactory because of the lack of a general and smooth account for these notions.

An approach to this issue, originated with Church and widely adopted in Logical Frameworks and proof assistants, is that of *Higher-Order Abstract Syntax* (HOAS) (Church, 1940; Harper *et al.*, 1993; Pfenning & Elliott, 1988; Miculan, 1997). The key idea is to represent variables and binders of object languages with variables and binders of a metalanguage based on some type-theoretic λ-calculus. In this way, we shift the treatment of variables and binders to the metalanguage, where where we can establish all the required properties and notions once and for all.

However, this approach has some drawbacks. First of all, being equated to metalanguage variables (i.e., *metavariables*), object level variables cannot be defined inductively without introducing *exotic terms*, i.e. metalevel terms which do not correspond to any object level term (Despeyroux *et al.*, 1995; Miculan, 1997). A

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

similar difficulty arises with term contexts, *i.e.* terms with holes, which are rendered as functional terms. Reasoning by induction and definition by recursion on object level terms is therefore problematic.

To address these problems, some of the authors proposed an *axiomatic* strategy. First, a small set of basic properties, aiming to characterize the "natural" behaviour of term contexts and names as intended in weak HOAS encodings, has been identified and proposed (Honsell *et al.*, 2001b) and generalized later (Honsell *et al.*, 2001a). These properties, called the *Theory of Contexts*, can be assumed (as "axioms") in existing logical frameworks and metalanguages based on type theories corresponding to (Intuitionistic) Higher Order Logics (such as the Calculus of Constructions, or $CC^{(Co)Ind}$), for gaining the extra expressive power required for reasoning about formalizations using weak HOAS. The Theory of Contexts has been shown to be practically useful and well-suited also for large, non trivial languages and systems: successful case studies include, but are not limited to, the metatheory of strong late bisimilarity of the π-calculus (Honsell *et al.*, 2001b), the metatheory of the λ-calculus, (Miculan, 2001), the metatheory of Mobile Ambients and its logic (Scagnetto & Miculan, 2002).

We have still to accomplish the second part of the "axiomatic" strategy: we have to prove the *consistency* of these axioms. This is the main aim of the present work.

More precisely, we present a basic logical framework, composed by a type theory *à la* Church and a classical Higher Order Logic extended with the Theory of Contexts. This simple yet expressive metalanguage can be used for representing a broad class of object languages and logics with binders; as an example, we show how to encode (a subset of) the π-calculus. Then, we present a sound model for this type theory and classical logic, which validates also the axioms of the Theory of Contexts. Moreover, we prove that suitable structural *induction* and *recursion principles* over terms with binders and even term contexts are validated by this model.

To achieve these results, we have to resort to rather sophisticated mathematical tools. Categorical models of languages with binders and local names have been around for a while. In most cases, they are based on *(covariant) (pre)sheaf categories*, i.e. *functors* of the form $\mathscr{C} \to Set$, for some suitable index category $\mathscr{C}$. Types are then interpreted as "stratified sets", indexed by the objects of $\mathscr{C}$ which are finite sets (of variables/names). The morphisms in $\mathscr{C}$ express the kind of "replacement law" we are requiring the terms to satisfy: if we are interested in modeling languages with variables (which can be unified), then morphisms are all (finite) functions (Fiore *et al.*, 1999; Crole, 2003); on the other hand, if we are interested in *names* (which cannot be unified), the morphisms are only injective functions (Moggi, 1993; Stark, 1994; Fiore & Turi, 2001).

However, using a functor category is not enough for modeling the Theory of Contexts. In fact, a peculiarity of the Theory of Contexts is that it contradicts the Axiom of Unique Choice (Honsell *et al.*, 2001b; Honsell *et al.*, 2001a). Since the Axiom of Unique Choice holds in *all* topoi, this means that *no* (pre)sheaf category alone can be used for building such a model.

The problem is that to model a metalanguage embodying the Theory of Contexts we must address at once two related, but different, aspects. The first one is the interpretation of terms of the metalanguage itself, whose *variables*, like for any

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

λ-calculus, may be subject to substitution and unification. The other is that variables ranging over the specific type of "names" are given a particular meaning by the axioms of the Theory of Contexts. Therefore, the properties of such variables must be obtained from some model of *names*, which cannot be unified.

The solution to this problem has been devised in Hofmann (1999), by "gluing" together a model of variables and a model of names. Essentially, we need to deal with *two* presheaf categories, over two different index categories. Datatypes of the metalanguage are interpreted as presheaves over the category of all variable substitutions, while predicates are interpreted as subsheaves in the *Schanuel topos*, which is a sheaf category of presheaves over *injective* variable substitutions only. The right technical notion for best describing this construction is that of *tripos*, a useful generalization of topoi (Hyland *et al.*, 1980; Pitts, 1999). However, our aim is to describe this construction in elementary terms, in order to make it accessible also to the reader with little knowledge of category theory.

In fact, this paper has a *pedagogical* purpose. It illustrates through a concrete example the novelty of the approach in Hofmann (1999) for modeling a logic for reasoning on systems in HOAS, using a tripos-like construction for interpreting predicates. The construction is described in full detail, without resorting to advanced categorical theoretic notions.

This paper has also the *technical* purpose of working out the details of the category-theoretic ideas and constructions outlined in §7. One of the crucial tools that we introduce to this end is a notion of *forcing* which allows us to streamline the computation of the truth value of a proposition. This methodology can be useful also for validating other theories of names and binders, and reasoning about other models based on functor categories.

Finally, this paper has also the *speculative* purpose of discussing, besides consistency, also independence and completeness of our axiomatization for variables and contexts in HOAS.

Synopsis

In section 2 we present Υ, a simple logical framework geared toward encodings in weak HOAS (and containing the Theory of Contexts).

In section 3 we define from scratch the model $\mathscr{U}$ and the interpretation of the term language and of predicates of Υ. In section 4 we will verify that this model validates the (classical) higher order logic, and the Theory of Contexts, thus proving its soundness. In section 5 we will clarify the connection of this model with tripos theory; in fact, the reader familiar with tripos theory is encouraged to read section 5 in parallel to (or even *before*) section 3.

Then, in section 6 we extend Υ with *recursion* and *induction* principles, possibly also over higher-order terms (i.e., term contexts).

Finally, a comparison of our approach with similar works in the literature is given in section 7. Concluding remarks and directions for future work are in section 8.

Some category-theoretic preliminaries, together with longer proofs, are gathered in an appendix available online at the JFP web site (Bucalo *et al.*, 2005).

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

2 The logical framework Υ

In this section we present Υ, a simple logical framework geared toward encodings in weak Higher Order Abstract Syntax.

Υ consists of a simply typed λ-calculus *a la* Church, suited for representing a broad class of object languages, and a Classical Higher Order Logic extended with the Theory of Contexts. This theory is "parametric" in the specific signature we are focusing on; thus, in order to exemplify the framework and the encoding methodology, we fix a simple object language, namely a fragment of the π-calculus. For a more general presentation we refer to Honsell *et al.* (2001a). Finally, we discuss some design choices and issues of the Theory of Contexts.

2.1 Theory of terms

The metalanguage Υ is a theory of Simple Types/Higher Order Logic *à la* Church over a particular signature encoding an object language. Many details of the underlying type theory are not strictly intrinsic. The machinery that we define in this section could have been based on any sufficiently expressive type theory, e.g. the Calculus of Inductive Constructions (INRIA, 2003). We picked Church's Simple Theory of Types only for simplicity. Thus, types, ranged over by σ, τ (possibly with indices or apices), are defined by the following abstract syntax:

$$\sigma ::= o \mid \iota \mid \upsilon \mid \sigma \to \tau$$

For each type there is a countably infinite disjoint set of variables $x, y, z \ldots$ Since these are variables of the metalanguage, we sometimes call them *metavariables*.

An object system is defined by a *signature* Σ, which is a finite collection of *constant symbols* together with their type. Given a signature Σ, the *(pre)terms* over Σ are defined by the following syntax:

$$\begin{aligned} M ::= {} & x \mid MN \mid \lambda x^{\sigma}.M \mid M \Rightarrow N \mid \forall_{\sigma}.M \\ & \mid c(M_1, \ldots, M_n) \qquad \text{where } c{:}\sigma_1 \to \ldots \sigma_n \to \sigma \in \Sigma \end{aligned}$$

As usual, we denote by $M[N/x]$ the capture-avoiding substitution. Terms are identified up-to α-conversion and are ranged over by M, N, P, Q, R (possibly with indices). In the case of terms of type o, we also use $p, q, r \ldots$.

Terms of types ι, o are intended to denote object-level *terms* and *propositions* of any given system. Terms of type υ represent variables and names of the object system, the exact behaviour being enforced by suitable logical assumptions, as we will see. We require signatures to do not have constructors of type υ; hence the only terms (in normal form) inhabiting υ can be metavariables.

In the following, we restrict our attention to *well-typed terms* only. An *environment* Γ is a finite set of typing assertions over distinct variables of Υ, denoted by $\{x_1 : \sigma_1, x_2 : \sigma_2, \ldots, x_n : \sigma_n\}$, possibly without curly brackets. Its *domain* is the set $\{x_1, \ldots, x_n\}$. Then, as usual, the *typing judgment* is of the form

$$\Gamma \vdash_{\Sigma} M : \sigma$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

$$\frac{}{\Gamma, x:\sigma \vdash_\Sigma x:\sigma}\ (\text{VAR}) \qquad \frac{\Gamma, x:\sigma' \vdash_\Sigma M:\sigma}{\Gamma \vdash_\Sigma \lambda x^{\sigma'}.M:\sigma' \to \sigma}\ (\text{ABS}) \qquad \frac{\Gamma \vdash_\Sigma M:\sigma' \to \sigma \quad \Gamma \vdash_\Sigma N:\sigma'}{\Gamma \vdash_\Sigma MN:\sigma}\ (\text{APP})$$

$$\frac{\Gamma \vdash_\Sigma M:\sigma \to o}{\Gamma \vdash_\Sigma \forall_\sigma.M:o}\ (\forall) \qquad \frac{\Gamma \vdash_\Sigma M:o \quad \Gamma \vdash_\Sigma N:o}{\Gamma \vdash_\Sigma M \Rightarrow N:o}\ (\Rightarrow)$$

$$\frac{\Gamma \vdash_\Sigma M_1:\sigma_1 \quad \ldots \quad \Gamma \vdash_\Sigma M_n:\sigma_n}{\Gamma \vdash_\Sigma c(M_1,\ldots,M_n):\sigma}(c:\sigma_1 \to \cdots \to \sigma_n \to \sigma) \in \Sigma\ (\text{CONST})$$

Fig. 1. Typing rules.

expressing the fact that M is a term of type σ, starting from environment Γ and using the typed constants in signature Σ. The rules for typing are listed in Figure 1.

The term language given so far allows for an adequate encoding of a broad class of object languages, following the second-order encoding procedure of the Edinburgh Logical Framework (Harper *et al.*, 1993; Miculan, 1997). As a simple example object system, here we consider a fragment of π-calculus (Milner *et al.*, 1992); a complete and general treatment can be found in Honsell *et al.* (2001a).

The object system: syntax: In the π-calculus there are two basic syntactical entities:

- *Names:* the set $\mathscr{N}$ is an infinite set of names;
- *Processes:* the set $\mathscr{P}$, ranged over by P, Q, is defined by the following abstract syntax, where the operators are listed in decreasing order of precedence:

$$P ::= 0 \mid \tau.P \mid P_1|P_2 \mid [x \neq y]P \mid (\nu x)P$$

It is worthwhile to recall here the difference between (object-level) *names* and (metalanguage) *variables*. A name is an atomic data structures, distinguished from all other names. On the other hand, a variable of the metalanguage acts as a placeholder: it can be instantiated by any term of the type it ranges over.

In the case of π-calculus, a variable of type υ (i.e., ranging over names) can be bound by the *restriction* operator ν. Processes are taken up to α-equivalence, and variables bound by ν's are intended to denote a fresh (i.e. local) name. Since we can always choose different variables for bound names, we can assume that different variables of type υ are intended to denote different names.

For each process P we can define in the standard way the sets of its *free names* $fn(P)$ as the set of free variables of type υ. Let X be a finite set of (variables denoting) names; then, $\mathscr{P}_X$ denotes the set $\{P \subset \mathscr{P} \mid fn(P) \subseteq X\}$. Capture-avoiding substitution of a single name y in place of x in P is denoted by $P[y/x]$. A *(process) context* is a process with a (possibly repeated) hole for a name.

This fragment has been chosen by striving for simplicity in order to highlight the problematic issues of reasoning about names in higher-order abstract syntax. It lacks the computational expressivity of the original system, since it features neither synchronization nor mobility of processes; still, the presence of the τ prefix leads to a a non-trivial theory of strong bisimulation. For a formalization of the full π-calculus, see Honsell *et al.* (2001b).

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

The above language can be encoded in Υ by the following signature Σ:

$$0 : \iota \qquad \tau : \iota \to \iota \qquad | : \iota \to \iota \to \iota$$
$$[\cdot \neq \cdot]\cdot : \upsilon \to \upsilon \to \iota \to \iota \qquad \nu : (\upsilon \to \iota) \to \iota$$

Let $X = \{x_1, \ldots, x_n\}$ a finite set of variables, and let us consider the object language terms $\mathscr{P}_X$. The corresponding terms of the metalanguage are the terms in long $\beta\eta$-normal form defined as follows, using infix notation:

$$x ::= x_1 \mid \ldots \mid x_n \qquad P ::= 0 \mid \tau P \mid P|Q \mid [x_1 \neq x_2]P \mid \nu\lambda y^{\upsilon}.P$$

We denote the set of such normal forms by $Proc_X$.

Proposition 2.1
There is a bijective correspondence between $\mathscr{P}_X$ and the normal forms of type ι in the signature Σ and in the environment $\Gamma_X \triangleq \{x_1 : \upsilon, \ldots, x_n : \upsilon\}$.

The proof (omitted) follows a standard induction on the syntax of terms and on the derivation of the typing judgment (Harper *et al.*, 1993; Miculan, 1997).

2.2 Theory of predicates

The main aim of Υ is to provide a formal setting for reasoning on the properties of the object language, especially those involving names, variables and binders. To this end, beside the theory of terms, Υ has to support also a sufficiently expressive *theory of predicates*. In fact, terms inhabiting the basic type o are supposed to represent *propositions* stating facts about the (terms representing the) object languages.

Since the main aim of this paper is to prove the consistency of the Theory of Contexts, whereas proof theory is not an issue, we choose to use a Hilbert-style deductive logical system, because this style is simpler to deal with for model-theoretical investigations. Of course, Natural-Deduction and sequent presentations are possible as well (Honsell *et al.*, 2001a). Thus, the *logical judgment* is of the form

$$\Gamma \rhd_\Sigma p \qquad \text{well-formed when } \Gamma \vdash_\Sigma p : o$$

to express the fact that a proposition p involving free variables from Γ is valid.

The rules for this judgment are given in two parts. The first part provides the general *logical* axioms and rules for *classical* higher-order logic; the second part consists of the three axioms of the Theory of Contexts. Only the latter depends on the particular object system (i.e., signature) we are interested in. (A third part, providing recursion and induction principles, will be considered in section 6).

The logical rules are given in Figure 3; we note that permutation, weakening, and contraction are admissible rules. In Figure 2 we define some other logical connectives used in the following, using the usual higher-order encodings in terms of $\forall_\sigma$ and $\Rightarrow$.

Before introducing the second part of the system, we need to define the auxiliary predicate of *occurrence-checking* of variables in (terms representing) object-level terms. Intuitively, this can be defined by induction on the structure of terms; in fact, we can define $\notin : \upsilon \to \iota \to o$ by means of an impredicative inductive definition for

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

$$\forall x^{\sigma}.p \triangleq \forall_{\sigma}(\lambda x^{\sigma}.p) \qquad p \wedge q \triangleq \neg(p \Rightarrow \neg q)$$
$$\bot \triangleq \forall x^{o}.x \qquad p \vee q \triangleq \neg p \Rightarrow q$$
$$\neg p \triangleq p \Rightarrow \bot \qquad p \Leftrightarrow q \triangleq (p \Rightarrow q) \wedge (q \Rightarrow p)$$
$$\exists x^{\sigma}.p \triangleq \neg\forall x^{\sigma}.\neg p \qquad M =^{\sigma} N \triangleq \forall x^{\sigma\to o}.\ xM \Rightarrow xN$$

Fig. 2. Syntactic abbreviations.

$$\frac{\Gamma \vdash_{\Sigma} p : o \quad \Gamma \vdash_{\Sigma} q : o \quad \Gamma \vdash_{\Sigma} r : o}{\Gamma \rhd_{\Sigma} (p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow q) \Rightarrow p \Rightarrow r} \ (\mathrm{S}) \qquad \frac{\Gamma, x : \sigma \vdash_{\Sigma} M : \sigma' \quad \Gamma \vdash_{\Sigma} N : \sigma}{\Gamma \rhd_{\Sigma} (\lambda x^{\sigma}.M)N =^{\sigma'} M[N/x]} \ (\beta)$$

$$\frac{\Gamma \vdash_{\Sigma} p : o \quad \Gamma \vdash_{\Sigma} q : o}{\Gamma \rhd_{\Sigma} p \Rightarrow q \Rightarrow p} \ (\mathrm{K}) \qquad \frac{\Gamma \vdash_{\Sigma} M : \sigma \to \sigma'}{\Gamma \rhd_{\Sigma} \lambda x^{\sigma}.Mx =^{\sigma\to\sigma'} M} x \notin FV(M) \ (\eta)$$

$$\frac{\Gamma \vdash_{\Sigma} P : \sigma \to o \quad \Gamma \vdash_{\Sigma} M : \sigma}{\Gamma \rhd_{\Sigma} \forall_{\sigma}(P) \Rightarrow PM} \ (\forall\text{-E}) \qquad \frac{\Gamma, x : \sigma \vdash_{\Sigma} M : \sigma' \quad \Gamma, x : \sigma \vdash_{\Sigma} N : \sigma'}{\Gamma \rhd_{\Sigma} (\forall x^{\sigma}.M =^{\sigma'} N) \Rightarrow \lambda x^{\sigma}.M =^{\sigma\to\sigma'} \lambda x^{\sigma}.N} \ (\xi)$$

$$\frac{\Gamma \vdash_{\Sigma} p : o}{\Gamma \rhd_{\Sigma} \neg\neg p \Rightarrow p} \ (\mathrm{DN}) \qquad \frac{\Gamma \vdash_{\Sigma} p : o \quad \Gamma, x : \sigma \rhd_{\Sigma} p \Rightarrow q}{\Gamma \rhd_{\Sigma} p \Rightarrow \forall x^{\sigma}.q} \ (\mathrm{Gen})$$

$$\frac{\Gamma \vdash_{\Sigma} p \Rightarrow q \quad \Gamma \vdash_{\Sigma} p}{\Gamma \rhd_{\Sigma} q} \ (\mathrm{MP})$$

Fig. 3. Logical axioms and rules.

a suitable "closure" operator $T_{\notin}$ (Figure 4). The definition of the operator $T_{\notin}$ is completely syntax driven, after the signature we have chosen.

The second part of the logical system of Υ is given by the axioms in Figure 5, which represent the *Theory of Contexts* (for the given signature Σ). These properties reflect in the metalogic some core properties of names and higher-order terms over names. In section 2.3.1 we illustrate how to apply the axioms of the Theory of Contexts in order to derive some fundamental metatheoretic properties about process algebras. According to our experience (Honsell *et al.*, 2001b; Scagnetto, 2002), these axioms are very useful for proving metatheoretic properties about encodings in higher-order abstract syntax. For a general account and discussion of these axioms we refer to Honsell *et al.* (2001a), (where *Fresh* was called *Unsat*).

It is worthwhile noticing that these additional axioms are not derivable from the logical ones; in fact, as we will see in section 3, the axioms of Figure 5 are not validated by any set-theoretic semantics.

$$\begin{aligned}
T_{\notin} &: (\upsilon \to \iota \to o) \to (\upsilon \to \iota \to o)\\
&\triangleq \lambda\mathscr{R}^{\upsilon\to\iota\to o}.\lambda x^{\upsilon}.\lambda P^{\iota}.P = 0 \vee\\
&\quad (\exists Q^{\iota}.P = \tau.Q \wedge (\mathscr{R}\ x\ Q)) \vee\\
&\quad (\exists P_1^{\iota}.\exists P_2^{\iota}.P = P_1|P_2 \wedge (\mathscr{R}\ x\ P_1) \wedge (\mathscr{R}\ x\ P_2)) \vee\\
&\quad (\exists Q^{\iota}.\exists y^{\upsilon}.\exists z^{\upsilon}.P = [y \neq z]Q \wedge \neg(x =^{\upsilon} y) \wedge \neg(x =^{\upsilon} z) \wedge (\mathscr{R}\ x\ Q)) \vee\\
&\quad (\exists Q^{\upsilon\to\iota}.P = \nu Q \wedge (\forall y^{\upsilon}.\neg(x =^{\upsilon} y) \Rightarrow (\mathscr{R}\ x\ (Q\ y))))\\
\notin &\triangleq \lambda x^{\upsilon}.\lambda P^{\iota}.\forall\mathscr{R}^{\upsilon\to\iota\to o}.(\forall y^{\upsilon}.\forall Q^{\iota}.(T_{\notin}\ \mathscr{R}\ y\ Q) \Rightarrow (\mathscr{R}\ y\ Q)) \Rightarrow (\mathscr{R}\ x\ P)\\
\in &\triangleq \lambda x^{\upsilon}.\lambda P^{\iota}.\neg(x \notin P)\\
\notin^{n} &\triangleq \lambda x^{\upsilon}.\lambda P^{\upsilon^{n}\to\iota}.x \notin (\nu\lambda x_1^{\upsilon}.\ldots\nu\lambda x_{n-1}^{\upsilon}.\nu(P\ x_1 \ldots x_{n-1})) \quad (n \geqslant 1)
\end{aligned}$$

Fig. 4. The occurrence-checking predicates.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

$$\frac{\Gamma \vdash_\Sigma P : \iota}{\Gamma \rhd_\Sigma \exists x^\upsilon . x \notin P} \quad (Fresh_\iota)$$

$$\frac{\Gamma \vdash_\Sigma P : \upsilon^{n+1} \to \iota \quad \Gamma \vdash_\Sigma Q : \upsilon^{n+1} \to \iota \quad \Gamma \vdash_\Sigma x : \upsilon}{\Gamma \rhd_\Sigma x \notin^{n+1} P \Rightarrow x \notin^{n+1} Q \Rightarrow (P\ x) =^{\upsilon^n \to \iota} (Q\ x) \Rightarrow P =^{\upsilon^{n+1} \to \iota} Q} \quad (Ext^{\upsilon^{n+1} \to \iota})$$

$$\frac{\Gamma \vdash_\Sigma P : \upsilon^n \to \iota \quad \Gamma \vdash_\Sigma x : \upsilon}{\Gamma \rhd_\Sigma \exists Q^{\upsilon^{n+1} \to \iota} . x \notin^{n+1} Q \wedge P =^{\upsilon^n \to \iota} (Q\ x)} \quad (\beta\_exp^{\upsilon^n \to \iota})$$

Fig. 5. Axioms of the Theory of Contexts.

The object system: semantics. We show how to use Υ to encode predicates of object systems. For the simplified π-calculus presented in the previous section, let us consider an operational semantics $\longrightarrow \subseteq \mathscr{P} \times \mathscr{P}$ inductively defined by the following rules:

$$\frac{-}{\tau.P \longrightarrow P}(\text{TAU}) \qquad \frac{P \longrightarrow P'}{(\nu y)P \longrightarrow (\nu y)P'}(\text{RES}) \qquad \frac{P \longrightarrow P'}{P|Q \longrightarrow P'|Q}(\text{PAR}_1)$$

$$\frac{Q \longrightarrow Q'}{P|Q \longrightarrow P|Q'}(\text{PAR}_2) \qquad \frac{P \longrightarrow P'}{[x \neq y]P \longrightarrow P'} x \neq y \ (\text{MISMATCH})$$

This operational semantics is a relation over $Proc \times Proc$ which can be encoded in Υ by a predicate $\longrightarrow : \iota \to \iota \to o$, defined by an impredicative inductive clause for a closure operator $T_{\longrightarrow}$ (Incidentally, this definition makes use of the $\notin$ previously defined, but this does not hold in general):

$$\begin{aligned}
&T_{\longrightarrow} : (\iota \to \iota \to o) \to (\iota \to \iota \to o) \\
&\quad \triangleq \lambda \mathscr{R}^{\iota \to \iota \to o} . \lambda P^\iota . \lambda Q^\iota . P = \tau.Q \vee \\
&\qquad (\exists P_1^\iota . \exists Q_1^\iota . \exists S^\iota . P = P_1|S \wedge Q = Q_1|S \wedge (\mathscr{R}\ P_1\ Q_1)) \vee \\
&\qquad (\exists P_2^\iota . \exists Q_2^\iota . \exists S^\iota . P = S|P_2 \wedge Q = S|Q_2 \wedge (\mathscr{R}\ P_2\ Q_2)) \vee \\
&\qquad (\exists P'^\iota . \exists x^\upsilon . \exists y^\upsilon . P = [x \neq y]P' \wedge \neg(x =^\upsilon y) \wedge (\mathscr{R}\ P'\ Q)) \vee \\
&\qquad (\exists P'^{\upsilon \to \iota} . \exists Q'^{\upsilon \to \iota} . P = \nu P' \wedge Q = \nu Q' \wedge (\forall x^\upsilon . x \notin P' \Rightarrow (\mathscr{R}\ (P'\ x)\ (Q'\ x)))) \\
&\longrightarrow \triangleq \lambda P^\iota . \lambda Q^\iota . \forall \mathscr{R}^{\iota \to \iota \to o} . (\forall P'^\iota . \forall Q'^\iota . (T_{\longrightarrow}\ \mathscr{R}\ P'\ Q') \Rightarrow (\mathscr{R}\ P'\ Q')) \Rightarrow (\mathscr{R}\ P\ Q)
\end{aligned}$$

This encoding technique applies to *coinductive* predicates as well, such as the quite common case of *bisimilarity*. Recall that a binary relation S on processes is a *simulation* iff, for all P, Q processes, if $P\ S\ Q$ and $P \longrightarrow P'$ then for some Q', $Q \longrightarrow Q'$ and $P'\ S\ Q'$. S is a *bisimulation* if both S and S^{-1} are simulations. *Bisimilarity* is the greatest bisimulation, that is the binary relation $\dot{\sim}$ defined by

$$P \dot{\sim} Q \iff \exists S . S \text{ bisimulation and } (P\ S\ Q).$$

The corresponding formalization in Υ is by means of a coinductive definition for a "denseness" operator "$T_{\sim}$":

$$\begin{aligned}
&T_{\sim} : (\iota \to \iota \to o) \to (\iota \to \iota \to o) \\
&\quad \triangleq \lambda \mathscr{R}^{\iota \to \iota \to o} . \lambda P^\iota . \lambda Q^\iota . (\forall P'^\iota . (P \longrightarrow P') \Rightarrow \exists Q'^\iota . (Q \longrightarrow Q') \wedge (\mathscr{R}\ P'\ Q')) \\
&\qquad\qquad \wedge (\forall Q'^\iota . (Q \longrightarrow Q') \Rightarrow \exists P'^\iota . (P \longrightarrow P') \wedge (\mathscr{R}\ P'\ Q'))
\end{aligned}$$

$$\dot{\sim} \triangleq \lambda P^\iota . \lambda Q^\iota . \exists \mathscr{R}^{\iota \to \iota \to o} . (\forall P'^\iota . \forall Q'^\iota . (\mathscr{R}\ P'\ Q') \Rightarrow (T_{\sim}\ \mathscr{R}\ P'\ Q')) \wedge (\mathscr{R}\ P\ Q)$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

In this paper we will not deal with the properties of $\longrightarrow$ and $\dot{\sim}$; we refer the interested reader to Honsell *et al.* (2001b).

2.3 Remarks on the design of Υ

2.3.1 Motivations and rationale

In this subsection we will try to convey the reader the rationale behind the Theory of Contexts. The axioms of this theory aim to reflect in the logic, some fundamental and natural properties of object level "term contexts", "variables" and "name", when these are represented by meta-level abstractions and variables following the weak HOAS paradigm. Their informal meaning is the following:

Freshness: for any term, there exists a variable which does not occur free in it.

Extensionality: two term contexts are equal if they are equal on a fresh variable; that is, if $M(x) = N(x)$ and $x \notin M(\cdot), N(\cdot)$, then $M = N$.

β-expansion: it is always possible to split a term into a context applied to a variable, that is: given a term M and a variable x, there is a context $N(\cdot)$ such that $N(x) = M$ and x does not occur in $N(\cdot)$.

These properties have been first suggested by practical reasoning about process calculi, and have been proved to be quite useful in a number of situation. A simple and recurring situation, faced in all the formal developments of the metatheory of process calculi (like the π-calculus), and in which the Theory of Contexts is put to work is the following: proving that crucial properties are preserved by *fresh renamings*, i.e., by replacing a given name with a fresh one. An example is Lemma 3 of Milner *et al.* (1992) (adapted to the π-calculus fragment used in this paper):

For all processes P, Q *and* $y \notin fn(P) \cup fn(Q)$, *if* $P \longrightarrow Q$ *then* $P\{y/x\} \longrightarrow Q\{y/x\}$.

All these lemmata are instances of the following general pattern of *renaming lemma* (expressed in natural deduction style):

$$\text{for all } C_1(\cdot), \ldots, C_n(\cdot) : \frac{\text{for some } x \notin \bigcup_{i=1}^{n} fn(C_i(\cdot)) : \mathscr{R}(C_1(x), \ldots, C_n(x))}{\text{for all } y \notin \bigcup_{i=1}^{n} fn(C_i(\cdot)) : \mathscr{R}(C_1(y), \ldots, C_n(y))} \qquad (1)$$

where $\mathscr{R} : \tau_1 \to \cdots \to \tau_n \to o$ is a given n-ary relation and $C_1(\cdot) \ldots, C_n(\cdot)$ are meta-variables ranging over contexts of terms (e.g., contexts of π-calculus processes). For instance, Lemma 3 above can be expressed in Υ as follows:

$$P : \upsilon \to \iota, Q : \upsilon \to \iota, x : \upsilon \triangleright_{\Sigma} x \not\in^1 P \Rightarrow x \not\in^1 Q \Rightarrow (P\ x) \longrightarrow (Q\ x) \Rightarrow$$
$$\forall y^{\upsilon}.y \not\in^1 P \Rightarrow y \not\in^1 Q \Rightarrow (P\ y) \longrightarrow (Q\ y)$$

here, $\mathscr{R} = \longrightarrow$, and $C_1 = P, C_2 = Q$.

"On paper", this kind of properties is usually proved by structural induction either on the derivation of the premise $\mathscr{R}(C_1(x), \ldots, C_n(x))$ or on one of the arguments $C_i(x)$ or else on a "measure" of an argument (e.g., the number of symbols it contains). The problem is that the term $C_i(x)$ is not a "plain" metavariable, but rather the application of the metavariable C_i to the metavariable x. Thus, we cannot apply the

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

induction principle over $C_i(x)$, because this would require a second-order unification which is not usually available in proof-assistants.

To circumvent this problem, we need to prove a preliminary version of the renaming lemma, where the necessary second-order unifications are all made explicit in order to recover sufficient inductive information on the structure of the contexts $C_i(\cdot)$ from their instantiations $C_i(x)$. In other words, we "lift" structural information to the level of functional terms, using the β-expansion and the extensionality axioms. This lifting follows a general pattern. First of all, we replace the original goal of the form (1) with the following one

$$\text{for all } C_1(\cdot),\dots,C_n(\cdot), \text{for all } T_1,\dots,T_n : \quad \frac{\text{for some } x \notin \bigcup_{i=1}^{n} fn(C_i(\cdot)) : T_1 = C_1(x),\dots,T_n = C_n(x) \Rightarrow \mathscr{R}(T_1,\dots,T_n)}{\text{for all } y \notin \bigcup_{i=1}^{n} fn(C_i(\cdot)) : \mathscr{R}(C_1(y),\dots,C_n(y))} \tag{2}$$

where $T_1,\dots,T_n$ are plain terms and $T_1 = C_1(x),\dots,T_n = C_n(x)$ are the required unifications. Clearly we can infer the former goal (1) from (2) by taking $T_i = C_i(x)$.

In proving (2), we can proceed by structural induction over T_i. We can take advantage of the structural information on $T_1,\dots,T_n$ given by the inductive hypothesis: using the *β-expansion* axiom, we can rewrite each term T_i as a context applied to x, yielding the equations $T_1 = T_1'(x),\dots,T_n = T_n'(x)$, where $x \notin \bigcup_{i=1}^{n} fn(T_i'(\cdot))$. Differently from $C_i(\cdot)$, $T_i'(\cdot)$ is not a variable, but a concrete λ-abstraction. By transitivity of equality, we obtain the equations $C_i(x) = T_i'(x)$; thus, by the *extensionality* axiom, we can infer $C_i(\cdot) = T_i'(\cdot)$, i.e., the structural information we needed on the variable $C_i(\cdot)$. Such an information can then be used in the instantiations over y in the current goal, in order to apply the suitable constructor of $\mathscr{R}$ and solve the subsequent subgoal by means of the inductive hypothesis.

It should be noticed that this kind of "fresh-renaming" properties cannot be derived in standard type theories, using HOAS-based encodings. In the proof sketch above, the use of *β-expansion* and *extensionality* is essential.

2.3.2 Independence

It is worth noticing that, according to our experience, in order to reason about the metatheory of nominal calculi, full classical logic is not strictly needed. Indeed, we could replace DN in Figure 3 with either an axiom stating the decidability of Leibniz equality over names or an axiom stating the decidability of occurrence predicates of names into terms. This is the approach we adopted in Honsell *et al.* (2001b). In our framework Υ, these two axioms can be rendered as follows:

$$\frac{\Gamma \vdash_\Sigma x : \upsilon \quad \Gamma \vdash_\Sigma y : \upsilon}{\Gamma \rhd_\Sigma x =^\upsilon y \vee x \neq^\upsilon y} \quad (EM_{=^\upsilon}) \qquad \frac{\Gamma \vdash_\Sigma x : \upsilon \quad \Gamma \vdash_\Sigma P : \iota}{\Gamma \rhd_\Sigma x \notin P \vee \neg(x \notin P)} \quad (EM_{\notin})$$

While $EM_{=^\upsilon}$ derives directly from $EM_{\notin}$, on the converse, $EM_{\notin}$ can be derived from $EM_{=^\upsilon}$ using $Fresh_\iota$ and the induction principles Ind^ι, $Ind^{\upsilon\to\iota}$ (section 6.3) over plain terms and contexts.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Thus, the minimal classical flavour that Υ must have in order to allow metatheoretic reasoning about the representation of the π-calculus amounts to decidability of equality of names. However, for simplicity we prefer to stick to full classical logic.

In Honsell *et al.* (2001b) the Theory of Contexts is enriched by another axiom stating the congruence of $\not\in$ with respect to ν (there called "monotonicity"):

$$\frac{\Gamma \vdash_\Sigma x : \upsilon \quad \Gamma \vdash_\Sigma y : \upsilon \quad \Gamma \vdash_\Sigma P : \upsilon \to \iota}{\Gamma \rhd_\Sigma x \not\in (P\ y) \Rightarrow x \not\in \nu P} \qquad (CONG_{\not\in})$$

Recently, we discovered that this law is indeed derivable from $Fresh_\iota$, $EM_{\not\in}$ and Ind^ι. Another possibility of deriving $CONG_{\not\in}$ is to exploit $Ind^{\upsilon\to\iota}$ without any other axioms, i.e., to reason by induction on the structure of the term context $P(\cdot)$ (for the details see Section 4.5.1 of Scagnetto, 2002).

We remark here that congruence of $\in$ w.r.t. ν, i.e.

$$\frac{\Gamma \vdash_\Sigma x : \upsilon \quad \Gamma \vdash_\Sigma y : \upsilon \quad \Gamma \vdash_\Sigma P : \upsilon \to \iota}{\Gamma \rhd_\Sigma \neg x =^\upsilon y \Rightarrow x \in (P\ y) \Rightarrow x \in \nu P} \qquad (CONG_{\in})$$

is trivially derivable by exploiting the inductive nature of $\not\in$.

Finally, for what concerns the axioms schemata $\beta\_exp^{\upsilon^n\to\iota}$ and $Ext^{\upsilon^{n+1}\to\iota}$, we have the following result:

Proposition 2.2
For all $n \in \mathbb{N}$: $Ind^{\upsilon^n\to\iota}$ allows to derive $\beta\_exp^{\upsilon^n\to\iota}$ from $\beta\_exp^{\upsilon^{n+1}\to\iota}$ and (if $n > 0$) $Ext^{\upsilon^n\to\iota}$ from $Ext^{\upsilon^{n+1}\to\iota}$.

Proof
By structural induction on contexts of type $\upsilon^n \to \iota$, using $Ind^{\upsilon^n\to\iota}$. Most cases are trivial; in the case of the ν constructor, we apply the axioms $\beta\_exp^{\upsilon^{n+1}\to\iota}$ and $Ext^{\upsilon^{n+1}\to\iota}$. □

2.3.3 *Simplicity, adequacy and expressiveness*

In the present work we define the predicate $\in$ in terms of $\not\in$ and $=^\upsilon$ rather than giving an independent constructive definition, like in Honsell *et al.* (2001b). This approach is motivated by the fact that in nominal calculi a crucial rôle is played by freshness (i.e. non-occurrence) of names within terms. However, it would be clearly possible to give a constructive definition of such a predicate as follows:

$$\in^c \triangleq \lambda x^\upsilon.\lambda P^\iota.\forall \mathscr{R}^{\upsilon\to\iota\to o}.(\forall y^\upsilon.\forall Q^\iota.(T_{\in^c}(\mathscr{R})\ y\ Q) \Rightarrow (\mathscr{R}\ y\ Q)) \Rightarrow (\mathscr{R}\ x\ P)$$

where

$$\begin{aligned} T_{\in^c} \quad &: \quad (\upsilon \to \iota \to o) \to (\upsilon \to \iota \to o) \\ &\triangleq \quad \lambda \mathscr{R}^{\upsilon\to\iota\to o}.\lambda x^\upsilon.\lambda P^\iota. \\ &\quad (\exists Q^\iota.P = \tau.Q \wedge (\mathscr{R}\ x\ Q)) \vee \\ &\quad (\exists P_1^\iota.\exists P_2^\iota.P = P_1|P_2 \wedge ((\mathscr{R}\ x\ P_1) \vee (\mathscr{R}\ x\ P_2))) \vee \\ &\quad (\exists Q^\iota.\exists y^\upsilon.\exists z^\upsilon.P = [y \neq z]Q \wedge (x =^\upsilon y \vee x =^\upsilon z \vee (\mathscr{R}\ x\ Q))) \vee \\ &\quad (\exists Q^{\upsilon\to\iota}.P = \nu Q \wedge (\forall y^\upsilon.(\mathscr{R}\ x\ (Q\ y)))) \end{aligned}$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

The two definitions are provably equivalent. We have in fact the following:

$$x : \upsilon, P : \iota \rhd_\Sigma x \in P \iff x \in^c P$$

where Σ denotes the signature introduced so far together with the operator $T_{\in^c}$ defined above. Without going into the details of the proof, we simply note that the left ($\Leftarrow$) implication required $Fresh_\iota$ (see Figure 5), while the right ($\Rightarrow$) direction has been proved by means of Ind^ι (see Figure 8), $EM_{=^\upsilon}$ (which is trivially derivable in classical logic), $Fresh_\iota$ and $CONG_{\notin}$ (which, as we pointed out before, can be derived from $Ind^{\upsilon\to\iota}$).

3 The construction of the model $\mathscr{U}$

Proving that the axioms of the Theory of Contexts are sound is not trivial. In fact, they cannot be validated in any set-theoretic interpretation of the metalanguage of Υ. Let us consider an interpretation of the types of Υ as sets, in particular o as the set of truth values $\{\bot, \top\}$, function spaces as sets of all functions, υ, ι as an arbitrary set, truth of a proposition as giving $\top$ under all valuations of the environment with $\forall, \Rightarrow$ given their usual meaning.

So that $Fresh_\iota$ be validated we must necessarily interpret υ as an infinite set, so suppose that $n_0, n_1, \ldots$ is an enumeration of (the interpretation of) υ. Let $Q : \upsilon \Rightarrow \iota$ be the function that maps n_i to $[n_{i+1} \neq n_0]0$. Then, by unfolding the definition of $\notin$ we obtain $n_i \in \nu(Q)$ for all i which is in conflict with $Fresh_\iota$.

More subtly even, under an arbitrary interpretation of function spaces we get a contradiction if we furthermore assume the *Axiom of Unique Choice*, as explained in Hofmann (1999):

$$\frac{\Gamma \vdash_\Sigma p : \tau_1 \to \tau_2 \to o}{\Gamma \rhd_\Sigma (\forall x^{\tau_1}.\exists y^{\tau_2}.p(x,y) \wedge \forall z^{\tau_2}.p(x,z) \Rightarrow y =^{\tau_2} z) \Rightarrow \exists f^{\tau_1\to\tau_2}.\forall x^{\tau_1}.p(x, f(x))}$$

Now, the Axiom of Unique Choice is validated in *any* topos, not only in set theory. Therefore, in order to model consistently the Theory of Contexts, we have to go beyond the theory of topoi.

The solution to this problem has been suggested in Hofmann (1999), by "gluing" together a model of variables and a model of names. The interpretation of types and environments as set-valued functors from the category of finite sets of names and functions. The meaning of a term depends on the set of names which can be associated to its free variables. The functor interpreting a type, therefore, gives the set of possible values for every set of names, while its action on a function between two sets of names corresponds to the capture-avoiding substitution of names in terms. The meaning of a well-typed term is then the interpretation of its typing judgment, which is a natural transformation from the meaning of the environment to the meaning of the type. Naturality ensures that this interpretation is compatible with all possible substitutions of names for interpreting free variables.

Given a set of names for interpreting free variables, the meaning of a formula is the set of name substitutions under which it is verified. Intuitively, a valid proposition must be satisfied under all injective substitutions, since these keep distinct the

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

meaning given to different variables. Therefore a proposition is valid if, for all sets of names, its intepretation contains at least all injective substitutions.

We will proceed as follows: in section 3.1 we introduce the base categories $\check{\mathscr{V}}$ and $\check{\mathscr{I}}$, which will be used in section 3.2 for interpreting the types of Υ. The interpretation of terms will be given in section 3.3. Finally in section 3.4 we will give the interpretation of the logical judgment.

We suppose the reader familiar at least with basic concepts and notions of the theory; e.g. see Mac Lane (1971) and Barr & Wells (1999) for a first introduction. An account of the basic notions and constructions we are going to use is in Appendix A.

A suitable model can be constructed directly using the theory of *triposes* (Pitts, 1999; Jacobs, 1999), an advanced categorical notion used for building models of higher-order logic. However, our aim is to describe the model, and the subtle techniques in it, also to readers with little knowledge of category theory; hence, we have decided to describe in detail the construction. The connections with tripos theory will be described in section 5; in fact, the *cognoscenti* is encouraged to read section 5 in parallel (or before) the present one.

3.1 The ambient categories $\check{\mathscr{V}}$ and $\check{\mathscr{I}}$

In this section we introduce the categories we will use to build the model and we state some useful properties. We will mainly work in $\check{\mathscr{V}} \triangleq \mathscr{S}et^{\mathscr{V}}$, where $\mathscr{V}$ is the category whose objects are finite sets of variables, ranged over by $X, Y, Z, \ldots$, and whose morphisms are functions between them. The intended meaning of morphisms is that of *variable substitutions*. We will use the fact that $\mathscr{V}$ has coproducts, given by disjoint union, and also that, by the Yoneda Lemma, for all $F \in \check{\mathscr{V}}$, $\check{\mathscr{V}}(\mathbf{1}, F) \cong F_{\emptyset}$.

Though the category $\check{\mathscr{V}}$ would suffice to interpret basic datatypes, in order to obtain a consistent model for our extra logical axioms ($Fresh_\iota$, $Ext^{\upsilon^{n+1}\to\iota}$ and $\beta\_exp^{\upsilon^n\to\iota}$), we must interpret the type of propositions o in a non-standard way. Following Hofmann (1999), we introduce the auxiliary notion of predicate over a given type exploiting the subcategory of $\mathscr{V}$ whose objects are the same of $\mathscr{V}$ and morphisms are injective functions. We will denote this category by $\mathscr{I}$; notice that it inherits coproducts from $\mathscr{V}$.

The following proposition is an instance of a general result on subcategories (see Mac Lane, 1971, § X.3). It will be fundamental in the construction of the model.

Proposition 3.1
There is an adjunction $((\_)^r, (\_)^*, \phi)$ from $\check{\mathscr{V}}$ to $\check{\mathscr{I}}$ with $(\_)^r$ the restriction to $\check{\mathscr{I}}$ of functors in $\check{\mathscr{V}}$ and the identity on morphisms, and $(\_)^*$ and ϕ defined as follows:

$(\_)^*$**:** for $G \in \check{\mathscr{I}}$, $G^* : \check{\mathscr{V}} \longrightarrow \mathscr{S}et$ is the functor whose action is

$$G^*_X \triangleq \check{\mathscr{I}}(\mathscr{V}(X, \_)^r, G), \qquad (G^*_f(t))_Z(h) \triangleq t_Z(h \circ f),$$

and for $s \in \check{\mathscr{V}}(F, G)$, $s^* \in \check{\mathscr{V}}(F^*, G^*)$ is the natural trasformation defined by

$$(s^*_X(m))_Y(f) \triangleq s_Y(m_Y(f)),$$

ϕ**:** for all $F \in \check{\mathscr{V}}$, $G \in \check{\mathscr{I}}$, $\alpha \in \check{\mathscr{V}}(F, G^*)$, $x \in F_X$: $(\phi_{FG}(\alpha))_X(x) \triangleq (\alpha_X(x))_X(\mathrm{id}_X)$.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

It will be useful to have the explicit definition of the inverse ψ of ϕ: for $F \in \check{\mathscr{V}}$, $G \in \check{\mathscr{I}}$, $\alpha \in \check{\mathscr{I}}(F^r, G)$, $X, Y \in \mathscr{V}$, $t \in F_X$ and $g \in \mathscr{V}(X, Y)$):

$$((\psi_{FG}(\alpha))_X(t))_Y(g) \triangleq \alpha_Y(F_g(t)).$$

3.2 Interpreting types

3.2.1 Variables and Processes

The interpretation of the type of *variables* is the functor $[\![\upsilon]\!] \triangleq Var : \mathscr{V} \longrightarrow \mathscr{S}et$ defined by $Var_X \triangleq X$ and, for $h \in \mathscr{V}(X, Y), x \in X$, $Var_h(x) \triangleq h(x)$. In other words, *Var* is simply the embedding of $\mathscr{V}$ into $\mathscr{S}et$. Note that it is isomorphic to the representable functor $\check{\mathscr{Y}}(\{\star\})$.

The interpretation $[\![\iota]\!]$ of *processes* is given by the functor *Proc*, which is defined by extending the previous definition $Proc_X$ (denoting the set of processes with free names in X) with the action on morphisms. Given $h : X \longrightarrow Y$, we define $Proc_h \triangleq \sigma$, where $\sigma : Proc_X \longrightarrow Proc_Y$ is the substitution function which replaces every X-indeterminate x in $t \in Proc_X$ with $h(x)$, yielding a term of $Proc_Y$. For this reason, sometimes in the following we will denote $Proc_h(t)$ by $t[h]$. This useful notation can be extended to any type, i.e., for all $F \in \check{\mathscr{V}}$ and $h \in \mathscr{V}(X, Y)$ and $t \in F_X$, we will often denote $F_h(t)$ by $t[h]$.

Notice that *Proc* is not representable; indeed if this were the case, then there would be a finite set of variables Z such that $[\![\iota]\!] \cong \check{\mathscr{Y}}(Z) \triangleq \mathscr{V}(Z, \_)$. From this we could infer that $[\![\iota]\!]_X \triangleq Proc_X \cong \check{\mathscr{Y}}(Z)_X \triangleq \mathscr{V}(Z, X)$, i.e., that the set of processes with free variables included in X would be isomorphic to the set of finite substitutions with domain Z and codomain X. This is clearly absurd since the cardinality of the latter set is finite and precisely $|Z| \cdot |X|$, while the cardinality of $Proc_X$ is infinite (since it is inhabited by the following succession of processes: $0,\ 0|0,\ 0|0|0, \dots$).

3.2.2 Propositions

As anticipated, we cannot interpret propositions in the standard way. Instead we will proceed as follows:

1. a functor $\mathbf{Pred}_{\mathscr{J}} : \check{\mathscr{I}}^{op} \longrightarrow \mathscr{S}et$ with suitable properties is introduced.
2. $\mathbf{Pred}_{\mathscr{J}}$ is extended to a functor $\mathbf{Pred} : \check{\mathscr{V}}^{op} \longrightarrow \mathscr{S}et$ by means of Proposition 3.1; the adjunction ensures that the properties of $\mathbf{Pred}_{\mathscr{J}}$ we are interested in are transferred to **Pred**. In particular **Pred** is representable.
3. $Prop : \mathscr{V} \longrightarrow \mathscr{S}et$ is defined as the functor representing **Pred**.

The whole construction is inspired by results related to the notion of *tripos* (Pitts, 1999). Indeed, the properties of **Pred** we are interested in essentially amount to the conditions ensuring that $\mathbf{Pred}_{\mathscr{J}}$ is a tripos on $\check{\mathscr{I}}$, so that we can interpret Higher Order Logic. However, to keep the construction of the model as elementary as possible, in this section we will not refer to tripos theory, but we will just introduce the notions needed to carry out a direct verification that our construction indeed

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

yields a model of Υ. In section 5 we will briefly discuss how our results can be set in the general setting of tripos theory.

First, we introduce $\mathbf{Pred}_{\check{\mathscr{I}}}$, which assigns to every functor $F \in \check{\mathscr{I}}$, a Boolean algebra of *predicates*. For this purpose we recall the following definition.

Definition 3.1
Given a functor $F : \mathscr{I} \longrightarrow \mathscr{S}et$, a *subfunctor* of F is a $\mathscr{I}$-indexed family of sets $(P_X)_{X \in \mathscr{I}}$ such that

$$\text{for } X \in \mathscr{I} : P_X \subseteq F_X \tag{Sub}$$

$$\text{for } h \in \mathscr{I}(X, Y), \text{if } t \in P_X \text{ then } F_h(t) \in P_Y. \tag{Func}$$

(Notice that by condition (Func), a subfunctor is a functor).

We will say that a subfunctor P is *closed* if it satisfies the following:

$$\text{for all } X, Y \in \mathscr{I},\ t \in F_X,\ \text{if } t[h] \in P_Y \text{ for some } h \in \mathscr{I}(X, Y),\ \text{then } t \in P_X \tag{Closure}$$

(As we will see in section 5, closed subfunctors of F are precisely the double negation closed predicates in the topos logic of $\check{\mathscr{I}}$.)

We will denote a subfunctor P of F by $P \rightarrowtail F$. With the usual abuse of language, we will identify subfunctors of F with the *subobjects*, or *predicates*, of F.

Now let $\mathbf{Pred}_{\check{\mathscr{I}}} : \check{\mathscr{I}}^{op} \longrightarrow \mathscr{S}et$ be defined as follows:

- for $F \in \check{\mathscr{I}}^{op}$, $\mathbf{Pred}_{\check{\mathscr{I}}}(F) \triangleq \{P \in \check{\mathscr{I}} \mid P \rightarrowtail F, P \text{ is closed}\}$;
- for $\alpha \in \check{\mathscr{I}}^{op}(F, G)$ and $P \in \mathbf{Pred}_{\check{\mathscr{I}}}(F)$, $\mathbf{Pred}_{\check{\mathscr{I}}}(\alpha)(P)$ is the subfunctor of G such that $(\mathbf{Pred}_{\check{\mathscr{I}}}(\alpha)(P))_X \triangleq \alpha_X^{-1}(P_X)$, and $(\mathbf{Pred}_{\check{\mathscr{I}}}(\alpha)(P))_f \triangleq F_f$.

It is a standard result that the previous conditions indeed define a functor, and moreover that the following holds:

Proposition 3.2
For all $F \in \check{\mathscr{I}}$, $\mathbf{Pred}_{\check{\mathscr{I}}}(F)$ is a boolean algebra w.r.t. the operations:

$$0_X \triangleq \emptyset \qquad (U \vee V)_X \triangleq U_X \cup V_X$$
$$1_X \triangleq F_X \qquad (U \wedge V)_X \triangleq U_X \cap V_X \qquad (\overline{U})_X \triangleq F_X \setminus U_X$$

and moreover for all $\alpha \in \check{\mathscr{I}}^{op}(F, G)$, $\mathbf{Pred}_{\check{\mathscr{I}}}(\alpha)$ preserves all boolean operations.

The proof of both previous statements is in Appendix B.1. In the following, we will denote by $\leqslant$ the order naturally arising from the operations of the algebra.

Now let $\Omega \in \check{\mathscr{I}}$ be the functor defined by $\Omega_X \triangleq \mathbf{Pred}_{\check{\mathscr{I}}}(\mathscr{I}(X, \_))$ and $\Omega_f \triangleq \mathscr{I}(f, \_)$. (The notation is reminiscent of the fact that this is the subobject classifier in the topos of $\neg\neg$-sheaves over $\mathscr{I}$.) Then:

Proposition 3.3
$\mathbf{Pred}_{\check{\mathscr{I}}}$ and $\check{\mathscr{I}}(\_, \Omega)$ are naturally isomorphic, so $\mathbf{Pred}_{\check{\mathscr{I}}}$ is representable.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

We just give the definition of the isomorphism: $\chi^{\check{\mathscr{J}}} : \mathbf{Pred}_{\check{\mathscr{J}}} \longrightarrow \check{\mathscr{J}}(\_, \Omega)$ and $\kappa^{\check{\mathscr{J}}} : \check{\mathscr{J}}(\_, \Omega) \longrightarrow \mathbf{Pred}_{\check{\mathscr{J}}}$ are defined by:

$$\begin{aligned}(\chi^{\check{\mathscr{J}}}_F(U))_X(t) &\triangleq (\{f \in \mathscr{I}(X,Y) \mid F_f(t) \in U_Y\})_{Y \in \mathscr{I}} \\ \kappa^{\check{\mathscr{J}}}_F(m) &\triangleq (\{t \in F_Y \mid m_Y(t) = \check{\mathscr{Y}}_{\mathscr{J}}(Y)\})_{Y \in \mathscr{I}}.\end{aligned}$$

Then the proof of the proposition is a routine check that these maps form an isomorphism (see Appendix B.2).

Now let us proceed to define the functor $\mathbf{Pred} : \check{\mathscr{V}}^{op} \longrightarrow \mathscr{S}et$ by setting $\mathbf{Pred}(F) \triangleq \mathbf{Pred}_{\check{\mathscr{J}}}(F^r)$ and $\mathbf{Pred}(\alpha) \triangleq \mathbf{Pred}_{\check{\mathscr{J}}}(\alpha^r)$. By Propositions 3.1 and 3.3 we have the following natural isomorphisms:

- for all $F \in \check{\mathscr{V}}$, $\quad \mathbf{Pred}_{\check{\mathscr{J}}}(F^r) \xrightarrow[\chi^{\check{\mathscr{J}}}_{F^r}]{\sim} \check{\mathscr{J}}(F^r, \Omega) \xrightarrow[\psi_{F,\Omega}]{\sim} \check{\mathscr{V}}(F, \Omega^*)$
- for all $X \in \mathscr{V}$, $\quad \gamma_X \triangleq \kappa^{\check{\mathscr{J}}}_{\mathscr{V}(X,\_)^r} : (\Omega^*)_X = \check{\mathscr{J}}(\mathscr{V}(X,\_)^r, \Omega) \longrightarrow \mathbf{Pred}_{\check{\mathscr{J}}}(\mathscr{V}(X,\_)^r)$.

Let *Prop* be defined by

$$Prop_X \triangleq \mathbf{Pred}(\mathscr{V}(X,\_)) \qquad Prop_f \triangleq \mathbf{Pred}(\mathscr{V}(f,\_)) = f \circ \_.$$

Then we obtain natural isomorphisms

$$\chi : \mathbf{Pred} \longrightarrow \check{\mathscr{V}}(\_, Prop) \quad \text{and} \quad \kappa : \check{\mathscr{V}}(\_, Prop) \longrightarrow \mathbf{Pred}$$

given by

$$\begin{aligned}(\chi_F(U))_X(t) &\triangleq (\gamma \circ \psi_{F,\Omega}(\chi^{\check{\mathscr{J}}}_{F^r}(U)))_X(t) = \big(\{g \in \mathscr{V}(X,Y) \mid t[g] \in U_Y\}\big)_{Y \in \mathscr{V}}, \\ \kappa_F(m) &\triangleq \kappa^{\check{\mathscr{J}}}_{F^r}(\phi_{F,\Omega}(\gamma^{-1} \circ m)) = \big(\{t \in F_X \mid m_X(t) \geqslant \mathscr{I}(X,\_)\}\big)_{X \in \mathscr{V}}.\end{aligned}$$

Now we can define the interpretation of the type of propositions as $[\![o]\!] \triangleq Prop$, i.e. the object representing **Pred**.

3.2.3 *Arrow types*

The interpretation $[\![\sigma \to \sigma']\!]$ is given by the functor $[\![\sigma]\!] \Rightarrow [\![\sigma']\!]$ where $\Rightarrow$ is in general the usual exponential in presheaf categories: for A, B in $\check{\mathscr{V}}$:

$$(A \Rightarrow B)_X \triangleq \check{\mathscr{V}}(A \times \mathscr{V}(X,\_), B)$$

$$\text{for } f : Y \to Z \text{ and } m : A \times \mathscr{V}(Y,\_) \longrightarrow B : \quad (A \Rightarrow B)_f(m) \triangleq m \circ (\mathrm{id}_A \times (\_ \circ f))$$

while evaluation $ev_{A,B} : A \times (A \Rightarrow B) \longrightarrow B$ and transposition $\ulcorner \_ \urcorner : \check{\mathscr{V}}(A \times B, C) \to \check{\mathscr{V}}(B, A \Rightarrow C)$ are given as follows, for all $X \in \mathscr{V}$, $a \in A_X$, $b \in B_X$:

$$\begin{aligned}&\text{for } m : A \times \mathscr{V}(X,\_) \longrightarrow B : \quad (ev_{A,B})_X(a,m) \triangleq m_X(a, \mathrm{id}_X), \\ &\text{for } f : A \times B \longrightarrow C : \quad (\ulcorner f \urcorner_X(b))_Y : A_Y \times \mathscr{V}(X,Y) \longrightarrow C_Y \\ &\qquad\qquad (a,h) \mapsto f_Y(a, b[h])\end{aligned}$$

Therefore, elements of $(A \Rightarrow B)_X$ are, in general, natural transformations in $\check{\mathscr{V}}$. However, a particularly important case is when the exponent A is a *representable* type, that is $A \cong \check{\mathscr{Y}}(X)$ for some $X \in \mathscr{V}$. In this case, the elements of $(A \Rightarrow B)_X$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

have a much simpler representation (which holds in any presheaf category $Set^{\mathscr{C}}$ as long as $\mathscr{C}$ has finite coproducts).

Proposition 3.4 (*(Hofmann, 1999)*)
For all $X, Y \in \mathscr{V}$, and B in $\check{\mathscr{V}}$: $(\check{y}(X) \Rightarrow B)_Y \cong B_{X \uplus Y}$.

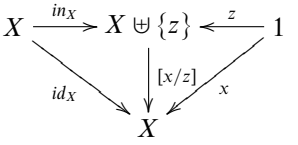
Proof

$$\begin{aligned}(\check{y}(X) \Rightarrow B)_Y &= \check{\mathscr{C}}(\check{y}(X) \times \check{y}(Y), B) && \text{by definition of } \Rightarrow \\ &\cong \check{\mathscr{C}}(\check{y}(X \uplus Y), B) && \text{since } \check{y} \text{ preserves coproducts} \\ &\cong B_{X \uplus Y} && \text{by Yoneda Lemma} \quad \square\end{aligned}$$

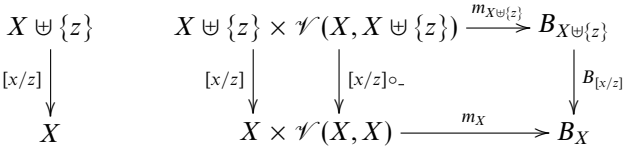
Therefore, each natural transformation $m : Var \times \mathscr{V}(Y, \_) \to B$ is represented by a unique element $\bar{m} \in B_{Y \uplus \{z\}}$, and *vice versa*. Explicitly, this $\bar{m}$ can be described as

$$\bar{m} = m_{X \uplus \{z\}}(z, in_X)$$

where $in_X \in \mathscr{V}(X, X \uplus \{z\})$ is the left inclusion in the following coproduct diagram



Notice that by naturality the following diagram commutes



Taking advantage of this naturality, we have that

$$\begin{aligned}(ev_{Var,B})_X(x, m) &\triangleq m_X(x, \mathrm{id}_X) \\ &= m_X(z[x/z], [x/z] \circ in_X) \\ &= (B_{[x/z]} \circ m_{X \uplus \{z\}})(z, in_X) = \bar{m}[x/z]\end{aligned}$$

In particular, the interpretation of $\upsilon \to \iota$ is $Var \Rightarrow Proc$ and hence, since $Var = \check{y}(1)$, by Proposition 3.4 we have

$$[\![\upsilon \to \iota]\!]_X \cong Proc_{X \uplus \{x\}} \tag{3}$$

In other words, a term of type $\upsilon \to \iota$ with variables in X is (equivalent to) a term in ι with variables in X plus an extra fresh variable x. This corresponds to the basic idea of higher-order abstract syntax, where term schemata correspond to terms with an extra (abstracted) variable.

The equivalence (3) is so fundamental for the construction of the model $\mathscr{U}$, that we can take it as the *real* definition of exponentials of the form $\upsilon \to \iota$. Since $Proc_{[x/z]}$ is simply the capture-avoiding substitution, the corresponding evaluation is

$$(ev_{Var,Proc})_X(x, P) \triangleq P[x/z] \qquad \text{for } P \in Proc_{X \uplus \{z\}} \tag{4}$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

3.3 Interpreting the typing judgment of terms

Typing judgments of the form $\Gamma \vdash M : \sigma$ will be interpreted as suitable natural transformations with domain $[\![\Gamma]\!]$ and codomain $[\![\sigma]\!]$. As usual, the interpretation $[\![\Gamma]\!]$ of an environment $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ is given by the functor $\prod_{i=1}^{n} [\![\sigma_i]\!]$.

We shall give the interpretation of the typing judgment by induction on the depth of the derivation of $\Gamma \vdash M : \sigma$.

Rule VAR: $[\![x_1 : \sigma_1, \ldots, x_i : \sigma_i, \ldots, x_n : \sigma_n \vdash_\Sigma x_i : \sigma_i]\!] \triangleq \pi_i : \prod_{j=1}^{n} [\![\sigma_j]\!] \longrightarrow [\![\sigma_i]\!]$

Rule CONST: for interpreting the judgments involving constants in Σ we introduce the following natural transformations (naturality is trivial to prove):

$$\begin{array}{rr} nil : \mathbf{1} \longrightarrow Proc & mismatch : Var \times Var \times Proc \longrightarrow Proc \\ nil_X : \mathbf{1}_X \longrightarrow Proc_X & mismatch_X : X \times X \times Proc_X \longrightarrow Proc_X \\ * \longmapsto 0 & \langle x, y, P \rangle \longmapsto [x \neq y]P \\ tau : Proc \longrightarrow Proc & par : Proc \times Proc \longrightarrow Proc \\ tau_X : Proc_X \longrightarrow Proc_X & par_X : Proc_X \times Proc_X \longrightarrow Proc_X \\ P \longmapsto \tau.P & \langle P, Q \rangle \longmapsto P \mid Q \end{array}$$

$$\begin{array}{r} new : Var \Rightarrow Proc \longrightarrow Proc \\ new_X : Proc_{X \uplus \{x\}} \longrightarrow Proc_X \\ P \longmapsto (\nu x)P \end{array}$$

We can now interpret judgments of the form $\Gamma \vdash_\Sigma c(M_1, \ldots, M_n) : \sigma$, for c a constant of the signature. Let Γ be an environment, x, y be variables, and P, Q be terms. Then:

- $[\![\Gamma \vdash_\Sigma 0 : \iota]\!] = nil \circ !_{[\![\Gamma]\!]}$, where $!_{[\![\Gamma]\!]}$ is the unique morphism from $[\![\Gamma]\!]$ to $\mathbf{1}$;
- $[\![\Gamma \vdash_\Sigma \tau P : \iota]\!] = tau \circ [\![\Gamma \vdash_\Sigma P : \iota]\!]$;
- $[\![\Gamma \vdash_\Sigma P|Q : \iota]\!] = par \circ \langle [\![\Gamma \vdash_\Sigma P : \iota]\!], [\![\Gamma \vdash_\Sigma Q : \iota]\!] \rangle$;
- $[\![\Gamma \vdash_\Sigma [x \neq y]P : \iota]\!] = mismatch \circ \langle [\![\Gamma \vdash_\Sigma x : \upsilon]\!], [\![\Gamma \vdash_\Sigma y : \upsilon]\!], [\![\Gamma \vdash_\Sigma P : \iota]\!] \rangle$;
- $[\![\Gamma \vdash_\Sigma \nu\lambda x^{\upsilon}.P : \iota]\!] = new \circ [\![\Gamma \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]$.

Rule APP: given $t_1 = [\![\Gamma \vdash_\Sigma M : \sigma' \to \sigma]\!] : [\![\Gamma]\!] \longrightarrow ([\![\sigma']\!] \Rightarrow [\![\sigma]\!])$ and $t_2 = [\![\Gamma \vdash_\Sigma N : \sigma']\!] : [\![\Gamma]\!] \longrightarrow [\![\sigma']\!]$, we define

$$[\![\Gamma \vdash_\Sigma MN : \sigma]\!] \triangleq ev_{[\![\sigma']\!],[\![\sigma]\!]} \circ \langle t_2, t_1 \rangle : [\![\Gamma]\!] \longrightarrow [\![\sigma]\!],$$

Rule ABS: given $t = [\![\Gamma, x : \sigma \vdash_\Sigma M : \sigma']\!] : [\![\Gamma]\!] \times [\![\sigma]\!] \longrightarrow [\![\sigma']\!]$, we define

$$[\![\Gamma \vdash_\Sigma \lambda x^{\sigma}.M : \sigma \to \sigma']\!] \triangleq \ulcorner t \urcorner,$$

where $\ulcorner t \urcorner : [\![\Gamma]\!] \longrightarrow ([\![\sigma]\!] \Rightarrow [\![\sigma']\!])$ is the exponential transpose of t.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Rule $\Rightarrow$: $[\![\Gamma \vdash_\Sigma p \Rightarrow q : o]\!] = imp \circ \langle [\![\Gamma \vdash_\Sigma p : o]\!], [\![\Gamma \vdash_\Sigma q : o]\!] \rangle$, where

$$\begin{array}{rcl} imp_X : Prop_X \times Prop_X & \longrightarrow & Prop_X \\ \langle U, V \rangle & \longmapsto & \overline{U} \vee V. \end{array}$$

Rule $\forall$: $[\![\Gamma \vdash_\Sigma \forall_\sigma p : o]\!] = forall_\sigma \circ [\![\Gamma \vdash_\Sigma p : \sigma \to o]\!]$, where

$$\begin{array}{rcl} (forall_\sigma)_X : ([\![\sigma]\!] \Rightarrow Prop)_X & \longrightarrow & Prop_X \\ m & \longmapsto & \forall_\pi(\kappa_{[\![\sigma]\!] \times \breve{y}(X)}(m)) \end{array}$$

and m is a natural transformation from $[\![\sigma]\!] \times \breve{y}(X)$ to $Prop$ (remember that $([\![\sigma]\!] \Rightarrow Prop)_X \triangleq \breve{\mathscr{C}}([\![\sigma]\!] \times \breve{y}(X), Prop)$), $\pi : [\![\sigma]\!] \times \breve{y}(X) \longrightarrow \breve{y}(X)$ is the projection and, for $F \in \mathbf{Pred}([\![\sigma]\!] \times \breve{y}(X))$,

$$\forall_\pi(F) \triangleq (\{f \in \mathscr{C}(X, Y) \mid \forall g \in \mathscr{I}(Y, Z).\pi_Z^{-1}(g \circ f) \subseteq F_Z\})_{Y \in \mathscr{C}}.$$

More explicitly

$$\begin{aligned} &(forall_\sigma)_X(m) = \\ &\quad = (\{f \in \mathscr{C}(X, Y) \mid \forall g \in \mathscr{I}(Y, Z).\forall t \in [\![\sigma]\!]_Z.\langle t, g \circ u \rangle \in \kappa_{[\![\sigma]\!] \times \breve{y}(X)}(m)_Z\})_{Y \in \mathscr{C}} \end{aligned}$$

Remark. Notice that, if $[\![\Gamma \vdash_\Sigma M : \sigma]\!]$ is defined and $x \notin dom(\Gamma)$, then $[\![\Gamma, x{:}\sigma' \vdash_\Sigma M : \sigma]\!]$ is the following natural transformation:

$$\begin{array}{rcl} ([\![\Gamma, x{:}\sigma' \vdash_\Sigma M : \sigma]\!])_X : [\![\Gamma]\!]_X \times [\![\sigma']\!]_X & \longrightarrow & [\![\sigma]\!]_X \\ \langle \eta, \eta_x \rangle & \longmapsto & [\![\Gamma \vdash_\Sigma M : \sigma]\!]_X(\eta) \end{array}$$

This means that the model $\mathscr{U}$ admits the weakening rule.

3.4 Interpreting logical judgments

As we said before, intuitively a proposition is valid if and only if it is verified under all possible injective substitutions of names. In our model, this means that the interpretation of a valid proposition contains all injective substitutions. More formally, let $[\![\Gamma \vdash_\Sigma p : o]\!]^\uparrow$ and $\top$ be defined by

$$\begin{array}{rcl} [\![\Gamma \vdash_\Sigma p : o]\!]^\uparrow : [\![\Gamma]\!] & \longrightarrow & Prop \\ [\![\Gamma \vdash_\Sigma p : o]\!]^\uparrow_X : [\![\Gamma]\!]_X & \longrightarrow & Prop_X \\ \eta & \longmapsto & [\![\Gamma \vdash_\Sigma p : o]\!]_X(\eta) \wedge \mathscr{I}(X, \_) \end{array} \qquad \begin{array}{rcl} \top : \mathbf{1} & \longrightarrow & Prop \\ \top_X : \mathbf{1}_X & \longrightarrow & Prop_X \\ * & \longmapsto & \mathscr{I}(X, \_) \end{array}$$

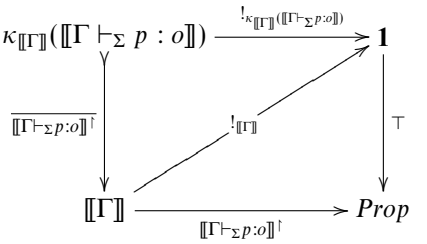
Then we give the following

Definition 3.2 (*Validity*)
We say that $\Gamma \rhd_\Sigma p$ *holds* in $\mathscr{U}$ if $[\![\Gamma \vdash_\Sigma p : o]\!]^\uparrow$ is the constant natural transformation

$$\begin{array}{rcl} (True_\Gamma)_X : [\![\Gamma]\!]_X & \longrightarrow & Prop_X \\ \eta & \longmapsto & \mathscr{I}(X, \_) \end{array}$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

This is equivalent to $[\![\Gamma \vdash_\Sigma p : o]\!]^\dagger = \top \circ !_{[\![\Gamma]\!]}$, i.e., the following diagram commutes:



where the outer square is the pullback of $\top$ along $[\![\Gamma \vdash_\Sigma p : o]\!]^\dagger$. Notice that in this case we have $\kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!]) = 1 \in \mathbf{Pred}([\![\Gamma]\!])$. One should also note that $\kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!]) = \kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!]^\dagger)$; indeed we have the following:

$$\begin{aligned}\kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!]) &\triangleq (\{t \in [\![\Gamma]\!]_X \mid \mathscr{I}(X, \_) \leqslant [\![\Gamma \vdash_\Sigma p : o]\!]_X(t)\})_{X \in \mathscr{V}} \\ &= (\{t \in [\![\Gamma]\!]_X \mid \mathscr{I}(X, \_) \leqslant [\![\Gamma \vdash_\Sigma p : o]\!]_X(t) \wedge \mathscr{I}(X, \_)\})_{X \in \mathscr{V}} \\ &\triangleq \kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!]^\dagger)\end{aligned}$$

4 $\mathscr{U}$ is a model of Υ

In this section we verify that the model defined in section 3 validates the axioms and rules of the framework Υ. In order to be able to streamline the computation of the truth value of a judgment $\Gamma \rhd_\Sigma p$ in the model $\mathscr{U}$, in section 4.1 we introduce an appropriate notion of *forcing*. By means of this useful tool, in section 4.2 we will give a characterisation of Leibniz equality; finally, in sections 4.3 and 4.4 we will verify that $\mathscr{U}$ is a model of Classical Higher-Order Logic and of the Theory of Contexts, respectively.

4.1 Forcing

Definition 4.1
Forcing judgments are statements of the shape

$$X \Vdash_{F,\eta} U$$

for $X \in \mathscr{V}$, $F \in \check{\mathscr{V}}$, $U \in \mathbf{Pred}(F)$, and $\eta \in F_X$. The intended meaning of $X \Vdash_{F,\eta} U$ is that $\eta \in U_X$.

When $F = [\![\Gamma]\!]$, $U = \kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!])$ and $\eta \in [\![\Gamma]\!]_X$, we will also write $X \Vdash_{\Gamma,\eta} p$ instead of $X \Vdash_{\Gamma,\eta} \kappa_{[\![\Gamma]\!]}([\![\Gamma \vdash_\Sigma p : o]\!])$. We will write $X \Vdash p$ to denote "for any Γ such that $\Gamma \vdash_\Sigma p : o$, for all $\eta \in [\![\Gamma]\!]_X$: $X \Vdash_{\Gamma,\eta} p$".

Hence we can rephrase the condition for a logical judgment to be valid in terms of the forcing relation, namely

Proposition 4.1
The judgment $\Gamma \rhd_\Sigma p$ holds in $\mathscr{U}$ iff for all $X \in \mathscr{V}$ and for all $\eta \in [\![\Gamma]\!]_X$ we have $X \Vdash_{\Gamma,\eta} p$.

Lemma 4.1
Let $P \in \mathbf{Pred}(\check{\mathscr{Y}}(X))$ such that $P \not\geqslant \mathscr{I}(X, \_)$, then $P_Y \cap \mathscr{I}(X, Y) = \emptyset$ for all $Y \in \mathscr{V}$.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Proof
We proceed by an absurdity argument: let us suppose that $P \not\geqslant \mathscr{I}(X, \_)$ and there exists $Y \in \mathscr{V}$ and $f \in \mathscr{I}(X, Y)$ such that $f \in P_Y$, we will show that, given any $Z \in \mathscr{V}$ and $g \in \mathscr{I}(X, Z)$, $g \in P_Z$.

Indeed, by property (Closure) of predicates (see Definition 3.1), we have that $\mathrm{id}_X \in P_X$ since $\check{\mathscr{Y}}(X)_f(\mathrm{id}_X) = f \circ \mathrm{id}_X = f \in P_Y$. Then, by property Func, we have that, for all $g \in \mathscr{I}(X, Z)$, $g \in P_Z$ since $\check{\mathscr{Y}}(X)_g(\mathrm{id}_X) = g \circ \mathrm{id}_X = g \in P_Z$. □

A number of useful propositions can now be easily stated.

Theorem 4.1
For all X, Γ, $\eta \in [\![\Gamma]\!]_X$,

1. $X \Vdash_{\Gamma,\eta} \forall x^\sigma.p$ if and only if for all Y, $h \in \mathscr{I}(X, Y)$, and for all $a \in [\![\sigma]\!]_Y$ we have that $Y \Vdash_{(\Gamma, x:\sigma),\ \langle [\![\Gamma]\!]_h(\eta), a\rangle} p$;
2. $X \Vdash_{\Gamma,\eta} p \Rightarrow q$ if and only if $X \Vdash_{\Gamma,\eta} p$ implies $X \Vdash_{\Gamma,\eta} q$;
3. $X \Vdash_{\Gamma,\eta} PM$ iff $\langle [\![\Gamma \vdash_\Sigma M : \sigma]\!]_X(\eta), \mathrm{id}_X\rangle \in \kappa_{[\![\sigma]\!] \times \check{\mathscr{Y}}(X)}([\![\Gamma \vdash_\Sigma P : \sigma \to o]\!]_X(\eta))$, iff $([\![\Gamma \vdash_\Sigma P : \sigma \to o]\!]_X(\eta))_X([\![\Gamma \vdash_\Sigma M : \sigma]\!]_X(\eta), \mathrm{id}_X) \geqslant \mathscr{I}(X, \_)$;
4. it is never the case that $X \Vdash_{\Gamma,\eta} \bot$.

Proof
The proof is a rather easy consequence of the definition of forcing and of the interpretation of logical judgments described in section 3.4. For the details, see Appendix B.3. □

Notice that the last statement of the previous theorem amounts to say that the model $\mathscr{U}$ is sound.

Corollary 4.1
1. $X \Vdash_{\Gamma,\eta} \neg p$ if and only if it is not the case that $X \Vdash_{\Gamma,\eta} p$;
2. $X \Vdash_{\Gamma,\eta} p \wedge q$ if and only if $X \Vdash_{\Gamma,\eta} p$ and $X \Vdash_{\Gamma,\eta} q$;
3. $X \Vdash_{\Gamma,\eta} p \vee q$ if and only if $X \Vdash_{\Gamma,\eta} p$ or $X \Vdash_{\Gamma,\eta} q$;
4. $X \Vdash_{\Gamma,\eta} \exists x^\sigma.p$ if and only if there exist Y, $h \in \mathscr{I}(X, Y)$ and $a \in [\![\sigma]\!]_Y$ such that $Y \Vdash_{(\Gamma, x:\sigma),\ \langle [\![\Gamma]\!]_h(\eta), a\rangle} p$.
5. $X \Vdash_{\Gamma,\eta} \forall x_1^{\sigma_1} \ldots \forall x_n^{\sigma_n}.p$ if and only if for all Y, $f \in \mathscr{I}(X, Y)$, $\eta_1 \in [\![\sigma_1]\!]_Y, \ldots \eta_n \in [\![\sigma_n]\!]_Y$ we have that $Y \Vdash_{(\Gamma, x_1:\sigma_1, \ldots, x_n:\sigma_n), \langle [\![\Gamma]\!]_f(\eta), \eta_1, \ldots, \eta_n\rangle} p$.

Proof
The proof follows from Theorem 4.1 and, in the case of point 5, from a straightforward induction on n. For the details, see Appendix B.4. □

4.2 Characterisation of Leibniz equality

Definition 4.2 (*Separatedness*)
An object $F \in \check{\mathscr{V}}$ is said to be *separated* if its diagonal $\Delta_F \leqslant F \times F$, defined as $(\Delta_F)_X \triangleq \{\langle t, t\rangle \mid t \in F_X\}$ and $(\Delta_F)_h \triangleq F_h \times F_h$, is a predicate of $F \times F$, i.e., $\Delta_F \in \mathbf{Pred}(F \times F)$.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

This definition is equivalent to those usually given about sheaves on textbooks, in the case of sheaves for the $\neg\neg$-topology (Mac LaneS & Moerdijk, 1994, p.223, Lemma V.2.3). As we will show below, for separated objects Leibniz equality coincides with true equality. First we need the following lemma:

Lemma 4.2
An object A in $\check{\mathscr{V}}$ is separated if and only if for every map $i \in \mathscr{I}(X, Y)$ the function $A_i : A_X \to A_Y$ is injective.

Proof
($\Rightarrow$) Let A be separated; then, by definition, Δ_A is a predicate of $A \times A$. Hence, by the condition (Closure), we have that for all X, $Y \in \mathscr{I}$ and $f = \langle a, b\rangle \in A_X \times A_X$, if $(A \times A)_h(f) \in (\Delta_A)_Y$ for some $h \in \mathscr{I}(X, Y)$, then $f \in (\Delta)_X$. Observing that $(A \times A)_h(f) = \langle A_h(a), A_h(b)\rangle$, $A_h(a) = A_h(b)$ (since $(A \times A)_h(f) \in (\Delta_A)_Y$) and $a = b$ (since $f \in (\Delta)_X$), we have proved that A_h is injective for any $h \in \mathscr{I}(X, Y)$.

($\Leftarrow$) It is trivial to verify that Δ_A satisfies both condition (Sub) and (Func). For condition (Closure), we observe that, for all X, $Y \in \mathscr{I}$ and $f = \langle a, b\rangle \in A_X \times A_X$, if $(A \times A)_h(f) \in (\Delta_A)_Y$ for some $h \in \mathscr{I}(X, Y)$, then we must have $A_h(a) = A_h(b)$. At this point, since we know that A_h is injective, we can deduce that $a = b$ holds, whence $f \in (\Delta_A)_X$. □

As a consequence, we have that A is separated iff for all Y, the function $A_? : A_\emptyset \to A_Y$ is injective, where $? : \emptyset \to Y$ is the empty function. In fact, for any $i \in \mathscr{I}(X, Y)$, if i has a left inverse p then A_p is a left inverse to A_i by functoriality, so in this case A_i is injective. Therefore, to establish separatedness it suffices to check injectivity of $A_?$. For example, the presheaf A given by $A_\emptyset = \{0, 1\}$ and $A_X = \{0\}$ otherwise, fails to be separated since $A_?$ is not injective.

Lemma 4.3
The objects *Var*, *Proc* and *Prop* are separated. If G is separated, so is $F \Rightarrow G$.

Proof
If $i \in \mathscr{I}(X, Y)$ and $x \in Var_X = X$ then $Var_i(x) = i(x)$, hence Var_i is injective. Similarly, if $p \in Proc_X$ then $Proc_i(p) = p[i]$, hence $Proc_i$ is injective.

For *Prop* we appeal to the above analysis and merely check that $Prop_?$ is injective. Indeed, $Prop_\emptyset$ contains exactly two elements corresponding to $\top$ and $\bot$ which are never identified.

Finally, assume $u, v \in (F \Rightarrow G)_X = \check{\mathscr{V}}(\check{\mathscr{Y}}(X) \times F, G)$, let $i : X \to Y$ be injective and assume $(F \Rightarrow G)_i(u) = (F \Rightarrow G)_i(v)$. To show $u = v$ assume a—not necessarily injective—map $f : X \to X'$ and $t \in F_{X'}$. We must show $u_{X'}(f, t) = v_{X'}(f, t)$. Now, we can find an injective map $j : X' \to Y'$ and arbitrary map $g : Y \to Y'$ such that $g \circ i = j \circ f$. Since G is separated, it suffices to show $G_j(u(f, t)) = G_j(v(f, t))$. But, $G_j(u(f, t)) = u(j \circ f, F_j(t)) = u(g \circ i, F_j(t)) = (F \Rightarrow G)_i(u)(g, F_j(t))$ which yields the desired conclusion by assumption and symmetry. □

Corollary 4.2
For all types σ, $[\![\sigma]\!]$ is separated.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Theorem 4.2

For all σ, Γ, M, N, X and $\eta \in [\![\Gamma]\!]_X$:

$$X \Vdash_{\Gamma,\eta} M =^{\sigma} N \quad \Longleftrightarrow \quad [\![\Gamma \vdash_\Sigma M : \sigma]\!]_X(\eta) = [\![\Gamma \vdash_\Sigma N : \sigma]\!]_X(\eta)$$

Proof

Let us denote by T the interpretation $[\![\sigma]\!]$ and by Γ' the environment $\Gamma, P : \sigma \to o$, for P a fresh variable. By definition of $=^\sigma$ and Theorem 4.1, $X \Vdash_{\Gamma,\eta} M =^\sigma N$ holds iff

$$\begin{array}{l}\text{for all } Y, h \in \mathscr{I}(X,Y), p \in (T \Rightarrow Prop)_Y : \\ \text{if } [\![\Gamma' \vdash_\Sigma PM : o]\!]_Y(\eta[h], p) \geqslant \mathscr{I}(Y,\_), \text{then } [\![\Gamma' \vdash_\Sigma PN : o]\!]_Y(\eta[h], p) \geqslant \mathscr{I}(Y,\_)\end{array}$$

iff

$$\begin{array}{l}\text{for all } Y, h \in \mathscr{I}(X,Y), p : T \times \mathscr{V}(Y,\_) \longrightarrow Prop : \\ \text{if } ([\![\Gamma' \vdash_\Sigma P : \sigma \to o]\!]_Y(\eta[h], p))_Y([\![\Gamma' \vdash_\Sigma M : \sigma]\!]_Y(\eta[h], p), \mathrm{id}_Y) \geqslant \mathscr{I}(Y,\_), \\ \text{then } ([\![\Gamma' \vdash_\Sigma P : \sigma \to o]\!]_Y(\eta[h], p))_Y([\![\Gamma' \vdash_\Sigma N : \sigma]\!]_Y(\eta[h], p), \mathrm{id}_Y) \geqslant \mathscr{I}(Y,\_)\end{array}$$

iff

$$\begin{array}{l}\text{for all } Y, h \in \mathscr{I}(X,Y), p : T \times \mathscr{V}(Y,\_) \longrightarrow Prop : \\ \text{if } p_Y(m_Y(\eta[h]), \mathrm{id}_Y) \geqslant \mathscr{I}(Y,\_), \text{ then } p_Y(n_Y(\eta[h]), \mathrm{id}_Y) \geqslant \mathscr{I}(Y,\_).\end{array} \tag{5}$$

where $m, n : [\![\Gamma]\!] \longrightarrow T$ denote the natural transformations $[\![\Gamma \vdash_\Sigma M : \sigma]\!]$ and $[\![\Gamma \vdash_\Sigma N : \sigma]\!]$, respectively. We have to prove that this is equivalent to

$$m_X(\eta) = n_X(\eta). \tag{6}$$

(5 $\Rightarrow$ 6) By Corollary 4.3, Δ_T is a predicate of $T \times T$. Let $\delta_T : T \times T \longrightarrow Prop$ be its characteristic map, i.e., for all X and $s, t \in T_X$: $(\delta_T)_X(s,t) \geqslant \mathscr{I}(X,\_)$ iff $s = t$.

Let $\bar{m} : \mathscr{V}(X,\_) \longrightarrow T$ be the natural transformation $\bar{m}_Z(h) \triangleq m_Z(\eta[h])$, and define $q \triangleq \delta_T \circ (\mathrm{id}_T \times \bar{m}) : T \times \mathscr{V}(X,\_) \longrightarrow Prop$. Then, for all $t \in T_X$:

$$q_X(t, \mathrm{id}_X) \geqslant \mathscr{I}(X,\_) \iff (\delta_T)_X(t, m_X(\eta)) \geqslant \mathscr{I}(X,\_) \iff t = m_X(\eta)$$

Instantiating (5) for $Y = X$, $h = \mathrm{id}_X$ and $p = q$, we have

$$\text{if } q_X(m_X(\eta), \mathrm{id}_X) \geqslant \mathscr{I}(X\_) \text{ then } q_X(n_X(\eta), \mathrm{id}_X) \geqslant \mathscr{I}(X,\_)$$

which is equivalent to

$$\text{if } m_X(\eta) = m_X(\eta) \text{ then } n_X(\eta) = m_X(\eta)$$

hence (6) holds.

(6 $\Rightarrow$ 5) By naturality, if $m_X(\eta) = n_X(\eta)$ then for all Y and $h \in \mathscr{I}(X,Y)$, we have $m_Y(\eta[h]) = n_Y(\eta[h])$, hence the thesis. □

We can generalize this result to a semantic form of Leibniz equality. In the following, given a stage X, an object A of $\check{\mathscr{V}}$ and $a_1, a_2 \in A_X$, we will denote by $X \Vdash a_1 =^A a_2$ the property "for all $U \in \mathbf{Pred}(A)$, if $a_1 \in U_X$ then $a_2 \in U_X$."

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Proposition 4.2
Let $A \in \check{\mathscr{V}}$ and $a_1, a_2 \in A_X$. The following are equivalent:

1. $X \Vdash a_1 =^A a_2$
2. If $X \neq \emptyset$ then $a_1 = a_2$; otherwise, $A_?(a_1) = A_?(a_2)$ for $? : \emptyset \to \{x\}$.

Proof
$(1 \Rightarrow 2)$ Leibniz equality is the least reflexive predicate, so it suffices to show that the property (2) is a reflexive predicate, but this is clear by inspection.

$(2 \Rightarrow 1)$ If $X \neq \emptyset$, it is trivial. Let $X = \emptyset$; then, $A_?(a_1) = A_?(a_2)$. Let $U \in \mathbf{Pred}(A)$ such that $a_1 \in U_\emptyset$. Then $A_?(a_1) \in U_{\{x\}}$, that is, $A_?(a_2) \in U_{\{x\}}$. By property (Closure) of predicates, we have $a_2 \in U_\emptyset$. □

Theorem 4.3
Let $F, G \in \check{\mathscr{V}}$. The following formula is valid in $\check{\mathscr{V}}$:

$$\forall u, v{:}F{\Rightarrow}G.(\forall x{:}F.u(x) =^G v(x)) \Rightarrow u =^{F\Rightarrow G} v$$

Proof
Let X be a stage and $u, v \in (F \Rightarrow G)_X = \check{\mathscr{V}}(\mathscr{V}(X, \_) \times F, G)$, such that for all $i \in \mathscr{I}(X, X')$ and $t \in F_{X'}$ one has $X' \Vdash u_{X'}(i,t) =^G v_{X'}(i,t)$. We must show $X \Vdash u =^{F\Rightarrow G} v$. Let us first assume $X \neq \emptyset$. In this case, by Proposition 4.2, we need to show $u = v$, so assume $h \in \mathscr{V}(X, Y)$ and $t \in F_Y$. We should prove $u_Y(h,t) = v_Y(h,t)$.

We write $h = p \circ e$ where $p : Z \to Y$ is surjective and $e : X \to Z$ is injective. This can be done by putting $Z \triangleq X + (Y \setminus \mathrm{Im}(h))$. Every surjective map has a right inverse, thus F_p is surjective, too. So we can find $t_0 \in F_Z$ with $F_p(t_0) = t$. By assumption $u_Y(e, t_0) = v_Y(e, t_0)$, so the claim follows by naturality of u and v.

If $X = \emptyset$ then, again by Prop. 4.2, we need to show that $(F{\Rightarrow}G)_?(u) = (F{\Rightarrow}G)_?(v)$ where $? : \emptyset \to \{x\}$. This is done in the same way. □

Unfortunately, the following extensionality principle for propositions

$$\forall p, q{:}Prop.(p \Leftrightarrow q) \Rightarrow p =^{Prop} q$$

is *not* validated by the model. For example, if $X = \{x, y\}$ then $p_Y \triangleq \mathscr{I}(X, Y)$ and $q_Y \triangleq \mathscr{V}(X, Y)$ are both elements of $Prop_X$ that moreover, are logically equivalent. However, they are clearly nonequal.

Nevertheless, if $u, v : A \longrightarrow Prop$ are maps in $\check{\mathscr{V}}$ and $\forall a \in A.u(a) \Leftrightarrow v(a)$ is valid then $u = v$. The assumption asserts that for all X and $a \in A_X$ if $\mathrm{id}_X \in u_X(a)$ then $\mathrm{id}_X \in v_X(a)$ and vice versa. To deduce $u = v$ assume $a \in A_X$ and $h \in u(a)_Y \subseteq \mathscr{V}(X, Y)$. By naturality of u this implies $\mathrm{id}_Y \in u(A_h(a))$, thus $\mathrm{id}_Y \in v(A_h(a))$ by assumption and $h \in v(a)$ by symmetry.

4.3 $\mathscr{U}$ models logical axioms and rules

Theorem 4.4
The model $\mathscr{U}$ validates all logical axioms and rules; indeed if $\Gamma \vdash_\Sigma p : o$, $\Gamma \vdash_\Sigma q : o$ and $\Gamma \vdash_\Sigma r : o$, then all the following statements hold in $\mathscr{U}$:

1. $\Gamma \rhd_\Sigma (p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow q) \Rightarrow p \Rightarrow r$
2. $\Gamma \rhd_\Sigma p \Rightarrow q \Rightarrow p$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

3. If $\Gamma \vdash_\Sigma P : \sigma \to o$ and $\Gamma \vdash_\Sigma M : \sigma$, then $\Gamma \rhd_\Sigma \forall_\sigma(P) \Rightarrow PM$
4. If $\Gamma, x : \sigma \vdash_\Sigma M : \sigma'$ and $\Gamma \vdash_\Sigma N : \sigma$, then $\Gamma \rhd_\Sigma (\lambda x^\sigma.M)N =^{\sigma'} M[N/x]$
5. If $\Gamma, x : \sigma \vdash_\Sigma M : \sigma'$, $\Gamma, x : \sigma \vdash_\Sigma N : \sigma'$,
   then $\Gamma \rhd_\Sigma (\forall x^\sigma.M =^{\sigma'} N) \Rightarrow \lambda x^\sigma.M =^{\sigma \to \sigma'} \lambda x^\sigma.N$
6. If $\Gamma \vdash_\Sigma M : \sigma \to \sigma$ and $x \notin FV(M)$, then $\Gamma \vdash_\Sigma \lambda x^{\sigma'}.Mx =^{\sigma' \to \sigma} M$
7. $\Gamma \rhd_\Sigma \neg\neg p \Rightarrow p$
8. If $\Gamma \rhd_\Sigma p \Rightarrow q$ and $\Gamma \rhd_\Sigma p$, then $\Gamma \rhd_\Sigma q$
9. If $\Gamma, x : \sigma \rhd_\Sigma p \Rightarrow q$, then $\Gamma \rhd_\Sigma p \Rightarrow \forall x^\sigma.q$

Proof
Follows from Theorem 4.1, Corollary 4.1 and Theorem 4.2 (see Appendix B.5). □

We conclude this section with a result about the $\notin$ predicate which will be useful in the following proofs.

Theorem 4.5
For all Γ, y, M, X and $\eta \in [\![\Gamma]\!]_X$, such that $\Gamma \vdash_\Sigma y : \upsilon$ and $\Gamma \vdash_\Sigma M : \iota$, we have:

$$X \Vdash_{\Gamma,\eta} y \notin M \iff \eta_y \notin FV([\![\Gamma \vdash_\Sigma M : \iota]\!]_X(\eta))$$

Proof
(⇒) By structural induction on the derivation of $\Gamma \vdash_\Sigma M : \iota$.
(⇐) By definition of $T_\notin$, $(T_\notin\ p\ z\ Q)$ is the following λ-term:

$$\begin{array}{l} Q = 0 \vee \\ (\exists P^\iota.Q = \sigma.P \wedge (p\ z\ P)) \vee \\ (\exists P_1^\iota.\exists P_2^\iota.Q = P_1 \mid P_2 \wedge (p\ z\ P_1) \wedge (p\ z\ P_2)) \vee \\ (\exists P^\iota.\exists y^\upsilon.\exists u^\upsilon.Q = [y \neq u]P \wedge \neg z =^\upsilon y \wedge \neg z =^\upsilon u \wedge (p\ z\ P)) \vee \\ (\exists P^{\upsilon \to \iota}.Q = \nu P \wedge (\forall y^\upsilon.\neg z =^\upsilon y \Rightarrow (p\ z\ (P\ y)))) \end{array}$$

Whence (by Corollary 4.1), to prove the premise, it suffices to show that one of the disjunctions holds. This can be achieved by structural induction on the derivation of $\Gamma \vdash_\Sigma M : \iota$.

For the details, see Appendix B.6. □

Corollary 4.3
For all Γ, y, M, X and $\eta \in [\![\Gamma]\!]_X$, such that $\Gamma \vdash_\Sigma y : \upsilon$ and $\Gamma \vdash_\Sigma M : \upsilon \to \iota$. Let $M' = [\![\Gamma \vdash_\Sigma M : \upsilon \to \iota]\!]_X(\eta) \in Proc_{X \uplus \{z\}}$ we have:

$$X \Vdash_{\Gamma,\eta} y \notin^1 M \iff \eta_y \notin FV(M') \setminus \{z\}$$

Proof
By unfolding the definition of $\notin^1$ and using Corollary 4.1 and Theorem 4.5. □

4.4 $\mathscr{U}$ models the Theory of Contexts

Lemma 4.4
The model $\mathscr{U}$ validates $Fresh_\iota$: if $\Gamma \vdash_\Sigma P : \iota$, then for all X, $\eta \in [\![\Gamma]\!]_X$, the following holds: $X \Vdash_{\Gamma,\eta} \exists x^\upsilon.x \notin P$.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Proof
Applying Corollary 4.1, we deduce that $X \Vdash_{\Gamma,\eta} \exists x^{\upsilon}.x \notin P$ holds if and only if there exist Z, $g \in \mathscr{I}(X,Z)$, $z \in [\![\upsilon]\!]_Z \triangleq Z$ such that $Z \Vdash_{(\Gamma,x:\upsilon),\langle [\![\Gamma]\!]_g(\eta),z\rangle} x \notin P$. By Theorem 4.5, this is equivalent to

$$z \notin FV([\![\Gamma, x{:}\upsilon \vdash_\Sigma P : \iota]\!]_Z(\langle [\![\Gamma]\!]_g(\eta), z\rangle)) = FV([\![\Gamma \vdash_\Sigma P : \iota]\!]_Z([\![\Gamma]\!]_g(\eta))).$$

Hence it is sufficient to take $Z \triangleq X \cup \{n\}$ where $n \notin X$ (which surely exists since X is a finite set), $z \triangleq n$ and $g \triangleq \mathrm{id}_X$. □

Lemma 4.5
The model $\mathscr{U}$ validates $Ext^{\upsilon\to\iota}$: if $\Gamma \vdash_\Sigma P : \upsilon \to \iota$, $\Gamma \vdash_\Sigma Q : \upsilon \to \iota$ and $\Gamma \vdash_\Sigma x : \upsilon$, then for all X, $\eta \in [\![\Gamma]\!]_X$, the following holds:

$$X \Vdash_{\Gamma,\eta} x \not\in^1 P \Rightarrow x \not\in^1 Q \Rightarrow (P\ x) =^{\iota} (Q\ x) \Rightarrow P =^{\upsilon\to\iota} Q.$$

Proof
By Theorem 4.1, we have to prove that if $X \Vdash_{\Gamma,\eta} x \not\in^1 P$, $X \Vdash_{\Gamma,\eta} x \not\in^1 Q$ and $X \Vdash_{\Gamma,\eta} (P\ x) =^{\iota} (Q\ x)$, then $X \Vdash_{\Gamma,\eta} P =^{\upsilon\to\iota} Q$. Let us denote

$$P' \triangleq [\![\Gamma \vdash_\Sigma P : \upsilon \to \iota]\!]_X(\eta) \qquad Q' \triangleq [\![\Gamma \vdash_\Sigma Q : \upsilon \to \iota]\!]_X(\eta)$$

in $(Var \Rightarrow Proc)_X = Proc_{X \uplus \{z\}}$. By hypothesis and Corollary 4.3, we have that $\eta_x \notin (FV(P') \cup FV(Q')) \setminus \{z\}$.

By definition of interpretation, we have

$$[\![\Gamma \vdash_\Sigma (P\ x) : \iota]\!]_X(\eta) = (ev_{Var,Proc})_X(\eta_x, [\![\Gamma \vdash_\Sigma P : \upsilon \to \iota]\!]_X(\eta)) = P'[\eta_x/z]$$

and similarly $[\![\Gamma \vdash_\Sigma (P\ x) : \iota]\!]_X(\eta) = Q'[\eta_x/z]$. By hypothesis and Theorem 4.2, therefore, it follows that $P'[\eta_x/z] = Q'[\eta_x/z]$.

We have to prove that the two processes P' and Q' are equal. Indeed:

$$\begin{array}{lll} P' = & P'[\eta_x/z][z/\eta_x] & \text{because } \eta_x \notin FV(P') \setminus \{z\} \\ = & Q'[\eta_x/z][z/\eta_x] & \text{by above} \\ = & Q' & \text{because } \eta_x \notin FV(Q') \setminus \{z\} \quad \square \end{array}$$

Lemma 4.6
The model $\mathscr{U}$ validates β_exp': if $\Gamma \vdash_\Sigma P : \iota$ and $\Gamma \vdash_\Sigma x : \upsilon$, then for all X, $\eta \in [\![\Gamma]\!]_X$, we have that $X \Vdash_{\Gamma,\eta} \exists Q^{\upsilon\to\iota}.x \not\in^1 Q \wedge P =^{\iota} (Q\ x)$ holds.

Proof
By Corollary 4.1, it is sufficient to prove that there exist $Z, g \in \mathscr{I}(X,Z), \eta_Q \in (Var \Rightarrow Proc)_Z$ such that $Z \Vdash_{\Delta,\mu} x \not\in^1 Q$ and $Z \Vdash_{\Delta,\mu} P =^{\iota} (Q\ x)$ hold (where $\Delta \triangleq \Gamma, Q : \upsilon \to \iota$ and $\mu \triangleq \langle [\![\Gamma]\!]_g(\eta), \eta_Q\rangle$). Hence we choose $Z \triangleq X$, $g \triangleq \mathrm{id}_X$ and $\eta_Q \triangleq [\![\Gamma \setminus \{x : \upsilon\} \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\eta')$ (where $\eta' \triangleq \eta_{\restriction_{dom(\Gamma \setminus \{x:\upsilon\})}}$). In order to prove the first forcing statement, we observe that it is equivalent, by Corollary 4.3, to $\eta_x \notin FV(Q') \setminus \{x\}$, where $\eta_x \triangleq [\![\Gamma \vdash_\Sigma x : \upsilon]\!]_X(\eta)$ and $Q' \triangleq [\![\Delta \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\mu_X) \in Proc_{X \uplus \{x\}}$. Hence, we can conclude since the following holds:

$$\begin{aligned} [\![\Delta \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\mu_X) &= [\![\Delta \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\langle \eta, \eta_Q\rangle) \\ &= [\![\Gamma \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\eta) \\ &= [\![\Gamma \setminus \{x : \upsilon\} \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\eta'). \end{aligned}$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Referring to the proof of $X \Vdash_{\Delta,\mu_X} P =^{\iota} (Q\ x)$, we observe that this statement holds if and only if $[\![\Delta \vdash_\Sigma P : \iota]\!]_X(\mu_X) = [\![\Delta \vdash_\Sigma (Q\ x) : \iota]\!]_X(\mu_X)$ holds. Then we have that $[\![\Delta \vdash_\Sigma P : \iota]\!]_X(\mu_X) = [\![\Gamma \vdash_\Sigma P : \iota]\!]_X(\eta)$; hence, we can conclude since

$$\begin{aligned}
[\![\Delta \vdash_\Sigma (Q\ x) : \iota]\!]_X(\mu_X) &= (ev_{Var,Proc})_X([\![\Delta \vdash_\Sigma x : \upsilon]\!]_X(\mu_X), [\![\Delta \vdash_\Sigma Q : \upsilon \to \iota]\!]_X(\mu_X)) \\
&= (ev_{Var,Proc})_X(\eta_x, [\![\Gamma \setminus \{x : \upsilon\} \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\eta')) \\
&= ([\![\Gamma \setminus \{x : \upsilon\} \vdash_\Sigma \lambda x^{\upsilon}.P : \upsilon \to \iota]\!]_X(\eta'))_X(\eta_x, \mathrm{id}_X) \\
&= [\![\Gamma \vdash_\Sigma P : \iota]\!]_X(\eta). \quad \square
\end{aligned}$$

As an immediate corollary of the results proved in this section, and by Theorem 4.1, we have one of the main achievement of this paper.

Theorem 4.6
The Theory of Contexts is consistent with (classical) higher-order logic.

5 Connections with tripos theory

In the previous sections, to be self-contained also to readers without a deep knowledge of category theory, we have illustrated the construction of the model $\mathscr{U}$ in full detail. In this section we will review briefly the basic steps in the construction of the model $\mathscr{U}$ from the point of view of *tripos theory*. The reader aware of the many categorical notions behind this model will benefit from this more abstract perspective which gives a more general justification to the definitions and results presented. This can suggest further developments in using functor categories to model other metalanguages and allows to relate this work with other recent research.

In the following we suppose the reader familiar with the notions of topos, Lawvere-Tierney topology and sheaf (Johnstone, 1977; Mac Lane & Moerdijk, 1994).

First, let us recall some standard notions and results in topos theory. Given a topos $\mathscr{E}$, there is a functor $Sub : \mathscr{E}^{op} \to Set$ which associates to every $X \in \mathscr{E}$ the set of its subobjects, and to every arrow $f : X \to Y$ in $\mathscr{E}$ the function $Sub(X) \to Sub(Y)$ defined by $Sub(f)(m) = f^{-1}(m)$. In general the partially ordered set $Sub(X)$ is a Heyting algebra and the function $Sub(f)$ is a Heyting algebra morphism. A topos is said *Boolean* if, for every $X \in \mathscr{E}$, the Heyting algebra $Sub(X)$ is a Boolean algebra, in this case $Sub(f)$ is a morphism of Boolean algebras.

Given a Lawvere-Tierney topology j on $\mathscr{E}$, the subobject classifier in the topos of j-sheaves $Sh_j(\mathscr{E})$ is the equalizer of id_Ω and j, say Ω_j. In fact Ω_j classifies the j-closed monomorphisms, and the subsheaves of a sheaf are exactly its closed subobjects. Moreover the inclusion I of $Sh_j(\mathscr{E})$ into $\mathscr{E}$ has a left adjoint L which preserves finite limits. These two facts imply that there is an isomorphism between j-closed subobjects of X (j-$Sub(X)$ in the following) and subsheaves of LX.

Now if *false* is the characteristic map of the unique arrow $0 \to 1$ and $\neg$ is the characteristic map of *false*, the morphism $\neg \circ \neg : \Omega \to \Omega$ is a Lawvere-Tierney topology on $\mathscr{E}$ and $Sh_{\neg\neg}(\mathscr{E})$ is a Boolean topos.

Finally if $\mathscr{E}$ is Set^C for some small category C, then the functor Ω defined by $\Omega_X = Sub(h^X)$ and $\Omega_f(F) = Sub(h^f)(F)$ is the subobject classifier of $\mathscr{E}$; so the

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

subobject classifier $\Omega_{\neg\neg}$ in the topos of $\neg\neg$-sheaves is given by $\Omega_{\neg\neg}(X) = \{F \mid F \text{ is a subobject } \neg\neg\text{-closed of } h^X\}$ (and the restriction of Ω on morphisms).

Now we show how these notions and results are related to the properties of $\mathbf{Pred}_{\mathscr{J}}$. First, notice that the closure condition (Closure) in the definition of $\mathbf{Pred}_{\mathscr{J}}$ is exactly the request that a subfunctor is closed w.r.t. the $\neg\neg$-topology. The verification is straightforward: by using twice the following description of $\neg U$, for $U \rightarrowtail A$ in $\check{\mathscr{I}}$:

$$(\neg U)_X = \{a \mid a \in A_X \text{ and, for all } h : X \to Y, A_h(a) \notin U_Y\}$$

one obtains

$$\neg(\neg U)_X = \{x \mid x \in F_X \text{ and, for all } f : X \to Y, \text{ there exists } Z \in \check{\mathscr{I}} \text{ and } g \in \check{\mathscr{I}}(Y, Z) \text{ such that } A_{g\circ f}(x) \in U_Z\}.$$

Thus, requiring that $\neg(\neg U) = U$ is equivalent to condition (Closure). As a consequence, $\mathbf{Pred}_{\mathscr{J}}$ is the functor Sub in the topos of $\neg\neg$-sheaves of $\check{\mathscr{I}}$. This immediately imply Proposition 3.2.

Now, as to $\Omega \in \check{\mathscr{I}}$, previous remarks show that it is precisely the subobject classifier in the topos $Sh_{\neg\neg}(\check{\mathscr{I}})$ so Proposition 3.3 follows from fact that in any topos $\mathscr{E}$ there is a natural isomorphism $Sub \cong \mathscr{E}(\_, \Omega)$. We remark that, actually, Ω_X is a two-element set.

To clarify in the present context the definition of **Pred** we need to introduce some more notions and results.

The principal one is the notion of *tripos*, a structure which generalizes realizability toposes. There are several slightly different definitions of tripos in the literature (Hyland *et al.*, 1980; Pitts, 1981; van Oosten, 1991; Jacobs, 1999); the following one, is good for our purpose:

Definition 5.1
Let $\mathscr{C}$ be a category with finite products. A *$\mathscr{C}$-tripos* is a functor $\mathscr{P} : \mathscr{C}^{op} \to Set$ such that

i) for each $A \in \mathscr{C}$, $\mathscr{P}(A)$ is a Heyting algebra
ii) for each $f \in \mathscr{C}(A, B)$
   (a) $\mathscr{P}(f)$ is a homomorphism of Heyting algebras
   (b) $\mathscr{P}(f)$ has left and right adjoints $\exists f$ and $\forall f$ which satisfy the Beck-Chevalley condition: if

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow{k} & D \end{array}$$

   is a pullback square then $\exists f \circ \mathscr{P}(k) = \mathscr{P}(g) \circ \exists h$ (and hence also the dual condition for $\forall$ holds)
iii) $\mathscr{P}$, when regarded as a set-valued functor, is representable, i.e., there is an object $Prop \in \mathscr{C}$ such that for all A: $\mathscr{P}(A) \cong \mathscr{C}(A, Prop)$.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

A fundamental property of triposes is that, if $\mathscr{C}$ models some metalanguage then a $\mathscr{C}$-tripos models intuitionistic higher order logic over that metalanguage. This means that there is a type for propositions, terms formers for implications and universal quantification. We can therefore interpret the logical judgment $\Gamma \rhd \phi$ which intuitively states that the proposition ϕ, involving variables from Γ, holds. (Pitts, 1981) proves that intuitionistic logic is sound w.r.t. this semantics

To show the connection between triposes and the functor **Pred** we will apply the following two results which we state without proof.

Proposition 5.1 (*(Pitts, 1981), Example 1.3 (i)*)
If $\mathscr{E}$ is a topos, the functor $Sub : \mathscr{E}^{op} \to Set$ carries the structure of a tripos.

Proposition 5.2 (*(van Oosten, 1991), Prop. 1.4*)
If $\mathscr{C}, \mathscr{D}$ are categories with finite products, $F \dashv G : \mathscr{C} \to \mathscr{D}$, F preserves products and $\mathbf{Pred}_{\mathscr{D}}$ is a tripos on $\mathscr{D}$, then the functor $\mathbf{Pred}_{\mathscr{C}}$ defined by

$$\begin{aligned} \mathbf{Pred}_{\mathscr{C}} : \mathscr{C}^{op} &\longrightarrow Set \\ X &\longmapsto \mathbf{Pred}_{\mathscr{D}}(F(X)) \\ (X \xrightarrow{f} Y) &\longmapsto \mathbf{Pred}_{\mathscr{D}}(F(f)) \end{aligned}$$

is a tripos on C.

Now we can show that

Proposition 5.3
Pred is a tripos on $\check{\mathscr{V}}$.

Proof
Consider the adjunctions $L \dashv I : \check{\mathscr{I}} \to Sh_{\neg\neg}(\check{\mathscr{I}})$ and $(\_)^r \dashv (\_)^* : \check{\mathscr{V}} \to \check{\mathscr{I}}$, where L and $(\_)^r$ preserve products. By Proposition 5.2 and the fact that $\neg\neg\text{-}Sub(F) \cong Sub_{Sh_{\neg\neg}(\check{\mathscr{I}})}(LF)$, the functor $\mathbf{Pred}_{\check{\mathscr{I}}}$ is a tripos. Another application of Proposition 5.2 immediately shows that **Pred** is a tripos on $\check{\mathscr{V}}$. □

The interpretation defined in the model $\mathscr{U}$ has been suggested by tripos semantics, therefore from general results it follows that all intuitionistic theorems hold in $\mathscr{U}$. Moreover, since $\mathbf{Pred}(F)$ is a Boolean algebra, the logic of **Pred** is the full higher-order classical logic. This is a consequence of the fact that we consider only the $\neg\neg$-subobjects. In other words, although we work in $\check{\mathscr{V}}$, our logical propositions ultimately live in $Sh_{\neg\neg}(\check{\mathscr{I}})$.

6 Recursion and induction

In the case of traditional first-order signatures, recursion and induction principles are easily derived directly from syntactic definitions. Unfortunately, this is not so clear when we consider second-order and higher-order signatures. The key issue is about the arguments of the binding constructors. Let us suppose to prove the

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

$$\frac{\Gamma \vdash_\Sigma f_1 : \sigma \quad \Gamma \vdash_\Sigma f_2 : \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_3 : \sigma \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_4 : \upsilon \to \upsilon \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_5 : (\upsilon \to \sigma) \to \sigma}{\Gamma \rhd_\Sigma (R\ 0) =^\sigma f_1} \quad (Rec^\iota_\sigma\_red_1)$$

$$\frac{\Gamma \vdash_\Sigma f_1 : \sigma \quad \Gamma \vdash_\Sigma f_2 : \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_3 : \sigma \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_4 : \upsilon \to \upsilon \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_5 : (\upsilon \to \sigma) \to \sigma}{\Gamma \rhd_\Sigma \forall P^\iota.(R\ \tau.P) =^\sigma (f_2\ (R\ P))} \quad (Rec^\iota_\sigma\_red_2)$$

$$\frac{\Gamma \vdash_\Sigma f_1 : \sigma \quad \Gamma \vdash_\Sigma f_2 : \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_3 : \sigma \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_4 : \upsilon \to \upsilon \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_5 : (\upsilon \to \sigma) \to \sigma}{\Gamma \rhd_\Sigma \forall P^\iota.\forall Q^\iota.(R\ P|Q) =^\sigma (f_3\ (R\ P)\ (R\ Q))} \quad (Rec^\iota_\sigma\_red_3)$$

$$\frac{\Gamma \vdash_\Sigma f_1 : \sigma \quad \Gamma \vdash_\Sigma f_2 : \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_3 : \sigma \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_4 : \upsilon \to \upsilon \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_5 : (\upsilon \to \sigma) \to \sigma}{\Gamma \rhd_\Sigma \forall x^\upsilon.\forall y^\upsilon.\forall P^\iota.(R\ [x \neq y]P) =^\sigma (f_4\ x\ y\ (R\ P))} \quad (Rec^\iota_\sigma\_red_4)$$

$$\frac{\Gamma \vdash_\Sigma f_1 : \sigma \quad \Gamma \vdash_\Sigma f_2 : \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_3 : \sigma \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_4 : \upsilon \to \upsilon \to \sigma \to \sigma \quad \Gamma \vdash_\Sigma f_5 : (\upsilon \to \sigma) \to \sigma}{\Gamma \rhd_\Sigma \forall P^{\upsilon \to \iota}.(R\ \nu P) =^\sigma (f_5\ \lambda x^\upsilon.(R\ (P\ x)))} \quad (Rec^\iota_\sigma\_red_5)$$

where R is a typographic shorthand for $(Rec^\iota_\sigma\ f_1\ f_2\ f_3\ f_4\ f_5)$;

Fig. 6. Reduction rules for first-order recursion.

validity of a given property for all processes (as defined in section 2). Proceeding by structural induction, it is not clear what the inductive hypothesis should be in the case of $\nu x.P$, because P is an abstraction (of type $\upsilon \to \iota$) and not a plain process.

This problem has been addressed in Hofmann (1999), where induction principles for higher-order abstract syntax have been introduced and justified in suitable presheaf categories. We aim to extend these constructions to our logical framework Υ, and within the tripos-based model $\mathscr{U}$ presented in the previous sections.

More precisely, in this section we extend the logical framework Υ with *recursion* and *induction* principles, also over second-order data types, that is, terms with "holes". We prove that these rules are justified by the model $\mathscr{U}$. Finally, we conclude the section with a more abstract (and shorter) treatment of these issues. The required categorical theoretic notions are those of *(initial) algebra*, *slice category* and *strong functor*; see Appendix A and Jacobs (1999).

6.1 First-order recursion

To be able to define recursive function over ι, we extend the signature Σ with a *recursor operator* Rec^ι_σ for any type σ:

$$Rec^\iota_\sigma : \sigma \to (\sigma \to \sigma) \to (\sigma \to \sigma \to \sigma) \to (\upsilon \to \upsilon \to \sigma \to \sigma) \to ((\upsilon \to \sigma) \to \sigma) \to \iota \to \sigma$$

and whose reduction rules are given in Figure 6.

To interpret the constant Rec^ι_σ it is sufficient to show that *Proc* can be seen as the initial algebra of the functor $T : \mathscr{V}^\wedge \longrightarrow \mathscr{V}^\wedge$ defined on objects by

$$TF \triangleq \mathbf{1} + F + (F \times F) + (Var \times Var \times F) + (Var \Rightarrow F),$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

and on morphisms $h : F \longrightarrow G$ by (at each stage $X \in \mathscr{V}$):

$$\begin{aligned}
(Th)_X : (TF)_X &\longrightarrow (TG)_X \\
in_1(*) &\longmapsto in_1(*) \\
in_2(a) &\longmapsto in_2(h_X(a)) \\
in_3(\langle a, b\rangle) &\longmapsto in_3(\langle h_X(a), h_X(b)\rangle) \\
in_4(\langle x, y, a\rangle) &\longmapsto in_4(\langle x, y, h_X(a)\rangle) \\
in_5(a) &\longmapsto in_5(h_{X \uplus \{x\}}(a)) \qquad \text{where } a \in F_{X \uplus \{x\}}\,.
\end{aligned}$$

Indeed, any type σ equipped with $f_1, \ldots, f_5$ as in Figure 6 induces a semantic type $[\![\sigma]\!]$ with a T-algebra structure on it (defined by the interpretation of $f_1, \ldots, f_5$). If *Proc* is the initial T-algebra, then there exists a unique homomorphism (i.e., a morphism of algebras) which can be used for interpreting the recursor R in Figure 6. Parameterizing this construction over σ and $f_1, \ldots, f_5$, we will obtained the interpretation of Rec^{ι}_{σ}. Let us spell out in detail this construction.

First, we define the algebra structure $\alpha : T\,Proc \longrightarrow Proc$ for *Proc* as the "natural term forming operation" at each stage $X \in \mathscr{V}$:

$$\begin{aligned}
\alpha_X(in_1(*)) &\triangleq 0 \\
\alpha_X(in_2(P)) &\triangleq \tau.P \\
\alpha_X(in_3(\langle P_1, P_2\rangle)) &\triangleq P_1 | P_2 \\
\alpha_X(in_4(\langle x, y, P\rangle)) &\triangleq [x \neq y]P \\
\alpha_X(in_5(P)) &\triangleq (\nu x)P \qquad \text{where } P \in Proc_{X \uplus \{x\}}
\end{aligned}$$

Proposition 6.1

$(Proc, \alpha)$ is an initial T-algebra.

Proof

Let (B, β) be an arbitrary T-algebra; then there is a unique homomorphism $f : (Proc, \alpha) \longrightarrow (B, \beta)$ of T-algebras such that $f \circ \alpha = \beta \circ Tf$. Given f, in order to prove the latter equality we must consider each component f_X for $X \in \mathscr{V}$. We define f by cases as follows:

$$\begin{aligned}
f_X(0) &\triangleq \beta_X(in_1(*)) \\
f_X(\tau.P) &\triangleq \beta_X(in_2(f_X(P))) \\
f_X(P_1 | P_2) &\triangleq \beta_X(in_3(\langle f_X(P_1), f_X(P_2)\rangle)) \\
f_X([x \neq y]P) &\triangleq \beta_X(in_4(\langle x, y, f_X(P)\rangle)) \\
f_X((\nu x)P)) &\triangleq \beta_X(in_5(f_{X \uplus \{x\}}(P))) \qquad (P \in Proc_{X \uplus \{x\}})
\end{aligned}$$

Then we can easily check that, for each $t \in (T\,Proc)_X$, we have $f_X(\alpha_X(t)) = \beta_X((Tf)_X(t))$:

$$\begin{aligned}
f_X(\alpha_X(in_1(*))) = f_X(0) \triangleq \beta_X(in_1(*)) = \beta_X((Tf)_X(in_1(*))) \\
f_X(\alpha_X(in_2(P))) = f_X(\tau.P) \triangleq \beta_X(in_2(f_X(P))) = \beta_X((Tf)_X(in_2(P)))
\end{aligned}$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

$$\begin{aligned}
f_X(\alpha_X(in_3(\langle P_1, P_2\rangle))) &= f_X(P_1|P_2) \triangleq \beta_X(in_3(\langle f_X(P_1), f_X(P_2)\rangle)) \\
&= \beta_X((Tf)_X(in_3(\langle P_1, P_2\rangle))) \\
f_X(\alpha_X(in_4(\langle x, y, P\rangle))) &= f_X([x \neq y]P) \triangleq \beta_X(in_4(\langle x, y, f_X(P)\rangle)) \\
&= \beta_X((Tf)_X(in_4(\langle x, y, P\rangle))) \\
f_X(\alpha_X(in_5(P))) &= f_X((\nu x)P) \qquad (P \in Proc_{X \uplus \{x\}}) \\
&\triangleq \beta_X(in_5(f_{X \uplus \{x\}}(P))) \\
&= \beta_X((Tf)_X(in_5(P)))
\end{aligned}$$

The uniqueness of f follows by its definition. Indeed, if there is another homomorphism $g : (Proc, \alpha) \longrightarrow (B, \beta)$ such that $g \circ \alpha = \beta \circ Tg$, then rewriting the previous equality and simplifying it according to the definition of α we would obtain exactly the definition of f by cases. □

Using this result we can interpret the recursor Rec_i^σ as follows. Let $A \triangleq [\![\sigma]\!]$, $G \triangleq [\![\Gamma]\!]$ and $\Gamma \vdash R : Proc \to \sigma$, where $R \triangleq (Rec_i^\sigma\ f_1\ f_2\ f_3\ f_4\ f_5)$. Let g_i be the meaning of f_i, as follows:

$$\begin{aligned}
g_1 &= [\![\Gamma \vdash f_1 : \sigma]\!] && : G \longrightarrow A \\
g_2 &= [\![\Gamma \vdash f_2 : \sigma \to \sigma]\!] && : G \longrightarrow A \Rightarrow A \\
g_3 &= [\![\Gamma \vdash f_3 : \sigma \to \sigma \to \sigma]\!] && : G \longrightarrow A \Rightarrow A \Rightarrow A \\
g_4 &= [\![\Gamma \vdash f_4 : \upsilon \to \upsilon \to \sigma \to \sigma]\!] && : G \longrightarrow Var \Rightarrow Var \Rightarrow A \Rightarrow A \\
g_5 &= [\![\Gamma \vdash f_5 : (\upsilon \to \sigma) \to \sigma]\!] && : G \longrightarrow (Var \Rightarrow A) \Rightarrow A
\end{aligned}$$

We define a natural transformation $m : T(G \Rightarrow A) \longrightarrow G \Rightarrow A$, that is, for $X \in \mathscr{V}$,

$$\begin{aligned}
m_X : \mathbf{1}_X + (G \Rightarrow A)_X + (G \Rightarrow A)_X \times (G \Rightarrow A)_X + Var_X \times Var_X \times (G \Rightarrow A)_X + \\
+ (Var \Rightarrow G \Rightarrow A)_X \longrightarrow (G \Rightarrow A)_X
\end{aligned}$$

by cases as follows, bearing in mind that $(G \Rightarrow A)_X = \check{\mathscr{V}}(G \times \mathscr{V}(X, \_), A)$: for a stage Y, $\eta \in G_Y$ and $h \in \mathscr{V}(X, Y)$,

$$\begin{aligned}
(m_X(in_1(*)))_Y(\eta, h) &\triangleq g_{1Y}(\eta) \\
(m_X(in_2(r)))_Y(\eta, h) &\triangleq (g_{2Y}(\eta))_Y(r_Y(\eta, h), \mathrm{id}_Y) \\
(m_X(in_3(\langle r_1, r_2\rangle)))_Y(\eta, h) &\triangleq ((g_{3Y}(\eta))_Y(r_{1Y}(\eta, h), \mathrm{id}_Y))_Y(r_{2Y}(\eta, h), \mathrm{id}_Y) \\
(m_X(in_4(\langle x, y, r\rangle)))_Y(\eta, h) &\triangleq (((g_{4Y}(\eta))_Y(h(x), \mathrm{id}_Y))_Y(h(y), \mathrm{id}_Y))_Y(r_Y(\eta, h), \mathrm{id}_Y)) \\
(m_X(in_5(r)))_Y(\eta, h) &\triangleq (g_{5Y}(\eta))_Y(r', \mathrm{id}_Y) \\
&\quad \text{where} \quad r' : Var \times \mathscr{V}(Y, _) \longrightarrow A \\
&\qquad\qquad r'_Z : Z \times \mathscr{V}(Y, Z) \longrightarrow A_Z \\
&\qquad\qquad\quad \langle z, k\rangle \longmapsto (r_Z(z, k \circ h))_Z(\eta[h], \mathrm{id}_Z)
\end{aligned}$$

Thus, $(G \Rightarrow A, m)$ is a T-algebra; therefore, there exists a unique natural transformation $\bar{m} : Proc \longrightarrow G \Rightarrow A$ such that $m \circ T\bar{m} = \bar{m} \circ \alpha$. By using a standard argument of cartesian closed categories, $\bar{m}$ can be converted into the morphism $G \longrightarrow Proc \Rightarrow A$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

we need. More explicitly, we interpret $\Gamma \vdash R : Proc \to \sigma$ as follows:

$$\begin{aligned}
[\![\Gamma \vdash R : Proc \to \sigma]\!] &: G \longrightarrow Proc \Rightarrow A \\
[\![\Gamma \vdash R : Proc \to \sigma]\!]_X &: G_X \longrightarrow (Proc \Rightarrow A)_X \\
[\![\Gamma \vdash R : Proc \to \sigma]\!]_X(\eta) &: Proc \times \mathscr{V}(X, _) \longrightarrow A \\
[\![\Gamma \vdash R : Proc \to \sigma]\!]_X(\eta)_Y &: Proc_Y \times \mathscr{V}(X, Y) \longrightarrow A_Y \\
&\quad \langle P, h \rangle \longmapsto (\bar{m}_Y(P))_Y(\eta[h], \mathrm{id}_Y)
\end{aligned}$$

We can now prove the soundness of the recursion principles.

Theorem 6.1
The model $\mathscr{U}$ validates $Rec^i_\sigma\_red_i$, for $i = 1 \ldots 5$.

Proof
Long unfolding of forcing definitions, using the universal property of the initial T-algebra *Proc*; see Appendix B.7. □

6.2 Second-order recursion

First-order recursion rules can be generalized to second-order processes, i.e. terms with holes for variables. Indeed, the initial algebra over *Proc* can be readily "lifted" to the types $Var \Rightarrow Proc$, $Var \Rightarrow Var \Rightarrow Proc$, ... Let us consider the functor $T' : \check{\mathscr{V}} \longrightarrow \check{\mathscr{V}}$ defined on objects by

$$T'_F \triangleq \mathbf{1} + F + F \times F + (Var \Rightarrow Var) \times (Var \Rightarrow Var) \times F + (Var \Rightarrow F),$$

and on morphisms in the obvious way. Then, the following holds:

Proposition 6.2
$Var \Rightarrow Proc$ has an initial T'-algebra structure, which is isomorphic to $Var \Rightarrow \alpha$.

Proof
Let $G : \check{\mathscr{V}} \longrightarrow \check{\mathscr{V}}$ be the functor $G(F) \triangleq Var \Rightarrow F$. G has a right adjoint, namely the functor $R : \check{\mathscr{V}} \longrightarrow \check{\mathscr{V}}$ defined on objects by

$$R(F)_X = \check{\mathscr{V}}(Var \Rightarrow \check{y}(X), F) \qquad R(F)_h = \_ \circ (Var \Rightarrow \check{y}(h)) \quad (h \in \mathscr{V}(X, Y))$$

and on natural transformations $t \in \check{\mathscr{V}}(F, F')$ by $R(t) : Var \Rightarrow F \longrightarrow Var \Rightarrow F'$, $R(t)_X(f) = t \circ f$ for $f \in \check{\mathscr{V}}(Var \Rightarrow \check{y}(X), F)$. Hence, for Theorem A.1, we need only to show that $T' \circ G \cong G \circ T$. Given any functor F in $\check{\mathscr{V}}$, we have that

$$\begin{aligned}
(T' \circ G)(F) = &T'(Var \Rightarrow F) = \mathbf{1} + Var \Rightarrow F + (Var \Rightarrow F) \times (Var \Rightarrow F) + \\
&+ (Var \Rightarrow Var) \times (Var \Rightarrow Var) \times (Var \Rightarrow F) + Var \Rightarrow (Var \Rightarrow F) \\
\cong &Var \Rightarrow \mathbf{1} + Var \Rightarrow F + Var \Rightarrow (F \times F) + \\
&+ Var \Rightarrow (Var \times Var \times F) + (Var \Rightarrow (Var \Rightarrow F)) \\
\cong &Var \Rightarrow (\mathbf{1} + F + F \times F + Var \times Var \times F + Var \Rightarrow F) \\
= &Var \Rightarrow TF = (G \circ T)_F
\end{aligned}$$

and similarly for the morphism part. □

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

We can elaborate the functor T' a step further, by noticing that

$$Var \Rightarrow Var \cong Var + \mathbf{1}.$$

Indeed, for all X, we have $(Var \Rightarrow Var)_X = Var_{X \uplus \{x\}} = X \uplus \{x\} \cong X+1 = (Var+\mathbf{1})_X$. Thus we can rewrite T' as follows:

$$T'_F \triangleq \mathbf{1} + F + F \times F + \underbrace{Var \times Var \times F + Var \times F + Var \times F + F}_{\cong (Var \Rightarrow Var) \times (Var \Rightarrow Var) \times F} + Var \Rightarrow F \quad (7)$$

and Proposition 6.2 still holds, that is, $Var \Rightarrow Proc$ is a T'-algebra. The intuitive meaning of the four cases arising from an abstraction $\lambda x.[y \neq z]P$ corresponds to the four situations when none, one or both y, z are exactly x, and hence are bound by the abstraction.

This argument can be generalized to an arbitrary number of "holes", so that all types $Var^n \Rightarrow Proc$ have an initial algebra structure for a suitable functor. In fact, it is easy to see that for all n:

$$Var^n \Rightarrow Var \cong Var + \underbrace{\mathbf{1} + \cdots + \mathbf{1}}_{n \text{ times}}$$

Hence we can generalize (7) at any number of holes, as follows:

$$T^{(n)}_F \triangleq \mathbf{1} + F + F \times F + Var \times Var \times F + \\ + \underbrace{Var \times F + \cdots + Var \times F}_{2n \text{ times}} + \underbrace{F + \cdots + F}_{n^2 \text{ times}} + Var \Rightarrow F \quad (8)$$

Correspondingly, Proposition 6.2 can be generalized as follows:

Theorem 6.2
For all n, $Var^n \Rightarrow Proc$ has an initial $T^{(n)}$-algebra structure.

From the definition of $T^{(n)}$ we can derive immediately that the recursor over second-order terms with n holes (i.e., contexts with n free variables) for type σ has the following type:

$$Rec^{\upsilon^n \to \iota}_{\sigma} : \sigma \to (\sigma \to \sigma) \to (\sigma \to \sigma \to \sigma) \to (\upsilon \to \upsilon \to \sigma \to \sigma) \to \\ \underbrace{(\upsilon \to \sigma \to \sigma) \to \cdots \to (\upsilon \to \sigma \to \sigma)}_{2n \text{ times}} \to \underbrace{(\sigma \to \sigma) \to \cdots \to (\sigma \to \sigma)}_{n^2 \text{ times}} \to \\ ((\upsilon \to \sigma) \to \sigma) \to (\upsilon^n \to \iota) \to \sigma$$

The reduction rules for second-order recursion are in Figure 7. Notice that when $n = 0$, these rules reduce to those for first-order terms (Figure 6).

Theorem 6.3
For all n, the model $\mathscr{U}$ validates all axioms in Figure 7.

Proof
Similarly to Theorem 6.1, also this proof is an application of the forcing relation, using the universal property of initial algebras. Obviously, this proof is longer and more complicated since there are many subtleties to deal with as far as the mismatch operator is involved (which leads to to the $n^2 + 2n + 1$ cases of Figure 7). □

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

$$\frac{H}{\Gamma \rhd_\Sigma (R\ \lambda\vec{x}^{\upsilon}.0) =^{\sigma} f_1} \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_1)$$

$$\frac{H}{\Gamma \rhd_\Sigma \forall P^{\upsilon^n \to \iota}.(R\ \lambda\vec{x}^{\upsilon}.\tau.(P\ \vec{x})) =^{\sigma} (f_2\ (R\ P))} \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_2)$$

$$\frac{H}{\Gamma \rhd_\Sigma \forall P^{\upsilon^n \to \iota}.\forall Q^{\upsilon^n \to \iota}.(R\ \lambda\vec{x}^{\upsilon}.(P\ \vec{x})|(Q\ \vec{x})) =^{\sigma} (f_3\ (R\ P)\ (R\ Q))} \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_3)$$

$$\frac{H}{\Gamma \rhd_\Sigma \forall y^{\upsilon}.\forall z^{\upsilon}.\forall P^{\upsilon^n \to \iota}.(R\ \lambda\vec{x}^{\upsilon}.[y \neq z](P\ \vec{x})) =^{\sigma} (f_{41}\ y\ z\ (R\ P))} \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_{41})$$

$$\frac{H}{\Gamma \rhd_\Sigma \forall y^{\upsilon}.\forall P^{\upsilon^n \to \iota}.(R\ \lambda\vec{x}^{\upsilon}.[y \neq x_j](P\ \vec{x})) =^{\sigma} (f_{42j}\ y\ (R\ P))}\ j = 1 \ldots n \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_{42j})$$

$$\frac{H}{\Gamma \rhd_\Sigma \forall z^{\upsilon}.\forall P^{\upsilon^n \to \iota}.(R\ \lambda\vec{x}^{\upsilon}.[x_i \neq z](P\ \vec{x})) =^{\sigma} (f_{43i}\ z\ (R\ P))}\ i = 1 \ldots n \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_{43i})$$

$$\frac{H}{\Gamma \rhd_\Sigma \forall P^{\upsilon^n \to \iota}.(R\ \lambda\vec{x}^{\upsilon}.[x_i \neq x_j](P\ \vec{x})) =^{\sigma} (f_{44ij}\ (R\ P))}\ i, j = 1 \ldots n \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_{44ij})$$

$$\frac{H}{\Gamma \rhd_\Sigma \forall P^{\upsilon^{n+1} \to \iota}.(R\ \lambda\vec{x}^{\upsilon}.(\nu\lambda y^{\upsilon}(P\ y\ \vec{x}))) =^{\sigma} (f_5\ \lambda y^{\upsilon}.(R\ (P\ y)))} \qquad (Rec_{\sigma}^{\upsilon^n \to \iota}\_red_5)$$

where H is a typographic shorthand for the following hypotheses

$$\Gamma \vdash f_1 : \sigma \quad \Gamma \vdash f_2 : \sigma \to \sigma \quad \Gamma \vdash f_3 : \sigma \to \sigma \to \sigma \quad \Gamma \vdash f_{41} : \upsilon \to \upsilon \to \sigma \to \sigma$$
$$\Gamma \vdash f_{42j} : \upsilon \to \sigma \to \sigma \quad \Gamma \vdash f_{43i} : \upsilon \to \sigma \to \sigma \quad \Gamma \vdash f_{44ij} : \sigma \to \sigma \quad (i, j = 1 \ldots n)$$

and R is a typographic shorthand for

$$(Rec_{\sigma}^{\upsilon^n \to \iota}\ f_1\ f_2\ f_3\ f_{41}\ f_{421} \ldots f_{42n}\ f_{431} \ldots f_{43n}\ f_{4411} \ldots f_{44nn}\ f_5)$$

Fig. 7. Reduction rules for second-order recursion.

$$\frac{\Gamma \vdash_\Sigma R : \iota \to o}{\begin{array}{l}\Gamma \rhd_\Sigma (R\ 0) \Rightarrow (\forall P^{\iota}.(R\ P) \Rightarrow (R\ \tau.P)) \Rightarrow \\ \quad (\forall P^{\iota}.(R\ P) \Rightarrow \forall Q^{\iota}.(R\ Q) \Rightarrow (R\ P|Q)) \Rightarrow \\ \quad (\forall y^{\upsilon}.\forall z^{\upsilon}.\forall P^{\iota}.(R\ P) \Rightarrow (R\ [y \neq z]P)) \Rightarrow \\ \quad (\forall P^{\upsilon \to \iota}.(\forall x^{\upsilon}.(R\ (P\ x))) \Rightarrow (R\ \nu P)) \Rightarrow \\ \quad \forall P^{\iota}.(R\ P)\end{array}} \qquad (Ind^{\iota})$$

Fig. 8. First-order induction principle.

6.3 First-order induction

The first-order induction principle we consider is presented in Figure 8.

Since the model $\mathscr{U}$ does not support the "proposition-as-types, proofs-as-λ-terms" interpretation, induction principles do not derive automatically from the recursion principles in Section 6.1. A problem we have to deal with, is the presence of parameters, represented by the environment Γ. Actually, induction with parameters in $\check{\mathscr{V}}$ can be recovered from the initial algebra property in a simple slice category defined from $\check{\mathscr{V}}$ (Definition A.4). In fact the signature functor T in $\check{\mathscr{V}}$ can be "transferred" in this category, so that it has an initial algebra corresponding to the initial algebra in $\check{\mathscr{V}}$ (Jacobs, 1995).

For the sake of simplicity, and without loss of generality, in the following we consider $\Gamma = R : \iota \to o$, where R is the predicate over terms, appearing in the induction principle.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

We proceed as follows. We will work in the simple slice category $\check{\mathscr{V}}/\!\!/G$, where $G \triangleq Proc \Rightarrow Prop$; over this category we will consider the functor $T/\!\!/G$ where $T : \check{\mathscr{V}} \longrightarrow \check{\mathscr{V}}$ is the signature functor defined in Section 6.1. We will prove that $(Proc, G^*(\alpha))$ is an initial $T/\!\!/G$-algebra. Then, the soundness of the induction principle will derive from a usual argument in the category $\check{\mathscr{V}}/\!\!/G$.

To prove the main statement, we need the following two results:

Proposition 6.3

The functor T is strong.

Proof

The strength $(st_{A,B})_X : A_X \times (TB)_X \longrightarrow (T(A \times B))_X$ is defined as follows

$$\begin{aligned}
(st_{A,B})_X(a, in_1(*)) &\triangleq in_1(*)\\
(st_{A,B})_X(a, in_2(b)) &\triangleq in_2(\langle a, b\rangle)\\
(st_{A,B})_X(a, in_3(\langle b_1, b_2\rangle)) &\triangleq in_3(\langle a, b_1, a, b_2\rangle)\\
(st_{A,B})_X(a, in_4(\langle x, y, b\rangle)) &\triangleq in_4(\langle x, y, a, b\rangle)\\
(st_{A,B})_X(a, in_5(b)) &\triangleq in_5(\overline{b}_a)
\end{aligned}$$

where $\overline{b}_a \in \check{\mathscr{V}}(Var \times \mathscr{V}(X, \_), A \times B)$ is the natural transformation such that $(\overline{b}_a)_Y(\langle y, g\rangle) \triangleq \langle a[g], b_Y(\langle y, g\rangle)\rangle)$. The commutativity of the two diagrams of Definition A.5 is proved by cases over b (see Appendix B.8). □

Proposition 6.4

For every $G \in \check{\mathscr{V}}$, $(Proc, G^*(\alpha))$ is an initial $T/\!\!/G$-algebra.

Proof

Since $\alpha \in \check{\mathscr{V}}(TProc, Proc)$, $G^*(\alpha) \in \check{\mathscr{V}}/\!\!/G((T/\!\!/G)_{Proc}, Proc)$, i.e., $(Proc, G^*(\alpha))$ is a $T/\!\!/G$-algebra. It remains to show that, given any other $T/\!\!/G$-algebra (B, β), there is a unique morphism f from $Proc$ to B such that the following diagram in $\check{\mathscr{V}}/\!\!/G$ commutes:

$$\begin{array}{ccc}
(T/\!\!/G)Proc & \xrightarrow{G^*(\alpha)} & Proc\\
{\scriptstyle (T/\!\!/G)_f}\downarrow & & \downarrow{\scriptstyle f}\\
(T/\!\!/G)B & \xrightarrow[\beta]{} & B
\end{array}$$

Notice that the same diagram can be read in $\check{\mathscr{V}}$ as follows:

$$\begin{array}{ccc}
G \times TProc & \xrightarrow{\mathrm{id}_G \times \alpha} & G \times Proc\\
{\scriptstyle \langle \pi, Tf \circ st_{G,Proc}\rangle}\downarrow & & \downarrow{\scriptstyle f}\\
G \times TB & \xrightarrow[\beta]{} & B
\end{array}$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

We define f as follows:

$$\begin{aligned}
f_X(\langle g, 0\rangle) &\triangleq \beta_X(\langle g, in_1(*)\rangle)\\
f_X(\langle g, \tau.P\rangle) &\triangleq \beta_X(\langle g, in_2(f_X(\langle g, P\rangle))\rangle)\\
f_X(\langle g, P_1|P_2\rangle) &\triangleq \beta_X(\langle g, in_3(\langle f_X(\langle g, P_1\rangle), f_X(\langle g, P_2\rangle)\rangle)\rangle)\\
f_X(\langle g, [x \neq y]P\rangle) &\triangleq \beta_X(\langle g, in_4(\langle x, y, f_X(\langle g, P\rangle)\rangle)\rangle)\\
f_X(\langle g, (\nu x)P\rangle) &\triangleq \beta_X(\langle g, in_5(\gamma_{B,X}(f_{X\uplus\{x\}}(\langle g[in_X], P\rangle))\rangle)
\end{aligned}$$

Commutativity of the previous diagram is proved by cases on P (Appendix B.9).

The uniqueness of f follows by its definition. Given any other homomorphism $g : (Proc, G^*(\alpha)) \longrightarrow (B, \beta)$ such that $g \bullet G^*(\alpha) = \beta \bullet (T /\!\!/ G)_g$, rewriting the previous equality and simplifying it according to the definitions of $G^*(\alpha)$ and $st_{G,Proc}$ we obtain exactly the definition by cases of f. □

Now we have the necessary tools for proving our goal:

Theorem 6.4

The model $\mathscr{U}$ validates Ind^ι, i.e., the following holds:

$$\begin{aligned}
\emptyset \rhd_\Sigma \quad & \forall R^{\iota\to o}.((R\ 0) \Rightarrow (\forall P^\iota.(R\ P) \Rightarrow (R\ \tau.P)) \Rightarrow\\
& (\forall P^\iota.(R\ P) \Rightarrow \forall Q^\iota.(R\ Q) \Rightarrow (R\ P|Q)) \Rightarrow\\
& (\forall y^\upsilon.\forall z^\upsilon.\forall P^\iota.(R\ P) \Rightarrow (R\ [y \neq z]P)) \Rightarrow\\
& (\forall P^{\upsilon\to\iota}.(\forall x^\upsilon.(R\ (P\ x))) \Rightarrow (R\ \nu P)) \Rightarrow\\
& \forall P^\iota.(R\ P))
\end{aligned}$$

Proof

By Proposition 4.1, we have to prove that for all $X \in \mathscr{V}$ the following holds:

$$\begin{aligned}
X \Vdash_{\emptyset,*} \quad & \forall R^{\iota\to o}.((R\ 0) \Rightarrow (\forall P^\iota.(R\ P) \Rightarrow (R\ \sigma.P)) \Rightarrow\\
& (\forall P^\iota.(R\ P) \Rightarrow \forall Q^\iota.(R\ Q) \Rightarrow (R\ P|Q)) \Rightarrow\\
& (\forall y^\upsilon.\forall z^\upsilon.\forall P^\iota.(R\ P) \Rightarrow (R\ [y \neq z]P)) \Rightarrow\\
& (\forall P^{\upsilon\to\iota}.(\forall x^\upsilon.(R\ (P\ x))) \Rightarrow (R\ \nu P)) \Rightarrow\\
& \forall P^\iota.(R\ P))
\end{aligned}$$

By Theorem 4.1, this is equivalent to prove that, under the following assumptions

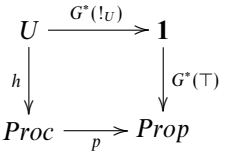
$$\begin{aligned}
& Y \Vdash_{R:\iota\to o,\eta_R} (R\ 0),\\
& Y \Vdash_{R:\iota\to o,\eta_R} (\forall P^\iota.(R\ P) \Rightarrow (R\ \tau.P)),\\
& Y \Vdash_{R:\iota\to o,\eta_R} (\forall P^\iota.(R\ P) \Rightarrow \forall Q^\iota.(R\ Q) \Rightarrow (R\ P|Q)),\\
& Y \Vdash_{R:\iota\to o,\eta_R} (\forall y^\upsilon.\forall z^\upsilon.\forall P^\iota.(R\ P) \Rightarrow (R\ [y \neq z]P)),\\
& Y \Vdash_{R:\iota\to o,\eta_R} (\forall P^{\upsilon\to\iota}.(\forall x^\upsilon.(R\ (P\ x))) \Rightarrow (R\ \nu P)),
\end{aligned}$$

we have that for all $Z \in \mathscr{V}$, for all $f \in \mathscr{I}(Y, Z)$ and for all $\eta_P \in Proc_Z$, the judgment $Z \Vdash_{(R:\iota\to o,P:\iota),\langle(Proc\Rightarrow Prop)_f(\eta_R),\eta_P\rangle} (R\ P)$ holds. This fact amounts to say that the following equation must hold:

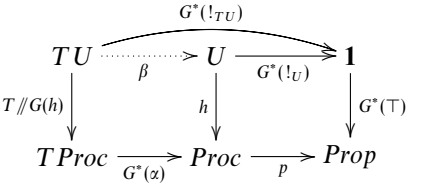
$$p \triangleq [\![R : \iota \to o, P : \iota \vdash_\Sigma (R\ P) : o]\!]^\dagger = \top \circ !_{[\![(R:\iota\to o,P:\iota)]\!]}. \tag{9}$$

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Consider the following pullback in $\check{\mathscr{V}}/\!\!/G$:



where $G \triangleq [\![R : \iota \to o]\!]$. Then, from the assumptions above, we have that the following diagram in $\check{\mathscr{V}}/\!\!/G$ commutes (see Appendix B.10):



Let $\beta : TU \to U$ be the unique map defined by the universal property of the pullback. Then, (U, β) is a $T/\!\!/G$-algebra; therefore, by initiality of *Proc* (existential part) there is a map $h' \in \check{\mathscr{V}}/\!\!/G(Proc, U)$. Moreover, again by initiality of *Proc* (unicity part) we have $h \bullet h' = G^*(\mathrm{id}_{Proc})$. Hence we have the following:

$$p = p \bullet G^*(\mathrm{id}_{Proc}) = p \bullet h \bullet h' = G^*(\top) \bullet G^*(!_U) \bullet h'$$

Translating the equation in terms of the composition in the category $\check{\mathscr{V}}$, we get

$$p = G^*(\top) \circ \langle \pi, G^*(!_U) \circ \langle \pi, h' \rangle \rangle = \top \circ !_U \circ h' = \top \circ !_{G \times Proc}$$

i.e., the thesis (9). □

6.4 Second-order induction

As in the case of recursion, also the induction principle can be generalized to second-order processes. The second-order induction principle is given in Figure 9. Notice that in the case of $n = 0$, this rule degenerates in that for first-order terms introduced above.

The proofs of the validity of second-order induction principles follow the same pattern of the first-order case, exploiting the initiality of the corresponding second-order initial algebra (see section 6.2) and the following result which extends Lemma 7.8 of Jacobs (1995) to the exponentiation of functors in the category $\check{\mathscr{V}}$:

Lemma 6.1
If $T : \check{\mathscr{V}} \longrightarrow \check{\mathscr{V}}$ is strong, then the functor $Var \Rightarrow T$ (whose action on objects is $A \mapsto Var \Rightarrow TA$) is strong as well.

Proof
Let *st* be the strength of T; we define (up-to suitable isomorphisms) the strength st' for $Var \Rightarrow T$ as $(st'_{A,B})_X \triangleq (st_{A,B})_{X \uplus \{x\}} \circ (A_{in} \times id)$, where $X \overset{in}{\hookrightarrow} X \uplus \{x\}$ is the

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

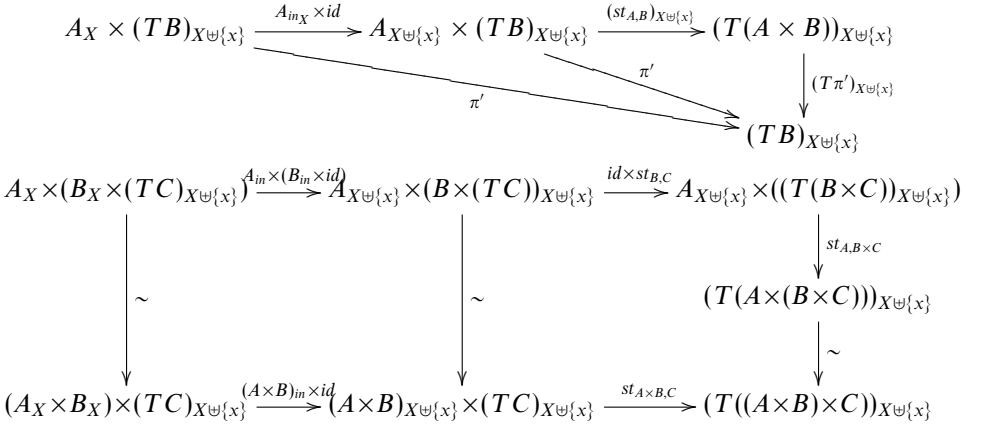
$$\frac{\Gamma \vdash_\Sigma R : (\upsilon^n \to \iota) \to o}{\begin{array}{l}\Gamma \rhd_\Sigma (R\ \lambda\vec{x}^\upsilon.0) \Rightarrow (\forall P^{\upsilon^n\to\iota}.(R\ P) \Rightarrow (R\ \lambda\vec{x}^\upsilon.(\tau.(P\ \vec{x})))) \Rightarrow \\ \quad (\forall P^{\upsilon^n\to\iota}.(R\ P) \Rightarrow \forall Q^{\upsilon^n\to\iota}.(R\ Q) \Rightarrow (R\ \lambda\vec{x}^\upsilon.(P\ \vec{x})|(Q\ \vec{x}))) \Rightarrow \\ \quad (\forall P^{\upsilon^n\to\iota}.\forall y^\upsilon.\forall z^\upsilon.(R\ P) \Rightarrow (R\ \lambda\vec{x}^\upsilon.[y \neq z](P\ \vec{x}))) \Rightarrow \\ \quad (\forall P^{\upsilon^n\to\iota}.\forall z^\upsilon.(R\ P) \Rightarrow \bigwedge_{i=1}^n (R\ \lambda\vec{x}^\upsilon.[x_i \neq z](P\ \vec{x}))) \Rightarrow \\ \quad (\forall P^{\upsilon^n\to\iota}.\forall y^\upsilon.(R\ P) \Rightarrow \bigwedge_{j=1}^n (R\ \lambda\vec{x}^\upsilon.[y \neq x_j](P\ \vec{x}))) \Rightarrow \\ \quad (\forall P^{\upsilon^n\to\iota}.(R\ P) \Rightarrow \bigwedge_{i,j=1}^n (R\ \lambda\vec{x}^\upsilon.[x_i \neq x_j](P\ \vec{x}))) \Rightarrow \\ \quad (\forall P^{\upsilon^{n+1}\to\iota}.(\forall y^\upsilon.(R\ \lambda\vec{x}^\upsilon.(P\ \vec{x}\ y))) \Rightarrow (R\ \lambda\vec{x}^\upsilon.v(P\ \vec{x}))) \Rightarrow \\ \quad \forall P^{\upsilon^n\to\iota}.(R\ P)\end{array}} \quad (Ind^{\upsilon^n\to\iota})$$

Fig. 9. Second-order induction principle.

obvious injection. More explicitly, for $A, B \in \check{\mathscr{V}}$ and $X \in Var$:

$$\begin{array}{rrcl} (st'_{A,B})_X : & A_X \times (Var \Rightarrow TB)_X & \longrightarrow & (Var \Rightarrow TA \times B)_X \\ (st'_{A,B})_X : & A_X \times (TB)_{X\uplus\{x\}} & \longrightarrow & (T(A \times B))_{X\uplus\{x\}} \\ & \langle a, b\rangle & \longmapsto & (st_{A,B})_{X\uplus\{x\}}(a[in], b) \end{array}$$

It is easy to check that st' is a strength for the functor $Var \Rightarrow T$, that is, the following diagrams commute:



In the latter diagram, the left square is the naturality of the associativity isomorphism of product, and the right part is the property of the strength *st*. □

6.5 An abstract view on initial algebras and induction

We end this section with a more abstract hence shorter account of the results in section 6.3. First, recall Theorem A.1: initial T-algebras can be lifted along functors having a right adjoint. Our aim is to extend this result to induction principles in an arbitrary tripos.

Definition 6.1

Let **Pred** be a tripos on some category $\mathscr{C}$ and let $T : \mathscr{C} \to \mathscr{C}$ be an endofunctor. An *action of T on predicates* consists of a Heyting algebra morphism T : **Pred**$(A) \to$ **Pred**(TA) for each object A compatible with substitution, i.e., if $f : A \to B$ and $P \in$ **Pred**(B) then $T($**Pred**$_f(P)) =$ **Pred**$_{Tf}(TP)$.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

An initial T-algebra (A, α) for a functor T with an action on predicates *has induction* if for each $P \in \mathbf{Pred}(A)$ one has $TP \leqslant P[\alpha]$ implies $P = \top$.

Consider, for example the case where $\mathscr{C} = \mathscr{S}et, \mathbf{Pred}(A) = \wp(A)$ and $TX = 1 + X$ and $TP = \{\mathrm{inl}(\star)\} \cup \mathrm{inr}(P)$. The initial algebra is the set of natural numbers and induction coincides with the usual one.

To define an action of T on predicates is tantamount to lifting the functor T to the category which has as objects pairs (A, P) with $P \in \mathbf{Pred}(A)$ and where a morphism from (A, P) to (B, Q) is a morphism $f : A \to B$ such that $P \leqslant Q[f]$. The latter category is the *total category* of the fibration associated with **Pred**. The fact that initial algebra $\alpha : TA \to A$ has induction then amounts to $(A, \top)$ being an initial algebra in the total category.

Thus, applying Theorem A.1 to the total category allows us to transport induction principles in the same way as recursion principles.

7 Related work

7.1 Semantics based on functor categories

The application of functor categories in the semantics of programming languages goes back to the early 1980s, when "variable" sets (i.e., objects of $\check{\mathscr{C}}$) were recognized as a useful tool to model variability of memory allocation in Algol-like languages (Reynolds, 1981; Oles, 1985). An important step towards the generalization of this approach has been the *monad for allocation* over the category $\check{\mathscr{C}}$ (Moggi, 1989). More recently, presheaf models have been extensively used for interpreting concurrency and mobility (Stark, 1996; Fiore *et al.*, 1996; Cattani *et al.*, 1997).

Recently, the use of functor categories as a semantics for HOAS has been advocated (Fiore *et al.*, 1999; Hofmann, 1999), the latter being the basis for the present paper. At the same time, an alternative approach based on Frænkel-Mostowski set theory has been presented (Gabbay & Pitts, 1999). Here we briefly illustrate the connections between these models.

In Hofmann (1999), functor categories are used for formally justifying several logical principles which have been previously proposed for reasoning about HOAS. In particular, metalanguage types are interpreted as suitable objects of $\mathscr{S}et^{\mathscr{C}^{op}}$, where the index category $\mathscr{C}$ depends on the nature of the metalanguage. A key feature of this approach (which we have exploited in Proposition 3.4) is that the interpretation of types which appear in negative position in the types of syntactic constructors must be representable. This allows to apply the following property:

$$\text{for } X \in \mathscr{C} \text{ and } F \in \mathscr{S}et^{\mathscr{C}^{op}} : (\mathscr{Y}(X) \Rightarrow F)_Y \cong F_{X+Y} .$$

For instance, in the case of untyped λ-calculus, whose syntax is defined by

$$\begin{aligned} \text{Inductive } tm : Set := {} & isvar : var \to tm \mid app : var \times var \to tm \\ & \mid lam : (var \to tm) \to tm. \end{aligned}$$

the second-order type $var \to tm$ is interpreted as the functor $Var \Rightarrow Tm$. Since $Var \cong \mathscr{Y}(\{x\})$, we have that $(Var \Rightarrow Tm)_X \cong Tm_{X \uplus \{x\}}$. In other words, functions

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

over variables correspond exactly to terms with an extra variable, which can be seen as the "hole" of the syntactic context. Thus, the interpretation of *tm* is the initial algebra of the functor $T(A) = Var + A \times A + Var \Rightarrow A$.

On the other hand, one cannot use the plain topos $\mathcal{S}et^{\mathcal{C}^{op}}$ for interpreting predicates, because the induction principles over second-order types contradict the Axiom of Unique Choice. The solution originally conceived in Hofmann (1999), and which have been fully developed in this work, is to resort to some tripos over the category of types.

Covariant presheaves are adopted in Fiore *et al.* (1999), where a general methodology is developed in order to associate to every *binding signature* a category of models which gives a notion of initial algebra semantics. The models are presheaves which are both algebras for the signature functor and monoids with respect to substitution. The choice of $\mathcal{F}$ (that is, the skeleton of the category of finite sets and functions) as the index category is motivated by the operations which are allowed on environments: name swapping, contraction and weakening. Indeed, the closure by composition of these operations generates exactly all the functions between finite cardinals. A key feature of the category $\mathcal{S}et^{\mathcal{F}}$ is that it has a type constructor

$$\delta : \mathcal{S}et^{\mathcal{F}} \to \mathcal{S}et^{\mathcal{F}} \qquad (\delta A)_X = A_{X+1} \quad (\delta A)_h = A_{h+\mathrm{id}_1}$$

which is used for interpreting second-order types like $var \to tm$ of the previous example. Thus, the interpretation of *tm* is the initial algebra of the functor $T(A) = Var + A \times A + \delta A$. Clearly, δ corresponds to the functor $Var \Rightarrow \_$ in Hofmann's approach, via the isomorphism previously described.

However, since $\check{\mathcal{F}}$ alone is proposed as a framework for higher-order abstract syntax, this work is fine for the purely algebraic aspect, i.e., terms and equations; as we have seen, in order to *reason* about HOAS, $\check{\mathcal{F}}$ alone is inadequate since for example equality of names cannot be expressed. The value of Fiore *et al.* (1999) is to have placed inductive types like *Proc* in $\check{\mathcal{V}}$ in the context of universal algebra.

A different perspective is taken in Gabbay & Pitts (1999, 2002), where a logic for specifying and reasoning about formal systems with name binders is introduced using as a semantic basis the Frænkel-Mostowski permutation model of set theory with atoms. (We will discuss briefly this logic in section 7.2 below.) This model has a specific set of "atoms" $\mathbb{A}$, and each set X is endowed with a primitive "atom-swapping" operation $(\_\ \_)\cdot\_ : \mathbb{A} \times \mathbb{A} \times X \to X$. All usual constructions of set theory (product, coproduct, function space,...) must respect this swapping operation, that is an atom appearing in an object can be safely replaced with another (fresh) atom without altering the behaviour of the object. (In particular, functions must be *equivariant*, that is they must commute with atom permutations.) Therefore, given an object x (of some FM-set X) and an atom a, the *abstraction* $a.x$ can be defined as the equivalence class of pairs (b, y) such that $(c\ a)\cdot x = (c\ b)\cdot y$ for any c fresh for x and y. The dual operation of *instantiation* $x@b$ is then the choice of a specific representative of the class, namely the one with the particular b in place of the abstracted atom. These notions of abstraction and instantiations are well-defined in virtue of the equivariance restriction of FM-sets.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

By gathering all the abstractions $a.x$ for $a \in \mathbb{A}$ and $x \in X$, we get the a new set denoted by $[\mathbb{A}]X$ and called the set of *abstractions* of elements of X. This new construction, specific of FM-set theory, is the key for interpreting binding operators in signatures: if X is the interpretation of some syntactic class, $[\mathbb{A}]X$ is the interpretation of contexts of type X. In our running example of λ-calculus, the interpretation of *tm* is the FM-set defined as the least fixed point of the (FM-set valued) function $F_\alpha(X) = \mathbb{A} + X \times X + [\mathbb{A}]X$. In other words, the quotient with respect to α-equivalence is applied only to the interpretation of binding constructors, instead of being applied to the whole initial algebra (like in the case of a pure first-order syntax approach). As a consequence, in this approach the usual arguments about least fixed points in set theory can be applied for deriving induction and recursion principles over the higher-order abstract syntax.

To highlight the close connection between this latter approach and the previous ones, notice that the universe of sets used in (Gabbay & Pitts, 1999) is the category of the perm($\mathbb{A}$)-sets with finite support and equivariant functions. As the authors of that work point out, this category is equivalent to the Schanuel topos, that is, the category of sheaves over $\mathscr{I}^{op}$ for the $\neg\neg$-topology § III.9, which, as we have noticed in section 5, is the topos we have used for the interpretation of logical judgments. Both $Sh_{\neg\neg}(\check{\mathscr{I}})$ and the Schanuel topos embed in $\check{\mathscr{I}}$, which is related to $\check{\mathscr{V}}$ by the adjunction of Proposition 3.1. The reason we could not use $Sh_{\neg\neg}(\check{\mathscr{I}})$ to interpret terms and functions and resorted to $\check{\mathscr{V}}$ instead was that datatypes with second-order constructors (such as $\nu : (\upsilon \to \iota) \to \iota$) would not be inductive.

A final remark is about the peculiar behaviour of the interpretation of abstraction and instantiation in (Gabbay & Pitts, 1999). In our approach both can be rendered naturally using the features of the metalanguage: the first as λ-abstraction, the latter as application. On the other hand, notice that instantiation $x@a$ in FM-set theory is only partially defined, *i.e.*, when a is not in the support of x, *i.e.*, the free variables of x. Actually, the abstraction set of FM has a clear corresponding in our categorial setting. Recall that in our model, the type constructor "$\upsilon \to \_$" is interpreted exactly as the exponentiation $Var \Rightarrow \_ : \check{\mathscr{V}} \longrightarrow \check{\mathscr{V}}$ (Section 3.2). The corresponding operation in $\check{\mathscr{I}}$ via the adjunction (*i.e.*, the restriction) is not the usual exponentiation of $\check{\mathscr{I}}$, but only a certain arrow $Var \multimap \_ : \check{\mathscr{I}} \longrightarrow \check{\mathscr{I}}$ which is the right adjunct of a suitable tensor product. This arrow corresponds exactly to the exponent in the Schanuel topos, and ultimately it corresponds to $[\mathbb{A}]\_$ of FM. Using this exponentiation has the advantage that we can interpret both terms and proposition in the very same Schanuel topos, where (as in any topos) the Axiom of Unique Choice holds and thus it can be employed consistently.

Another difference between the present work and Gabbay & Pitts (2002) is that the metalanguage we adopt is ordinary higher-order logic; this means that the Theory of Contexts can be consistently used in our preferred logic and implementation (as long as it does not enforce the Axiom of Unique Choice), such as the Calculus of Constructions (implemented in Coq). No extensions to the syntax other than constants and axioms, and no modification of the system are required; in particular, we do not need a new kind of quantifier.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

7.2 Logics for nominal calculi

The Theory of Contexts and Isabelle/HOL. The Theory of Contexts can be used in many different logical frameworks in order to reason about higher-order abstract syntax. A HOAS-based encoding of the syntax of π-calculus processes in Isabelle/HOL is given in Röckl *et al.* (2001). For types of the form $\upsilon^n \to \iota$, inductively defined well-formedness predicates delineate members that correspond to terms with free names in the object syntax. An instantiation of the axioms of the Theory of Contexts, suitably adapted for their case study, can then be proved by induction on the definition of these well-formedness predicates.

In particular, this allows for the axioms to be proved within the theory, i.e., no non-standard interpretation of the logic is required to establish soundness. On the other hand, for each term in question one first has to assert well-formedness which in view of the rules defining well-formedness is tantamount to giving the term in de Bruijn notation.

The Nominal Logic. A first order logic for reasoning about languages with binders has been proposed in Gabbay & Pitts (1999) and later extended and called *Nominal Logic* in Pitts (2003). This logic is based on the Frænkel-Mostowski permutation model of set theory, which we have already discussed in section 7.1 above. Nominal Logic features a specific sort ν of "atoms" representing variables names, and for each type τ a primitive *atom-swapping operation* $swap_\tau : \nu \to \nu \to \tau \to \tau$. The axioms of the logic compel terms, functions and propositions to be *equivariant*, that is "stable" under atom permutations. This means that an atom a appearing in a term t can be safely replaced by any other (fresh) atom, without affecting the behaviour (meaning) of t. The equivariance constraint applies also to propositions; namely, a proposition which holds for some fresh atom, will hold for *any* fresh atom. This observation leads to the introduction of a special quantifier И for expressing freshness of names: the intuitive meaning of И$a.p$ is "p holds for a some/any fresh name". И resembles both $\forall$ and $\exists$, it satisfies the rules:

$$\frac{\Gamma, a\#\vec{b} \vdash p}{\Gamma \vdash И a.p} \qquad \frac{\Gamma \vdash И a.p \quad \Gamma, p, a\#\vec{b} \vdash q}{\Gamma \vdash q}$$

where $\vec{b}$ is the "support" (i.e., the set of "free names") of p, and $\#$ is an atomic predicate stating *freshness* of atoms with respect to terms.

Essentially, the main difference with our approach is that in the Theory of Contexts, terms with fresh names are modelled as functions $\upsilon \to \iota$, whereas in Nominal Logic they are modelled as equivalence classes of name-term pairs. Predicates as И$a.p$ and $a\#\vec{b}$ can be translated in the Theory of Contexts as follows:

$$И a.p \triangleq \forall a^\upsilon.a \notin^{\upsilon\to o} (\lambda a^\upsilon.p) \Rightarrow p \qquad a\#\vec{b} \triangleq a \notin^o p$$

Rules, corresponding to the ones above, can then be easily derived using the Theory of Contexts. Correspondingly, suitable adaptations of our Theory of Contexts are validated in the FM.

In a nutshell one can say that our approach works in the standard setting of higher-order logic and type theory, whereas for FM new syntactic constructs are

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

needed. On the other hand, FM has the advantage that axioms about π-calculus can be derived from more primitive concepts so that it would more easily carry over to different settings.

Meta-metalogics. In the approaches we discussed so far, the logical level belongs to the same metalanguage which is used for the representation of the syntax. A different perspective is to add explicitly an extra logical level for reasoning over metalogics. One of these meta-metalogic is $FO\lambda^{\Delta N}$ (McDowell & Miller, 2002), a higher-order intuitionistic logic extended with definitions and higher order quantification over simply typed λ-terms. Induction on types is recovered from induction on natural numbers via appropriate notions of measure.

A similar approach, but with different aims, is behind $\mathscr{M}_2$ (Pfenning & Schürmann, 1999), which is a constructive first-order logic based on the Edinburgh LF and implemented in *Twelf*. At the meta-metalevel, $\mathscr{M}_2$ offers higher-order induction and recursion for reasoning over (possibly open) objects of a LF encoding.

8 Conclusions

In this work we have proved the consistency of the Theory of Contexts, working out in full detail the constructions of a categorical model based on a tripos on functor categories. The technical machinery we have presented should be suitable for reasoning about models with a similar structure. In our opinion, this construction could be adopted for validating other theories of names and binders.

The first important application of the consistency of the Theory of Contexts is that it can be safely embedded in existing logical frameworks (as far as their logics do not entail the Axiom of Unique Choice). For instance, this theory has been used fruitfully for developing the (meta)theory of several object languages in the proof assistant Coq (INRIA, 2003); see Honsell *et al.* (2001b) and Miculan (2001) for the case of π-calculus and λ-calculus, respectively.

At least two possible developments are stemming from this work. An open question concerns *completeness* of the Theory of Contexts. It is not clear which class of properties can be derived from our axioms; a suitable characterization is needed. Another direction is to extend the model in order to handle more expressive metalanguages. For instance, one could take into account a theory of *dependent* and *impredicative types*. The expressive power of such a metalanguage would allow the representation and the manipulation of *proof objects*, via the usual "propositions-as-types" paradigm. An example of object theories which could be dealt with in this case are Natural Deduction-style proof systems; then, the well-known *Inversion Lemma* could be proved by induction over proof objects, using (a suitable extension of) the Theory of Contexts.

Acknowledgments

We thank an anonymous referee for useful remarks and hints on the preliminary version. The first author is grateful to Pino Rosolini for useful conversations about tripos theory.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

References

Ambler, S., Crole, R. and Momigliano, A. (eds). (2001) *Mechanized reasoning about languages with variable binding*. Electronic Notes in Theoretical Computer Science, vol. 58.1. Elsevier.

Barr, M. and Wells, C. F. (1999) *Category theory for computing science*. Les Publications CMR.

Bucalo, A., Hofmann, M., Honsell, F., Miculan, M. and Scagnetto, I. (2005) *Appendices of "Consistency of the Theory of Contexts"*. Available online at the *Journal of Functional Programming* web site.

Cattani, G. L., Stark, I. and Winskel, G. (1997) Presheaf models for the π-calculus. *Proc. CTCS*.

Church, A. (1940) A formulation of the simple theory of types. *Journal of Symbolic Logic*, **5**, 56–68.

Crole, R. L. (2003) Basic category theory for models of syntax. *Pages 133–177 of:* Backhouse, R. C. and Gibbons, J. (eds), *Generic Programming*. Lecture Notes in Computer Science, vol. 2793. Springer.

Despeyroux, J., Felty, A. and Hirschowitz, A. (1995) Higher-order syntax in Coq. *Proc. of tlca'95*. Lecture Notes in Computer Science, vol. 905. Springer-Verlag.

Fiore, M. and Turi, D. (2001) Semantics of name and value passing. *Pages 93–104 of:* Mairson, H. (ed), *Proc. 16th LICS*. Boston, USA: IEEE.

Fiore, M., Moggi, E. and Sangiorgi, D. (1996) A fully-abstract model for the π-calculus. *Proc. 11th LICS*. IEEE.

Fiore, M. P., Plotkin, G. D. and Turi, D. (1999) Abstract syntax and variable binding. In: Longo, G. (ed.), *Proceedings, 14th Annual IEEE Symposium on Logic in Computer Science*. IEEE.

Gabbay, M. J. and Pitts, A. M. (1999) A new approach to abstract syntax involving binders. In: Longo, G. (ed.), *Proceedings, 14th Annual IEEE Symposium on Logic in Computer Science*. IEEE.

Gabbay, M. J. and Pitts, A. M. (2002) A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, **13**, 341–363.

Harper, R., Honsell, F. and Plotkin, G. (1993) A framework for defining logics. *Journal of the ACM*, **40**(1), 143–184.

Hofmann, M. (1999) Semantical analysis of higher-order abstract syntax. In: Longo, G. (ed.), *Proceedings, 14th Annual IEEE Symposium on Logic in Computer Science*. IEEE.

Honsell, F., Miculan, M. and Scagnetto, I. (2001a) An axiomatic approach to metareasoning on systems in higher-order abstract syntax. *Pages 963–978 of: Proc. ICALP'01*. Lecture Notes in Computer Science, vol. 2076. Springer-Verlag.

Honsell, F., Miculan, M. and Scagnetto, I. (2001b) π-calculus in (co)inductive type theory. *Theoretical Computer Science*, **253**(2), 239–285.

Hyland, J. M. E., Johnstone, P. T. and Pitts, A. M. (1980) Tripos theory. *Math. Proc. Cambridge Philos. Soc.*, **88**, 205–232.

INRIA (2003) *The Coq proof assistant*. `http://coq.inria.fr/doc/main.html` .

Jacobs, B. (1995) Parameters and parametrization in specification using distributive categories. *Fund. Informaticae*, **24(3)**.

Jacobs, B. (1999) *Categorical logic and type theory*. Studies in Logic and the Foundations of Mathematics, vol. 141. Elsevier.

Johnstone, P. (1977) *Topos theory*. London Mathematical Society Monographs, no. 10. London: Academic Press.

Longo, G (ed.) (1999) *Proceedings, 14th Annual IEEE Symposium on Logic in Computer Science*. IEEE.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press

Mac Lane, S. (1971) *Categories for the Working Mathematician.* Springer-Verlag.

Mac Lane, S. and Moerdijk, I. (1994) *Sheaves in geometry and logic: a first introduction to topos theory*. Universitext. Springer-Verlag.

McDowell, R. and Miller, D. (2002) Reasoning with higher-order abstract syntax in a logical framework. *ACM Trans. Computational Logic*, **3**(1), 80–136.

Miculan, M. (1997) *Encoding logical theories of programs.* PhD thesis, Dipartimento di Informatica, Università di Pisa, Italy.

Miculan, M. (2001) Developing (meta)theory of lambda-calculus in the theory of contexts. In: Ambler, S., Crole, R. & Momigliano, A. (eds.), *Mechanized reasoning about languages with variable binding*. Electronic Notes in Theoretical Computer Science, vol. 58.1. Elsevier.

Milner, R., Parrow, J. and Walker, D. (1992) A calculus of mobile processes. *Inform. and Comput*. **100**(1), 1–77.

Moggi, E. (1989) *An abstract view of programming languages.* Tech. rept. ECS-LFCS-90-113. LFCS, University of Edinburgh.

Moggi, E. (1993) Notions of computation and monads. *Inform. and Comput*., **1**.

Oles, F. J. (1985) Type categories, functor categories and block structure. In: Nivat, M. and Reynolds, J. C. (eds.), *Algebraic Semantics.* Cambridge University Press.

Pfenning, F. and Elliott, C. (1988) Higher-order abstract syntax. *Pages 199–208 of: Proc. of ACM SIGPLAN '88 Symposium on Language Design and Implementation.*

Pfenning, F. and Schürmann, C. (1999) System description: Twelf — A meta-logical framework for deductive systems. *Pages 202–206 of:* Ganzinger, H. (ed.), *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*. LNAI, vol. 1632. Trento, Italy: Springer-Verlag.

Pitts, A. M. (1981) *The theory of triposes.* PhD thesis, Cambridge Univ.

Pitts, A. M. (1999) Tripos theory in retrospect. In: Birkedal, L. and Rosolini, G. (eds.), *Tutorial workshop on realizability semantics, FLoC'99,* Trento, Italy. Electronic Notes in Theoretical Computer Science, vol. 23. Elsevier.

Pitts, A. M. (2003) Nominal logic, a first order theory of names and binding. *Information and Computation*, **186**, 165–193.

Reynolds, J. C. (1981) The essence of Algol. *Pages 345–372 of:* de Bakker and van Vliet (eds.), *Algorithmic Languages. Proc. ACM Annual Conference.* North-Holland.

Röckl, C., Hirschkoff, D. and Berghofer, S. (2001) Higher-order abstract syntax with induction in Isabelle/HOL: Formalising the π-calculus and mechanizing the theory of contexts. *Pages 359–373 of:* Honsell, F. and Miculan, M. (eds.), *Proc. fossacs 2001.* LNCS, vol. 2030. Springer-Verlag.

Scagnetto, I. (2002) *Reasoning about names in higher-order abstract syntax.* PhD thesis, Dipartimento di Matematica e Informatica, Università di Udine, Italy.

Scagnetto, I. and Miculan, M. (2002) Ambient calculus and its logic in the calculus of inductive constructions. In: Pfenning, F. (ed.), *Proc. Third International Workshop on Logical Frameworks and Meta-languages (LFM'02)*. Electronic Notes in Theoretical Computer Science, vol. 70.2. Elsevier.

Stark, I. (1994) *Names and higher-order functions.* PhD thesis, University of Cambridge. Available as Technical Report 363, University of Cambridge Computer Laboratory.

Stark, I. (1996) A fully abstract domain model for the π-calculus. *Pages 36–42 of: Proc. LICS'96.* IEEE.

van Oosten, J. (1991) *Exercises in realizability*. PhD thesis, Department of Mathematics and Computer Science, University of Amsterdam.

https://doi.org/10.1017/S0956796806005892 Published online by Cambridge University Press