A presheaf model of parametric type theory

Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin

Chalmers University of Technology and University of Gothenburg
{bernardy,coquand,mouling}@chalmers.se

Abstract

We propose a new type theory with internalized parametricity. Compared to previous similar proposals, this version comes with a denotational semantics which is a refinement of the standard presheaf semantics of dependent type theory. Further, this presheaf semantics is a refinement of the one used to interpret nominal sets with restriction. The present calculus is a candidate for the core of a proof assistant with internalized parametricity.

1998 ACM Subject Classification F.4.1. Mathematical logic. Lambda calculus and related systems.

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Reynolds [1983] proved a general *abstraction theorem* (sometimes called *parametricity theorem*) about polymorphic functions. His argument is about a set theoretic semantic. As he stated it, *the underlying idea is that the meanings of an expression in "related" environments will be "related" values.* For instance, he proves that if t_X is a term of type $X \to X$ and if we consider two sets A_0, A_1 and a relation $R \subseteq A_0 \times A_1$ then we have $R([t_X]_{X=A_0}(a_0), [t_X]_{X=A_1}(a_1))$ whenever $R(a_0, a_1)$, where $[t_X]_{X=A}$ denotes the meaning of the expression t_X where X is interpreted by the set A. As he noted, one can replace in this statement binary relations by n-ary relations, and in particular unary relations (predicates). In the latter case, the statement is the following: if A is a set and P is a predicate on A, then we have $P([t_X]_{X=A}(a))$ whenever $P(a)$ holds. Wadler [1989] illustrates by many examples how this result is useful for reasoning about functional programs.

The argument and result of Reynolds are model-theoretic in nature. In lambda-calculi with dependent types, it is possible to state such an abstraction result in a purely syntactical way. One states for example that if a function f has type $(A : U) \to U \to U$ — the type of the polymorphic identity — then the following proposition holds:

$$(A : U) \to (P : A \to U) \to (x : A) \to Px \to P(f\,A\,x)$$

Indeed Bernardy et al. [2012] proves such a result as a (syntactical) meta-theorem about type systems. However this result is not provable internally, *i.e.*, the following is not provable:

$$(f : (A : U) \to A \to A) \to (A : U) \to (P : A \to U) \to (x : A) \to Px \to P(f\,A\,x)$$

Several attempts have been made [Bernardy and Moulin, 2012, 2013] — or are currently developed [Altenkirch and Kaposi, 2014] — for designing an extension of dependent type theory in which such an internal form of parametricity holds. We propose another such system here. Our technical contributions are as follows:

- We present a type theory (section 2 on the following page) which internalizes parametricity (as we show in section 3 on page 5) and can be seen as a simplification and generalization of the systems of Bernardy and Moulin [2012, 2013]

- We provide a *denotational* semantics, in the form of a presheaf model, for this type theory (section 4 on page 8). This model is a refinement of the presheaf semantics used to interpret nominal sets with restrictions [Bezem et al., 2014, Pitts, 2014].
- We conjecture that conversion and type-checking are decidable for this system.

2 Syntax

In this section we define the syntax and typing rules of our parametric type theory, as well as the equality judgment.

We assume a special symbol '0', and a countable infinite set of other symbols, called *colors*. The metasyntactic variables $i, j, \ldots$ range over colors, while $I, J, \ldots$ range over *finite* sets of colors. We further assume a fixed function $\mathsf{fresh}(\cdot)$ such that $\mathsf{fresh}(I) \notin I$ for any finite color set I. The main innovation of the type theory presented here is that terms may depend on (a finite number of) colors.

▶ **Definition 1** (Syntax of terms and contexts).

$t, u, A, B \coloneqq x$	variable
$\mid U$	universe
$\mid \lvert A \rvert$	code
$\mid \mathrm{El}(A)$	decode
$\mid \lambda x : A.t$	abstraction
$\mid t\ u$	application
$\mid (x : A) \to B$	product
$\mid (t,_i u)$	colored pair
$\mid (x : A) \times_i B$	colored type pair
$\mid \langle t,_i u \rangle$	colored function pair
$\mid A \ni_i u$	parametricity type
$\mid t{\cdot}i$	parametricity proof
$\Gamma, \Delta \coloneqq () \mid \Gamma, x : A$	

We give a few intuitions to interpret the novel syntax, before giving formally the typing rules of the system.

1. Reynolds [1983] associates each type with a predicate. Here, each type is associated not a single predicate, but many: one for every color. Furthermore this predicate is accessible from the logic. The type $A \ni_i u$ expresses that u satisfies the parametricity predicate associated with the type A on color i.
2. The term $a{\cdot}i$ yields a proof of $A \ni_i a\,(i\,0)$.
3. In the above, the term $a\,(i\,0)$ denotes a realizer of a, obtained by erasing the color i. (Erasure is detailed in Def. 5.)
4. The forms $(t,_i u)$, $(x : A) \times_i B$ and $\langle t,_i u \rangle$ allow to locally associate parametricity proofs with a given realizer.

▶ **Definition 2** (Typing judgements — à la Tarski). We mutually define three judgments:

- $\Gamma \vdash_I$ (Is the context Γ is well-formed, assuming the color set I?).
- $\Gamma \vdash_I A$ (Is the type A well-formed in Γ, assuming the color set I?)
- $\Gamma \vdash_I a : A$ (Does the term a have type A in the context Γ, assuming the color set I?)

$\boxed{\Gamma \vdash_I}$

$$\textsc{Empty}\quad \frac{}{() \vdash_I} \qquad \textsc{NewVar}\quad \frac{\Gamma \vdash_I \qquad \Gamma \vdash_I A}{\Gamma, x : A \vdash_I}$$

$\boxed{\Gamma \vdash_I A}$

$$\textsc{Universe}\quad \frac{}{\Gamma \vdash_I U} \qquad \textsc{Decode}\quad \frac{\Gamma \vdash_I A : U}{\Gamma \vdash_I \mathrm{El}(A)}$$

$$\textsc{Pi}\quad \frac{\Gamma \vdash_I A \qquad \Gamma, x : A \vdash_I B}{\Gamma \vdash_I (x : A) \to B} \qquad \textsc{Out}\quad \frac{\Gamma \vdash_{I,i} T \qquad \Gamma \vdash_I a : T\,(i\,0)}{\Gamma \vdash_I T \ni_i a} \qquad \textsc{In-Pred}\quad \frac{\Gamma \vdash_I A \qquad \Gamma, x : A \vdash_I B}{\Gamma \vdash_{I,i} (x : A) \times_i B}$$

$\boxed{\Gamma \vdash_I a : A}$

$$\textsc{Conv}\quad \frac{\Gamma \vdash_I t : A \qquad A = B}{\Gamma \vdash_I t : B} \qquad \textsc{Var}\quad \frac{\Gamma \vdash_I \qquad x : A \in \Gamma}{\Gamma \vdash_I x : A} \qquad \textsc{Code}\quad \frac{\Gamma \vdash_I A}{\Gamma \vdash_I |A| : U}$$

$$\textsc{Lam}\quad \frac{\Gamma, x : A \vdash_I b : B}{\Gamma \vdash_I \lambda x : A.b : (x : A) \to B} \qquad \textsc{App}\quad \frac{\Gamma \vdash_I t : (x : A) \to B[x] \qquad \Gamma \vdash_I u : A}{\Gamma \vdash_I t\ u : B[u]}$$

$$\textsc{In-Abs}\quad \frac{\Gamma \vdash_I a : T\,(i\,0) \qquad \Gamma \vdash_I p : T \ni_i a}{\Gamma \vdash_{I,i} (a,_i p) : T}$$

$$\textsc{In-Fun}\quad \frac{\begin{array}{c}\Gamma \vdash_I t : ((x : A) \to P[x])\,(i\,0) \\ \Gamma \vdash_I u : (x : A\,(i\,0)) \to (x' : A \ni_i x) \to P[(x,_i x')] \ni_i tx\end{array}}{\Gamma \vdash_{I,i} \langle t,_i u\rangle : (x : A) \to P[x]} \qquad \textsc{Color-Elim}\quad \frac{\Gamma \vdash_{I,i} a : T}{\Gamma \vdash_I a{\cdot}i : T \ni_i a\,(i\,0)}$$

The parametricity constructions ($\cdot$ and $\ni$) act like color binders (they bring colors into scope), while the pairing constructs remove colors from scope. The equality relation used in the Conv rule is detailed below in Def. 8.

Additionally, for the above system to be well-founded, we need to distinguish small and big types, and allow only small types to be encoded in U. Small types are closed under product, $\times_i$ and $\ni_i$. The distinction between big and small types being standard, and to keep the presentation concise, we leave it implicit in the syntax.

▶ **Definition 3.** A *color map* $f : I \to J$ is a function $I \to J \cup \{0\}$ such that $f(i_1) = f(i_2)$ iff. $i_1 = i_2$ whenever $f(i_1) = f(i_2) \in J$.

▶ **Definition 4** (Category **pI**). Let objects be objects be finite color sets and morphisms be color maps. If $f : I \to J$ and $g : J \to K$, we define the composition $fg : I \to K$ as $fg(i) = 0$ if $f(i) = 0$ and $fg(i) = g(f(i))$ if $f(i) \in J$. We write $1_I : I \to I$ for the identity map, and define it as $1_I(i) = i$ for each $i \in I$. It is easy to check that **pI** is a category (see [Pitts, 2013, ex. 9.7 p. 176] for another description of this category), which is is equivalent to the category **Res** of nominal restriction sets [Pitts, 2013, rem. 9.9 p. 161].

We note

- $(i\,0) : I, i \to I$ the partial identity: $(i\,0)(i) = 0$ and $(i\,0)(j) = j$ for each $j \in I$;
- $\iota_i : I \to I, i$ the inclusion: $\iota_i(j) = j$ for each $j \in I$; and
- $f^{ij} : I, i \to J, j$ the color map such that $f^{ij}(i) = j$ and $f^{ij}(k) = f(k)$ for all $k \in I$ (if $f : I \to J$ with $i \notin I$ and $j \notin J$).

▶ **Definition 5** (Color substitution). We consider a color map $f : I \to J$ as a (color) substitution on terms, and define af by structural induction on a.

$$\begin{aligned}
xf &= x \\
Uf &= U \\
(\lambda(x : A).t)f &= \lambda(x : Af).tf \\
(t\,u)f &= (tf)\,(uf) \\
((x : A) \to B)f &= (x : Af) \to (Bf) \\
(a,_i\, p)f &= (ag,_j\, pg) && \text{if } f(i) = j \in J, \text{ where g} = \iota_i f(j\,0) \\
&= a(\iota_i f) && \text{if } f(i) = 0 \\
(A \times_i B)f &= (Ag) \times_j (Bg) && \text{if } f(i) = j \in J, \text{ where g} = \iota_i f(j\,0) \\
&= A(\iota_i f) && \text{if } f(i) = 0 \\
\langle t,_i\, u\rangle f &= \langle tg,_j\, ug\rangle && \text{if } f(i) = j \in J, \text{ where g} = \iota_i f(j\,0) \\
&= u(\iota_i f) && \text{if } f(i) = 0 \\
(A \ni_i a)f &= (Af^{ij}) \ni_j (af) && \text{where } j = \mathsf{fresh}(J) \\
(a \cdot i)f &= (af^{ij}) \cdot j && \text{where } j = \mathsf{fresh}(J)
\end{aligned}$$

We extend the definition to contexts in the obvious way:

$$\begin{aligned}
()f &= () \\
(\Gamma, x : A)f &= \Gamma f, x : Af
\end{aligned}$$

We leave color substitution undefined if a color appears free in the term but is not in the domain of f.

▶ **Theorem 6.**

- *$a1_I = a$*
- *$(af)g = a(fg)$ for any $f : I \to J$ and $g : J \to K$*

Proof. By structural induction on a. ◀

▶ **Theorem 7** (Color substitution preserves typing). *If $\Gamma \vdash_I a : A$ then the term af is defined and $\Gamma f \vdash_J af : Af$.*

Proof. By induction on the typing judgment. ◀

▶ **Definition 8** (Conversion). The convertibility of types used in the CONV rule and written simply $(=)$ is defined as the smallest congruence containing the following rules.

$$\textsc{Pair-App}\quad \langle t,_i u\rangle\, a = (t\, a\, (i\, 0),_i u\, a\, (i\, 0)\, (a\cdot i)) \qquad \textsc{Pair-Param}\quad (a,_i p)\cdot i = p \qquad \textsc{Pair-Pred}\quad ((x : A) \times_i B[x]) \ni_i a = B[a]$$

$$\mathrm{El}(|A|) = A \qquad |\mathrm{El}(A)| = A \qquad \beta\quad (\lambda x : A.u[x])t = u[t] \qquad \eta\quad \frac{t\ x = u}{t = \lambda x : A.u}$$

$$\textsc{Surj-Param}\ \frac{t\, (i\, 0) = a \qquad t\cdot i = p}{t = (a,_i p)} \qquad \textsc{Surj-Fun}\ \frac{t\, (i\, 0) = u \qquad (t(x,_i y))\cdot i = v x y}{t = \langle u,_i v\rangle} \qquad \textsc{Surj-Typ}\ \frac{T\, (i\, 0) = A \qquad T \ni_i x = B}{T = (x : A) \times_i B}$$

$$\textsc{Refl}\ \frac{}{a = a} \qquad \textsc{Sym}\ \frac{a = b}{b = a} \qquad \textsc{Trans}\ \frac{a = b \qquad b = c}{a = c}$$

▶ **Corollary 9** (Any term can be seen as a pair of a realizer and a parametricity proof)**.**
- $a = (a\, (i\, 0),_i a\cdot i)$
- $T = (x : T\, (i\, 0)) \times_i (T \ni_i x)$
- $t = \langle t\, (i\, 0),_i \lambda x x'.(t(x,_i x'))\cdot i\rangle$

Our conversion relation is intentional for functions, but extensional when it comes to dependencies on colors. Because there is at any point only a finite number of colors to consider, we conjecture that our conversion relation is decidable.

3 Parametricity

In this section we prove that our system properly internalizes parametricity. We also illustrate the system by giving a few simple proofs relying on parametricity (including iterated parametricity).

Contrary to previous type theories with internalized parametricity [Bernardy and Moulin, 2012, 2013], the system presented here lacks equalities which allow to compute parametricity types. Expressed in our syntax, those equalities would become the conversion rules:

$$U \ni_i A = A \to U$$

and

$$((x : A) \to B[x]) \ni_i f = (x : A) \to (x' : A \ni_i x) \to B[(x,_i x')] \ni_i (f x).$$

The absence of the above equalities allows for a simpler system, but how can we ensure that all parametricity theorems hold? The answer is that the above relationships hold as isomorphisms.

We say that A is isomorphic to B iff.
1. There exist $f : A \to B$
2. There exist $g : B \to A$
3. For any x, $f\, (g\, x) = x$
4. For any x, $g\, (f\, x) = x$

This notion of isomorphism is quite strong, because the equality used in its definition is the conversion relation (Def. 8).

▶ **Theorem 10.** *$U \ni_i A$ is isomorphic to $A \to U$.*

Proof.
1. $f : (Q : U \ni_i A) \to A \to U$
   $f\,Q\,x = (A,_i Q) \ni_i x$
2. $g : (P : A \to U) \to U \ni_i A$
   $g\,P = ((x : A) \times_i (Px)) \cdot i$
3. $(A,_i ((y : A) \times_i (Py)) \cdot i) \ni_i x = ((y : A) \times_i (Px) \ni_i x = Px$ By η-contraction we get the desired result.
4. $((x : A) \times_i (A,_i Q) \ni_i x).i = Q$ if $(x : A) \times_i (A,_i Q) \ni_i x = (A,_i Q)$. We then use equality for $\times_i$. The first components are obviously equal. For the second components we are left with $(A,_i Q) \ni_i x = (A,_i Q) \ni_i x$, which holds by reflexivity. ◀

▶ **Theorem 11.** $((x : A) \to B[x]) \ni_i f$ *is isomorphic to*

$$(x : A) \to (x' : A \ni_i x) \to B[(x,_i x')] \ni_i (f\,x)$$

Proof.
1. $f : (q : ((x : A) \to B[x]) \ni_i f) \to (x : A) \to (x' : A \ni_i x) \to B[(x,_i x')] \ni_i (fx)$
   $f\,q\,x\,x' = ((f,_i q)(x,_i x')) \cdot i$
2. $g : ((x : A) \to (x' : A \ni_i x) \to B[(x,_i x')] \ni_i (f\,x)) \to ((x : A) \to B[x]) \ni_i f$
   $g\,p = \langle f,_i p \rangle \cdot i$
3. $((f,_i \langle f,_i p \rangle \cdot i)(x,_i x)') \cdot i = (\langle f,_i p \rangle (x,_i x')) \cdot i = (f\,x,_i p\,x\,x') \cdot i = p\,x\,x'$
4. $\langle f,_i \lambda x x'.((f,_i q)(x,_i x')) \cdot i \rangle \cdot i$ iff. $\langle f,_i \lambda x x'.((f,_i q)(x,_i x')) \cdot i \rangle = (f,_i q)$, which is true by the equality rule for function pairing. ◀

In practice, when carrying out parametricity proofs, many of the steps of the above isomorphisms cancel each other and one obtains a simpler proof. This behaviour is illustrated by the following example: parametricity for Church-encoded natural numbers. (For the sake of simplicity, in the remainder of this section, we leave out the distinction between types and their codes.)

▶ **Example 12.** Let $N = (X : U) \to X \to (X \to X) \to X$. Proving (unary) parametricity for N means that, assuming
- $f : N$
- $A : U$
- $P : A \to U$
- $z : A$
- $z' : P\,z$
- $s : A \to A$
- $s' : (x : A) \to P\,x \to P\,(s\,x)$

we can prove $P\,(f\,A\,z\,s)$.

Indeed, a proof term is the following:

$$(f((x : A) \times_i (Px))(z,_i z')\langle s,_i s' \rangle) \cdot i$$

3.1 Iterating Parametricity

In our system, one can use parametricity generically as follows:

$$p : (A : U) \to (x : A) \to A \ni_i x$$
$$p\,x = x \cdot i$$

We have already seen that $A \ni_i$ corresponds to the parametricity predicate for type A. We can iterate this operator to construct relations between parametricity witnesses. That is, given

$$\begin{aligned}&x : A\\&y : A \ni_i x\\&z : A \ni_i x\end{aligned}$$

Then the type $A \ni_i (x,_j y) \ni_j z$ is well formed ($\ni$ is left associative), and can be understood as a relation between the parametricity proofs y and z. The following results about this relation illustrate the expressivity of our system.

▶ **Theorem 13.** *If the type A does not depend on either i or j, the relation $\lambda yz.A \ni_i (x,_j y) \ni_j z$ is symmetric.*

Proof. We first construct the proof term:

$$\begin{aligned}&\sigma_1 : (x : A) \to (y : A \ni_i x) \to (z : A \ni_i x) \to (w : A \ni_i (x,_j y) \ni_j z) \to A \ni_j (x,_i z) \ni_i y\\&\sigma_1\, x\, y\, z\, w = ((x,_j y),_i (z,_j w))\cdot j \cdot i\end{aligned}$$

And, by α-equivalence on colors, $A \ni_j (x,_i z) \ni_i y = A \ni_i (x,_j z) \ni_j y$. ◀

▶ **Theorem 14.** *The function σ_1 (defined above) is involutive in its last argument:*

$$\sigma_1\, y\, x\, z\, (\sigma_1\, x\, y\, z\, w) = w$$

Proof. Let

$$\begin{aligned}t &= ((x,_j y),_i (z,_j w))\\w' &= t\cdot j\cdot i\\t' &= ((x,_i z),_j (y,_i w'))\end{aligned}$$

Then

$$\begin{aligned}t'\,(i\,0) &= (x,_j y) = t\,(i\,0)\\t'\,(j\,0) &= (x,_i z) = t\,(j\,0)\\(t\cdot j)\,(i\,0) &= y\end{aligned}$$

We continue to reason by deduction:

$$\begin{aligned}w' &= t\cdot j\cdot i && \text{By def}\\(y,_i w') &= t\cdot j && \text{Because } (t\cdot j)\,(i\,0) = y\\t'\cdot j &= t\cdot j && \text{By def}\\t' &= t && \text{Because } t'\,(j\,0) = t\,(j\,0)\\t' &= ((x,_j y),_i (z,_j w)) && \text{By def}\\t'\cdot i &= (z,_j w)\\t'\cdot i\cdot j &= w\end{aligned}$$ ◀

▶ **Corollary 15.** *The types $A \ni_i (x,_j y) \ni_j z$ and $A \ni_j (x,_i z) \ni_i y$ are isomorphic.*

▶ Remark. At this point one may wonder if the system could have been set up to have $t{\cdot}i{\cdot}j = t{\cdot}j{\cdot}i$, and the equality between $A \ni_i (x,_j y) \ni_j z$ and $A \ni_j (x,_i z) \ni_i y$ rather than an isomorphism. The answer is that the equation

$$A \ni_i (x,_j y) \ni_j z = A \ni_j (x,_i z) \ni_i y$$

is inconsistent: in particular for $A = U$ one gets

$$U \ni_i (X,_j P) \ni_j Q = U \ni_j (X,_i Q) \ni_i P$$

for arbitrary P and Q of type $U \ni_i X$. The above equality in turn implies

$$(x : X) \to Px \to Qx \to U = (x : X) \to Qx \to Px \to U$$

for arbitrary predicates P and Q over X, which is obviously inconsistent.

▶ **Theorem 16.** *If the type A and the term a do not depend on either i or j, any proof a' of $A \ni_i a$ (not depending on i or j either) is related to the canonical proof $(a{\cdot}i)$, i.e., formally $A \ni_i (a,_j a{\cdot}i) \ni_j a'$.*

Proof. We can construct the following closed term:

$$\begin{aligned}
&q : (A : U) \to (x : A) \to (x' : A \ni_i x) \to A \ni_i (x,_j x{\cdot}i) \ni_j x' \\
&q : (A : U) \to (x : A) \to (x' : A \ni_i x) \to A \ni_i x \ni_j x' && \text{by Corollary 9} \\
&q\, A\, x\, x' = x'{\cdot}j
\end{aligned}$$

The result is obtained by substituting a for x and a' for x'. ◀

To conclude the section we note that by iterating parametricity n times, one creates n-ary relations between proofs of relations of arity $n-1$. Furthermore, the above results carry over to the n-ary case. That is, for each $k < n$, one can construct a function σ_k, which exchanges the arguments k and $k+1$ of a relation. Furthermore, these functions satisfy the laws of the generators of the symmetric group.

4 Presheaf model

In this section we show how to interpret our type theory by a presheaf model. Recall **pI** (Def. 4), the category of color maps.

▶ **Definition 17** (Projection). We say that a morphism $\alpha : I \to I_\alpha$ is a *projection* if $I_\alpha \subseteq I$, $\alpha(i) = 0$ for each $i \in I \backslash I_\alpha$, and $\alpha(i) = i$ for each $i \in I_\alpha$.

▶ **Definition 18** (Total maps). Injective morphisms, noted $h : I \rightarrowtail J$, are the *total* ones, *i.e.*, those verifying $h(i) \neq 0$ for all $i \in I$.

▶ Remark (Morphism decomposition). Any morphism $f : I \to J$ has a unique decomposition into a projection map $\alpha : I \to I_\alpha$ and a total map $h : I_\alpha \rightarrowtail J$.

$$\begin{tikzcd}
I \arrow[d, dotted, "\alpha"'] \arrow[dr, "f"] & \\
I_\alpha \arrow[r, dotted, tail, "h"'] & J
\end{tikzcd}$$

▶ **Definition 19** (I-set)**.** We call I-element any tuple indexed by the subsets of I: $(u_J)_{J\subseteq I}$. An I-set is a set of I-elements. For instance, the elements of a $\{i,j\}$-set are of the form $u = (u_\varnothing, u_i, u_j, u_{i,j})$. Alternatively, such an element can be seen as a tuple (u_α) indexed by the projections $\alpha : I \to I_\alpha$.

If a, b are I-elements and $j \notin I$, we define the (I,j)-element $(a,_j b)$ as $(a,_j b)_J := a_J$ if $j \notin J$ and $(a,_j b)_{J,j} := b_J$. Any (I,i)-element can be written $u = (u_J)_{J\subseteq\{I,i\}} = (u_J)_{J\subseteq I} \cup (u_{J,i})_{J\subseteq I}$; We can therefore define the I-elements $u(i\,0) := (u_J)_{J\subseteq I}$ and $u \cdot i := (u_{J,i})_{J\subseteq I}$. (Hence by definition $u = (u(i\,0),_i u \cdot i)$.)

Recall that a *presheaf* F on $\mathbf{pI}^{\mathrm{op}}$ is given by a family of sets $F(I)$ together with restriction maps $F(I) \to F(J)$, $u \mapsto uf$ for $f : I \to J$ satisfying $u1 = u$ and $(uf)g = u(fg)$. We use a refined presheaf on $\mathbf{pI}^{\mathrm{op}}$ by requiring two further conditions:

1. for any object I, $F(I)$ is an I-set; and
2. for any projection map $\alpha : I \to I_\alpha$, the restriction map $F(I) \to F(I_\alpha)$, $u \mapsto u\alpha$ is the projection operation, *i.e.*, $u\alpha_J = u_J$ for any $J \subseteq I$.

Seeing an I-element u as a tuple indexed by projection maps $\alpha : I \to I_\alpha$, the second requirement can be written $(u\alpha)_\beta = u_{\alpha\beta}$.

A context $\Gamma \vdash_I$ is interpreted as a presheaf on the slice category $\mathbf{pI}^{\mathrm{op}}/I$, *i.e.*, by a family of J-sets Γf for any map $f : I \to J$ together with restriction maps $\Gamma f \to \Gamma fg$, $\rho \mapsto \rho g$ for $g : J \to K$ satisfying the conditions $\rho 1 = \rho$ and $(\rho g)h = \rho(gh)$. Furthermore the map $\Gamma f \to \Gamma(f\alpha)$, $\rho \mapsto \rho\alpha$ is the projection operation.

A type $\Gamma \vdash_I A$ is interpreted as follows. For each map $f : I \to J$ and $\rho \in \Gamma f$ we give a J-set $A(f,\rho)$ together with restriction maps $A(f,\rho) \to A(fg, \rho g)$, $u \mapsto ug$ if $g : J \to K$ satisfying $u1 = u$ and $(ug)h = u(gh)$ for any $h : K \to L$. Furthermore the map $A(f,\rho) \to A(f\alpha, \rho\alpha)$, $u \mapsto u\alpha$ is the projection operation.

A term $\Gamma \vdash_I a : A$ is interpreted by a J-element $a(f,\rho) \in A(f,\rho)$ for each $f : I \to J$ and $\rho \in \Gamma f$, such that $a(f,\rho)g = a(fg, \rho g)$ for any $g : J \to K$.

If $\Gamma \vdash_I A$ we define the interpretation of $\Gamma, x : A \vdash_I$ by taking $\langle \rho, x = u\rangle_\alpha = \langle \rho_\alpha, x = u_\alpha\rangle$, where and $\rho \in \Gamma f$ and $u \in A(f,\rho)$. The restriction map is defined by $\langle \rho, x = u\rangle g = \langle \rho g, x = ug\rangle$.

The above refinement on presheaves is necessary for the interpretation of some of our syntactic constructions. Indeed, without this refinement, it is not clear how to validate the equality $((x : A) \times_i B[x]) \ni_i a = B[a]$.

The semantics we define satisfies the substitution law. That is, if $\Gamma, x : A \vdash_I B$ and $\Gamma \vdash_I a : A$ then for any $f : I \to J$ and $\rho \in \Gamma f$ we have $B[a](f,\rho) = B(f, \langle \rho, x = a(f,\rho)\rangle)$. It also satisfies the substitution law on colors, *i.e.*, if $\Gamma \vdash_I A$ and $f : I \to J$ then for any $g : J \to K$ we have $\Gamma' g = \Gamma(fg)$, where Γ' is the result of performing the substitution f in Γ, and if $\rho \in \Gamma fg$ we have $Af(g,\rho) = A(fg,\rho)$. For establishing these properties, we proceed as Aczel [1998].

We proceed to interpret each type construction.

Pi. Assume $f : I \to J$ and $\rho \in \Gamma f$. We define $((x : A) \to B)(f, \rho)$ as a J-set. A J-element of $((x : A) \to B)(f, \rho)$ is defined as a tuple $\lambda = (\lambda_\alpha)$, where each λ_α is a family of elements indexed by a total map $g : J_\alpha \rightarrowtail K$:

$$\lambda_{\alpha g} \in \prod_{u \in A(f\alpha g, \rho\alpha g)} B(f\alpha g, \langle \rho\alpha g, x = u\rangle)$$

such that $\mathsf{app}(\lambda_{\alpha g}, u)h = \mathsf{app}(\lambda_{\alpha gh}, uh)$ for $g : J \rightarrowtail K$ total and for *any* $h : K \to L$. Because any map $J \to K$ has an unique decomposition as a projection and a total map, we can consider λ_h for an arbitrary map $h : J \to K$.

If $g : J \to K$ is an arbitrary map, we define λg to be the tuple (λg_β) where λg_β is the family $\lambda g_{\beta h} = \lambda_{g\beta h}$.

With this definition, we directly have $\lambda\alpha_\beta = \lambda_{\alpha\beta}$.

This is similar to the usual interpretation of dependent product in presheaf models [Hofmann, 1997, Bezem et al., 2014]; but to satisfy our first extra condition on presheaves we present each element as a tuple, which can be done naturally by repartitioning the family as follows: $(\lambda_f)_{f:I\to J} = (\lambda_{\alpha g})_{I_\alpha \subseteq I, g:I_\alpha \rightarrowtail J} \cong ((\lambda_{\alpha g})_{g:I_\alpha \rightarrowtail J})_{I_\alpha \subseteq I}$

Universe. The universe U is interpreted as a presheaf over $\mathbf{pI}$. An element A of $U(I)$ is a tuple (A_α) where each A_α is a family of sets $A_{\alpha f}$ for $f : I_\alpha \rightarrowtail J$ *total* together with restriction maps $A_{\alpha f} \to A_{\alpha f g}$, $u \mapsto ug$ for $f : I_\alpha \rightarrowtail J$ total and $g : J \to K$ arbitrary, such that $u1 = u$ and $(ug)h = u(gh)$.

As before, such data define a set A_f for an arbitrary map $f : I \to J$ with restriction maps $A_f \to A_{fg}$ if $g : J \to K$.

If $g : I \to J$ is an arbitrary map, we define Ag by taking $Ag_{\beta h}$ to be the set $A_{g\beta h}$, together with restriction maps $Ag_{\beta h} \to Ag_{\beta hl}$ defined as the given maps $A_{g\beta h} \to A_{g\beta hl}$. We can then check, as before, that we have $A\alpha_\beta = A_{\alpha\beta}$

As before, this is similar to the usual interpretation of universe in presheaf models, where each element is presented as a tuple.

Out. Assume $f : I \to J$ and assume $\rho \in \Gamma f$. We need to define the J-set $(P \ni_i a)(f, \rho)$. Let $j = \mathsf{fresh}(J)$. Recall that we note the inclusions $\iota_i : I \to I, i$ and $\iota_j : J \to J, j$. By the induction hypotheses we get a (J, j)-set $P(f^{ij}, \rho\iota_i)$, and the J-element $a(f, \rho)$ belongs to $P(i\,0)(f, \rho) = P((i\,0)f, \rho) = P(\iota_i f, \rho\iota_j)(j\,0)$. We define $(P \ni_i a)(f, \rho)$ to be the set of J-elements v such that $(a(f, \rho),_j v) \in P(f^{ij}, \rho\iota_j)$. If v is such an element and $g : J \to K$ and $k = \mathsf{fresh}(K)$, then vg is defined by the equation $(a(f, \rho)g,_k vg) = (a(f, \rho),_j v)g^{jk}$.

In-Pred. Assume $f : I, i \to J$, and $\rho \in \Gamma(\iota_i f)$. We need to define the J-set $((x : A) \times_i B)(f, \rho)$. Let $\iota_i : I \to I, i$ be the inclusion. There are two cases. If $f(i) = 0$, then $((x : A) \times_i B)(f, \rho)$ is defined to be the J-set $A(\iota_i f)$. Otherwise, if $f(i) = j \in J$, then we define $((x : A) \times_i B)(f, \rho)$ to be the J-set of $(u,_j v)$ where u is a $J\backslash\{j\}$-element in $A(\iota_i f(j\,0), \rho(j\,0))$ and v is an element in $B(\iota_i f(j\,0), \langle\rho(j\,0), x = u\rangle)$.

Decode. Assume $f : I \to J$ and $\rho \in \Gamma f$. We have $A(f, \rho) \in U(J)$ and we define $\mathrm{El}(A)(f, \rho)$ to be the set $A(f, \rho)_1$. The restriction map $\mathrm{El}(A)(f, \rho) \to \mathrm{El}(A)(fg, \rho g)$, $u \mapsto ug$ is defined using the restriction map $A(f, \rho)_1 \to A(f, \rho)_g$ and the fact that we have $A(f, \rho)_g = A(fg, \rho g)_1$.

▶ Remark. Our calculus does not have any base type, but they could be interpreted by modifying their usual interpretation as a constant presheaf into an isomorphic I-set. For instance, the base type of natural numbers would be interpreted as the I-set of $(n_J)_{J\subseteq I}$ where $n_\varnothing \in \mathbb{N}$ and $n_J = 0$ for any non-empty $J \subseteq I$.

We now describe how to interpret terms.

Var. We define $x(f,\rho)$ to be $\rho(x)$

Lam. We define $\mathsf{app}((\lambda x : A.b)(f,\rho)_g, u)$ to be $b(f, \langle \rho g, x = u\rangle)$

App. We define $(t\,u)(f,\rho)$ to be $\mathsf{app}(t(f,\rho)_1, u(f,\rho))$

In-Abs. Let $f : I, i \to J$ and $\rho \in \Gamma\iota_i f$ be given. We define $(a,_i\, p)(f,\rho)$ by case analysis on $f(i)$. If $f(i) = 0$, we take $(a,_i\, p)(f,\rho)$ to be $a(\iota_i f, \rho)$. If $f(i) = j \in J$, we take $(a,_i\, p)(f,\rho)$ to be $(a(\iota_i f(j0), \rho),_j\, p(\iota_i f(j0), \rho))$

In-Fun. Let $f : I, i \to J$ and $\rho \in \Gamma\iota_i f$ be given. We define $\langle t,_i\, u\rangle(f,\rho)_g$ by case analysis on $g(f(i))$. If $g(f(i)) = 0$, we take $\langle t,_i\, u\rangle(f,\rho)_g$ to be $t(\iota_i f, \rho)_g$. If $g(f(i)) = j \in K$, we define $w = \langle t,_i\, u\rangle(f,\rho)_g$ by

$$\mathsf{app}(w, (a,_j\, b)) = (\mathsf{app}(t\,(\iota_i f g(j0), \rho g(j0)), a),_j\, \mathsf{app}(\mathsf{app}(u\,(\iota_i f g(j0), \rho g(j0)), a), b))$$

Color-Elim. Let $f : I \to J$ and $\rho \in \Gamma f$ be given. We define $(a\cdot i)(f,\rho)$ to be $a(f^{ij}, \rho\iota_j)\cdot j$ where $j = \mathsf{fresh}(J)$.

▶ **Theorem 20** (Convertible terms are semantically equal)**.**

- *If $\Gamma \vdash_I A_1$ and $\Gamma \vdash_I A_2$ with $A_1 = A_2$, then $A_1(f,\rho) = A_2(f,\rho)$ for any $f : I \to J$ and $\rho : \Gamma f$.*
- *If $\Gamma \vdash_I a_1 : A$ and $\Gamma \vdash_I a_2 : A$ with $a_1 = a_2$, then $a_1(f,\rho) = a_2(f,\rho)$ for any $f : I \to J$ and $\rho : \Gamma f$.*

Proof. By simultaneous induction on the derivation. We only show the conversion rules Pair-Param and Pair-Pred here; other rules involving colors can be proven in a similar fashion, while β and η can be proven in the usual way.

Pair-Param. Let $f : I \to J$ and $j = \mathsf{fresh}(J)$. We have

$$\begin{aligned}
& v \in (((x : A) \times_i B) \ni_i a)(f,\rho)\\
\text{iff. } & (a(f,\rho),_j\, v) \in ((x : A) \times_i B)(f^{ij}, \rho\iota_j)\\
\text{iff. } & (a(f,\rho),_j\, v) \in \{(u,_j\, w) \mid u \in A(f,\rho), w \in B(f, \langle \rho, x = u\rangle)\}\\
\text{iff. } & v \in B(f, \langle \rho, x = a(f,\rho)\rangle)\\
\text{iff. } & v \in B[a](f,\rho)
\end{aligned}$$

Pair-Pred. Let $f : I \to J$ and $j = \mathsf{fresh}(J)$. We have

$$\begin{aligned}
& ((a,_i\, p)\cdot i)(f,\rho)\\
= {} & (a,_i\, p)(f^{ij}, \rho\iota_j) \cdot j\\
= {} & (a(\iota_i f^{ij}(j\,0), \rho),_j\, p(\iota_i f^{ij}(j\,0), \rho)) \cdot j\\
= {} & (a(f,\rho),_j\, p(f,\rho)) \cdot j\\
= {} & p(f,\rho)
\end{aligned}$$

◀

▶ Remark. As noted earlier, the types $U \ni_i (X,_j\, P) \ni_j Q$ and $U \ni_j (X,_i\, Q) \ni_i P$ are not convertible. Their semantic interpretations are not equal either. Indeed taking $f = 1_\varnothing$, $k = \mathsf{fresh}(\varnothing)$ and $l = \mathsf{fresh}(\{k\})$, we have (leaving out the context interpretation ρ for the sake of readability) on the one hand

$$\begin{aligned}
& v \in (U \ni_i (X,_j\, P) \ni_j Q)f\\
\text{iff. } & (Qf,_k\, v) \in (U \ni_i (X,_j\, P))f^{jk}\\
\text{iff. } & ((X,_j\, P)f^{jk},_l\, (Qf,_k\, v)) \in U(l,k)\\
\text{iff. } & ((X,_k\, P),_l\, (Q,_k\, v)) \in U(l,k)
\end{aligned}$$

and on the other hand

$$\begin{aligned} & v \in (U \ni_j (X,_i Q) \ni_i P) f \\ \text{iff. } & (Pf,_k v) \in (U \ni_j (X,_i Q)) f^{ik} \\ \text{iff. } & ((X,_i Q) f^{ik},_l (Pf,_k v)) \in U(k,l) \\ \text{iff. } & ((X,_k Q),_l (P,_k v)) \in U(k,l) \end{aligned}$$

Hence $(U \ni_i (X,_j P) \ni_j Q) f \neq (U \ni_j (X,_i Q) \ni_i P) f$ since the map $U(l,k) \to U(k,l)$, $u \mapsto ug$ where $g(k) = l$ and $g(l) = k$ is not the identity.

▶ **Theorem 21** (Validity). *For any $f : I \to J$ and any $\rho \in \Gamma f$, if $\Gamma \vdash_I a : A$ then $a(f,\rho) \in A(f,\rho)$.*

Proof. By induction on the typing judgment. We only show the cases In-Abs and Color-Elim. In-Fun is similar to the former, and the other cases match the usual proof (using Th. 20 for Conv).

In-Abs. Let $f : I, i \to J$ and $\rho \in \Gamma f$. We need to show that $(a,_i p)(f,\rho) : T(f,\rho)$. If $f(i) = 0$, we have by definition $(a,_i p)(f,\rho) = a(\iota_i f, \rho)$, which by induction hypothesis belongs to the J-set $T(i\,0)(\iota_i f, \rho)$; but by color substitution $T(i\,0)(\iota_i f,\rho) = T((i\,0)\iota_i f, \rho) = T(f,\rho)$. If $f(i) = j \in J$, we have $f = g^{ij}$ where $g = \iota_i f(j\,0)$; by induction hypothesis $p(g,\rho) \in (T \ni_i a)(g,\rho)$ hence by definition $(a(g,\rho),_j p(g,\rho)) \in T(g^{ij}, \rho\iota_j)$ then $(a,_i p)(f,\rho) \in T(f,\rho)$.

Color-Elim. Let $f : I \to J$ and $\rho \in \Gamma f$. We need to show that $(a \cdot i)(f,\rho) \in (T \ni_i a(i\,0))(f,\rho)$, *i.e.*, that $(a(i\,0)(f,\rho),_j (a\cdot i)(f,\rho)) = (a((i\,0)f,\rho),_j a(f^{ij},\rho\iota_j)\cdot j) \in T(f^{ij},\rho\iota_j)$ where $j = \mathsf{fresh}(J)$. By induction hypothesis $a(f^{ij},\rho\iota_j) \in T(f^{ij},\rho\iota_j)$, and because it is a (J,j)-element we have $a(f^{ij},\rho\iota_j) = (a(f^{ij},\rho\iota_j)(j\,0),_j a(f^{ij},\rho\iota_j) \cdot j)$ We conclude by remarking that $a(f^{ij},\rho\iota_j)(j\,0) = a((i\,0)f,\rho)$ holds by color substitution and definition of f^{ij}. ◀

5 Related Work

5.1 Our own line of work

This work continues a line of work aiming at a smooth integration of parametricity with dependent types [Bernardy et al., 2010, Bernardy and Lasson, 2011, Bernardy et al., 2012, Bernardy and Moulin, 2012, 2013]. The present work offers two improvements over previous publications: 1. a denotational semantics, and 2. a much simplified syntax, suitable as the basis of a proof assistant.

The simplification of syntax is allowed by foregoing the preservation of functions by parametricity. We call preservation of functions by parametricity the property that if f were a function, then the canonical proof that f is parametric (denoted $f \cdot i$ here) is also a function. To our knowledge, following Reynolds [1983], all parametric *models* of parametricity (both syntactical and semantical ones) have this property. However, having this property in the *syntax* implies that certain function arguments must be swapped when performing the substitution of beta reduction, as identified by Bernardy and Moulin [2012]. In the present system, the parametric interpretation of functions is instead merely isomorphic to a function, thanks to the In-Fun rule (Th. 11). This isomorphism (rather than equality) means on the one hand that the swapping of arguments is handled by the usual rules of logic, instead of special-purpose ones. On the other hand, obtaining the classical parametric interpretation of types requires some purely mechanical work by the user of the logic.

5.2 Parametric Models of Type Theory vs. Parametric Type Theories

Two pieces of work propose alternative parametric models of type theory [Atkey et al., 2014, Krishnaswami and Dreyer, 2013], but do not integrate parametricity in the syntax of the calculus. This means that, while certain consequences of parametricity can be made available in the logic, via constants validated by the model, parametricity itself is not available. In this paper, we not only propose a parametric model, but also show how it can be used to interpret parametricity straight up in the syntax of the type theory.

5.3 Various kinds of models

Another characterizing feature of proposals for parametricity is the kind of model underlying the semantics. Krishnaswami and Dreyer [2013] propose a model based on Q-PER. Atkey et al. [2014] propose a model based on reflexive graphs. The model that we use is based on cubes (functions from subsets of colors). In our 2012 work the cubes were reified as syntax in an underlying calculus, while in the present work they refine a presheaf structure.

5.4 Presheaf models

The presheaf construction used in this paper follows a known template, used for example by Bezem et al. [2014], Pitts [2014] to model univalence in type theory. Not only both models use a presheaf, but they also have the same underlying category **pI**. This means as all these models have an additional cubical structure. We find remarkable that cubical structures are useful for modeling both parametricity and univalence. Altenkirch and Kaposi [2014] give a syntax for Bezem et al.'s Cubical Type Theory, effectively modelling univalence by internalization of their model. The present work further refines the model by interpreting terms as I-elements, which is essential to interpret our special-purpose pairing constructions.

6 Future work and conclusion

We have defined a new type theory with internalized parametricity. Thanks to our model construction, we have proved the consistency of the system. The missing piece to construct a type-checker is a decision algorithm for the conversion relation. This checker could then be used as a minimal proof assistant for a type theory with parametricity.

Acknowledgment: The fact that the category of partial bijections **pI** should be relevant for internalization of parametricity became apparent through discussions between Thorsten Altenkirch and the second author about the paper [Bernardy and Moulin, 2012].

References

P. Aczel. On relating type theories and set theories. In *Proceedings of TYPES'98, volume 1657 of Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 1998.

T. Altenkirch and A. Kaposi. A syntax for cubical type theory. 2014. URL `http://www.cs.nott.ac.uk/~txa/publ/ctt.pdf`. Draft.

R. Atkey, N. Ghani, and P. Johann. A relationally parametric model of dependent type theory. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 503–516, 2014. 10.1145/2535838.2535852. URL `http://doi.acm.org/10.1145/2535838.2535852`.

J.-P. Bernardy and M. Lasson. Realizability and parametricity in pure type systems. In M. Hofmann, editor, *Foundations Of Software Science And Computational Structures*, volume 6604 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2011.

J.-P. Bernardy and G. Moulin. A computational interpretation of parametricity. In *LICS*. IEEE Computer Society, 2012.

J.-P. Bernardy and G. Moulin. Type-theory in color. In *Proceedings of the 18th ACM SIGPLAN international conference on Functional Programming*, pages 61–72, 2013.

J.-P. Bernardy, P. Jansson, and R. Paterson. Parametricity and dependent types. In *Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 345–356, Baltimore, Maryland, 2010. ACM. 10.1145/1863543.1863592.

J.-P. Bernardy, P. Jansson, and R. Paterson. Proofs for free — parametricity for dependent types. *Journal of Functional Programming*, 22(02):107–152, 2012. 10.1017/S0956796812000056.

M. Bezem, T. Coquand, and S. Huber. A model of type theory in cubical sets. In *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26, pages 107–128, 2014.

M. Hofmann. Syntax and semantics of dependent types. In *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997.

N. R. Krishnaswami and D. Dreyer. Internalizing relational parametricity in the extensional calculus of constructions. In *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, pages 432–451, 2013. 10.4230/LIPIcs.CSL.2013.432. URL `http://dx.doi.org/10.4230/LIPIcs.CSL.2013.432`.

A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2013. ISBN 9781107017788.

A. M. Pitts. An equivalent presentation of the bezem-coquand-huber category of cubical sets. *CoRR*, abs/1401.7807, 2014. URL `http://arxiv.org/abs/1401.7807`.

J. C. Reynolds. Types, abstraction and parametric polymorphism. *Information processing*, 83(1):513–523, 1983.

P. Wadler. Theorems for free! In *Proceedings of the fourth international conference on Functional programming languages and computer architecture*, pages 347–359, Imperial College, London, United Kingdom, 1989. ACM. ISBN 0-89791-328-0. 10.1145/99370.99404. URL `http://portal.acm.org/citation.cfm?id=99404`.