Fresh Parametricity

ERIC BOND, University of Michigan, USA
MAX NEW, University of Michigan, USA

ABSTRACT **Prop**

CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Additional Key Words and Phrases: Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Eric Bond and Max New. 2018. Fresh Parametricity. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

<eric-no "type instantiations"> <eric-fresh param instead of non-standard>

1.1 Parametricity

<eric-decide how you want to introduce parametrictiy, then work on flow> Parametricity states <eric-wording> that any term of a universal or existential type must behave uniformly for any possible type instantiation. Existential types can be used to encode abstract data types like Queues or Graphs. If we know our language enjoys parametric polymorphism, then we have a guarantee that any two instance of a Queue, as encoded by an existential type, are observationally equivalent. Meaning, we can swap implementations of the Queue data type without changing the correctness of the overall program. (ignoring resource constraints, time/space) Universal types in a parametrically polymorphic language allow for a sound implementation of church encoded data. Theorems for free <eric- Address this first, then provide examples?>

1.2 Parametricity vs Intensional Type Analysis

<eric-Dyn or ?> While parametricity is a desirable property, it is hard to maintain with certain language features that use *intensional type analysis* [5]. By Intensional type analysis, we mean the ability to inspect the run time representaiton of a type and dispatch based on the result. <eric-wording> An example of this behavior is the casting rules in a gradual typed language[9].

Authors' Contact Information: Eric Bond, bonderic@umich.edu, University of Michigan, Ann Arbor, USA; Max New, University of Michigan, Ann Arbor, USA.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1.2.1 Gradual Typing. Gradual typing allows programmers gradually migrate from <eric-dynamically/untyped> code to typed code by including a type *Dyn* representing a dynamically checked term. The interface between dynamically typed code and statically typed code is handled by casting. A value of any type A can be up cast $\uparrow^?_A$ to the dynamic type. Downcasting $\downarrow^?_A$ from the dynamic type can introduce errors if the value stored in Dyn is not of the requested type. Naively adding Dyn, up casting, and down casting to a parametrically polymorphic language breaks parametricity. Consider a polymorphic gradually typed language which can error.

$$t : \forall X.X \to \mathbb{B} := (\Lambda X.\lambda(x : X).\ \downarrow^?_{\mathbb{B}} \uparrow^?_X (x))$$

<eric-$\mathbb{B}$ and Bool used here> If we assume that t behaved uniformly for all possible type instantiations, then t should either error or return a boolean that is not dependent on the argument of type X. But this is not the case <eric-grammar> when instantiated at type Bool, t will return the given boolean value $t[\mathbb{B}](b : \mathbb{B}) \rightsquigarrow^* b$ for any other type A it will error $t[A](x : A) \rightsquigarrow^* \Omega$ Therefore t does not behave uniformly for all types. The issue here is that casting inspects the type tag used in the values stored in dyn, a case of intensional type analysis.

1.3 Preserving Parametricity

The difficulties of combining parametric polymorphism and language features relying on type analysis are well known[8][9][11][1]. Solutions that preserve parametricity usually involve a form of *dynamic sealing* or generation of *fresh type tags*. Reason being, the ability to inspect a type and dispatch based on the result is in direct conflict with the information hiding principle provided in a language with parametric polymorphism. By hiding information about the runtime representation of a type, we can restore the property that polymorphic types behave uniformly across all type instantiations.

While such techniques have demonstrated they can preserve the expected information hiding properties, they do so using a *non-standard* formulation of parametricity. Specifically, the approach used by Niels[8] and New[9] invoke a *Type-World* logical relation where the freshly allocated type variables/tags, α, are associated with concrete types, (A, A'), and a relation, $R : Rel[A, A']$, on those types. This mapping, $\alpha \mapsto (A, A', R)$, of dynamically allocated type variable to concrete types and a relation are threaded through the logical relation via a Kripke world. A question remains, what is the relative strength of this non-standard formulation of parametricity compared to traditional definitions? In particular, does the *type-world* formulation allows us to prove the *free-theorems* and *data-abstraction* properties we would should expect?<eric-introduce vocab? non-starndard param as type world param?>

1.4 Failure of Full Abstraction

This was partially answered in the affirmative by Nies et al[8]. The argument proceeds by introducing two languages: An effectful variant of CBV System F with a standard notion of parametricity, and System G, an extension of the former language with type casts and fresh type name allocation supporting a *non-standard*, type world, formulation of parametricity. To compare the languages, they provide a type preserving embedding, $[\![\_]\!]$, from System F to System G.

Theorem 1.1. *Preservation of Typing.*

$$\vdash_F V : A \implies \vdash_G [\![V]\!] : [\![A]\!]$$

The embedding, $[\![\vdash_F V : A]\!] = \mathcal{W}_A \circ [\![\vdash_F V : A]\!]^L$, is decomposed into a lifting, $[\![\_]\!]^L$, and a type directed wrapping function $\mathcal{W}_A$. The lifting preserves syntactic equality of terms and types, so $V \equiv [\![V]\!]^L$ and $A \equiv [\![A]\!]^L$. The wrapping function is a deep η expansion which strategically inserts fresh type allocation in the elimination forms of the universal and existential types. The purpose of

the wrapping is to ensure that lifted polymorphic terms of System F are simultaneously safe from non-parametric uses and forced to behave parametrically in the presence of System G's type cast operation. More details can be found in Section 2.3.

Conjecture 1.2. ***False**: Full Abstraction of* $[\![\_]\!]$

$$\forall V_1, V_2. V_1 \cong_F V_2 \implies [\![V_1]\!] \cong_G [\![V_2]\!]$$

To demonstrate the relative strengths of the different notions of parametricity, they conjecture that the embeddings is *fully-abstract*, preserving all the contextual equivalences of the original language. However, this conjecture was later disproven[3] [4]. The counter example relies on the fact that the type $Univ := \exists Y. \forall X. (X \to Y) \times (Y \to X)$ must be *degenerate* in effectful System F, but there are non-degenerate inhabitants of this type in System G.

1.5 Our Result

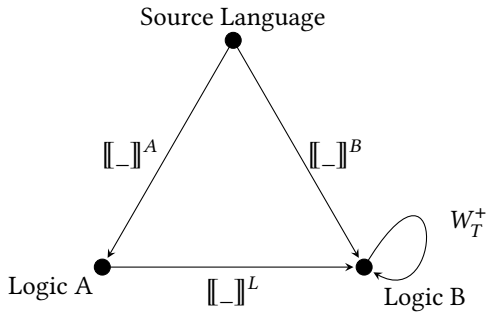
<eric-rename section, more connective tissue> Our aim is to demonstrate that, while the transformation does not preserve contextual equivalence, it preserves a weaker notion we call *Logical Equivalence*. By Logical Equivalence, we mean to say that any results proven in a *Parametricity Logic*[10][2][7] for effectful System F can be transfered to a parametricity logic backed by the non-standard, type-world, notion of parametricity.

<eric-TODO: rename "Logic A" and "Logic B"(source/target?)>

Theorem 1.3. *Logical Equivalence:*

$$\Gamma \vdash_A M = N \implies [\![\Gamma]\!] \vdash_B [\![M]\!] = [\![N]\!]$$

<eric-placeholder figure, refactor>



1.6 Overview

<eric-bullet format or paragraph?> To demonstrate the preservation of logical equivalence we first define an effectful CBV variant of System F in Section 2.1 for which we want our theorem to apply to. We then define the *object languages* of our paremetricity logics in Section 2.2 which are effectful, CBPV [6] variants of System F. We then define the translations of the source language to each of these object languages <eric-cref> . The translations will largely follow the usual[6] CBV to CBPV translation, with a few exceptions, notably the parametricity preserving wrapping in Section 2.3. In Section 3, we will introduce our parametricity logics<eric-cref> and the translation <eric-cref> between the logics. <eric-compositionality issues.. oblivious predicate..adequacy..example proof in Logic..discussion/relatedwork.. order TBD here.. come back to this>

2 LANGUAGES

- section overview.
- why have a source language
- why CBPV

2.1 Source Language

<eric-remove existential for now>

- why "jumbo" connective (not jumbo, just greater airity)
- why general recursion and error? (OSum can encode these)

$$\frac{}{\Gamma, X \vdash X\ Type} \qquad \frac{}{\Gamma \vdash \mathbb{B}\ Type} \qquad \frac{}{\Gamma \vdash \mathbb{N}\ Type} \qquad \frac{\Gamma \vdash A\ Type \qquad \Gamma \vdash A'\ Type}{\Gamma \vdash A \times A'\ Type} \qquad \frac{\Gamma, X \vdash A\ Type}{\Gamma \vdash \exists X.A}$$

$$\frac{\Gamma, \overrightarrow{X}_n \vdash A_i\ Type\ \ \forall i \in \{m\} \qquad \Gamma, \overrightarrow{X}_n \vdash A\ Type}{\Gamma \vdash \forall[\overrightarrow{X}_n].(\overrightarrow{A}_m) \to A\ Type}$$

Fig. 1. Source Language Type Formers

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \vdash true : \mathbb{B}} \qquad \frac{}{\Gamma \vdash false : \mathbb{B}} \qquad \frac{\Gamma \vdash b : \mathbb{B} \qquad \Gamma \vdash M : A \qquad \Gamma \vdash N : A}{\Gamma \vdash rec_{\mathbb{B}}\ b\ \{M \mid N\} : A}$$

$$\frac{}{\Gamma \vdash z : \mathbb{N}} \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash s\ n : \mathbb{N}} \qquad \frac{\Gamma \vdash n : \mathbb{N} \qquad \Gamma \vdash M : A \qquad \Gamma, x : \mathbb{N} \vdash N : A}{\Gamma \vdash rec_{\mathbb{N}}\ n\ \{M \mid x.\ N\} : A}$$

$$\frac{\Gamma, \overrightarrow{X}_n \vdash A_i\ Type\ \ \forall i \in \{m\} \qquad \Gamma, \overrightarrow{X}_n \vdash A'\ Type \qquad \Gamma, \overrightarrow{X}_n, x_1 : A_1, \cdots, x_m : A_m \vdash M : A'}{\Gamma \vdash \Lambda[\overrightarrow{X}_n](x_1 : A_1, \cdots, x_m : A_m).M : \forall[\overrightarrow{X}_n].(\overrightarrow{A}_m) \to A'}$$

$$\frac{\Gamma \vdash A_i\ Type\ \ \forall i \in \{n\} \qquad \Gamma \vdash M : \forall[\overrightarrow{X}_n].(\overrightarrow{A}_m) \to A' \qquad \Gamma \vdash N_i : B_i[\overrightarrow{A}/\overrightarrow{X}]\ \forall i \in \{m\}}{\Gamma \vdash M[\overrightarrow{A}_n](N_1 : B_1, \cdots N_m : B_m) : A'}$$

$$\frac{\Gamma \vdash V : T[A/X]}{\Gamma \vdash pack\langle A, V\rangle : \exists X.T} \qquad \frac{\Gamma \vdash W : \exists X.T \qquad \Gamma, A, V : T[A/X] \vdash M : A'}{\Gamma \vdash unpack\langle A, V\rangle = W; M : A'} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : A}{\Gamma \vdash (M, N) : A \times A}$$

$$\frac{\Gamma \vdash M : A \times A' \qquad \Gamma, x : A, y : A' \vdash N : A}{\Gamma \vdash rec_{\times}\ M\ \{x, y.\ N\} : A''} \qquad \frac{}{\Gamma \vdash \Omega_A : A} \qquad \frac{\Gamma, x : A \vdash V : A'}{\Gamma \vdash \mu x.V : A'}$$

Fig. 2. Source Language Typing Rules

2.2 Object Languages

- remark: mixed term/type context
- explain: Why $\forall \underline{X}.\underline{B}$ and no F? Justify church encoding here or later? Provide definition of church encoded *ret* and *bind*? will likely use later..
- describe: *OSum*, *Case A*
- remark: how *new* is presented as an effect? (its church encoding)

- remark: inclusion of Observation type (b/c church encoded F)
- <eric-TODO: remove exists>
- <eric-TODO: include complex values (disclose motivation? the wrapping definition)>
- <eric-operational behavior..? perhaps not if we don't discuss models>
- <eric-equational theory here? or in the Logic section?> equations link

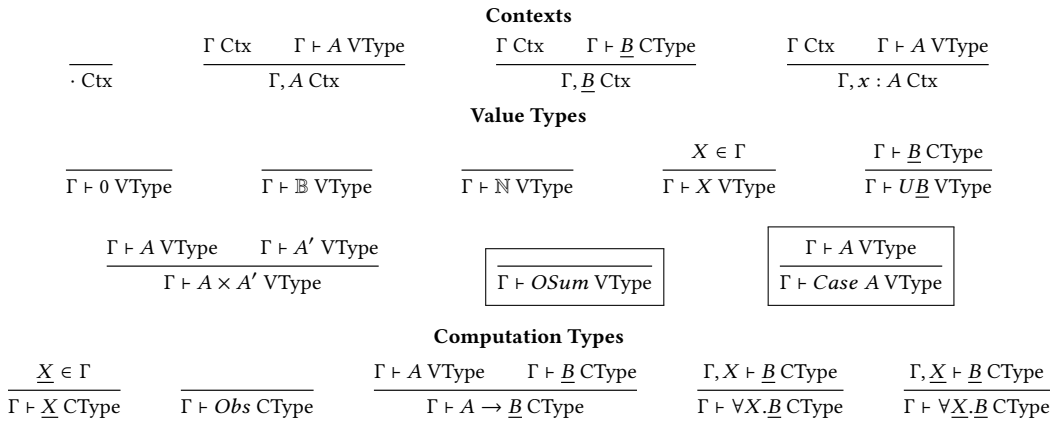


Fig. 3. Object Language Contexts, Value Types, and Computation Types

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \mid x : \underline{B} \vdash x : \underline{B}} \qquad \frac{}{\Gamma \vdash absurd : 0 \to \underline{B}}$$

$$\frac{}{\Gamma \vdash true : \mathbb{B}} \quad \frac{}{\Gamma \vdash false : \mathbb{B}} \quad \frac{\Gamma \vdash b : \mathbb{B} \qquad \Gamma \mid \cdot \vdash M : \underline{B} \qquad \Gamma \mid \cdot \vdash N : \underline{B}}{\Gamma \mid \cdot \vdash rec_{\mathbb{B}}\ b\{M|N\} : \underline{B}}$$

$$\frac{}{\Gamma \vdash z : \mathbb{N}} \quad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash s\ n : \mathbb{N}} \quad \frac{\Gamma \vdash n : \mathbb{N} \qquad \Gamma \mid \cdot \vdash M : \underline{B} \qquad \Gamma, N : U\underline{B} \mid \cdot \vdash N' : \underline{B}}{\Gamma \mid \cdot \vdash rec_{\mathbb{N}}\ n\{M|N.N'\} : \underline{B}} \qquad \frac{\Gamma \mid \cdot \vdash M : \underline{B}}{\Gamma \vdash thunk\ M : U\underline{B}} \quad \frac{\Gamma \vdash V : U\underline{B}}{\Gamma \mid \cdot \vdash force\ V : \underline{B}}$$

$$\frac{\Gamma \vdash V : A \qquad \Gamma \vdash W : A'}{\Gamma \vdash (V, W) : A \times A'} \quad \frac{\Gamma \vdash V : A \times A' \qquad \Gamma, x : A, y : A' \mid \cdot \vdash M : \underline{B}}{\Gamma \mid \cdot \vdash rec_{\times} V\{x, y.M\} : \underline{B}}$$

$$\frac{\Gamma \vdash V : T[A/X]}{\Gamma \vdash pack\langle A, V\rangle : \exists X.T} \quad \frac{\Gamma \vdash W : \exists X.T \qquad \Gamma, A, V : T[A/X] \mid \cdot \vdash M : \underline{B}}{\Gamma \mid \cdot \vdash unpack\langle A, V\rangle = W; M : \underline{B}}$$

$$\frac{\Gamma \vdash V : T[\underline{B}/\underline{X}]}{\Gamma \vdash pack\langle \underline{B}, V\rangle : \exists \underline{X}.T} \quad \frac{\Gamma \vdash W : \exists \underline{X}.T \qquad \Gamma, \underline{B}, V : T[\underline{B}/\underline{X}] \mid \cdot \vdash M : \underline{B'}}{\Gamma \mid \cdot \vdash unpack\langle \underline{B}, V\rangle = W; M : \underline{B'}}$$

$$\frac{\Gamma \vdash \sigma : Case\ A \qquad \Gamma \vdash V : A}{\Gamma \vdash inj_{\sigma}\ V : OSum} \qquad \frac{\Gamma \vdash \sigma : Case\ A \qquad \Gamma \vdash V : OSum \qquad \Gamma, x : A \mid \cdot \vdash M : \underline{B} \qquad \Gamma \mid \cdot \vdash N : \underline{B}}{\Gamma \mid \cdot \vdash rec_{OSum}\ \sigma, V\{x.M|N\} : \underline{B}}$$

$$\frac{\Gamma \vdash b : \mathbb{B}}{\Gamma \mid \cdot \vdash hault\ b : Obs} \quad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \mid \cdot \vdash hault\ n : Obs} \qquad \frac{}{\Gamma \mid \cdot \vdash \Omega : F0} \qquad \frac{}{\Gamma \mid \cdot \vdash fix : \forall \underline{B}.U(U\underline{B} \to \underline{B}) \to \underline{B}}$$

$$\frac{\Gamma, x : A \mid \cdot \vdash M : \underline{B}}{\Gamma \mid \cdot \vdash \lambda(x : A).M : A \to \underline{B}} \quad \frac{\Gamma \mid \Delta \vdash M : A \to \underline{B} \qquad \Gamma \vdash V : A}{\Gamma \mid \Delta \vdash MV : \underline{B}}$$

$$\frac{\Gamma, Y \mid \cdot \vdash M : \underline{B}[Y/X] \qquad Y \notin ftv(\Gamma)}{\Gamma \mid \cdot \vdash \Lambda Y.M : \forall X.\underline{B}} \quad \frac{\Gamma \mid \Delta \vdash M : \forall X.\underline{B}}{\Gamma \mid \Delta \vdash M[A] : \underline{B}[A/X]}$$

$$\frac{\Gamma, \underline{Y} \mid \cdot \vdash M : \underline{B}[\underline{Y}/\underline{X}] \qquad \underline{Y} \notin ftv(\Gamma)}{\Gamma \mid \cdot \vdash \Lambda \underline{Y}.M : \forall \underline{X}.\underline{B}} \quad \frac{\Gamma \mid \Delta \vdash M : \forall \underline{X}.\underline{B}}{\Gamma \mid \Delta \vdash M[\underline{B'}] : \underline{B}[\underline{B'}/\underline{X}]}$$

Fig. 4. Object Language Typing

2.3 Wrapping

- explain: interesting case, $\forall X.\underline{B}$
- example: demonstrate the protection wrapping provides (use this example?)
- explain: why we get rid of polarity
- remark: on usage of complex values?
- <eric-TODO: remove exists>

$$
\begin{aligned}
&\mathcal{W}_X(t) \coloneqq t\\
&\mathcal{W}_{\mathbb{B}}(t) \coloneqq t\\
&\mathcal{W}_{\mathbb{N}}(t) \coloneqq t\\
&\mathcal{W}_{Obs}(t) \coloneqq t\\
&\mathcal{W}_{A\times A'}(t) \coloneqq rec_\times\ t\ \{x, y.\ (\mathcal{W}_A(x), \mathcal{W}_{A'}(y))\}\\
&\mathcal{W}_{\exists X.A}(t) \coloneqq unpack\langle X, x\rangle = t;\\
&\quad pack\langle X, thunk(\lambda.\sigma_X.\sigma'_X \leftarrow new_X; x \leftarrow (force\ x)(\sigma'_X); ret\ \mathcal{W}_A(x))\rangle\\
&\mathcal{W}_{U\underline{B}}(t) \coloneqq thunk(\mathcal{W}_{\underline{B}}(force\ t))\\
\\
&\mathcal{W}_{\underline{X}}(t) \coloneqq t\\
&\mathcal{W}_{A\to\underline{B}}(t) \coloneqq \lambda(x : A).\mathcal{W}_{\underline{B}}(t(\mathcal{W}_A(x)))\\
&\mathcal{W}_{\forall\underline{X}.\underline{B}}(t) \coloneqq \Lambda\underline{X}.\mathcal{W}_{\underline{B}}(t[\underline{X}])\\
&\mathcal{W}_{\forall X.\underline{B}}(t) \coloneqq \Lambda X.\lambda\sigma_X.\mathcal{W}_{\underline{B}}(\sigma'_X \leftarrow new_X;\ t[X](\sigma'_X))
\end{aligned}
$$

Fig. 5. Wrapping

2.4 Translations

•

$$
\begin{aligned}
[\![\Gamma \vdash X]\!] &= X\\
[\![\Gamma \vdash \mathbb{B}]\!] &= \mathbb{B}\\
[\![\Gamma \vdash \mathbb{N}]\!] &= \mathbb{N}\\
[\![\Gamma \vdash A \times A']\!] &= [\![\Gamma \vdash A]\!] \times [\![\Gamma \vdash A']\!]\\
[\![\Gamma \vdash \exists X.A]\!] &= \exists X.UF([\![\Gamma, X \vdash A]\!])\\
[\![\Gamma \vdash \forall[\overrightarrow{X}_n].(\overrightarrow{A}_m) \to A']\!] &= U\Big(\forall X_1 \cdots X_n.\ ([\![\Gamma, \overrightarrow{X} \vdash A_1]\!], \ldots, [\![\Gamma, \overrightarrow{X} \vdash A_m]\!]) \to F[\![\Gamma, \overrightarrow{X} \vdash A']\!]\Big)
\end{aligned}
$$

Fig. 6. Source Types to Logic A Types

$$
\begin{aligned}
[\![\emptyset]\!]^L_{ctx} &= \emptyset \\
[\![\Gamma, X]\!]^L_{ctx} &= [\![\Gamma]\!]^L_{ctx}, X, \sigma_X : Case\ X \\
[\![\Gamma, \underline{X}]\!]^L_{ctx} &= [\![\Gamma]\!]^L_{ctx}, \underline{X}, \underline{X}' \\
[\![\Gamma, x : A]\!]^L_{ctx} &= [\![\Gamma]\!]^L_{ctx}, x : [\![\Gamma \vdash A]\!]^L_{ty}
\end{aligned}
$$

$$
\begin{aligned}
[\![\Gamma \vdash \mathbb{B}]\!]^L_{ty} &= \mathbb{B} \\
[\![\Gamma \vdash \mathbb{N}]\!]^L_{ty} &= \mathbb{N} \\
[\![\Gamma \vdash X]\!]^L_{ty} &= X \quad (X \in \Gamma) \\
[\![\Gamma \vdash U\underline{B}]\!]^L_{ty} &= U[\![\Gamma \vdash \underline{B}]\!]^L_{ty} \\
[\![\Gamma \vdash A \times A']\!]^L_{ty} &= [\![\Gamma \vdash A]\!]^L_{ty} \times [\![\Gamma \vdash A']\!]^L_{ty} \\
[\![\Gamma \vdash \exists X.A]\!]^L_{ty} &= \exists X.U(Case\ X \to F[\![\Gamma, X \vdash A]\!]^L_{ty}) \\
[\![\Gamma \vdash \exists \underline{X}.A]\!]^L_{ty} &= \exists \underline{X}.[\![\Gamma, \underline{X} \vdash A]\!]^L_{ty} \\
[\![\Gamma \vdash \underline{X}]\!]^L_{ty} &= \underline{X} \quad (\underline{X} \in \Gamma) \\
[\![\Gamma \vdash Obs]\!]^L_{ty} &= Obs \\
[\![\Gamma \vdash A \to \underline{B}]\!]^L_{ty} &= [\![\Gamma \vdash A]\!]^L_{ty} \to [\![\Gamma \vdash \underline{B}]\!]^L_{ty} \\
[\![\Gamma \vdash \forall X.\underline{B}]\!]^L_{ty} &= \forall X.Case\ X \to [\![\Gamma, X \vdash \underline{B}]\!]^L_{ty} \\
[\![\Gamma \vdash \forall \underline{X}.\underline{B}]\!]^L_{ty} &= \forall \underline{X}.[\![\Gamma, \underline{X} \vdash \underline{B}]\!]^L_{ty} \\
[\![\Gamma \vdash FA]\!]^L_{ty} &= F[\![\Gamma \vdash A]\!]^L_{ty}
\end{aligned}
$$

Fig. 7. Logic Translation - Contexts & Types

Values:

$$\begin{aligned}
[\![\Gamma \vdash x]\!] &= x \quad (x \in [\![\Gamma]\!]^L_{ctx}) \\
[\![\Gamma \vdash true]\!] &= true \\
[\![\Gamma \vdash false]\!] &= false \\
[\![\Gamma \vdash z]\!] &= z \\
[\![\Gamma \vdash sV]\!] &= s[\![\Gamma \vdash V]\!] \\
[\![\Gamma \vdash thunk\ M]\!] &= thunk\ [\![\Gamma \mid \cdot \vdash M]\!] \\
[\![\Gamma \vdash (V, W)]\!] &= ([\![\Gamma \vdash V]\!], [\![\Gamma \vdash W]\!]) \\
[\![\Gamma \vdash pack\langle A, V\rangle]\!] &= pack\langle [\![\Gamma \vdash A]\!], thunk(\lambda\sigma : Case\ [\![\Gamma \vdash A]\!].\ ret\ [\![\Gamma \vdash V]\!])\rangle \\
[\![\Gamma \vdash pack\langle \underline{B}, V\rangle]\!] &= pack\langle [\![\Gamma \vdash \underline{B}]\!], [\![\Gamma \vdash V]\!]\rangle
\end{aligned}$$

Computations:

$$\begin{aligned}
[\![\Gamma \vdash rec_{\mathbb{B}}\ V\{M \mid N\}]\!] &= rec_{\mathbb{B}}\ [\![\Gamma \vdash V]\!]\ \{[\![\Gamma \vdash M]\!] \mid [\![\Gamma \vdash N]\!]\} \\
[\![\Gamma \vdash rec_{\mathbb{N}}\ V\ \{M \mid x.N\}]\!] &= rec_{\mathbb{N}}\ [\![\Gamma \vdash V]\!]\ \{[\![\Gamma \vdash M]\!] \mid x.[\![\Gamma, x \vdash N]\!]\} \\
[\![\Gamma \vdash fix]\!] &= fix \\
[\![\Gamma \vdash force\ V]\!] &= force\ [\![\Gamma \vdash V]\!] \\
[\![\Gamma \vdash hault\ V]\!] &= hault\ [\![\Gamma \vdash V]\!] \\
[\![\Gamma \vdash \lambda x : A.M]\!] &= \lambda x : [\![\Gamma \vdash A]\!].[\![\Gamma, x : A \vdash M]\!] \\
[\![\Gamma \vdash MV]\!] &= [\![\Gamma \vdash M]\!][\![\Gamma \vdash V]\!] \\
[\![\Gamma \vdash \Lambda X.M]\!] &= \Lambda X.\lambda\sigma : Case\ X.[\![\Gamma, X \mid \Delta \vdash M]\!] \\
[\![\Gamma \vdash M[A]]\!] &= \sigma \leftarrow new_{[\![\Gamma \vdash A]\!]}; [\![\Gamma \mid \Delta \vdash M]\!][[\![\Gamma \vdash A]\!]](\sigma) \\
[\![\Gamma \vdash \Lambda\underline{X}.M]\!] &= \Lambda\underline{X}.[\![\Gamma, \underline{X} \vdash M]\!] \\
[\![\Gamma \vdash M[\underline{B}]]\!] &= [\![\Gamma \vdash M]\!][[\![\Gamma \vdash \underline{B}]\!]] \\
[\![\Gamma \vdash rec_{\times}\ V\ \{x, y.\ M\}]\!] &= rec_{\times}\ [\![\Gamma \vdash V]\!]\ \{x, y.\ [\![\Gamma, x, y \vdash M]\!]\} \\
[\![\Gamma \vdash unpack\langle A, V\rangle = W; M]\!] &= unpack\langle A, V\rangle = [\![\Gamma \vdash W]\!]; \sigma \leftarrow new_A; v \leftarrow (force\ V)\sigma; [\![\Gamma, A, v \mid \cdot \vdash M]\!] \\
[\![\Gamma \vdash unpack\langle \underline{B}, V\rangle = W; M]\!] &= unpack\langle \underline{B}', V'\rangle = [\![\Gamma \vdash W]\!]; [\![\Gamma, \underline{B}', V' \vdash M]\!]
\end{aligned}$$

Fig. 8. Logic Translation - Terms

3 LOGICS

4 LOGICAL EQUIVALENCE

5 EXAMPLE PROOFS

6 DISCUSSION/RELATED WORK

7 REFLECTION/CONCLUSION

REFERENCES

[1] Amal Ahmed, Dustin Jamner, Jeremy G. Siek, and Philip Wadler. 2017. Theorems for free for free: parametricity, with and without types. *Proc. ACM Program. Lang.* 1, ICFP, Article 39 (Aug. 2017), 28 pages. https://doi.org/10.1145/3110283

[2] Lars Birkedal, Rasmus Ejlers Møgelberg, and Rasmus Lerchedahl Petersen. 2006. Linear Abadi and Plotkin Logic. *Log. Methods Comput. Sci.* 2 (2006). https://api.semanticscholar.org/CorpusID:14086681

[3] Dominique Devriese, Marco Patrignani, and Frank Piessens. 2017. Parametricity versus the universal type. *Proc. ACM Program. Lang.* 2, POPL, Article 38 (Dec. 2017), 23 pages. https://doi.org/10.1145/3158126

[4] Dominique Devriese, Marco Patrignani, and Frank Piessens. 2022. Two Parametricities Versus Three Universal Types. *ACM Trans. Program. Lang. Syst.* 44, 4, Article 23 (Sept. 2022), 43 pages. https://doi.org/10.1145/3539657

[5] Robert Harper and Greg Morrisett. 1995. Compiling polymorphism using intensional type analysis. In *Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Francisco, California, USA)

(POPL '95). Association for Computing Machinery, New York, NY, USA, 130–141. https://doi.org/10.1145/199448.199475

[6] Paul Blain Levy. 2004. *Call-By-Push-Value: A Functional/Imperative Synthesis (Semantics Structures in Computation, V. 2)*. Kluwer Academic Publishers, USA.

[7] Rasmus Ejlers Møgelberg and Alex Simpson. 2008. A Logic for Parametric Polymorphism with Effects. In *Types for Proofs and Programs*, Marino Miculan, Ivan Scagnetto, and Furio Honsell (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 142–156.

[8] Georg Neis, Derek Dreyer, and Andreas Rossberg. 2009. Non-parametric parametricity. *SIGPLAN Not.* 44, 9 (Aug. 2009), 135–148. https://doi.org/10.1145/1631687.1596572

[9] Max S. New, Dustin Jamner, and Amal Ahmed. 2019. Graduality and parametricity: together again for the first time. *Proc. ACM Program. Lang.* 4, POPL, Article 46 (Dec. 2019), 32 pages. https://doi.org/10.1145/3371114

[10] Gordon D. Plotkin and Martín Abadi. 1993. A Logic for Parametric Polymorphism. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications (TLCA '93)*. Springer-Verlag, Berlin, Heidelberg, 361–375.

[11] Matías Toro, Elizabeth Labrada, and Éric Tanter. 2019. Gradual parametricity, revisited. *Proc. ACM Program. Lang.* 3, POPL, Article 17 (Jan. 2019), 30 pages. https://doi.org/10.1145/3290330

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009