Deciding equivalence with sums and the empty type

Gabriel Scherer

Deciding equivalence with sums and the empty type

Gabriel Scherer
Northeastern University, USA
gabriel.scherer@gmail.com

Abstract

The logical technique of *focusing* can be applied to the λ-calculus; in a simple type system with atomic types and negative type formers (functions, products, the unit type), its normal forms coincide with $\beta\eta$-normal forms. Introducing a *saturation* phase gives a notion of quasi-normal forms in presence of positive types (sum types and the empty type). This rich structure let us prove the decidability of $\beta\eta$-equivalence in presence of the empty type, the fact that it coincides with contextual equivalence, and a finite model property.

1. Introduction

1.1 Notion of equivalences

For a given type system, there may be several notions of program equivalence of interest. One may define a notion of syntactic equivalence by a system of equations between terms, such as β-equivalence and η-equivalence, or their union $\beta\eta$-equivalence. A more extensional notion of equivalence is contextual or observational equivalence, which checks that the two terms behave in the same way under all contexts. Finally, a semantic notion of equivalence is given by interpreting terms in a certain mathematical space, typically morphisms in categories with a certain structure, and considering two terms equivalent if they are equal under all interpretations. One can generally prove that considering all interpretations in the category of sets suffices to distinguish two observably distinct terms, and certain type systems have the stronger *finite model* property that considering all interpretations in *finite* sets suffices to distinguish observably distinct terms.

Contextual equivalence has a clear, compact definition, it is the notion of equivalence that corresponds to programmers' intuition, but it is difficult to prove and reason about. Syntactic equivalence can be used as part of syntactic typing judgments, and may be easier to prove decidable. Semantic models provide a more abstract point of view on the identity of programs, and may enable the powerful method of normalization-by-evaluation.

For the β-normalizable fragment of the untyped λ-calculus, Böhm (1968) proved that $\beta\eta$-equivalence and observational equivalence coincide. On arbitrary untyped terms, the two relations are undecidable.

For the simply-typed λ-calculus with atomic types, functions and pairs – what we call the *negative* fragment – we also know that these notions of equivalence coincide. Furthermore, typed equivalence is decidable: we can define and compute β-short η-long normal forms, and two terms are $\beta\eta$-equivalent exactly when they have the same normal form. Friedman (1975) proved that two terms equal in all set-theoretic models are equivalent, and Statman (1982) sharpened the result by proving that finite sets suffice to distinguish inequivalent terms – the finite model property.

This pleasant setting is quickly eroded by moving to richer programming languages and type systems. Adding notions of side-effects, for example, makes the general $\beta\eta$-equivalence unsound. Even in the realm of pure, total programming languages, adding parametric polymorphism (System F and beyond) makes $\beta\eta$-equivalence strictly weaker than contextual equivalence.

1.2 Sums and empty type

An interesting middle-ground is the *full* simply-typed λ-calculus, with not only atomic types, functions, pairs and the unit type 1 but also sum types and the empty type 0. There, results on program equivalence have been surprisingly long to come, because of deep difficulties caused by mixing functions (negative connectives) and sums (positive connectives), and the strong behavior of the empty type: in an inconsistent context, all terms are equal.

The first decidability result for the system with non-empty sums was Ghani (1995), using advanced rewriting techniques later simplified by Lindley (2007). It was followed by normalization-by-evaluation results (Altenkirch, Dybjer, Hofmann, and Scott 2001; Balat, Di Cosmo, and Fiore 2004) that also established decidability of equivalence in the non-empty case using the categorical structure introduced in Fiore and Simpson (1999). Decidability in the presence of the empty type is still open – we propose a proof.

Finally, the only known completeness result for set-theoretic models is Dougherty and Subrahmanyam (2000); it only holds for non-empty sums, and relies in an essential way on infinite sets. The finite model property is only conjectured – we propose a proof, including in presence of the empty type.

1.3 Focusing

Focusing (Andreoli 1992) is a general technique that uses the notion of invertibility of inference rules to define a *focused* subset of any logic that is complete but removes some redundant proofs.

A recent branch of work on *maximal multi-focusing* (Chaudhuri, Miller, and Saurin 2008a; Chaudhuri, Hetzl, and Miller 2012) has demonstrated that focused proofs can be further restricted to become even more canonical: in each application to a specific logic, the resulting representations are equivalent to existing representations capturing the identity of proofs – proof nets for linear logic, expansion proofs for classical logic.

Scherer and Rémy (2015) applied focusing to the λ-calculus with non-empty sums. Completeness of focusing there means that any term has a $\beta\eta$-equivalent focused term. Their counterpart of maximal multi-focusing is *saturation*: saturated terms of the focused λ-calculus introduce and deconstruct neutral terms of sum type as early as possible – when they are *new*, they were not derivable in the previous saturation phase. Canonicity of this representation gives yet another decision procedure for $\beta\eta$-equivalence of λ-terms with non-empty sums.

The present work extends saturated focusing to the full simply-typed lambda calculus, with units and in particular the empty type. The suitably extended notion of saturated form retains its canonicity properties in the full system. This means that $\beta\eta$-equivalence with empty types is decidable – by converting terms to their saturated focused form. From two distinct saturated forms, one can furthermore build a well-typed context that distinguishes them.

This proves that contextual equivalence implies equality of normal forms, and thus $\beta\eta$-equivalence: any term is $\beta\eta$-equivalent to its normal form, so two terms with the same normal form are $\beta\eta$-equivalent. Our distinguishing context needs only instantiate atomic types with finite sets, giving a finite model property.

Extending saturated focusing to the empty type requires adding a requirement that saturation phases be complete for provability: not only must they introduce all new neutrals of positive type for use in the rest of the term, but they must at least introduce one such neutral for each type deducible in the current context. As a consequence, we can prove a Saturation Consistency theorem which is key to our canonicity: if two saturated terms are distinct, then their context must be consistent.

1.4 Contributions

We establish the following results in the simply-typed λ-calculus with atoms, functions, pairs, the unit type, sums and the empty type $\mathsf{AC}(X, \to, \times, 1, +, 0)$:

- Saturated terms provide a notion of *quasi*-normal form; equivalent quasi-normal forms are not necessarily α-equivalent, but are related by a local, easily decidable relation of *invertible commutation conversions* ($\approx_{\mathtt{icc}}$).
- $\beta\eta$-equivalence is decidable.
- $\beta\eta$-equivalence and contextual equivalence coincide, along with set-theoretic equivalence in all models where atomic types are interpreted by closed types.
- The finite model property holds – as closed types in this system have finitely many inhabitants.

Our notion of $\beta\eta$-equivalence is the *strong* equivalence on sums. It corresponds to equality of morphisms in the free bi-cartesian closed category – see Scherer (2016), Section 3.2.2.

1.5 Plan

Section 2 (Equivalences in the full λ-calculus) introduces the full λ-calculus we shall consider in this work, along with the various notions of equivalence ($\beta\eta$, contextual, set-theoretic) we will discuss. We prove some elementary results: $\beta\eta$-equivalence implies contextual equivalence in all closed models, which coincides with set-theoretic equivalence in all closed models.

Section 3 (Focusing) presents focusing, before detailing the focused λ-calculus extended with the unit type and empty type. We formulate the computational counterpart of the completeness theorem: any λ-term has a $\beta\eta$-equivalent focused term.

Section 4 (Saturated focused λ-calculus) presents the saturated subset of the focused λ-calculus as defined by a different system of inference rules. Compared to the simpler system of Scherer and Rémy (2015), this saturated system is parametrized over a *selection function* that selects the neutrals to split over during saturation. The saturated system is, again, computationally complete with respect to focused terms.

Section 5 (Saturation consistency) establishes the main meta-theoretic property of saturated terms in presence of the empty type, namely that saturating an inconsistent context will always find a proof of 0. In other words, if two saturated terms differ after a saturation phase instead of both ending on an absurdity elimination, we know that they differ in consistent contexts. This result is key to extending the distinguishability result of Dougherty and Subrahmanyam (2000) to a system with empty types.

Finally, Section 6 (Canonicity) establishes the central result of this work: if two saturated λ-terms $t \not\approx_{\mathtt{icc}} u$ are syntactically distinct (modulo invertible commuting conversions), then there exists a closed type model $\mathcal{M}$ in which we have a distinguishing context $C\,[\Box]$ such $C\,[\mathcal{M}(t)], C\,[\mathcal{M}(u)]$ are closed booleans $(1 + 1)$, one of them equal to `true` and the other to `false`. By contraposition, this proves that two terms contextually equivalent in all models have the same saturated normal forms – giving decidability – and in particular are $\beta\eta$-equivalent.

For lack of space, our statements do not come with their proofs, but proof outlines are given in Appendix B (Proof outlines).

Sections 1 to 5, in particular the presentations of focusing and saturated normal forms, correspond to content that was presented in the thesis manuscript Scherer (2016). In the present article, definitions and proofs in these sections are given with minimal amount of details for economy of space; they are presented in full in the manuscript. In contrast, Section 6, which presents the key canonicity result enabling the contributions of this article, is entirely new, and presented in more detail.

2. Equivalences in the full λ-calculus

2.1 Typing rules and $\beta\eta$-equivalence

$$A, B, C ::= X, Y, Z \mid A \to B \mid A_1 \times A_2 \mid 1 \mid A_1 + A_2 \mid 0$$

$$t, u, r ::= x, y, z \mid \lambda x.\, t \mid t\ u \mid (t_1, t_2) \mid \pi_i\ t \mid () \mid \sigma_i\ t \mid \mathtt{match}\ t\ \mathtt{with}\ (\sigma_i\ x_i \to u_i)^i \mid \mathtt{absurd}(t)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.\, t : A \to B} \qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash (t, u) : A \times B} \qquad \frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \pi_i\ t : A_i}$$

$$\frac{\Gamma \vdash t : A_i}{\Gamma \vdash \sigma_i\ t : A_1 + A_2} \qquad \frac{\Gamma \vdash t : A_1 + A_2 \qquad (\Gamma, x_i : A_i \vdash u_i : C)^i}{\Gamma \vdash \mathtt{match}\ t\ \mathtt{with}\ (\sigma_i\ x_i \to u_i)^i : C}$$

$$\frac{}{\Gamma \vdash () : 1} \qquad \frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash t : 0}{\Gamma \vdash \mathtt{absurd}(t) : A}$$

Figure 1. Full simply-typed lambda-calculus $\mathsf{AC}(X, \to, \times, 1, +, 0)$

$$(\lambda x.\, t)\ u \triangleright_\beta t[u/x] \qquad (t : A \to B) \triangleright_\eta \lambda(x : A).\, t\ x$$

$$\pi_i\ (t_1, t_2) \triangleright_\beta t_i \qquad (t : A_1 \times A_2) \triangleright_\eta (\pi_1\ t, \pi_2\ t)$$

$$\mathtt{match}\ \sigma_j\ t\ \mathtt{with}\ (\sigma_i\ x_i \to u_i)^i \triangleright_\beta u_j[t/x_j]$$

$$t[u : A_1 + A_2/x] \triangleright_\eta \mathtt{match}\ u\ \mathtt{with}\ (\sigma_i\ y_i \to t[\sigma_i\ y_i/x])^i$$

$$\frac{\Gamma \vdash t : 1}{\Gamma \vdash t \triangleright_\eta () : 1} \qquad \frac{\Gamma \vdash u : 0 \qquad \Gamma, x : 0 \vdash t : A}{\Gamma \vdash t[u/x] \triangleright_\eta \mathtt{absurd}(u) : A}$$

Derived rules: $\dfrac{}{\Gamma \vdash t_1 \approx_\eta t_2 : 1} \qquad \dfrac{\Gamma \vdash u : 0}{\Gamma \vdash t_1 \approx_\eta t_2 : A}$

Figure 2. $\beta\eta$-equivalence for $\mathsf{AC}(X, \to, \times, 1, +, 0)$

Figure 1 gives the grammar and typing rules for the full simply-typed λ-calculus $\mathsf{AC}(X, \to, \times, 1, +, 0)$. By symmetry with the pair projections $\pi_i\ t$, we use $\sigma_i\ t$ for sum injection. We use $(\dots)^{i \in I}$ for a family of objects indexed by $i \in I$. The common indexing family, when dealing with syntactic binary operations, is $\{1, 2\}$, and we will most of the time leave it implicit. Finally, $\mathtt{match}\ t\ \mathtt{with}\ (\sigma_i\ x_i \to u_i)^i$ is a compact syntax for our full sum-elimination syntax, $(\mathtt{match}\ t\ \mathtt{with}\ \mid \sigma_1\ x_1 \to u_1 \mid \sigma_2\ x_2 \to u_2)$.

Definition 1. *A* closed *type does not contain atomic types.*[1]

Definition 2. *We write* $\Gamma \vdash A$ *when* t *exists such that* $\Gamma \vdash t : A$.

We define $\beta\eta$-equivalence as the smallest congruent relation ($\approx_{\beta\eta}$) closed by the union of the β-reduction relation ($\triangleright_\beta$) and the η-expansion relation ($\triangleright_\eta$) of Figure 2. We have not explicited the full typing assumptions, but only type-preserving rewrites are considered. The derived rules are not primitives, they are derivable from η-expansion at unit and empty type.

If t_1, t_2 are of type 1, then they both rewrite to $()$ and are thus equal. The derivation for equality under absurdity is less direct: if the current typing context Γ is inconsistent, that is there is a derivation $(u : 0)$, then any term of *any* type $(t : A)$ can be seen as the result of the substitution $t[u/x]$ for a variable x that does not appear in t, and is thus equal to $\texttt{absurd}(u)$; in particular, any two terms of the same type are equal.

Theorem 1 (Strong normalization)**.** *β-reduction is strongly normalizing in the full simply-typed λ-calculus $\mathsf{\Lambda C}(X, \to, \times, 1, +, 0)$.*

Theorem 2 (Confluence)**.** *β-reduction is confluent for the full simply-typed λ-calculus $\mathsf{\Lambda C}(X, \to, \times, 1, +, 0)$: each term has a unique β-short normal form.*

Lemma 1 (Inversion)**.** *A closed β-normal form (in an empty context) starts with an introduction form.*

2.2 Contextual equivalence

A common definition of contextual equivalence, for System F for example, is that two terms $\Gamma \vdash t, u : A$ are contextually equivalent if there exists no separating context $\emptyset \vdash C\,[\Gamma \vdash \Box : A] : 1 + 1$ such that $C\,[t] \approx_\beta \sigma_1\,()$ and $C\,[u] \approx_\beta \sigma_2\,()$. For a trivial example, if $\Gamma \stackrel{\text{def}}{=} (x : 1 + 1, y : 1 + 1)$, then $C\,[\Box] \stackrel{\text{def}}{=} (\lambda x.\, \lambda y.\, \Box)\ (\sigma_1\,())\ (\sigma_2\,())$ separates x and y.

This definition is too weak in presence of atomic types. Consider the context $\Gamma \stackrel{\text{def}}{=} (x : X, y : X)$ and the terms $\Gamma \vdash x, y : X$. We want a definition of contextual equivalence that declares these terms inequivalent, but there is no distinguishing context in the sense above as we have no information on X, and thus no way to provide distinct values for x and y. The variables x and y *could* be distinct depending on what is the unknown type represented by the abstract type X.

Definition 3. *A* model $\mathcal{M}$ *is a mapping from atomic types to closed types.*

If x is some syntactic object containing types, we write $\mathcal{M}(x)$ for the result of replacing each atomic type in x by its image in the model $\mathcal{M}$. If $\Gamma \vdash t : A$ holds, then it is also the case that $\mathcal{M}(\Gamma) \vdash t : \mathcal{M}(A)$ holds; we may write the term $\mathcal{M}(t)$ as well, to emphasize that we look at its typing in the model.

Definition 4. *If $\Gamma \vdash t, u : A$ and for a given model $\mathcal{M}$, we say that t and u are* contextually equivalent *in $\mathcal{M}$, written $t \approx_{\mathtt{ctx}(\mathcal{M})} u$, if*

$$\forall C,\ \emptyset \vdash C\,[\mathcal{M}(\Gamma) \vdash \Box : \mathcal{M}(A)] : 1 + 1 \Rightarrow C\,[\mathcal{M}(t)] \approx_\beta C\,[\mathcal{M}(u)]$$

We say that t and u are contextually equivalent*, written $t \approx_{\mathtt{ctx}} u$, if they are contextually equivalent in all models.*

2.3 Semantic equivalence

Definition 5 (Semantics of types)**.** *For a closed type A we define the set of* semantic values *of A, written $[\![A]\!]$, by induction on A as follows:*

$$\begin{array}{lcl}
[\![A \to B]\!] & \stackrel{\text{def}}{=} & \text{total functions from } [\![A]\!] \text{ to } [\![B]\!] \\
[\![A \times B]\!] & \stackrel{\text{def}}{=} & \{(v, w) \mid v \in [\![A]\!], w \in [\![B]\!]\} \\
[\![1]\!] & \stackrel{\text{def}}{=} & \{\star\} \\
[\![A + B]\!] & \stackrel{\text{def}}{=} & \{(1, v) \mid v \in [\![A]\!]\} \uplus \{(2, w) \mid w \in [\![B]\!]\} \\
[\![0]\!] & \stackrel{\text{def}}{=} & \emptyset
\end{array}$$

For an arbitrary type A we write $[\![A]\!]_{\mathcal{M}}$ for $[\![\mathcal{M}(A)]\!]$.

We remark that $[\![A]\!]_{\mathcal{M}}$ is always a finite type, whose elements can be enumerated. It is thus decidable whether two elements of $[\![A]\!]_{\mathcal{M}}$ are equal as mathematical objects; at function types, one compares two functions pointwise on their finite domain.

Definition 6 (Semantics of environments)**.** *For a closed typing environment Γ, we define the set $[\![\Gamma]\!]$ of* semantic valuations*, functions from the domain of Γ to semantic values such that:*

$$[\![\Gamma]\!] \quad \stackrel{\text{def}}{=} \quad \{G \mid \forall x : A \in \Gamma,\quad G(x) \in [\![A]\!]\}$$

We write $[\![\Gamma]\!]_{\mathcal{M}}$ for $[\![\mathcal{M}(\Gamma)]\!]$.

Definition 7 (Semantics of typing judgments)**.** *We write $[\![\Gamma \vdash A]\!]$ for the set of functions from semantic valuations of Γ to semantic values in A: $[\![\Gamma \vdash A]\!] \stackrel{\text{def}}{=} [\![\Gamma]\!] \to [\![A]\!]$. We write $[\![\Gamma \vdash A]\!]_{\mathcal{M}}$ for $[\![\mathcal{M}(\Gamma) \vdash \mathcal{M}(A)]\!]$.*

Definition 8 (Semantics of terms)**.** *For a term $\Gamma \vdash t : A$ in a judgment with closed types, we write $[\![t]\!]$ for the set-theoretic semantics of t, as an object of $[\![\Gamma \vdash A]\!]$. The (natural) definition is given in full in Appendix A (Semantics of terms), but for example*

$$[\![\lambda x.\, t]\!](G) \stackrel{\text{def}}{=} (v \mapsto [\![t]\!](G, x \mapsto v))$$

We write $[\![t]\!]_{\mathcal{M}}$ for $[\![\mathcal{M}(t)]\!]$.

Definition 9 (Semantic equivalence)**.** *For any terms $\Gamma \vdash t, u : A$ and model $\mathcal{M}$, we say that t and u are* semantically equivalent *in $\mathcal{M}$, written $t \approx_{\mathtt{sem}(\mathcal{M})} u$, if their semantics are (pointwise) equal.*

$$t \approx_{\mathtt{sem}(\mathcal{M})} u \stackrel{\text{def}}{=} (\forall G \in [\![\Gamma]\!]_{\mathcal{M}},\ [\![t]\!]_{\mathcal{M}}(G) = [\![u]\!]_{\mathcal{M}}(G) \in [\![A]\!]_{\mathcal{M}})$$

We say that t and u are semantically equivalent*, written $t \approx_{\mathtt{sem}} u$, if they are semantically equivalent in all models $\mathcal{M}$.*

2.4 Easy relations between equivalences

Theorem 3 ($\beta\eta$ is semantically sound)**.** *If $t \approx_{\beta\eta} u$ then $t \approx_{\mathtt{sem}} u$.*

Theorem 4 (Semantic equivalence implies contextual equivalence)**.** *If $t \approx_{\mathtt{sem}} u$ then $t \approx_{\mathtt{ctx}} u$.*

On models using closed types (Long version) Our definition of a model $\mathcal{M}$ allows to instantiate atomic types with any closed type; this instantiation happens in the world of syntax, before we interpret types as sets. It is more common, when giving set-theoretic semantics, to instantiate atoms only in the semantics, defining models as mapping from atomic types to arbitrary *sets*.

We are fortunate that our grammar of closed types is expressive enough to describe any finite set; if we did not have units (1 and 0), for example, we could not do this. Our more syntactic notion of model can be shared by our definition of contextual and semantic equivalence, which is very pleasing.

Note that, as a consequence, the finite model property is sort of built into our definition of the semantic and contextual equivalences: we may only distinguish atoms by instantiating them with *finite* sets, not arbitrary sets. One may wonder, then, whether a notion of semantic equivalence that allows to instantiate atoms with *infinite* sets would behave differently – did we really prove the finite model property, or just enforce it by stifling our notion of equivalence?

[1] A reviewer remarks that the terminology "atomic type" is awkward; if "atom" means "indecomposable", then unit types 1, 0 could arguably be considered atomic. In retrospect, we agree that "type variable", as naturally used in polymorphic type systems, would be a better terminology, and plan to use it in the future.

The coincidence of the finite and infinite interpretations is a consequence of the later results of this work on the coincidence of semantic and $\beta\eta$-equivalence. If we added the ability to instantiate atoms by infinite sets, we would distinguish more: we could not prove more terms equal, but would only prove more terms *different*. But any pair of terms equal in the finite-set semantics is $\beta\eta$-equivalent, and Theorem 3 ($\beta\eta$ is semantically sound) seamlessly extends to infinite sets – those terms must still be equal in an infinite-set semantics.

2.5 Fun-less types and reification

To prove that contextual equivalence implies semantic equivalence, we build a reification procedure $\mathtt{reify}_{\mathcal{M}}(v)$ that goes from $[\![A]\!]_{\mathcal{M}}$ to closed terms in $\mathcal{M}(A)$ and satisfies the following properties:

$$\forall v,\ [\![\mathtt{reify}_{\mathcal{M}}(v)]\!]_{\mathcal{M}} = v \qquad \forall(\emptyset \vdash t : A),\ \mathtt{reify}_{\mathcal{M}}([\![t]\!]_{\mathcal{M}}) \approx_{\beta\eta} t$$

This is difficult in the general case, as it corresponds to a normalization-by-evaluation procedure – when you can furthermore prove that $\mathtt{reify}_{\mathcal{M}}(v)$ is always a normal term. The reification of finite sums and products is straightforward, but function types are delicate; intuitively, to reify a function one builds a decision tree on its input, which requires an enumeration procedure for the input type (Altenkirch and Uustalu 2004) which may itself be a function, etc.

In the present case, however, the fact that we work with closed types (no atoms) enables a useful hack: we can eliminate function types by rewriting them in isomorphic types expressed in $\mathsf{AC}(\times, 1, +, 0)$ only. This was inspired by an idea of Danko Ilik (see for example Ilik (2015)), which removes sum types rather than function types. In presence of atomic or infinite types, neither sum nor function types can be fully removed. In absence of atoms, function types can be fully removed, but sum types cannot – there is no type isomorphic to $1 + 1$ in $\mathsf{AC}(\to, \times, 1)$.

Figure 3 (Fun-less data types) defines the fun-less $\lfloor A \rfloor$ for each closed type A. Its definition is structurally recursive, and uses an auxiliary definition $\lfloor\!\lfloor A \to B \rfloor\!\rfloor$ that takes a function type whose left-hand side A is fun-less, and is structurally recursive on this left-hand side. We also define pair of transformations $\lfloor \_ \rfloor_A$ from A to $\lfloor A \rfloor$ and $\lceil \_ \rceil_A$ from $\lfloor A \rfloor$ to A, on both terms and semantic values. On semantic values they are inverse; on terms they are inverse modulo $\beta\eta$-equivalence. The (natural) definitions are given in full in Appendix A (Fun-less types and reification). It uses auxiliary definitions $\lfloor\!\lfloor \_ \rfloor\!\rfloor_{A\to B}$ and $\lceil\!\lceil \_ \rceil\!\rceil_{A\to B}$, and has for example $\lfloor v \rfloor_{A\to B} \stackrel{\text{def}}{=} \lfloor\!\lfloor w \mapsto \lfloor v(\lceil w \rceil_A) \rfloor_B \rfloor\!\rfloor_{A\to B}$ and $\lfloor\!\lfloor t \rfloor\!\rfloor_{A_1\times A_2\to B} \stackrel{\text{def}}{=} \lfloor\!\lfloor \lambda x_1.\, \lfloor\!\lfloor \lambda x_2.\, t\ (x_1, x_2) \rfloor\!\rfloor_{A_2\to B} \rfloor\!\rfloor_{A_1\to(A_2\to B)}$.

Finally, we also have that the isomorphisms on semantic values and ground terms commute.

$$\lfloor A \to B \rfloor \stackrel{\text{def}}{=} \lfloor\!\lfloor \lfloor A \rfloor \to \lfloor B \rfloor \rfloor\!\rfloor \qquad \lfloor A_1 \times A_2 \rfloor \stackrel{\text{def}}{=} \lfloor A_1 \rfloor \times \lfloor A_2 \rfloor$$

$$\lfloor 1 \rfloor \stackrel{\text{def}}{=} 1 \qquad \lfloor A_1 + A_2 \rfloor \stackrel{\text{def}}{=} \lfloor A_1 \rfloor + \lfloor A_2 \rfloor \qquad \lfloor 0 \rfloor \stackrel{\text{def}}{=} 0$$

$$\begin{aligned}
\lfloor\!\lfloor (A_1 \times A_2) \to C \rfloor\!\rfloor &\stackrel{\text{def}}{=} \lfloor\!\lfloor A_1 \to \lfloor\!\lfloor A_2 \to C \rfloor\!\rfloor \rfloor\!\rfloor \\
\lfloor\!\lfloor 1 \to B \rfloor\!\rfloor &\stackrel{\text{def}}{=} B \\
\lfloor\!\lfloor (A_1 + A_2) \to B \rfloor\!\rfloor &\stackrel{\text{def}}{=} \lfloor\!\lfloor A_1 \to B \rfloor\!\rfloor \times \lfloor\!\lfloor A_2 \to B \rfloor\!\rfloor \\
\lfloor\!\lfloor 0 \to B \rfloor\!\rfloor &\stackrel{\text{def}}{=} 1
\end{aligned}$$

$$\forall v, \lceil \lfloor v \rfloor_A \rceil_A = v \qquad \forall t, \lceil \lfloor t \rfloor_A \rceil_A \approx_{\beta\eta} t$$

$$\forall t, \quad \lfloor [\![t]\!] \rfloor_A = [\![\lfloor t \rfloor_A]\!] \ \wedge\ \lceil [\![t]\!] \rceil_A = [\![\lceil t \rceil_A]\!]$$

Figure 3. Fun-less data types

Theorem 5 (Reification). *For each semantic value v in $[\![A]\!]$ we can define a closed term $\mathtt{reify}(v)$ such that $[\![\mathtt{reify}(v)]\!] = v$.*

Theorem 5 (Reification) establishes that semantic inhabitation and provability coincide at closed types.

Corollary 1 (Inhabited or inconsistent). *For Γ, A closed, if $\Gamma \vdash A$ then either $\emptyset \vdash A$ or $\Gamma \vdash 0$.*

Lemma 2. *For any closed term of closed type $\emptyset \vdash t : A$ we have $\mathtt{reify}([\![t]\!]) \approx_{\beta\eta} t$.*

Theorem 6 (Contextual equivalence implies semantic equivalence). *If $t \approx_{\mathtt{ctx}} u$ then $t \approx_{\mathtt{sem}} u$.*

3. Focusing

Consider the usual description of β-normal forms in the purely negative fragment of the simply-typed λ-calculus, $\mathsf{AC}(X, \to, \times, 1)$.

$$\begin{array}{lrcl} \text{values} & t & ::= & \lambda x.\, t \mid (t_1, t_2) \mid () \mid n \\ \text{neutrals} & n & ::= & n\ t \mid \pi_i\ n \mid x \end{array}$$

Values, the story goes, are a sequence of *constructors* applied to a neutral, which is a sequence of *destructors* applied to a variable. It is even possible and easy to capture the set of β-short η-long normal forms by adding a typing restriction to this grammar, asking for the first neutral term n found in a value to be of atomic type $(n : X)$.

Unfortunately, adding sum types to this picture shatters it irreparably. If $\sigma_i\ \_$ is a constructor, it should go in values, and $\mathtt{match}\ \_\ \mathtt{with}\ \ldots$ is a destructor, it should be a neutral term former. Adding $\sigma_i\ t$ to the grammar of values seems innocuous, but adding `match` to neutrals raises a question: should we ask the branches to be neutrals $\mathtt{match}\ n\ \mathtt{with}\ (\sigma_i\ x_i \to n_i)^i$ or values $\mathtt{match}\ n\ \mathtt{with}\ (\sigma_i\ x_i \to t_i)^i$? Neither choices work very well.

Asking branches to be neutrals means that the term

$$x : X + Y \vdash (\mathtt{match}\ x\ \mathtt{with}\ \mid \sigma_1\ y \to \sigma_2\ y \mid \sigma_2\ y \to \sigma_1\ y) : Y + X$$

is not a valid normal form, and in fact has no valid normal form! We cannot force all constructors to occur outside branches, as in this example we fundamentally need to choose a different constructor in each branch – committing to either $\sigma_1\ \_$ or $\sigma_2\ \_$ before matching on x would make us stuck, unable to complete our attempt with a well-formed term.

On the other hand, letting branches be any value introduces normal forms that really should not be normal forms, such as $\pi_1\ (\mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ y_i \to (n, y_i))^i)$, clearly equivalent, for any value of x, to the neutral n.

The solution to this problem comes from logic. Logicians remark that some inference rules are *invertible* and some are *non-invertible*. A rule is invertible when, used during goal-directed proof search, it preserves provability: if the conclusion was provable (maybe using another rule), then applying this rule results in premises that are also provable. For example, consider the implication and disjunction introduction rules:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 + A_2}$$

Implication introduction is invertible – this can be proved by inverting the rule, showing a derivation of $\Gamma, A \vdash B$ with open premise $\Gamma \vdash A \to B$. Disjunction introduction is not: if one decides to prove, say, A_1, one may get stuck while having chosen to prove A_2 would have worked. Or maybe one needs to delay this choice until some hypothesis of the contexts is explored – which is the heart of our $X + Y \vdash Y + X$ example.

Andreoli's focusing (Andreoli 1992) is a technique to restrict a logic, make its proof term more canonical, by imposing additional restrictions based on the invertibility of rules. One easy restriction is that invertible rules, when they can be applied to a judgment, should be applied as early as possible. The more interesting restriction is that when one starts applying non-invertible rules, focusing

forces us to apply them as long as possible, as long as the formula introduced in premises remain at a type where a non-invertible rule exists. For a complete reference on focusing in intuitionistic logic, see Liang and Miller (2007).

In programming terms, the fact that the right implication rule is invertible corresponds to an *inversion principle* on values: without loss of generality, one can consider that any value of type $A \to B$ is of the form $\lambda x.\, t$. Any value of type $A_1 \times A_2$ is of the form (t_1, t_2). This is strictly true for closed values in the empty context, but it is true *modulo equivalence* even in non-empty contexts, as is witnessed by the η-expansion principles. If a value $(t : A \to B)$ is not a λ-abstraction, we can consider the equivalent term $\lambda x.\, t\ x$.

But it is *not* the case that any value of type $A + B$ is of the form $\sigma_i\ t$, as our example $X + Y \vdash Y + X$ demonstrated. Inspired by focusing we look back at our grammar of $\beta\eta$-normal forms: it is not about constructors and destructors, it is about term-formers that correspond to invertible rules and those that do not. To gracefully insert sums into this picture, the non-invertible $\sigma_i\ \_$ should go into the *neutrals*, and case-splitting should be a *value*. Scherer and Rémy (2015) introduce focusing in more details, and present a grammar of *focused* normal forms is, lightly rephrased, as follows:

values	t	$::=$	$\lambda x.\, t \mid (t_1, t_2) \mid () \mid f$
			$\mid \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ y_i \to t_i)^i$
choice terms	f	$::=$	$(n : X) \mid \mathtt{let}\ x : A + B = n\ \mathtt{in}\ t \mid p$
negative neutrals	n	$::=$	$n\ p \mid \pi_i\ n \mid x$
positive neutrals	p	$::=$	$\sigma_i\ p \mid (t : N)$

The type N on the last line denotes a *negative* type, defined as a type whose head connective has an invertible right-introduction rule: $A \to B$ or $A \times B$ or 1. This means that if the argument of an injection $\sigma_i\ \_$ is itself of sum type, it must be of the form $\sigma_j\ \_$ as well; this enforces the focusing restriction that non-invertible rules are applied as long as the type allows.

It is interesting to compare this grammar to bidirectional type systems – when used to understand canonical forms rather than for type inference. Focusing generalizes the idea that some parts of the term structure (constructors) are canonically determined by type information, while some parts (neutrals) are not. It generalizes bidirectional typing by taking the typing environment into account as well as the goal type (variables of sum type are split during the inversion phase), and refines the application syntax $n\ t$ into the sharper $n\ p$ where both sub-terms have a neutral spine.

Our work builds on this focused representation, easily extended with an empty type 0. Our presentation of the type system is different, farther away from the standard λ-calculi and closer to recent presentation of focused systems, by using polarized syntax for types with explicit shifts – it clarifies the structure of focused systems. Instead of distinguishing positive and negative types based on their head connectives, we define two disjoint syntactic categories P and N, with explicit embeddings $\langle N \rangle^+$, $\langle P \rangle^-$ to go from one to the other. In particular, atoms are split in two groups, the positive atoms of the form X^+ and the negative atoms X^- – there is a global mapping from atoms X to polarities, a given atom is either always positive or always negative. Sometimes we need to consider either types of a given polarity or atoms of any polarity; we use P_a for positive types or negative atoms, and N_a for negative types of positive atoms.

We present this focused λ-calculus in Figure 4 (Cut-free focused λ-terms). The focusing discipline is enforced by the inference rules which alternate between four different judgments:

- The *invertible* judgment $\Gamma_\mathsf{na}; \Sigma_\mathsf{p} \vdash_\mathsf{inv} t : N \mid Q_\mathsf{a}$ corresponds to invertible phases in focused proof search. Γ_na is a typing environment mapping variables to negative types N or positive atoms X^+. Σ_p contains only positive types; it is the part of the context that must be decomposed by invertible rules before the end of the phase. The two positions $N \mid Q_\mathsf{a}$ in the goal are either formulas or empty ($\emptyset$), and exactly one of them is non-empty in any valid judgment. If the goal is a negative formula N, it has yet to be introduced by invertible rules during this phase; once it becomes atomic or positive it moves to the other position Q_a.
- The *negative focus* judgment $\Gamma_\mathsf{na} \vdash n \Downarrow N$ corresponds to a non-invertible phase focused on a (negative) formula in the context.
- The *positive focus* judgment $\Gamma_\mathsf{na} \vdash p \Uparrow P$ corresponds to a non-invertible phase focused on a (positive) formula in the goal.
- The *choice-of-focusing* judgment $\Gamma_\mathsf{na} \vdash_\mathsf{foc} f : P_\mathsf{a}$ corresponds to the moment of the proof search (reading from the conclusion to the premises) where the invertible phase is finished, but no choice of focus has been made yet. Focusing on the goal on the right uses a positive neutral to prove a positive type – FOCLC-CONCL-POS. Focusing on the left uses a negative neutral. If the neutral has a positive type, it is `let`-bound to a variable and the proof continue with an invertible phase – FOCLC-LET-POS. If it has a negative atomic type, then it must be equal to the goal type and the proof is done – FOCLC-CONCL-NEG.

The notation $\langle \_ \rangle^+_\mathsf{a}$ takes a negative-or-atomic type N_a and returns a positive type. It is used in the rule FOCLC-INV-FOC that concludes an invertible phase and starts the choice-of-focusing phase. It may only be applied when the positive context Σ_p is of the form $\langle \Gamma'_\mathsf{na} \rangle^+_\mathsf{a}$ for some Γ'_na, that is, when it only contains negative or atomic formulas – it has been fully decomposed.

Notice that the sum-elimination rule in the invertible judgment eliminates a variable x, and not an arbitrary term, and re-introduces variables with the same name x, shadowing the previous hypothesis of sum type: there is no need to refer to it anymore as we learned its value. This cute trick is not fundamental for a focused calculus, but it corresponds to the intuition of the corresponding sequent-calculus rule, and let us actually remove positive types from the context to have a negative-or-atomic context at the end of the phase.

For any judgment, for example $\Gamma_\mathsf{na} \vdash p \Uparrow P$, we use the version without a term position, for example $\Gamma_\mathsf{na} \Uparrow P$, as the proposition that there exists a well-typed term: $\exists p,\ \Gamma_\mathsf{na} \vdash p \Uparrow P$. This is also an invitation to think of the derivation as a logic proof rather than a typed program.

Our focused terms are *cut-free* in the sense that they contain no β-redexes, even modulo commuting conversions. The rule FOCLC-LET-POS does look like a cut, proving $\Gamma_\mathsf{na} \vdash_\mathsf{foc} Q_\mathsf{a}$ from $\Gamma_\mathsf{na} \vdash n \Downarrow \langle P \rangle^-$ and $\Gamma_\mathsf{na}; x : P \vdash_\mathsf{inv} t : \emptyset \mid Q_\mathsf{a}$, but notice that substituting the negative neutral n inside the invertible proof t would not create a β-redex: we know that x is matched over during the invertible phase, but n cannot start with a constructor so $\mathtt{match}\ n\ \mathtt{with}\ \ldots$ cannot reduce. If you are interested in focused systems that do have cuts and interesting dynamic semantics, then the abstract machine calculi of Curien, Fiore, and Munch-Maccagnoni (2016) are a better starting point.

3.1 (Non-)canonicity

When we look at the purely negative fragment of our calculus $\Lambda\mathsf{C}(X^-, \to, \times, 1)$, we can prove that the focused λ-terms correspond exactly to the usual notion of β-short η-long normal forms. For example, consider the valid terms for the judgment $x : Y^+ \to Z_1^- \times Z_2^-; \emptyset \vdash_\mathsf{inv} ? : Y^+ \to Z_1^- \times Z_2^- \mid \emptyset$. Neither x nor $\lambda y.\, x\ y$, that would be well-typed for the corresponding un-focused judgment, are valid according to our inference rules. There is exactly one valid derivation in our system, for the term $\lambda y.\, (\pi_1\ (x\ y), \pi_2\ (x\ y))$ which is the η-long normal form of x at this type.

$$\begin{array}{ll}
\text{negative types} & N, M ::= X^-, Y^-, Z^- \mid P \to N \mid N_1 \times N_2 \mid 1 \mid \langle P \rangle^- \\
\text{positive types} & P, Q \ ::= X^+, Y^+, Z^+ \mid P_1 + P_2 \mid 0 \mid \langle N \rangle^+
\end{array}$$

$$P_{\mathsf{a}}, Q_{\mathsf{a}} ::= P, Q \mid X^-, Y^- \quad N_{\mathsf{a}}, M_{\mathsf{a}} ::= N, M \mid X^+, Y^+$$

$$\begin{array}{ll}
\text{invertible terms} & t, u, r ::= \lambda x.\, t \mid () \mid (t_1, t_2) \mid (f : P) \\
& \qquad \mid \mathtt{absurd}(x) \mid \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t_i)^i \\
\text{focusing terms} & f, g \ ::= \mathtt{let}\ (x : P) = n\ \mathtt{in}\ t \mid (n : X^-) \mid p \\
\text{negative neutrals} & n, m \ ::= (x : N) \mid n\ p \mid \pi_i\ n \\
\text{positive neutrals} & p, q \ ::= \sigma_i\ p \mid (x : X^+)
\end{array}$$

$$\frac{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : P \vdash_{\mathsf{inv}} t : N \mid \emptyset}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} \lambda x.\, t : P \to N \mid \emptyset}$$

$$\frac{(\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t_i : N_i \mid \emptyset)^i}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} (t_1, t_2) : N_1 \times N_2 \mid \emptyset}$$

$$\frac{(\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : Q_i \vdash_{\mathsf{inv}} t_i : N \mid P_{\mathsf{a}})^i}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : Q_1 + Q_2 \vdash_{\mathsf{inv}} \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t_i)^i : N \mid P_{\mathsf{a}}}$$

$$\frac{}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : 0 \vdash_{\mathsf{inv}} \mathtt{absurd}(x) : N \mid P_{\mathsf{a}}} \qquad \frac{}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} () : 1 \mid \emptyset}$$

FOCLC-INV-FOC
$$\frac{\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}} \vdash_{\mathsf{foc}} f : (P_{\mathsf{a}} \mid Q_{\mathsf{a}})}{\Gamma_{\mathsf{na}}; \langle \Gamma'_{\mathsf{na}} \rangle^+_{\mathsf{a}} \vdash_{\mathsf{inv}} f : \langle P_{\mathsf{a}} \rangle^-_{\mathsf{a}} \mid Q_{\mathsf{a}}}$$

FOCLC-CONCL-POS
$$\frac{\Gamma_{\mathsf{na}} \vdash p \Uparrow P}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} p : P}$$

FOCLC-CONCL-NEG
$$\frac{\Gamma_{\mathsf{na}} \vdash n \Downarrow X^-}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} n : X^-}$$

FOCLC-LET-POS
$$\frac{\Gamma_{\mathsf{na}} \vdash n \Downarrow \langle P \rangle^- \qquad \Gamma_{\mathsf{na}}; x : P \vdash_{\mathsf{inv}} t : \emptyset \mid Q_{\mathsf{a}}}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} \mathtt{let}\ x = n\ \mathtt{in}\ t : Q_{\mathsf{a}}}$$

$$\frac{}{\Gamma_{\mathsf{na}}, x : N \vdash x \Downarrow N} \qquad \frac{}{\Gamma_{\mathsf{na}}, x : X^+ \vdash x \Uparrow X^+}$$

$$\frac{\Gamma_{\mathsf{na}} \vdash n \Downarrow N_1 \times N_2}{\Gamma_{\mathsf{na}} \vdash \pi_i\ n \Downarrow N_i} \qquad \frac{\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathsf{inv}} t : N \mid \emptyset}{\Gamma_{\mathsf{na}} \vdash t \Uparrow \langle N \rangle^+}$$

$$\frac{\Gamma_{\mathsf{na}} \vdash n \Downarrow P \to N \qquad \Gamma_{\mathsf{na}} \vdash p \Uparrow P}{\Gamma_{\mathsf{na}} \vdash n\ p \Downarrow N} \qquad \frac{\Gamma_{\mathsf{na}} \vdash p \Uparrow P_i}{\Gamma_{\mathsf{na}} \vdash \sigma_i\ p \Uparrow P_1 + P_2}$$

shift-or-atom notations
$$\langle N \rangle^+_{\mathsf{a}} \stackrel{\mathsf{def}}{=} \langle N \rangle^+ \quad \langle X^+ \rangle^+_{\mathsf{a}} \stackrel{\mathsf{def}}{=} X^+ \qquad \langle P \rangle^-_{\mathsf{a}} \stackrel{\mathsf{def}}{=} \langle P \rangle^- \quad \langle X^- \rangle^-_{\mathsf{a}} \stackrel{\mathsf{def}}{=} X^-$$

Figure 4. Cut-free focused λ-terms

A consequence of this result is that the focused λ-calculus is canonical for the purely negative fragment (or, in fact, the purely positive fragment): if we have $\Gamma_{\mathtt{n}}; \emptyset \vdash_{\mathsf{inv}} t, u : N \mid X^-$ with $t \neq_\alpha u$, then $t \not\approx_{\beta\eta} u$ and $t \not\approx_{\mathtt{ctx}} u$ – these are known to be equivalent in the negative fragment.

Focusing is not canonical anymore in mixed-polarity settings. The first source of non-canonicity is that there may a free choice of ordering of invertible rules in a phase; consider the judgment $\Gamma_{\mathsf{na}}; x : P_1 + P_2 \vdash_{\mathsf{inv}} ? : Q \to N \mid \emptyset$ for example, one may either do a case-split on $P_1 + P_2$ or introduce a λ-abstraction for the function type $Q \to N$: $\mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to \lambda y.\, ?_i)^i$ or $\lambda y.\, \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to ?_i)^i$ are both valid term prefixes. This is solved by declaring that we do not care about the ordering of invertible rules within a single phase.

Definition 10. *We define the equivalence relation* $(\approx_{\mathtt{icc}})$ *as allowing well-typed permutations of two adjacent invertible rules. For example we have* $\mathtt{absurd}(x) \approx_{\mathtt{icc}} \lambda y.\, \mathtt{absurd}(x)$.

From now on any notion of normal form being discussed should be understood as a *quasi*-normal form, a normal form modulo invertible commuting conversions ($\approx_{\mathtt{icc}}$). This is a reasonable approximation of the idea of normal form, as it is easily decidable. Indeed, while in general commuting conversions may relate very different terms, they can be easily decided inside a single invertible phase, for example by imposing a fixed ordering on invertible rules. By definition of invertibility, any ordering preserves completeness.

The other more fundamental source of non-canonicity is that two non-invertible phases may be independent from each other, and thus be ordered in several possible ways, giving distinct but equivalent terms. For example, $\mathtt{let}\ x_1 = n_1\ \mathtt{in}\ \mathtt{let}\ x_2 = n_2\ \mathtt{in}\ (x_1, x_2)$ and $\mathtt{let}\ x_2 = n_2\ \mathtt{in}\ \mathtt{let}\ x_1 = n_1\ \mathtt{in}\ (x_1, x_2)$ are equivalent at type $X^- \times Y^-$ if $x_1 \notin n_2, x_2 \notin n_1$. This source of redundancy is non-local – the permutable `let`-bindings may be miles away inside the term. It requires a global approach to recover canonicity, which we discuss in Section 4 (Saturated focused λ-calculus).

3.2 Computational completeness

We can define a *depolarization* operation $\lfloor \_ \rfloor_\pm$ that takes polarized types and erases polarity information. The definition is given in full in Appendix A (Depolarization), but for example we have $\lfloor X^- \rfloor_\pm \stackrel{\mathsf{def}}{=} X$, $\lfloor P \to N \rfloor_\pm \stackrel{\mathsf{def}}{=} \lfloor P \rfloor_\pm \to \lfloor N \rfloor_\pm$, and $\lfloor \langle N \rangle^+ \rfloor_\pm \stackrel{\mathsf{def}}{=} \lfloor N \rfloor_\pm$.

This erasure operation can be extended to a *defocusing* operation $\lfloor \_ \rfloor_{\mathtt{foc}}$ on focused terms that preserves typing modulo depolarization. For example, if $\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t : N \mid Q_{\mathsf{a}}$ holds, then in the un-focused system $\lfloor \Gamma_{\mathsf{na}} \rfloor_\pm, \lfloor \Sigma_{\mathsf{p}} \rfloor_\pm \vdash \lfloor t \rfloor_{\mathtt{foc}} : (\lfloor N \rfloor_\pm \mid \lfloor Q_{\mathsf{a}} \rfloor_\pm)$ holds – with $(A \mid \emptyset) \stackrel{\mathsf{def}}{=} A$ and conversely. This operation is defined on terms as a direct mapping on all λ-term formers, except the `let`-definition form which does not exist in the unfocused calculus and is substituted away: $\lfloor \mathtt{let}\ x = n\ \mathtt{in}\ t \rfloor_{\mathtt{foc}} \stackrel{\mathsf{def}}{=} \lfloor t \rfloor_{\mathtt{foc}} [\lfloor n \rfloor_{\mathtt{foc}} / x]$.

Going from a focused system to a system with less restrictions is easy. The more interesting statement is the converse, that for any un-focused term t there exists an equivalent focused term t'.

Theorem 7 (Completeness of focusing)**.**

$$\lfloor \Gamma_{\mathsf{na}} \rfloor_\pm, \lfloor \Sigma_{\mathsf{p}} \rfloor_\pm \vdash t : (\lfloor N \rfloor_\pm \mid \lfloor Q_{\mathsf{a}} \rfloor_\pm)$$
$$\implies \quad \exists t', \quad \lfloor t' \rfloor_{\mathtt{foc}} \approx_{\beta\eta} t \quad \wedge \quad \Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t' : N \mid Q_{\mathsf{a}}$$

Proof. Completeness of focusing is a non-trivial result, but it is independent to the contributions of the current work, and in particular extends gracefully to the presence of an empty type. See for example proofs in Liang and Miller (2007); Ahmad, Licata, and Harper (2010); Simmons (2011). □

3.3 Choices of polarization

We mentioned that a given un-polarized atom X must either appear always positively X^+ or always negatively X^- in our judgments. Violating this restriction would break completeness, as for example $X^+ \vdash X^-$ is not provable – they are considered distinct atoms. But the global choice of polarization of each atom is completely free: completeness holds whatever choice is made. Those choices influence the operational behavior of proof search: Chaudhuri, Pfenning, and Price (2008b) shows that using the negative polarization for all atoms corresponds to backward proof search, whereas using the positive polarization corresponds to forward proof search.

Similarly, there is some leeway in insertion of the polarity shifts $\langle \_ \rangle^+$ and $\langle \_ \rangle^-$; for example, $0 + X^+$ and $0 + \left\langle \langle X^+ \rangle^- \right\rangle^+$ depolarize to the same formula, but admit fairly different terms – the double-shifting allows an invertible phase to start right after $(\sigma_2\ \_)$. When transforming a non-polarized type into a polarized type, two strategies for inserting shifts are notable. One is to insert

as few shifts as possible; the terms inhabiting the minimally-shifted judgment are in one-to-one correspondence with terms of the unfocused system that "respect the focusing restriction". The other is to insert double-shifts under each connective; the terms inhabiting these double-shifted judgments are in one-to-one correspondence with unfocused sequent terms – Zeilberger (2013) relates this to double-negation translations from classical to intuitionistic logic.

4. Saturated focused λ-calculus

In Section 3.1 ((Non-)canonicity) we explained that the essential source of non-canonicity in focused term systems is that distinct non-invertible phases may be independent from each other: reordering them gives syntactically distinct terms that are observably equivalent in a pure calculus. Being such a reordering of another term is a highly global property, that cannot be decided locally like invertible commuting conversions.

Logicians introduced *maximal multi-focusing* (Chaudhuri, Miller, and Saurin 2008a) to quotient over those reorderings, and Scherer and Rémy (2015) expressed this in a programming setting as *saturation*. The idea of maximal multi-focusing is to force each non-invertible phase to happen *as early as possible* in a term, in parallel, removing the potential for reordering them. However, in general there is no goal-directed proof search (or term enumeration) procedure that generates only maximally multi-focused derivations, as one cannot guess in advance what non-invertible phases will be useful in the rest of the term – to introduce them as early as possible. Saturation is a technique specific to intuitionistic logic: when a non-invertible phase starts, instead of trying to guess which non-invertible phases would be useful later, one saturates the context by performing *all* the possible left-focused phases, `let`-binding all the neutrals that might be used in the rest of the term. One can think of a neutral of positive type $(n : P)$ as an *observation* of the current environment: we are saturating by performing all possible observations before making a choice – a right focusing phase. This strategy would be invalid in an effectful language, or a resource-aware logic where introducing unused sub-derivations can consume necessary resources and get you stuck.

In general there may be infinitely many distinct observations that can be made in a saturation phase – consider the context $(z : X^+, s : X^+ \to \langle X^+ \rangle^-)$, and a type system that would enforce complete saturation would then have to admit infinite terms. Instead, Scherer and Rémy (2015) relax the definition of saturation by allowing saturation phases to introduce only a finite subset of deducible neutrals $(n : P)$. They prove that canonicity holds (in a system without the empty type) in the following sense: if two saturated terms made the same saturation choices, then they are equivalent if and only if they are syntactically the same – modulo $(\approx_{\mathtt{icc}})$. In their work, the notion of equivalence is $\beta\eta$-equivalence of the defocused forms.

4.1 The saturated type system

The saturated type system is given in Figure 5 (Cut-free saturated focused type system). The neutral judgments are identical to the focused type system of Figure 4 (Cut-free focused λ-terms), and most of the invertible rules are also identical. The only change is that the rule SINV-SAT moving from the invertible phase to the focused phase, instead of merging the two contexts $\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}}$ in a single context position as FOCLC-INV-FOC, now keeps them separate.

This second position Γ'_{na} represents the fragment of the context that is *new* during the following saturation phase. The saturation rule SAT requires that any introduced neutral n use at least one variable of this new context ($\exists x \in \Gamma'_{\mathsf{na}},\ x \in n$). This guarantees that a single neutral term cannot be introduced twice by distinct saturation phases: the second time it will not be new anymore.

$$\frac{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : P \vdash_{\mathit{sinv}} t : N \mid \emptyset}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathit{sinv}} \lambda x.\, t : P \to N \mid \emptyset} \qquad \frac{(\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathit{sinv}} t_i : N_i \mid \emptyset)^i}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathit{sinv}} (t_1, t_2) : N_1 \times N_2 \mid \emptyset}$$

$$\frac{}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathit{sinv}} () : 1 \mid \emptyset} \qquad \frac{}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : 0 \vdash_{\mathit{sinv}} \mathtt{absurd}(x) : N \mid Q_{\mathsf{a}}}$$

$$\frac{(\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : P_i \vdash_{\mathit{sinv}} t_i : N \mid Q_{\mathsf{a}})^i}{\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}}, x : P_1 + P_2 \vdash_{\mathit{sinv}} \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t_i)^i : N \mid Q_{\mathsf{a}}}$$

SINV-SAT
$$\frac{\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}} \vdash_{\mathsf{sat}} f : (P_{\mathsf{a}} \mid Q_{\mathsf{a}})}{\Gamma_{\mathsf{na}}; \langle \Gamma'_{\mathsf{na}} \rangle^+_{\mathsf{a}} \vdash_{\mathit{sinv}} f : \langle P_{\mathsf{a}} \rangle^-_{\mathsf{a}} \mid Q_{\mathsf{a}}}$$

SAT
$$\frac{(\bar{n}, \bar{P}) \stackrel{\mathsf{def}}{=} \mathtt{Select}_{\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}}}\left(\left\{(n, P) \mid \begin{array}{c}(\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}} \vdash_{\mathsf{s}} n \Downarrow \langle P \rangle^-) \\ \wedge\ \exists x \in \Gamma'_{\mathsf{na}}, x \in n\end{array}\right\}\right) \qquad \Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}}; \bar{x} : \bar{P} \vdash_{\mathit{sinv}} t : \emptyset \mid Q_{\mathsf{a}}}{\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}} \vdash_{\mathsf{sat}} \mathtt{let}\ \bar{x} = \bar{n}\ \mathtt{in}\ t : Q_{\mathsf{a}}}$$

SAT-UP
$$\frac{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} p \Uparrow P}{\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathsf{sat}} p : P}$$

SAT-DOWN
$$\frac{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} n \Downarrow X^-}{\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathsf{sat}} n : X^-}$$

$$\frac{\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathit{sinv}} t : N \mid \emptyset}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} t \Uparrow \langle N \rangle^+} \qquad \frac{}{\Gamma_{\mathsf{na}}, x : X^+ \vdash_{\mathsf{s}} x \Uparrow X^+} \qquad \frac{}{\Gamma_{\mathsf{na}}, x : N \vdash_{\mathsf{s}} x \Downarrow N}$$

$$\frac{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} n \Downarrow N_1 \times N_2}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} \pi_i\ n \Downarrow N_i} \qquad \frac{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} p \Uparrow P_i}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} \sigma_i\ p \Uparrow P_1 + P_2}$$

$$\frac{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} n \Downarrow P \to N \qquad \Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} p \Uparrow P}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{s}} n\ p \Downarrow N}$$

Figure 5. Cut-free saturated focused type system

This new context is also used to know when saturation stops: if an instance of the SAT rule does not introduce any new neutral, then on the next saturation phase the new context Γ'_{na} will be the empty context $\emptyset$, allowing saturation to proceed to prove the goal with one of the two other choice-of-focusing rules.

This aspect of the saturation judgment is reused, unchanged, from Scherer and Rémy (2015). On the other hand, the formulation of the saturation rule SAT is different. We pass the (potentially infinite) set S of new introducible neutrals to a selection function $\mathtt{Select}_{\Gamma_{\mathsf{na}}}(S)$, which returns a finite subset of neutrals to introduce in a given context. $\mathtt{Select}_{\Gamma_{\mathsf{na}}}(S)$ may not return any subset, we give the requirement for a selection function to be valid in Section 4.3 (Selection function).

The notation $\mathtt{let}\ \bar{x} = \bar{n}\ \mathtt{in}\ t$ denotes simultaneous binding of a (finite) set of neutral terms – our notion of syntactic α-equivalence is considered to test the (decidable) set equality.

4.2 Strong positive neutrals

To understand and formalize saturation it is interesting to compare and contrast the various notions of deductions (seeing our type systems as logics) at play; how to prove A in a context Γ?

- The more general notion of deduction is the unfocused notion of proof $\Gamma \vdash A$ – proof terms have no restriction. In the focused system, it would correspond to looking for a proof of an invertible judgment $\emptyset; \Gamma \vdash_{\mathsf{inv}} A \mid \emptyset$.
- The neutral judgments $\Gamma \vdash n \Downarrow N$ and $\Gamma \vdash p \Uparrow P$ correspond to a less expressive notion of "simple deduction step", which are iterated by saturation. For example, $\langle X + Y \rangle^- \Uparrow Y + X$ does *not* hold, it requires more complex reasoning than a chain of

eliminations from the context variables. Focusing decomposes a non-focused reasoning into a sequence of such simple deduction steps, separated by invertible phases of blind proof search.

One notion that is missing is the notion of what is "already known by the context". With the usual non-focused logic, to know if a positive formula P has been introduced before, we simply check if P is in the context. But the focusing discipline decomposes positive formulas and removes them from the context.

One could use the judgment $\Gamma_{\mathsf{na}} \Uparrow P$ instead – $X^+, Y^+ \Uparrow X^+ + Y^+$ as intended. But $\Gamma_{\mathsf{na}} \Uparrow P$ is too strong for the purpose of just retrieving information from the context, as it calls the general invertible judgment at the end of the focused phase. $\Gamma_{\mathsf{na}} \Uparrow \langle N \rangle^+$ holds whenever N is provable from Γ_{na}, not only when N is an hypothesis in Γ_{na}.

To capture this idea of what can be "retrieved" from the context without any reasoning, we introduce in Figure 6 (Strong positive judgment $\Gamma_{\mathsf{na}} \vdash p \Uparrow P$) the *strong* positive judgment $\Gamma_{\mathsf{na}} \Uparrow P$.

$$\frac{}{\Gamma_{\mathsf{na}}, x : N \vdash x \Uparrow \langle N \rangle^+} \qquad \frac{}{\Gamma_{\mathsf{na}}, x : X^+ \vdash x \Uparrow X^+}$$

$$\frac{\Gamma_{\mathsf{na}} \vdash p \Uparrow P_i}{\Gamma_{\mathsf{na}} \vdash \sigma_i\ p \Uparrow P_1 + P_2}$$

Figure 6. Strong positive judgment $\Gamma_{\mathsf{na}} \vdash p \Uparrow P$

Strong positive neutrals correspond to the positive patterns of Zeilberger (2009). Those patterns describe the spine of a non-invertible phase, but they can also characterize invertible phases: an invertible phase, presented in a higher-order style, provides a derivation of the goal for any possible positive pattern passed by the environment. The two following results witness this relation.

Lemma 3 (Strong decomposition of invertible phases)**.** *Consider an invertible derivation $\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{sinv}} t : N \mid Q_{\mathsf{a}}$: it starts with invertible rules, until we reach a (possibly empty) "frontier" of saturated subterms f to which the rule* SINV-SAT *is applied. Let $\left(\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}\,k} \vdash_{\mathsf{sat}} f_k : Q'_{\mathsf{a}\,k}\right)^{k \in K}$ be the family of such subterms. Then the $\Gamma'_{\mathsf{na}\,k}$ are exactly the contexts such that*

$$\forall P \in \Sigma_{\mathsf{p}}, \qquad \forall k,\ \Gamma'_{\mathsf{na}\,k} \Uparrow P$$

Lemma 4 (Strong positive cut)**.**
If both $\Gamma_{\mathsf{na}} \vdash p \Uparrow P$ and $\Gamma_{\mathsf{na}}; P \vdash_{\mathsf{sinv}} t : \emptyset \mid Q_{\mathsf{a}}$ hold, then there exists a subterm f of t such that $\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} f : Q_{\mathsf{a}}$ holds.

This result would not be provable for the more expressive judgment $\Gamma_{\mathsf{na}} \vdash p \Uparrow P$: there is no obvious way to substitute a general $u : N$ through t that would respect the focusing structure – and return a strict subterm. $\Gamma_{\mathsf{na}} \Uparrow P \implies \Gamma_{\mathsf{na}} \Uparrow P$ is true but non-trivial, it relies on the compleness result – identity expansion.

4.3 Selection function

Contrarily to the simpler setting with sums but no empty type, not all ways to select neutrals for saturation preserve canonicity in presence of the empty type. Consider for example the focused terms

$$\mathtt{let}\ x = f\ ()\ \mathtt{in}\ \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to \sigma_1\ ())^i$$

$$\mathtt{let}\ x = f\ ()\ \mathtt{in}\ \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to \sigma_2\ ())^i$$

at the typing $f : \langle 1 \rangle^+ \to \langle 1 + 1 \rangle^-, g : \langle 1 \rangle^+ \to \langle 0 \rangle^- \vdash 1 + 1$. The set of potential observations is $\{f\ (), g\ ()\}$, and both terms made the same choice of observing only $f\ ()$. The first term always returns $\sigma_1\ ()$ and the second $\sigma_2\ ()$, so they are syntactically distinct even modulo ($\approx_{\mathtt{icc}}$). Yet they are $\beta\eta$-equivalent as the context is inconsistent. Note that if $\mathtt{let}\ y = g\ ()\ \mathtt{in}\ \_$ had been introduced during saturation, the immediately following invertible phase would necessarily have been $\mathtt{absurd}(y)$, and the two terms would thus be syntactially equal.

To make saturation canonical again, we need a provability completeness requirement: if there is a possible proof of 0, we want saturation to find it. One could cheat, knowing that provability of any formula is decidable in propositional logic, and test explicitly for the absence of proof of 0; but saturation is already doing proof search[2], and we can extend it gracefully to have this property.

We define our requirement on the selection function in Figure 7 (Specification of saturation selection functions). We require that, for any type P that is part of the deducible observations S (by a neutral $(n : \langle P \rangle^-)$), either P is already retrievable from the context Γ_{na} (no need to introduce it then) or it is the type of a neutral n' selected by the function. We do not require the same neutral n to be selected: there may be infinitely many different neutrals deducible at P, but just having one of them in the returned set suffices. This definition is not natural, it will be validated by the results from Section 5 (Saturation consistency).

Note that the types P that can be simply deduced from the context $\Gamma_{\mathsf{na}} \Downarrow P$ are subformulas of Γ_{na}. We know by the subformula property that they are also subformulas of the root judgment of the global derivation. In particular, there is only a finite number of such deducible types P – this would not hold in a second-order type system. Valid selection functions exist thanks to this finiteness.

SELECT-SPECIF

$$\frac{\forall \Gamma_{\mathsf{na}}, S, P,\quad n : \langle P \rangle^- \in S \implies \Gamma_{\mathsf{na}} \Uparrow P \vee \exists (n' : \langle P \rangle^-),\ n' \in \mathtt{Select}_{\Gamma_{\mathsf{na}}}(S)}{\mathtt{Select}\_(\_) \text{ is a valid selection function}}$$

Figure 7. Specification of saturation selection functions

Note that two valid selection functions can be merged into a valid selection function, by taking the union of their outputs.

4.4 Completeness of saturation

Completeness of saturation is relative to a specific choice of selection function. Indeed, consider the context

$$\Gamma_{\mathsf{na}} \qquad \stackrel{\mathsf{def}}{=} \qquad (x : X^+, y : \langle 1 \rangle^+ \to \langle X^+ \rangle^-)$$

In this context, the only deducible positive formula is X^+, and it is already retrievable from Γ_{na}. This means that a selection function that would satisfy $\mathtt{Select}_{\Gamma_{\mathsf{na}}}(S) = \emptyset$ would be a valid selection function. However, saturating with such a selection function is not computationally complete: the saturated term $x : X^+$ has a valid derivation, but $\mathtt{let}\ z = y\ ()\ \mathtt{in}\ z$ does not – nor does any equivalent term.

We can order selection functions by pointwise subset ordering: a function f is above g if it selects at least all of g's neutrals for each context. A set of saturation functions is upward-closed if, for any saturation function in the set, any function above it is in the set.

Theorem 8 (Completeness of saturation)**.** *For any focused term $\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t : N \mid P_{\mathsf{a}}$, there exists an upward-closed set of selections functions such that $\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{sinv}} t' : N \mid P_{\mathsf{a}}$ holds, for a (computable) saturated term t' such that $\lfloor t \rfloor_{\mathtt{foc}} \approx_{\beta\eta} \lfloor t' \rfloor_{\mathtt{foc}}$.*

Note that, given *two* focused term t_1, t_2, we can merge the selection functions used in the theorem above to get a single selection function for which there exists saturated terms for both t_1 and t_2.

[2] Saturation synthetizes new sub-terms and can thus decide equivalence with 0, unlike previous rewriting-based methods that would only reorder the subterms of the compared terms.

5. Saturation consistency

In this section, we prove the main result of this extension of saturation to the empty type: if a context is inconsistent, then the saturation phase will eventually introduce a variable of the empty type 0 in the context. This is key to obtaining a canonicity result – if saturation sometimes missed proofs of 0, it could continue with distinct neutral terms and result in distinct but equivalent saturated terms.

The informal view of the different ways to deduce a positive formula presented in Section 4.2 (Strong positive neutrals) (general proof, simple deduction, retrieval from context) gives a specification of what saturation is doing. From a high-level or big-step point of view, saturation is trying all possible new simple deductions iteratively, until all the positives deducible from the context have been added to it. The following characterization is more fine-grained, as it describes the state of an intermediary saturation judgment $\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}} \vdash_{\mathsf{sat}} f : P_{\mathsf{a}}$.

The characterization is as follows: any formula that can be "simply deduced" from the old context Γ_{na} becomes "retrievable" in the larger context $\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}}$. This gives a precise meaning to the intuition that Γ_{na} is "old". What we mean when saying that Γ'_{na} is "new" can be deduced negatively: it is the part of the context that is still fresh, its deductions are not stored in the knowledge base yet.

Theorem 9 (Saturation)**.** *If a saturated proof starts from a judgment of the form* $\emptyset; \Gamma_{\mathsf{na}0} \vdash_{\mathsf{sat}} f : Q_{\mathsf{a}}$ *or* $\emptyset; \Sigma_{\mathsf{p}0} \vdash_{\mathsf{sinv}} t : N \mid Q_{\mathsf{a}}$ *then for any sub-derivation of the form* $\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}} \vdash_{\mathsf{sat}} f : Q_{\mathsf{a}}$ *we have the following property:*

$$\forall P, \qquad \Gamma_{\mathsf{na}} \Downarrow \langle P \rangle^- \qquad \Longrightarrow \qquad \Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}} \Uparrow P$$

Definition 11. Γ_{na} *is* saturated *if* $\Gamma_{\mathsf{na}} \Downarrow \langle P \rangle^-$ *implies* $\Gamma_{\mathsf{na}} \Uparrow P$.

Corollary 2 (Saturation)**.** *If a saturated proof starts from a judgment of the form* $\emptyset; \Gamma_{\mathsf{na}0} \vdash_{\mathsf{sat}} f : Q_{\mathsf{a}}$ *or* $\emptyset; \Sigma_{\mathsf{p}0} \vdash_{\mathsf{sinv}} t : N \mid Q_{\mathsf{a}}$ *then for any sub-derivation of the form* $\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathsf{sat}} f : Q_{\mathsf{a}}$ *the environment* Γ_{na} *is saturated.*

Lemma 5 (Saturated consistency)**.** *If* Γ_{na} *is saturated, then* $\Gamma_{\mathsf{na}} \nvdash 0$.

Theorem 10 (Inconsistent canonicity)**.** *If* $\Gamma_{\mathsf{na}} \vdash 0$*, then for any* f, f' *such that* $\emptyset; \Gamma_{\mathsf{na}} \vdash_{\mathsf{sat}} f, f' : P_{\mathsf{a}}$ *we have* $f \approx_{\mathtt{icc}} f'$.

6. Canonicity

In this section we establish the main result of this article. If two saturated terms $\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{sinv}} t, t' : N \mid Q_{\mathsf{a}}$ are not syntactically equivalent ($t \not\approx_{\mathtt{icc}} t'$), then there exists a model $\mathcal{M}$ in which a context distinguishes t from t': they are not contextually equivalent.

(We build distinguishing contexts in the un-focused λ-calculus, so technically we are distinguishing the defocused forms $\lfloor t \rfloor_{\mathtt{foc}}, \lfloor t' \rfloor_{\mathtt{foc}}$; the proof crucially relies on the saturated structure of its inputs, but the code we generate for computation and separation is more easily expressed unfocused.)

6.1 Sketch of the proof

Intuition It is helpful to first get some intuition of what a pair of syntactically distinct normal forms looks like, and what the corresponding distinguishing context will look like. Suppose we have $\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{sinv}} t \not\approx_{\mathtt{icc}} t' : N \mid Q_{\mathsf{a}}$. We can explore t and t' simultaneously, until we find the source of their inequality.

The source of inequality cannot be in an invertible phase, given that the term formers in invertible phase are completely determined by the typing (modulo invertible commuting conversions); for example, if N is $N_1 \times N_2$, we know that t is of the form (t_1, t_2), and t' of the form (t'_1, t'_2), with $t_i \not\approx_{\mathtt{icc}} t'_i$ for some i – so we can continue exploring $t_i \not\approx_{\mathtt{icc}} t'_i$. Same thing if the term starts with a sum elimination (modulo $(\approx_{\mathtt{icc}})$ one can assume that they eliminate the same variable), $\mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t_i)^i \not\approx_{\mathtt{icc}} \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t'_i)^i$: the subterms t_i, t'_i in at least one of the two branches differ.

Similarly, the source of inequality cannot be in the saturation phase, where both terms saturate on neutrals that are completely determined by the typing context – and the saturation selection function – they are both of the form $\mathtt{let}\ \bar{x} = \bar{n}\ \mathtt{in}\ \_$ for the same set of neutrals on each side. The end of this saturation phase is also type-directed, so both terms stop saturating (they get an empty Γ'_{na} context) at the same time. The difference must then be in the neutrals used in the SAT-UP or SAT-DOWN rules, $n \not\approx_{\mathtt{icc}} n'$ or $p \not\approx_{\mathtt{icc}} p'$. Note that we cannot get a positive neutral on one side and a negative neutral on the other, as usage of those rules is directed by whether the goal type is a negative atom X^- or a positive type P.

Now, two neutrals $n \not\approx_{\mathtt{icc}} n'$ or $p \not\approx_{\mathtt{icc}} p'$ may differ because their *spine* differ, or because their sub-terms that are outside the non-invertible phase differ. In the latter case, finding the source of inequality is a matter of traversing the common structure towards the sub-terms that differ. The former case is more interesting – this pair of neutrals with distinct spines is what we call *source of inequality*.

In the positive neutral case, we end up on either $\sigma_i\ p$ and $\sigma_j\ p'$ with $i \neq j$, or distinct variables $x \neq y$ of atomic type X^+. In the negative neutral case, one may encounter distinct neutrals with distinct structure, for example $n\ p \neq x \neq \pi_i\ m$ at the same type N; negative neutrals should be looked "upside down", as in System L (Curien, Fiore, and Munch-Maccagnoni 2016): either their head variables differ, or the same variable is applied a different sequence of elimination rules.

In the easy case where the source of inequality is a sum constructor $(\sigma_i\ \_) \not\approx_{\mathtt{icc}} (\sigma_j\ \_)$, obtaining a distinguishing context looks relatively easy: we need a context $C\,[\Box]$ that corresponds to the term traversal we performed to reach this source of inequality. For example, if we had $(t_1, t_2) \not\approx_{\mathtt{icc}} (t'_1, t'_2)$ because $t_2 \not\approx_{\mathtt{icc}} t'_2$, the context fragment corresponding to this reasoning step would be $\pi_2\ \Box$. This is trickier in the sum-elimination case: if we have $\mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t_i)^i \not\approx_{\mathtt{icc}} \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t'_i)^i$ then we need our context to instantiate the variable x with the right value $\sigma_i\ p$ so that the branch we want is taken – the one with $t_i \not\approx_{\mathtt{icc}} t'_i$. This is easy if x is a formal variable introduced by a λ-abstraction: at the point where our context needs to distinguish the two λ-abstraction $\lambda x.\, t \not\approx_{\mathtt{icc}} \lambda x.\, t'$, we need to use an application context of the form $\Box\ (\sigma_i\ p)$. But x may have been introduced by a left-focusing step $\mathtt{let}\ x = n\ \mathtt{in}\ t \not\approx_{\mathtt{icc}} \mathtt{let}\ x = n\ \mathtt{in}\ t'$; then we need to instantiate the variables of the observation n *just so* that we get the desired result $\sigma_i\ t'$. When the source of inequality is on negative neutrals with heads $x : N$, $y : M$ or positive variables $x \neq y : X^+$, we need the distinguishing context to pass values in the same way to get to this source of inequality, and also to instantiate the variables x, y to get an inequality. If those variables are at an atomic type, we must pick a model that replaces this atomic type by a closed type, to instantiate them by distinguishable values at this closed type.

Positive simplification In order to simplify the following arguments, we will suppose that the context and types of the two saturated proof to distinguish do not use negative atoms, only positive atoms. In other words, the results only holds for the focused system in the fragment $\mathsf{\Lambda C}(X^+, \to, \times, 1, +, 0)$.

This is a perfectly reasonable simplification in view of our goal, which is to distinguish inequivalent non-focused term that have distinct saturated normal forms: we know that any choice of polarization for the non-focused atoms preserves the existence of normal forms, so we can make them all positives – see Section 3.3 (Choices of polarization).

In particular, the source of inequality (distinct neutrals whose spine differs) is always a pair of neutrals $p \not\approx_{\mathtt{icc}} p'$, who contain a syntactic difference ($\sigma_i \_ \not\approx_{\mathtt{icc}} \sigma_j \_$ with $i \neq j$, or $x \neq y : X^+$) before the end of the non-invertible phase. Negative neutrals are only used during saturation.

Neutral model If S is a finite set, let us write $\mathtt{Fin}\,(S)$ for the type $1 + \ldots + 1$ that is in bijection with S (same number of elements), witnessed by embeddings $\mathtt{in}_{\mathtt{Fin}}(\_) : S \to \mathtt{Fin}\,(S)$ and $\mathtt{out}_{\mathtt{Fin}}(\_) : \mathtt{Fin}\,(S) \to S$ such that

$$\mathtt{out}_{\mathtt{Fin}}(\mathtt{in}_{\mathtt{Fin}}(x)) = x \in S \;\wedge\; \emptyset \vdash \mathtt{in}_{\mathtt{Fin}}(\mathtt{out}_{\mathtt{Fin}}(t)) \approx_{\beta\eta} t : \mathtt{Fin}\,(S)$$

For any two distinct elements $x \neq y \in S$ there exists a distinguishing context for $\mathtt{in}_{\mathtt{Fin}}(x) \not\approx_{\mathtt{ctx}} \mathtt{in}_{\mathtt{Fin}}(y)$.

Definition 12 (Neutral model)**.** *Given two syntactically distinct saturated terms* $\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t_0 \not\approx_{\mathtt{icc}} t'_0 : N \mid P_{\mathsf{a}}$ *(with positive atoms only) with a source of inequality of the form*

$$\Gamma'_{\mathsf{na}} \vdash p \not\approx_{\mathtt{icc}} p' \Uparrow P$$

we define the neutral model $\mathcal{N}_{t_0,t'_0}$ *(or just* $\mathcal{N}$*) by*

$$\mathcal{N}_{t_0,t'_0}(Y^+) \stackrel{\mathsf{def}}{=} \mathtt{Fin}\left(\{x \mid (x : Y^+) \in \Gamma'_{\mathsf{na}}\}\right)$$

We say that $\mathcal{N}(X)$ contains a *code* for each atomic variable bound at the source of inequality.

Distinguishing outline The general idea of our distinguishing context construction is to instantiate variables just so that each variable of atomic type $(x : X^+)$ evaluates to its code $\mathtt{in}_{\mathtt{Fin}}(x)$. Thus, when we end up on distinct neutrals $p \not\approx_{\mathtt{icc}} p'$, we know that our context will send them to distinguishables values.

There are two moments where building a distinguishing context requires synthesizing a closed value of a type: to instantiate the open variables in the context of the two terms, and when distinguishing invertible terms at a function type $P \to N$, which we know to be of the shape $\lambda(x : P).\,(\_ : N)$.

Synthesizing a value for a variable of atomic type $(x : X^+)$ is obvious, we just pick $\mathtt{in}_{\mathtt{Fin}}(x)$ – this guarantees that, under this context, x will reduce to $\mathtt{in}_{\mathtt{Fin}}(x)$ as expected. For a variable of sum type $P + Q$, we have to choose the value to make sure that the correct branch of the two terms (the one containing the source of inequality) will be explored, as previously explained. For a variable x of negative type N, we have to make sure that any observation of x, any neutral term n whose head variable is x (in the syntax of Curien, Fiore, and Munch-Maccagnoni (2016) they are the $\langle x \mid S \rangle$), will reduce to the value we need: if we have $\mathtt{let}\; y : P = n\; \mathtt{in}\; \ldots$, the neutral n should reduce to $\mathtt{in}_{\mathtt{Fin}}(y)$. In other words, much in the spirit of higher-order focusing Zeilberger (2009), we specify the instantiation of $(x : N)$ by a mapping over all the observations over x that are made in t_0, t'_0.

Example Consider for example:

$$n : (1 + X^+) \to \langle X^+ \rangle^- \vdash \begin{array}{l} \mathtt{let}\; z = n\; (\sigma_1\; ())\; \mathtt{in} \\ \quad \mathtt{let}\; o = n\; (\sigma_2\; z)\; \mathtt{in}\; z \\ \qquad \not\approx_{\mathtt{icc}} \\ \mathtt{let}\; z = n\; (\sigma_1\; ())\; \mathtt{in} \\ \quad \mathtt{let}\; o = n\; (\sigma_2\; z)\; \mathtt{in}\; o \end{array} : X^+$$

The shared context in this example is

$$\mathtt{let}\; z = n\; \sigma_1\; ()\; \mathtt{in}\; \mathtt{let}\; o = n\; (\sigma_2\; z)\; \mathtt{in}\; \Box$$

and the source of inequality is $z \not\approx_{\mathtt{icc}} o$. The atomic variables in the context at this point are z and o, so we have $\mathcal{N}(X^+) = \mathtt{Fin}\,(\{z, o\})$. We have to provide a value for the negative variable n; its "observations", the arguments it is called with, are $\sigma_1\; ()$ and $\sigma_2\; z$, so we define it on these inputs following the general scheme:

$$\hat{n} \stackrel{\mathsf{def}}{=} \begin{cases} \sigma_1\; () & \mapsto & \mathtt{in}_{\mathtt{Fin}}(z) \\ \sigma_2\; \mathtt{in}_{\mathtt{Fin}}(z) & \mapsto & \mathtt{in}_{\mathtt{Fin}}(o) \end{cases}$$

The value of $\hat{n}$ on the last element of $\mathcal{N}(1 + X^+)$, namely $\sigma_2\; (\mathtt{in}_{\mathtt{Fin}}(o))$, is not specified; the return type is inhabited so a value can be chosen, and the specific choice does not matter.

It is easy to check that the context $C\,[\Box] \stackrel{\mathsf{def}}{=} (\lambda n.\,\Box)\; \hat{n}$ is such that plugging both terms will result in distinct closed values of $\mathcal{N}(X^+)$, namely $\mathtt{in}_{\mathtt{Fin}}(z)$ and $\mathtt{in}_{\mathtt{Fin}}(o)$. From there, building a distinguishing context returning a boolean is trivial.

Technical challenge We outlined the general argument and demonstrated it on an example. Unfortunately, we found that scaling it to a rigorous, general proof is very challenging.

When we define instantiation choices for negative types as mappings from observations to their results, we implicitly rely on the fact that the observations are distinct from each other. This is obvious when the domain of these observations is made of first-order datatypes (no functions), but delicate when some of those observations have function types – consider observations against $(x : \langle P \to N \rangle^+ \to \langle X^+ \rangle^-)$.

The natural idea is to inductively invoke the distinguishability result: if $x\; (\lambda y.\, t) \not\approx_{\mathtt{icc}} x\; (\lambda y.\, t')$, then $t \not\approx_{\mathtt{icc}} t'$ are distinguishable and $\hat{x}$ can distinguish those two arguments by passing the right instantiation for y. However, making this intuition precise gets us into a quagmire of self-references: to define instantiation of the variable x, we may need to instantiate any such variable y whose scope it dominates, but the argument for distinguishability is really on the values that $\lambda y.\, t$, $\lambda y.\, t'$ have reduced to by the time they are passed to $\hat{x}$; but the natural way to denote those values makes a reference to the instances passed for all the variables in their scope, x included...

We are convinced that there is a general inductive argument to be found that would play in a beautiful way with general polarized type structure. We have not found it yet, and shall for now use (a focused version of) the function-removal hack from Section 2.5 (Fun-less types and reification).

6.2 Saturated inequivalence

We argued that if we have $t \not\approx_{\mathtt{icc}} t'$, then those terms must be of the form $D\,[n], D\,[n']$ or $D\,[p], D\,[p']$ where the neutrals $n \not\approx_{\mathtt{icc}} n' : X^-$ or $p \not\approx_{\mathtt{icc}} p' : P$ have distinct *spines* (invertible phases), that is, they differ even when ignoring their invertible subterms. With the simplifying assumption that all atoms are positively polarized, only the case $p \not\approx_{\mathtt{icc}} p'$ may arise.

This structure is crucial in the proof of canonicity, so we introduce in Figure 8 (Saturated inequivalence judgment) a precise inductive definition of this decomposition as a *saturated inequivalence* judgment

$$\Gamma \mid \Xi \vdash_{\mathsf{inv}} D\left[\Gamma' \mid \Xi' \vdash_{\mathtt{ne}} p \neq p' : P\right] : N \mid P_{\mathsf{a}}$$

which represents constructive evidence of the fact that $D\,[p] \not\approx_{\mathtt{icc}} D\,[p']$, where p, p' are the source of inequality as witnessed by the new judgment $\Gamma' \vdash_{\mathtt{ne}} p \neq p' \Uparrow P$.

The new structure Ξ is a *constraint environment*, a list of equalities of the form $p = x : P$ or $x = n : P$ that correspond to knowledge that was accumulated during the traversal of D: under a binding $\mathtt{let}\; x = n\; \mathtt{in}\; t$ we remember $x = n$, and when doing a case-split on a variable $x : P_1 + P_2$ we remember which branch leads to the source of inequality by $\sigma_i\; x' = x$. Together, $\Gamma' \mid \Xi'$ form a *constrained environment* as used in Fiore and Simpson (1999). Note that when decomposing the variable $x : P_1 + P_2$ we mention it in the constraint environment, so we also keep it in Γ' for scoping purposes: we use general contexts with types of any polarity, not just negative or atomic contexts Γ_{na}.

Two side-conditions in this judgment capture the essential properties of saturated terms required to obtain canonicity. In the saturation case, the condition $\forall n \in \bar{n},\; n \notin \Xi$ enforces that $\mathtt{let}$-bindings in the constraint environment Ξ all bind distinct neutrals. At the

source of inequality, the condition $\Gamma \not\vdash 0$ enforces that the context is consistent.

constraint environment $\quad \Xi ::= \emptyset \mid \Xi, p = x \mid \Xi, x = n$

$$\boxed{\Gamma \mid \Xi \vdash_{\mathsf{inv}} D\,[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P'] : N \mid Q_{\mathsf{a}}}$$

$$\frac{\Gamma, x : P_1 + P_2, x_i : P_i \mid \Xi, \sigma_i\ x_i = x \vdash_{\mathsf{inv}} D\,[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P'] : N \mid Q_{\mathsf{a}}}{\Gamma, x : P_1 + P_2 \mid \Xi \vdash_{\mathsf{inv}} \left(\begin{array}{l}\mathtt{match}\ x\ \mathtt{with} \\ \mid\ \sigma_i\ x_i \to D \\ \mid\ \sigma_j\ x_{j\neq i} \to t\end{array}\right) [\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P'] : N \mid Q_{\mathsf{a}}}$$

(other cases omitted for space)

$$\boxed{\Gamma \mid \Xi \vdash_{\mathsf{foc}} D\,[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P'] : Q_{\mathsf{a}}}$$

$$\frac{\forall n \in \bar{n},\ n \notin \Xi \qquad \Gamma, \bar{x} : \bar{P} \mid \Xi, \bar{x} = \bar{n} \vdash_{\mathsf{inv}} D\,[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P'] : \emptyset \mid Q_{\mathsf{a}}}{\Gamma \mid \Xi \vdash_{\mathsf{foc}} (\mathtt{let}\ (\bar{x} : \bar{P}) = \bar{n}\ \mathtt{in}\ D)\ [(\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P')] : Q_{\mathsf{a}}}$$

$$\frac{\Gamma \vdash_{\mathsf{ne}} p \neq p' \Uparrow P \qquad \Gamma \not\vdash 0}{\Gamma \mid \Xi \vdash_{\mathsf{foc}} \Box\,[\Gamma \mid \Xi \vdash_{\mathsf{ne}} p \neq p' : P] : P}$$ (other cases omitted for space)

$$\boxed{\Gamma \vdash_{\mathsf{ne}} p \neq p' \Uparrow P}$$

$$\frac{x \neq y}{\Gamma, x : X^+, y : X^+ \vdash_{\mathsf{ne}} x \neq y \Uparrow X^+} \qquad \frac{(i \neq j) \ \vee\ \Gamma \vdash_{\mathsf{ne}} p \neq p' \Uparrow P_{i=j}}{\Gamma \vdash_{\mathsf{ne}} \sigma_i\ p \neq \sigma_j\ p' \Uparrow P_1 + P_2}$$

no case for $\Gamma \vdash_{\mathsf{ne}} t \neq t' \Uparrow \langle N \rangle^+$

Figure 8. Saturated inequivalence judgment

Definition 13. *A constrained environment* $\Gamma \mid \Xi$ *is* valid *if*

$$n \in \Xi \implies \Gamma \vdash n \Downarrow \Xi \qquad p \in \Xi \implies \Gamma \vdash p \Uparrow \Xi$$

$$(p = x) \in \Xi, (p' = y) \in \Xi, x =_\alpha y \implies p =_\alpha p'$$

$$(x = n) \in \Xi, (y = n') \in \Xi, n =_\alpha n' \implies x =_\alpha y$$

Lemma 6 (Saturated inequivalence)**.**
If $\emptyset; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t \not\approx_{\mathtt{icc}} t' : N \mid Q_{\mathsf{a}}$ *with positive atoms only, then there exists* $D\,[\Box], p, p'$ *and a valid* Ξ *such that*

$$\Sigma_{\mathsf{p}} \mid \emptyset \vdash_{\mathsf{inv}} D\,[\Gamma'_{\mathsf{na}} \mid \Xi \vdash_{\mathsf{ne}} p \neq p' : P] : N \mid Q_{\mathsf{a}}$$

$$t \approx_{\mathtt{icc}} D\,[p] \qquad t' \approx_{\mathtt{icc}} D\,[p']$$

Let us write $(t \not\approx_{\mathtt{icc}} t') \rightsquigarrow D\,[p \neq p']$ *when this relation holds.*

6.3 Focused fun removal

In this section, we describe how to transform any pair of distinct saturated terms $t_0 \not\approx_{\mathtt{icc}} t'_0$ in $\mathsf{\Lambda C}(X^+, \to, \times, 1, +, 0)$ into a pair of distinct saturated terms in the restricted type system $\mathsf{\Lambda C}(\times, 1, +, 0)$. We know that we can apply the neutral model $\mathcal{N}_{t_0,t'_0}$ to get a pair of terms without atoms – $\mathsf{\Lambda C}(\to, \times, 1, +, 0)$ – and then the bijections of Section 2.5 (Fun-less types and reification) give us terms in $\mathsf{\Lambda C}(\times, 1, +, 0)$. But the impact of these transformations on the focused and saturated term structure must be studied carefully.

Reduction to closed types When applying the neutral model $\mathcal{N}_{t_0,t'_0}$, we turn atomic types X^+ into strictly positive types $\mathcal{N}(X^+)$ of the form $1 + \ldots + 1$. Consider a binding site of the form $(\lambda(x : X^+).\, t)$, which becomes $\lambda(x : 1 + \ldots + 1).\, \mathcal{N}(t)$ after transformation. To recover a valid focused term, we need to insert a big sum elimination to remove the positive variable x from the context:

$$\lambda x.\, \mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t[\sigma_i\ ()/x])^{i \in \mathcal{N}(X^+)}$$

(the family notation here denotes a cascade of sum-eliminations, with as many cases in total as elements in $\mathcal{N}(X^+)$). We also perform such an elimination on each variable in context at the root.

The substitution of $\sigma_i\ ()$ for $(x : X^+)$ in t may not create β-redexes, as a variable x of atomic type may not occur in eliminated-sum position. By inspection of the typing rules of our focused system, such a variable can only appear in a judgment of the form $\Gamma_{\mathsf{na}} \vdash x \Uparrow X^+$; the focused structure is preserved by replacing it by the derivation $\Gamma_{\mathsf{na}} \vdash \sigma_i\ () \Uparrow 1 + \ldots + 1$.

For a focused term t, let us simply write $\mathcal{N}(t)$ for this transformed focused term, splitting over each variable of type $\mathcal{N}(X^+)$.

The *saturated* structure, however, is not preserved by this change. The problem is that replacing a new variable x by a closed term $\sigma_i\ ()$ may break the condition that only "new" neutrals are introduced during a saturation phase: any neutral that would only use x as the new variable introduced by the last invertible phase is not new anymore.

However, we can show that the *saturated inequivalence* structure is preserved by this transformation. There is a shared context to a source of inequality in the transformed terms, where the `let`-bindings might not be new at introduction time, but they are not redundant, as requested by the inequivalence structure. This is the property of saturated proofs (besides saturation consistency) that our distinguishability result relies on.

Theorem 11 (Inequivalence in the model)**.** *Suppose we have saturated forms* t_0, t'_0 *with positive atoms only. Let us define* $t_1 \stackrel{\mathsf{def}}{=} \mathcal{N}_{t_0,t'_0}(t_0)$ *and* $t'_1 \stackrel{\mathsf{def}}{=} \mathcal{N}_{t_0,t'_0}(t'_0)$.

If $(t_0 \not\approx_{\mathtt{icc}} t'_0) \rightsquigarrow D_0\,[p_0 \neq {p'}_0]$*, then there exists* D_1*,* p_1*,* ${p'}_1$ *such that* $(t_1 \not\approx_{\mathtt{icc}} {t'}_1) \rightsquigarrow D_1\,[p_1 \neq {p'}_1]$

Function elimination To turn $t_1 \not\approx_{\mathtt{icc}} t'_1$ into terms at function-less types, we use the transformations presented in Figure 9 (Fun-less focused forms). They correspond to focused versions of the term transformation $\lfloor \_ \rfloor_A$ and $[\![\_]\!]_A$ of Figure 3 (Fun-less data types), in two interdependent ways: their definition assumes the transformed terms respect the focused structures, which gives us rich information on term shapes, and their result remain in focused form. The specification of these transformations is as follows:

$$\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t : N \mid P_{\mathsf{a}} \Rightarrow \lfloor \Gamma_{\mathsf{na}} \rfloor; \lfloor \Sigma_{\mathsf{p}} \rfloor \vdash_{\mathsf{inv}} \lfloor t \rfloor_{(N|P_{\mathsf{a}})} : \lfloor N \rfloor \mid \lfloor P_{\mathsf{a}} \rfloor$$

$$\Gamma_{\mathsf{na}}; \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} t : N \mid P_{\mathsf{a}} \Rightarrow [\![t]\!]_{(N|P_{\mathsf{a}})} [\![N]\!] [\![P_{\mathsf{a}}]\!]$$

$$\frac{\Gamma_{\mathsf{na}} \vdash n \Downarrow N \qquad \ldots}{\Gamma_{\mathsf{na}} \vdash m \Downarrow N'} \Rightarrow \exists m', \frac{[\![\Gamma_{\mathsf{na}}]\!] \vdash [\![n]\!]_N \Downarrow [\![N]\!] \qquad \ldots}{[\![\Gamma_{\mathsf{na}}]\!] \vdash m' \Downarrow N'}$$

The transformation of negative neutrals is more elegantly expressed in the sequent-calculus syntax of Curien, Fiore, and Munch-Maccagnoni (2016): we transform a command $\langle n \mid_N S \rangle : \Gamma_{\mathsf{na}} \vdash M$ cutting on a type N into a command $[\![\langle n \mid_N S \rangle]\!]_N : [\![\Gamma_{\mathsf{na}}]\!] \vdash M$ cutting on $[\![N]\!]$. We will write $m\,[n]$ when the neutral m has head n – it is of the form $\langle n \mid S \rangle$.

The definition strongly relies on the inversion of term structure of focused terms modulo $(\approx_{\mathtt{icc}})$. For example, when we define $[\![\lambda x.\, t]\!]_{(P_1+P_2)\to N}$, we know that $x : P_1 + P_2$ and can thus assume modulo $(\approx_{\mathtt{icc}})$ that t is of the form $\mathtt{match}\ x\ \mathtt{with}\ (\sigma_i\ x \to t_i)^i$. Similarly in the neutral case, we know that any neutral $\Gamma_{\mathsf{na}} \vdash n \Downarrow (P_1 + P_2) \to N$ can only appear in a term applied to an argument $\Gamma_{\mathsf{na}} \vdash p \Uparrow (P_1 + P_2)$ – non-invertible phases are as long as possible, so they cannot stop on a negative function type – and we know that such an p must be of the form $\sigma_i\ p'$.

$$\lfloor t \rfloor_A \qquad \text{applies } \lfloor \_ \rfloor_A \text{ on all subterms}$$

$$\llbracket \lambda x.\ \begin{array}{|l} \texttt{match}\ x\ \texttt{with} \\ \sigma_1\ x \to t_1 \\ \sigma_2\ x \to t_2 \end{array} \rrbracket_{(P_1+P_2)\to N} \stackrel{\text{def}}{=} \left(\begin{array}{l} \llbracket \lambda x.\, t_1 \rrbracket_{P_1 \to N} \\ , \\ \llbracket \lambda x.\, t_2 \rrbracket_{P_2 \to N} \end{array} \right)$$

$$\llbracket n \rrbracket_{(P_1+P_2)\to N}\ (\sigma_i\ p) \stackrel{\text{def}}{=} \llbracket \pi_i\ n \rrbracket_{P_i \to N}\ p$$

$$\llbracket \lambda x.\, \texttt{absurd}(x) \rrbracket_{0\to N} \stackrel{\text{def}}{=} () \qquad \llbracket n \rrbracket_{0\to N}\ p \text{ impossible}$$

$$\llbracket \lambda x.\, t \rrbracket_{\langle 1 \rangle^+ \to N} \stackrel{\text{def}}{=} t \qquad \llbracket n \rrbracket_{\langle 1 \rangle^+ \to N}\ () \stackrel{\text{def}}{=} n$$

$$\llbracket \lambda x.\, t \rrbracket_{\langle N_1 \times N_2 \rangle^+ \to M} \stackrel{\text{def}}{=} \llbracket \lambda x_1.\, \llbracket \lambda x_2.\, t[x_i/\pi_i\ x]^i \rrbracket_{N_2 \to \_} \rrbracket_{N_1 \to \_}$$

$$\llbracket n \rrbracket_{\langle N_1 \times N_2 \rangle^+ \to M}\ (t_1, t_2) \stackrel{\text{def}}{=} \llbracket \llbracket n \rrbracket_{N_1 \to \_}\ t_1 \rrbracket_{N_2 \to \_}\ t_2$$

$$\llbracket \lambda x.\, t \rrbracket_{\langle \langle P \rangle^- \rangle^+ \to N} \stackrel{\text{def}}{=} \llbracket \lambda x.\, t \rrbracket_{P \to N}$$

$$\llbracket n \rrbracket_{\langle \langle P \rangle^- \rangle^+ \to N}\ (\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathsf{inv}} p : \langle P \rangle^- \mid \emptyset) \stackrel{\text{def}}{=} \llbracket n \rrbracket_{P \to N}\ p$$

$$\llbracket x \rrbracket_N \stackrel{\text{def}}{=} (x : \llbracket N \rrbracket)$$

Figure 9. Fun-less focused forms

The way the focused structure conspires to make this transformation possible is, in fact, rather miraculous – is it more than just a hack? In the $\lambda x.\, t$ case of $\langle N_1 \times N_2 \rangle^+ \to M$, we know that a derivation of the form $\Gamma_{\mathsf{na}} \vdash x \Downarrow N_1 \times N_2$ can only appear inside a larger derivation $\Gamma_{\mathsf{na}} \vdash \pi_i\ x \Downarrow N_i$, which we can replace by $\llbracket \Gamma_{\mathsf{na}} \rrbracket \vdash x_i \Downarrow N_i$. In the neutral case, a $(n : \langle N_1 \times N_2 \rangle^+ \to \_)$ can only appear applied to an argument $\Gamma_{\mathsf{na}} \vdash p \Uparrow \langle N_1 \times N_2 \rangle^+$, but as this is a shifted negative type we know that p is in fact an invertible form $\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathsf{inv}} t : N_1 \times N_2 \mid \emptyset$, and this must be a pair (t_1, t_2), exactly what we needed to define the transformation. The same phenomenon occurs on the argument $\Gamma'_{\mathsf{na}} \vdash p \Uparrow \langle \langle P \rangle^- \rangle^+$ of the double-shifted case.

Lemma 7 (Function elimination preserves inequivalence)**.**

$$(\exists D, p, p',\ (t \not\approx_{\mathtt{icc}} t') \rightsquigarrow D\ [p \neq p'])$$

$$\implies (\exists D, p, p',\ (\lfloor t \rfloor \not\approx_{\mathtt{icc}} \lfloor t' \rfloor) \rightsquigarrow D\ [p \neq p'])$$

(The converse implication also holds, but we don't need it.)

Lemma 8.

$$\left[\!\left[\lfloor \lfloor t \rfloor_A \rfloor_{\mathtt{foc}} \right]\!\right] = \left[\!\left[\lfloor \lfloor t \rfloor_{\mathtt{foc}} \rfloor_{\lfloor A \rfloor_\pm} \right]\!\right] = \lfloor \left[\!\left[\lfloor t \rfloor_{\mathtt{foc}} \right]\!\right] \rfloor_{\lfloor \lfloor A \rfloor_\pm \rfloor} \in \left[\!\left[\lfloor A \rfloor_\pm \right]\!\right]$$

Corollary 3 (Fun-elimination preserves semantic equivalence)**.**

$$[\![t]\!] = [\![t']\!] \qquad \iff \qquad [\![\lfloor t \rfloor_A]\!] = [\![\lfloor t' \rfloor_A]\!]$$

6.4 Canonicity

Definition 14. *Let a* closed substitution ρ *be a mapping from variables to closed terms of closed types (types without atoms).*

We write $\rho : \Gamma$ *when* Γ *is a closed context and, for any* $(x : A) \in \Gamma$ *we have* $\emptyset \vdash x[\rho] : A$.

If Ξ *is a constraint environment, we say that* $\Xi[\rho]$ *holds when, for any equation* $t = u$ *in* Ξ*, we have* $t[\rho] \approx_{\mathtt{sem}} u[\rho]$.

We write $\rho : (\Gamma \mid \Xi)$ *if* $\rho : \Gamma$ *and* $\Xi[\rho]$ *hold.*

Theorem 12 (Canonicity)**.** *Assume* Γ *is consistent,* Ξ *is valid, and*

$$\Gamma' \mid \emptyset \vdash_{\mathsf{inv}} D\ [\Gamma \mid \Xi \vdash_{\mathtt{ne}} p \neq p' : P] : N \mid P_{\mathsf{a}}$$

is a judgment of closed function-less types. Then there exists a closed substitution $\rho : (\Gamma \mid \Xi)$ *such that* $p[\rho] \not\approx_{\mathtt{sem}} p'[\rho]$.

Corollary 4 (Canonicity)**.** *In the sub-system* $\mathsf{\Lambda C}(\times, 1, +, 0)$*:*

$$\Gamma \mid \emptyset \vdash_{\mathsf{inv}} D\ [\Gamma' \mid \Xi' \vdash_{\mathtt{ne}} p \neq p' : P] : N \mid P_{\mathsf{a}} \Rightarrow D\ [p] \not\approx_{\mathtt{ctx}} D\ [p']$$

6.5 Results

Theorem 13 (Saturated terms are canonical)**.** *In the system with only positive atoms, if* $\emptyset; \Sigma_{\mathsf{p}} \vdash_{\mathsf{sinv}} t \not\approx_{\mathtt{icc}} t' : N \mid P_{\mathsf{a}}$ *then* $t \not\approx_{\mathtt{ctx}} t'$.

Corollary 5 (Contextual equivalence implies equality of saturated forms)**.** *If* $\Gamma \vdash t, t' : A$ *are (non-focused) terms of the full simply-typed lambda-calculus* $\mathsf{\Lambda C}(X, \to, \times, 1, +, 0)$ *with* $t \approx_{\mathtt{ctx}} t'$*, then for any* Σ_{p}, N *with no positive atoms and* $\lfloor \Sigma_{\mathsf{p}} \rfloor_\pm = \Gamma, \lfloor N \rfloor_\pm = A$ *and any saturated terms* $\emptyset; \Sigma_{\mathsf{p}} \vdash_{\mathsf{sinv}} u, u' : N \mid \emptyset$ *such that* $t \approx_{\beta\eta} \lfloor u \rfloor_{\mathtt{foc}}$ *and* $t' \approx_{\beta\eta} \lfloor u' \rfloor_{\mathtt{foc}}$ *we have* $u \approx_{\mathtt{icc}} u'$.

Corollary 6. *Contextual and* $\beta\eta$*-equivalence coincide*

Corollary 7. *Equivalence in the full simply-typed* λ*-calculus with sums and the empty type is decidable.*

Corollary 8. *The full simply-typed* λ*-calculus with sums and the empty type has the finite model property.*

7. Conclusion

7.1 Other Related Work

Background material For the reader looking for more historical perspective Dougherty and Subrahmanyam (2000) gives an interesting, detailed presentation of the state of the art in separation theorems in absence of sum types, and of why their approach are difficult to extend to sums. Simpson (1995) gives an enjoyable, accessible exposition of what "completeness" exactly means in the setting of categorical semantics; in particular, while we can prove that two inequivalent terms can be distinguished by a choice of finite sets, there is no fixed choice of finite sets that could separate all pairs of terms. It also discusses the notion of "typical ambiguity", and the idea that $\beta\eta$-equality is shown to be the maximal consistent relation. At the time of writing, Simpson's statement of the 15th TLCA open problem is also one of the clearest expositions of the relation between these questions.

Focusing The main field of application of focusing to programming is the area of *logic programming*, where operational semantics correspond to search strategies, which can often be described as specific choices of polarization in a focused logic (Chaudhuri, Pfenning, and Price 2008b). Focusing has been used to study programming language principles in Zeilberger (2009); when considering non-normal forms (logics with a strong cut rule) it let us reason finely about evaluation order.

This suggests the general notion of *polarization*, an approach of programming calculi where compositions (cuts) are non-associative (Curien, Fiore, and Munch-Maccagnoni 2016), modeling effects and resources. In the present work we consider only focused normal forms (except for stating the completeness of focusing), which only captures the pure, strongly normalizing fragment, and thus corresponds to a *depolarized* system.

Guillaume Munch-Maccagnoni's work on polarized abstract machine calculi harps on many ideas close to the present work, in particular Munch-Maccagnoni and Scherer (2015). It is conducted in a syntax that is inspired by the sequent calculus rather than the λ-calculus; this choice gives a beautiful dynamic semantics to polarized systems. In contrast, the focused λ-calculus as presented is a system of normal forms, and defining an internal reduction for it would be awkward. While there is no direct correspondence between sequent proofs and natural deduction proofs in general,

their normal forms are in one-to-one mapping, so in our context the choice of abstract machine or λ-terms-inspired syntax matters less.

Atom-less systems The structure of normal forms has been studied in Altenkirch and Uustalu (2004) in the special case of the type system $\mathsf{\Lambda C}(\to, +, 2)$, with boolean types instead of general sums, and no atoms. While conceptually easier to follow than the Grothendieck logical relations of the more general normalization-by-evaluation work on sums, they remain challenging in presence of higher-order functions.

In unpublished work, Ahmad, Licata, and Harper (2010) work with the type system $\mathsf{\Lambda C}(\to, \times, 1, +, 0)$: no atoms, but everything else. They use focusing as a guiding principle to generalize the normal forms of $\mathsf{\Lambda C}(\to, +, 2)$ to everything else – to our knowledge this is the first work to use focusing to approach sum types. The result obtained, namely decidability of observational equivalence in this system, is not striking on its own: in absence of atoms, all types are finitely inhabited, so two functions can be compared by testing them on all their input domain. But it is conducted in a rigorous, inspiring way that shows the promises of the focused structure. Our own proof of distinguishability of distinct normal forms is not as elegant as this development, as it uses the inelegant shortcut of function type elimination. We are convinced that there exists a beautiful proof that weaves the distinguishability structure through higher-order abstractions in their style, but have yet to find it.

7.2 Future Work

Direct distinguishability proof The use of the positive atoms simplification and the detour through types without functions are a bit disappointing. There should be a proof in which all steps are as general and general as possible: completeness of focusing in the explicitly polarized system, completeness of saturated forms, and then canonicity of saturated forms at all types.

Categorical semantics We wonder what is the relation between the saturated structure and the existing work on categorical semantics of the λ-calculus with finite sums.

Direct comparison algorithm We prove decidability by reducing equivalence of arbitrary λ-terms to equivalence of saturated forms. This algorithm can be implemented, but we would rather use an algorithm that does not need to compute full saturated normal forms before returning a result – in particular on inequivalent inputs.

It is reasonably easy to formulate an equivalence algorithm on the focused forms directly, that would perform the saturation "on the fly": at the beginning of each saturation phase, look for all the neutrals that can be defined in the current context, recursively compute their equivalence classes, and replace each class by a distinct free variable – then continue on the neutral structure. Proving this algorithm correct, however, turns out to be challenging.

Finally, it would be interesting to have an algorithm expressed directly on the non-focused terms, that would perform a focusing transformation on the fly, as much as each step of equivalence requires. The difficulty then is that neutral subterms are not as long as possible (they may be interrupted by commuting conversions), so it is possible to look for neutrals sub-terms definable in the current context and miss some of them – future reasoning stages may un-stuck sum-eliminations that in turn un-block neutrals that should have been bound now. For the type system with non-empty sums, control operators have been used (Balat, Di Cosmo, and Fiore 2004) to solve that issue; can the technique be extended?

Acknowledgments

Amal Ahmed made this work possible by giving the time, motivation and advice that let it foster. Andrew Pitts gave warm encouragements to look at the specific question of the empty type, and Alex Simpson enthusiastically sat through an intense whiteboard session at Schloss Dagstuhl. Max New, Didier Rémy and the anonymous referees provided very useful feedback on the article.

This research was supported in part by the National Science Foundation (grant CCF-1422133).

Errata

We list here errors that are not typos, but really technical mistakes, that appeared in the first publicly released version of this document and have since been fixed.

[January 27, 2017] Domenico Ruoppolo kindly pointed out a mistake in the presentation of the result of Böhm (1968): we wrote that $\beta\eta$ and contextual equivalence coincide on the untyped λ-calculus, but it is only true for β-normalizable terms. Non-terminating terms may be indistinguishable yet syntactically distinct, consider the applications $\Omega\ 0$ and $\Omega\ 1$ where Ω is the usual non-normalizing term $(\lambda x.\, x\ x)\ (\lambda x.\, x\ x)$.

References

Arbob Ahmad, Daniel R. Licata, and Robert Harper. Deciding coproduct equality with focusing. Online draft, 2010. 6, 13

Thorsten Altenkirch and Tarmo Uustalu. Normalization by evaluation for lambda^{-2}. In *FLOPS*, 2004. 4, 13

Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Philip J. Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *LICS*, 2001. 1

Jean-Marc Andreoli. Logic Programming with Focusing Proof in Linear Logic. *Journal of Logic and Computation*, 2(3), 1992. 1, 4

Vincent Balat, Roberto Di Cosmo, and Marcelo P. Fiore. Extensional normalisation and type-directed partial evaluation for typed lambda calculus with sums. In *POPL*, 2004. 1, 13

Corrado Böhm. Alcune proprieta delle forme normali nel k-calcolo. *IAC Pubbl*, 696119, 1968. 1, 13

Kaustuv Chaudhuri, Dale Miller, and Alexis Saurin. Canonical sequent proofs via multi-focusing. In *IFIP TCS*, 2008a. 1, 7

Kaustuv Chaudhuri, Frank Pfenning, and Greg Price. A logical characterization of forward and backward chaining in the inverse method. *J. Autom. Reasoning*, 40(2-3), 2008b. 6, 12

Kaustuv Chaudhuri, Stefan Hetzl, and Dale Miller. A Systematic Approach to Canonicity in the Classical Sequent Calculus. In *CSL*, 2012. 1

Pierre-Louis Curien, Marcelo Fiore, and Guillaume Munch-Maccagnoni. A Theory of Effects and Resources: Adjunction Models and Polarised Calculi. In *Proc. POPL*, 2016. doi: 10.1145/2837614.2837652. 5, 9, 10, 11, 12

Daniel J Dougherty and Ramesh Subrahmanyam. Equality between functionals in the presence of coproducts. *Information and Computation*, 157 (1), 2000. 1, 2, 12

Marcelo Fiore and Alex Simpson. Lambda definability with sums via grothendieck logical relations. In *TLCA*, 1999. 1, 10

Harvey Friedman. Equality between functionals. In *Logic Colloquium*, 1975. 1

Neil Ghani. Beta-Eta Equality for Coproducts. In *TLCA*, 1995. 1

Danko Ilik. The exp-log normal form of types and canonical terms for lambda calculus with sums. *CoRR*, `arxiv:1502.04634`, 2015. URL `http://arxiv.org/abs/1502.04634`. 4

Chuck Liang and Dale Miller. Focusing and polarization in intuitionistic logic. *CoRR*, `arxiv:0708.2252`, 2007. URL `http://arxiv.org/abs/0708.2252`. 5, 6

Sam Lindley. Extensional rewriting with sums. In *TLCA*, 2007. 1

Guillaume Munch-Maccagnoni and Gabriel Scherer. Polarised Intermediate Representation of Lambda Calculus with Sums. In *LICS*, 2015. 12

Gabriel Scherer. *Which types have a unique inhabitant? Focusing on pure program equivalence.* PhD thesis, Université Paris-Diderot, 2016. 2

Gabriel Scherer and Didier Rémy. Which simple types have a unique inhabitant? In *ICFP*, 2015. 1, 2, 5, 7, 15

Robert J. Simmons. Structural focalization. *CoRR*, `arxiv:1109.6273`, 2011. URL `http://arxiv.org/abs/1109.6273`. 6

Alex Simpson. Categorical completeness results for the simply-typed lambda-calculus. In *TLCA*, 1995. 12

Richard Statman. Completeness, equivalence and lambda-definability. *Journal of Symbolic Logic*, 47(1), 1982. 1

Noam Zeilberger. *The Logical Basis of Evaluation Order and Pattern-Matching*. PhD thesis, Carnegie Mellon University, 2009. 8, 10, 12

Noam Zeilberger. Polarity in proof theory and programming, 2013. URL `http://noamz.org/talks/logpolpro.pdf`. Lecture Notes for the Summer School on Linear Logic and Geometry of Interaction in Torino, Italy. 7

A. Full definitions

Definition (Semantics of terms)**.** *We define the following naive semantics for term formers:*

$$\mathtt{var}_x : [\![\Gamma, x : A \vdash A]\!]_{\mathcal{M}}$$
$$\mathtt{var}_x(G) \stackrel{\text{def}}{=} G(x)$$

$$\mathtt{pair} : [\![\Gamma \vdash A_1]\!]_{\mathcal{M}} \times [\![\Gamma \vdash A_2]\!]_{\mathcal{M}} \to [\![\Gamma \vdash A_1 \times A_2]\!]_{\mathcal{M}}$$
$$\mathtt{pair}(f_1, f_2)(G) \stackrel{\text{def}}{=} (f_1(G), f_2(G))$$

$$\mathtt{proj}_i : [\![\Gamma \vdash A_1 \times A_2]\!]_{\mathcal{M}} \to [\![\Gamma \vdash A_i]\!]_{\mathcal{M}}$$
$$\mathtt{proj}_i(f)(G) \stackrel{\text{def}}{=} v_i \quad \textit{where } f(G) = (v_1, v_2)$$

$$\mathtt{lam} : [\![\Gamma, x : A \vdash B]\!]_{\mathcal{M}} \to [\![\Gamma \vdash A \to B]\!]_{\mathcal{M}}$$
$$\mathtt{lam}(f)(G) \stackrel{\text{def}}{=} (v \in [\![A]\!]_{\mathcal{M}}) \mapsto f(G, x \mapsto v)$$

$$\mathtt{app} : [\![\Gamma \vdash A \to B]\!]_{\mathcal{M}} \times [\![\Gamma \vdash A]\!]_{\mathcal{M}} \to [\![\Gamma \vdash B]\!]_{\mathcal{M}}$$
$$\mathtt{app}(f, g)(G) \stackrel{\text{def}}{=} f(G)(g(G))$$

$$\mathtt{unit} : [\![\Gamma \vdash 1]\!]_{\mathcal{M}}$$
$$\mathtt{unit}(G) \stackrel{\text{def}}{=} \star$$

$$\mathtt{inj}_i : [\![\Gamma \vdash A_i]\!]_{\mathcal{M}} \to [\![\Gamma \vdash A_1 + A_2]\!]_{\mathcal{M}}$$
$$\mathtt{inj}_i(f)(G) \stackrel{\text{def}}{=} (i, f(G))$$

$$\mathtt{match} : [\![\Gamma \vdash A_1 + A_2]\!]_{\mathcal{M}} \times [\![\Gamma, x : A_1 \vdash B]\!]_{\mathcal{M}} \times [\![\Gamma, x : A_2 \vdash B]\!]_{\mathcal{M}} \to [\![\Gamma \vdash B]\!]_{\mathcal{M}}$$
$$\mathtt{match}(f, g_1, g_2)(G) \stackrel{\text{def}}{=} g_i(G, x \mapsto v) \quad \textit{where } f(G) = (i, v)$$

$$\mathtt{absurd} : [\![\Gamma \vdash \emptyset]\!]_{\mathcal{M}} \to [\![\Gamma \vdash A]\!]_{\mathcal{M}}$$
$$\mathtt{absurd} \stackrel{\text{def}}{=} \emptyset$$

By composing together the semantics of the term formers in the obvious way, we obtain semantics for terms t and one-hole contexts C:

$$[\![\Gamma \vdash t : A]\!]_{\mathcal{M}} \in [\![\Gamma \vdash A]\!]_{\mathcal{M}}$$

$$[\![\Gamma \vdash C\,[\Gamma' \vdash \Box : A'] : A']\!]_{\mathcal{M}} \in [\![\Gamma, \Gamma' \vdash A']\!]_{\mathcal{M}} \to [\![\Gamma \vdash A]\!]_{\mathcal{M}}$$

For example we have $[\![(t_1, t_2)]\!]_{\mathcal{M}} = \mathtt{pair}([\![t_1]\!]_{\mathcal{M}}, [\![t_2]\!]_{\mathcal{M}})$ and $[\![(t, \Box)]\!]_{\mathcal{M}}(f) = \mathtt{pair}([\![t_1]\!]_{\mathcal{M}}, f)$. In particular, $[\![\Box]\!]_{\mathcal{M}}$ is the identity function.

Definition (Fun-less types and reification)**.**

(The definitions of $\lfloor A \rfloor$ and $[\![A]\!]$ were given in Figure 3 (Fun-less data types).)

$$\lfloor \_ \rfloor_A \quad : \quad A \to \lfloor A \rfloor$$
$$\lfloor (t_1, t_2) \rfloor \stackrel{\text{def}}{=} (\lfloor t_1 \rfloor, \lfloor t_2 \rfloor)$$
$$\lfloor \sigma_i\ t \rfloor \stackrel{\text{def}}{=} \sigma_i\ \lfloor t \rfloor$$
$$\lfloor () \rfloor \stackrel{\text{def}}{=} ()$$
$$\lfloor t \rfloor_{A \to B} \stackrel{\text{def}}{=} [\![\lambda x.\, \lfloor t\ \lceil x \rceil_A \rfloor_B]\!]_{A \to B}$$

$$\lceil \_ \rceil_A \quad : \quad \lfloor A \rfloor \to A$$
$$\lceil (t_1, t_2) \rceil \stackrel{\text{def}}{=} (\lceil t_1 \rceil, \lceil t_2 \rceil)$$
$$\lceil \sigma_i\ t \rceil \stackrel{\text{def}}{=} \sigma_i\ \lceil t \rceil$$
$$\lceil () \rceil \stackrel{\text{def}}{=} ()$$
$$\lceil t \rceil_{\lfloor A \rfloor \to \lfloor B \rfloor} \stackrel{\text{def}}{=} \lceil \lambda x.\, \lceil t\ \lfloor x \rfloor_A \rceil_B \rceil$$

$$\lfloor \_ \rfloor_A \quad : \quad [\![A]\!] \to [\![\lfloor A \rfloor]\!]$$
$$\lfloor (v_1, v_2) \rfloor \stackrel{\text{def}}{=} (\lfloor v_1 \rfloor, \lfloor v_2 \rfloor)$$
$$\lfloor (i, v) \rfloor \stackrel{\text{def}}{=} (i, \lfloor v \rfloor)$$
$$\lfloor \star \rfloor \stackrel{\text{def}}{=} \star$$
$$\lfloor v \rfloor_{A \to B} \stackrel{\text{def}}{=} [\![w \mapsto \lfloor v(\lceil w \rceil_A) \rfloor_B]\!]_{A \to B}$$

$$\lceil \_ \rceil_A \quad : \quad [\![\lfloor A \rfloor]\!] \to [\![A]\!]$$
$$\lceil (v_1, v_2) \rceil \stackrel{\text{def}}{=} (\lceil v_1 \rceil, \lceil v_2 \rceil)$$
$$\lceil (i, v) \rceil \stackrel{\text{def}}{=} (i, \lceil v \rceil)$$
$$\lceil \star \rceil \stackrel{\text{def}}{=} \star$$
$$\lceil v \rceil_{A \to B} \stackrel{\text{def}}{=} [\![w \mapsto \lceil v(\lfloor w \rfloor_A) \rceil_B]\!]_{A \to B}$$

$$\begin{aligned}
&\llbracket t \rrbracket_{A_1 \times A_2 \to B} \stackrel{\text{def}}{=} \llbracket \lambda x_1.\, \llbracket \lambda x_2.\, t\ (x_1, x_2) \rrbracket_{A_2 \to B} \rrbracket_{A_1 \to \llbracket A_2 \to B \rrbracket} \\
&\llbracket t \rrbracket_{1 \to B} \stackrel{\text{def}}{=} t\ () \\
&\llbracket t \rrbracket_{A_1 + A_2 \to B} \stackrel{\text{def}}{=} \left(\llbracket \lambda x.\, t\ (\sigma_1\ t) \rrbracket_{A_1 \to B}, \llbracket \lambda x.\, t\ (\sigma_2\ t) \rrbracket_{A_2 \to B} \right) \\
&\llbracket t \rrbracket_{0 \to B} \stackrel{\text{def}}{=} () \\
\\
&\lceil t \rceil_{A_1 \times A_2 \to B} \stackrel{\text{def}}{=} \lambda x.\, \lceil \lceil t \rceil_{A_1 \to \llbracket A_2 \to B \rrbracket}\ (\pi_1\ x) \rceil_{A_2 \to B}\ (\pi_2\ x) \\
&\lceil t \rceil_{1 \to B} \stackrel{\text{def}}{=} \lambda x.\, t \\
&\lceil t \rceil_{A_1 + A_2 \to B} \stackrel{\text{def}}{=} \lambda x.\, \texttt{match}\ x\ \texttt{with} \mid \begin{array}{l} \sigma_1\ x_1 \to \lceil \pi_1\ t \rceil_{A_1 \to B}\ x_1 \\ \sigma_2\ x_2 \to \lceil \pi_2\ t \rceil_{A_2 \to B}\ x_2 \end{array} \\
&\lceil t \rceil_{0 \to B} \stackrel{\text{def}}{=} \lambda x.\, \texttt{absurd}(x) \\
\\
&\llbracket v \rrbracket_{A_1 \times A_2 \to B} \stackrel{\text{def}}{=} \llbracket w_1 \mapsto \llbracket w_2 \mapsto v((w_1, w_2)) \rrbracket_{A_2 \to B} \rrbracket_{A_1 \to \llbracket A_2 \to B \rrbracket} \\
&\llbracket v \rrbracket_{1 \to B} \stackrel{\text{def}}{=} v(\star) \\
&\llbracket v \rrbracket_{A_1 + A_2 \to B} \stackrel{\text{def}}{=} \left(\llbracket w \mapsto v(((1, v))) \rrbracket_{A_1 \to B}, \llbracket w \mapsto v(((2, v))) \rrbracket_{A_2 \to B} \right) \\
&\llbracket v \rrbracket_{0 \to B} \stackrel{\text{def}}{=} \star \\
\\
&\lceil v \rceil_{A_1 \times A_2 \to B} \stackrel{\text{def}}{=} (w_1, w_2) \mapsto \lceil \lceil v \rceil_{A_1 \to \llbracket A_2 \to B \rrbracket}(w_1) \rceil_{A_2 \to B}(w_2) \\
&\lceil v \rceil_{1 \to B} \stackrel{\text{def}}{=} \star \mapsto v \\
&\lceil (v_1, v_2) \rceil_{A_1 + A_2 \to B} \stackrel{\text{def}}{=} (i, w) \mapsto \lceil v_i \rceil_{A_i \to B}(w) \\
&\lceil v \rceil_{0 \to B} \stackrel{\text{def}}{=} \emptyset
\end{aligned}$$

Definition (Depolarization)**.**

$$\begin{array}{lcl@{\qquad}lcl}
\lfloor X^+ \rfloor_\pm & \stackrel{\text{def}}{=} & X & \lfloor Y^- \rfloor_\pm & \stackrel{\text{def}}{=} & Y \\
\lfloor P + Q \rfloor_\pm & \stackrel{\text{def}}{=} & \lfloor P \rfloor_\pm + \lfloor Q \rfloor_\pm & \lfloor P \to N \rfloor_\pm & \stackrel{\text{def}}{=} & \lfloor P \rfloor_\pm \to \lfloor N \rfloor_\pm \\
\lfloor 0 \rfloor_\pm & \stackrel{\text{def}}{=} & 0 & \lfloor N \times M \rfloor_\pm & \stackrel{\text{def}}{=} & \lfloor N \rfloor_\pm \times \lfloor M \rfloor_\pm \\
\left\lfloor \langle N \rangle^+ \right\rfloor_\pm & \stackrel{\text{def}}{=} & \lfloor N \rfloor_\pm & \lfloor 1 \rfloor_\pm & \stackrel{\text{def}}{=} & 1 \\
 & & & \left\lfloor \langle P \rangle^- \right\rfloor_\pm & \stackrel{\text{def}}{=} & \lfloor P \rfloor_\pm
\end{array}$$

B. Proof outlines

Theorem 3 ($\beta\eta$ is semantically sound). This is proved by direct induction on the evidence that $t \approx_{\beta\eta} u$; semantic equivalence is a congruence, and β-reduction and η-expansions are identities in the semantic domain. □

Theorem 4 (Semantic equivalence implies contextual equivalence). For any model $\mathcal{M}$ and context $\emptyset \vdash C\,[\Box] : 1 + 1$, the closed term $C\,[\mathcal{M}(t)]$ must have a normal form $\sigma_i\ ()$ by Lemma 1 (Inversion), and $C\,[\mathcal{M}(u)]$ a normal form $\sigma_j\ ()$. The semantic interpretation of $C\,[t]$ (technically, $\llbracket C \rrbracket(\llbracket t \rrbracket_{\mathcal{M}})$ for the natural semantics of contexts with a hole) and of $C\,[u]$ are equal by our assumption $t \approx_{\texttt{sem}} u$ and the fact that semantic equality is a congruence. They are also are respectively equal to $(i, \star)$ and $(j, \star)$ by Theorem 3 ($\beta\eta$ is semantically sound), so we must have $i = j$. $C\,[\Box]$ is not separating. □

Theorem 5 (Reification). We define $\texttt{reify}(v) \stackrel{\text{def}}{=} \lceil \texttt{reify}'(\lfloor v \rfloor_A) \rceil_A$ using an (obvious) auxiliary function $\texttt{reify}'(v)$ on values at fun-less types, on which the inversion property is immediate; for $\texttt{reify}(\_)$ it is then proved using the fact that $\llbracket \_ \rrbracket$ and $\lfloor \_ \rfloor_A$, $\lceil \_ \rceil_A$ commute. □

Lemma 2. By Theorem 3 ($\beta\eta$ is semantically sound), we can without loss of generality consider β-normal forms only. For the fun-less $\texttt{reify}'(\_)$, direct induction proves $\texttt{reify}'(\llbracket t \rrbracket) = t$ on normal-forms t – note that the fact that our types have no functions is key to preserve the inductive invariant that our terms are closed, necessary to use the inversion lemma to reason on the shape of t. The result for $\texttt{reify}(\_)$ holds by commutation of $\llbracket \_ \rrbracket$ and $\lfloor \_ \rfloor_A$, $\lceil \_ \rceil_A$. □

Theorem 6 (Contextual equivalence implies semantic equivalence). We prove the contrapositive: if $\llbracket t \rrbracket_{\mathcal{M}} \neq \llbracket t' \rrbracket_{\mathcal{M}}$ in some model $\mathcal{M}$, we build a discriminating context $C\,[\Box]$ in $\mathcal{M}$. Our induction on a type A has a slightly stronger hypothesis. We assume $\mathcal{M}(\Gamma) \vdash t, t' : A(\mathcal{M})$ and $\llbracket t \rrbracket \neq \llbracket t' \rrbracket$: the semantically distinct terms are well-typed at the closed typing $\mathcal{M}(\Gamma) \vdash \_ : \Gamma(A)$, but need not be well-typed at $\Gamma \vdash \_ : A$.

In the function case $A \to B$ we have an input $w \in \llbracket A \rrbracket_{\mathcal{M}}$ where the two semantics differ, we build the discriminating context $C\,[(\Box\ (\texttt{reify}_A(w)))]$, where C distinguishes at B by induction hypothesis.

In the sum case, $\llbracket t \rrbracket_{\mathcal{M}}$ is of the form (i, v) and $\llbracket t' \rrbracket_{\mathcal{M}}$ of the form (j, v') and we have either $i \neq j$ or $v \neq v'$. In the second case, suppose for example $i = j = 1$, we use a context $(\texttt{match}\ \Box\ \texttt{with} \mid \sigma_1\ x_1 \to C\,[x_1] \mid \sigma_2\ x_2 \to \sigma_k\ ())$, using the fact that the return type $1 + 1$ is inhabited to build a dummy value for the unused branch ($k \in \{1, 2\}$ is arbitrary). □

Lemma 4 (Strong positive cut). By simultaneous induction on p and t. In the variable case, it's just a variable/variable substitution. □

Theorem 8 (Completeness of saturation). The proof of Scherer and Rémy (2015) can be reused almost as-is. It defines a rewriting relation from focused terms to saturated terms, that proceeds by moving `let`-bindings around: at any saturation point in some context $\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}}$, it looks for all the negative neutral terms in t that would be well-typed in $\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}}$, and saturates over exactly those neutral terms.

However, just selecting the neutral sub-terms of t does not make a valid selection function. We also have to add enough neutrals at each saturation step to respect the validity criterion; by the subformula property, a finite number of neutrals always suffices, so there exists a valid selection function with the neutral sub-terms of t.

Finally, any selection function above this one also satisfies completeness. Introducing more neutrals either does not change the term shape (after the next invertible phase) or exposes a neutral proof of inconsistency $(n : 0)$; in the latter case, more terms become $\beta\eta$-equivalent, which helps us proving our goal $\lfloor t \rfloor_{\texttt{foc}} \approx_{\beta\eta} \lfloor t' \rfloor_{\texttt{foc}}$. □

Theorem 9 (Saturation). By induction on the derivation. In the base case, the old context is empty so the result is trivial – there is no P such that $\emptyset \Downarrow \langle P \rangle^-$. The induction case considers the application of the SAT rule to deduce $\Gamma_{\mathsf{na}}; \Gamma'_{\mathsf{na}} \vdash_{\mathsf{sat}} \texttt{let}\ \bar{x} = \bar{n}\ \texttt{in}\ t : Q_{\mathsf{a}}$. We have to prove that any sub-derivation of t of the form $\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}}; \Gamma''_{\mathsf{na}\,k} \vdash_{\mathsf{sat}} f_k : Q_{\mathsf{a}\,k}$ satisfies the expected property for any P.

If $\Gamma_{\mathsf{na}} \Downarrow P$, this is immediate by induction hypothesis. If we have $\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}} \Downarrow P$ but not $\Gamma_{\mathsf{na}} \Downarrow P$, then P is only provable by a "new" neutral n using variables from Γ'_{na}, so it is passed to the selection function. By validity of the selection function (Figure 7 (Specification of saturation selection functions)) we know that some neutral $n' : P$ must then selected, so P occurs in the positive context of the invertible term t. By Lemma 3 (Strong decomposition of invertible phases) we thus have $\Gamma''_{\mathsf{na}\,k} \Uparrow P$ as required. □

Lemma 5 (Saturated consistency). By (logical) completeness of focusing it suffices to prove that there is no *focused* proof of $\Gamma_{\mathsf{na}}; \emptyset \vdash_{\mathsf{inv}} t : \emptyset \mid 0$.

By inversion on the possible derivations, we see that the invertible phase must be empty, and FOCLC-LET-POS is the only possible choice of focusing: t is of the form $\texttt{let}\ x = n\ \texttt{in}\ u$ with

$$\frac{\Gamma_{\mathsf{na}} \vdash n \Downarrow \langle Q \rangle^- \qquad \Gamma_{\mathsf{na}}; Q \vdash_{\mathsf{inv}} u : \emptyset \mid 0}{\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} \texttt{let}\ x = n\ \texttt{in}\ u : 0}$$

Γ_{na} is saturated so $\Gamma_{\mathsf{na}} \Downarrow \langle Q \rangle^-$ implies $\Gamma_{\mathsf{na}} \Uparrow Q$. But then, by Lemma 4 (Strong positive cut), we know that u has a subterm $\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} f : 0$.

We have proved that any term in the judgment $\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} 0$ has a strictly smaller subterm itself in the judgment $\Gamma_{\mathsf{na}} \vdash_{\mathsf{foc}} 0$. There is no such proof term. □

Theorem 10 (Inconsistent canonicity). The inversible and saturation phase are purely type-directed, so f and f' necessarily use the same term-formers during those phases. They could only differ on a neutral term, but they cannot contain a derivation of the form $\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}} \vdash_{\mathsf{s}} n \Downarrow X^-$ or $\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}} \vdash_{\mathsf{s}} p \Uparrow P$ as, by Corollary 2 (Saturation), $\Gamma_{\mathsf{na}}, \Gamma'_{\mathsf{na}}$ would be saturated, which is incompatible with our assumption $\Gamma_{\mathsf{na}} \vdash 0$ and Lemma 5 (Saturated consistency). □

Lemma 6 (Saturated inequivalence). By direct induction on $t \not\approx_{\mathtt{icc}} t'$. In particular, the two side-conditions we mentioned are direct consequences of the saturated term structure.

The fact that the $x = n$ bindings are never redundant is a direct consequence of the "new"-ness condition on saturated neutrals: n cannot be added in Ξ in a future phase as it would not be new anymore.

The fact that the context is consistent at the source of inequality is a consequence of Corollary 2 and Lemma 5 (Saturated consistency). □

Theorem 11 (Inequivalence in the model). We prove this by simultaneous induction on the inequivalence evidence for $t_0 \not\approx_{\mathtt{icc}} t'_0$, $\Gamma_{\mathsf{na}} \mid \Sigma_{\mathsf{p}} \vdash_{\mathsf{inv}} D_0\,[\Gamma'_{\mathsf{na}} \mid \Xi \vdash_{\mathtt{ne}} p_0 \neq {p'}_0 : P] : N \mid Q_{\mathsf{a}}$ and the sub-terms of the two terms t_1, t'_1, building inequivalence evidence for $t_1 \not\approx_{\mathtt{icc}} t'_1$. There are three interesting cases: when the current term-former introduces a variable $(x : X^+)$ in context (or at the root for the variables of $\Gamma_{\mathsf{na}}, \Sigma_{\mathsf{p}}$), the saturation step (where non-redundancy is checked), and when the source of inequality $\Gamma'_{\mathsf{na}} \mid \Xi \vdash_{\mathtt{ne}} p \neq p' : P$ is finally encountered.

When we enter a binding $x : X^+$, the corresponding subterms of t_1, t'_1 start with a large pattern-matching on the possible values of $x : \mathcal{N}(X^+)$. We decide to take the branch corresponding to the case $\mathtt{in}_{\mathtt{Fin}}(x)$, maintaining the inductive invariant that the sub-terms of t_1, t'_1 are equal to the sub-terms of t_0, t'_0 under the family of substitutions $([\mathtt{in}_{\mathtt{Fin}}(x)/x])^x$, for all variables x of atomic type introduced so far.

In a saturation step, we must check that the newly added bindings are not already present in the constrained environment $\Gamma_{\mathsf{na}} \mid \Xi$. The neutrals introduced in $t_1 \not\approx_{\mathtt{icc}} t'_1$ are of the form $n[\mathtt{in}_{\mathtt{Fin}}(x)/x]^x$, and this substitution preserves α-inequivalence, as it replaces distinct variables by distinct values.

When we reach the source of inequality $\Gamma'_{\mathsf{na}} \mid \Xi \vdash_{\mathtt{ne}} p \neq p' : P$, the derivation of spine inequivalence $\Gamma'_{\mathsf{na}} \vdash_{\mathtt{ne}} p \neq p' \Uparrow P$ either ends up on a $\Gamma'_{\mathsf{na}} \vdash_{\mathtt{ne}} \sigma_i\ p \neq \sigma_j\ p' \Uparrow P$ with $i \neq j$, in which case the corresponding neutral subterms of t_1, t'_1 are also of this form, or $\Gamma'_{\mathsf{na}} \vdash_{\mathtt{ne}} x \neq y \Uparrow X^+$. In this case the subterms of t_1, t'_1 are $\mathtt{in}_{\mathtt{Fin}}(x), \mathtt{in}_{\mathtt{Fin}}(y)$, by our inductive invariant: they are also distinct positive neutrals, and form a source of inequality.

At the source of inequality, we also have to prove that $\mathcal{N}_{t_0,t'_0}(\Gamma') \nvdash 0$ from the hypothesis $\Gamma' \nvdash 0$. Note that this could be wrong if we applied some other model $\mathcal{M}$; for example, the context $(x : X^+, y : Y^+ \to 0)$ is consistent, but applying the model $\{X^+ \mapsto 1, Y^+ \mapsto 1\}$ results in an inconsistent context.

Fortunately, the definition of the neutral model Definition 12 (Neutral model) is such that the positive atoms that become inhabited are exactly those that have bound variables in the context at the source of inequality $\Gamma'_{\mathsf{na}} \mid \Xi \vdash_{\mathtt{ne}} p \neq p' : P$. And, by saturated consistency (Corollary 2 (Saturation)), we know that those positive atoms are exactly those that are *provable* in Γ'_{na}.

We can thus can prove $\mathcal{N}(\Gamma'_{\mathsf{na}}) \nvdash 0$ by contradiction as follows. Suppose we have a proof of $\mathcal{N}(\Gamma'_{\mathsf{na}}) \vdash 0$. Any subproof of the form $\mathcal{N}(\Gamma'_{\mathsf{na}}) \vdash \mathcal{N}(X^+)$, for one of the X^+ inhabited in $\mathcal{N}$, can be replaced by a subproof of the form $\mathcal{N}(\Gamma'_{\mathsf{na}} \vdash X^+)$, as we know that all such X^+ are derivable in Γ'_{na}. The transformed proof does not rely on the definition of the X^+, so it is also a valid proof of $\Gamma'_{\mathsf{na}} \vdash 0$, which is impossible. □

Corollary 3 (Fun-elimination preserves semantic equivalence). This is immediate from Lemma 8 and the fact that the function-elimination transformation is a bijection on semantics value – Figure 3 (Fun-less data types). □

Theorem 12 (Canonicity). We remark that $p[\rho] \not\approx_{\mathtt{sem}} p'[\rho]$ is a trivial property if P is a closed type (no atoms X^+): by inspecting the $\Gamma \vdash_{\mathtt{ne}} p \neq p' \Uparrow P$ judgment (or the strong positive neutral structure), we know that p and p' differ on a constructor, and this property is preserved by any substitution.

From $\Gamma \nvdash \emptyset$ we know that for any $A \in \Gamma$, A is inhabited by some closed values – Corollary 1 (Inhabited or inconsistent). We can always pick an arbitrary closed value of type A.

We build ρ by peeling off the variables of the environment Γ, from the most recently introduced to the first introduced. At any point during our induction, we have processed a fragment Δ' of the context, and Δ remains unprocessed, with $\Gamma = (\Delta, \Delta')$, and we have a substitution ρ that binds the variables of Δ' to closed terms, such that $\Xi[\rho]$ contains no inconsistent equation. Initially we have $\Delta \stackrel{\mathsf{def}}{=} \Gamma$, $\Delta' \stackrel{\mathsf{def}}{=} \emptyset$, $\rho \stackrel{\mathsf{def}}{=} \emptyset$.

When Γ is of the form $\Gamma', x : A_1 + A_2$, we look for equations of the form $\sigma_i\ x_i = x$ in Ξ; by validity of Ξ (Definition 13), there is at most one. If there is none, we choose an arbitrary value for x in ρ; if there is one, we choose $x_i[\rho]$; this is a closed term as x_i must be defined after x in the context.

When Γ is of the form $\Gamma', x : N$, we look for equations the family of equations $(y_i = n_i\,[x])^{i \in I}$ – remember that we write $n\,[x]$ when the neutral n has head x. Let us write $\Xi\,[x]$ for this set of equations about x.

We define the value for x in ρ as $\mathtt{Val}(x : N \mid (y_i = n_i)^{i \in I})$ where $\mathtt{Val}(n : N \mid \Xi)$, defined below, is such that $\Xi[\mathtt{Val}(n : N \mid \Xi)/n]$ always holds, if all equations in Ξ are of the form $y = m\,[n]$.

$$\mathtt{Val}(n : 1 \mid \Xi) \stackrel{\mathsf{def}}{=} ()$$

$$\mathtt{Val}(n : N_1 \times N_2 \mid \Xi) \stackrel{\mathsf{def}}{=} \begin{pmatrix} \mathtt{Val}(\pi_1\ n : N_1 \mid \{(x = m\,[\pi_1\ n]) \in \Xi\}) \\ , \\ \mathtt{Val}(\pi_2\ n : N_2 \mid \{(x = m\,[\pi_2\ n]) \in \Xi\}) \end{pmatrix}$$

$$\mathtt{Val}(n : \langle P \rangle^- \mid \{y = n\}) \stackrel{\mathsf{def}}{=} y \qquad \mathtt{Val}(n : \langle P \rangle^- \mid \emptyset) \stackrel{\mathsf{def}}{=} \text{arbitrary}$$

From the validity of Ξ, we know that $\Xi\,[x]$ contains at most one equation of the form $(y : P) = n$ for each possible neutral $\vdash n \Downarrow \langle P \rangle^-$, which justifies the only two cases in the definition of $\mathtt{Val}(n : \langle P \rangle^- \mid \Xi)$ above. □

Corollary 4 (Canonicity). We use a stronger induction hypothesis on the inequivalence judgment: if

$$\Gamma \mid \Xi \vdash_{\mathsf{inv}} D\,[\Gamma' \mid \Xi' \vdash_{\mathtt{ne}} p \neq p' : P] : N \mid P_{\mathsf{a}}$$

then there is a substitution $\rho : (\Gamma \mid \Xi)$ and a context $C\,[\Box]$ such as

$$\emptyset \vdash C\,[t[\rho]] \not\approx_{\beta} C\,[t'[\rho]] : 1 + 1$$

where $t \stackrel{\mathsf{def}}{=} D\,[p]$ and $t' \stackrel{\mathsf{def}}{=} D\,[p']$.

In the base case $D \stackrel{\mathsf{def}}{=} \Box$, we get a substitution $\rho : (\Gamma' \mid \Xi')$ from Theorem 12 (Canonicity); the theorem also gives us $p[\rho] \not\approx_{\mathtt{sem}} p'[\rho]$, so there is a distinguishing context C as required as contextual and semantic equivalence coincide.

In the sum-elimination case

$$\frac{\Gamma, x : P_1 + P_2, x_i : P_i \mid \Xi, \sigma_i\ x_i = x \vdash_{\mathsf{inv}} D\left[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P'\right] : N \mid Q_{\mathsf{a}}}{\Gamma, x : P_1 + P_2 \mid \Xi \vdash_{\mathsf{inv}} \left(\begin{array}{l}\mathtt{match}\ x\ \mathtt{with} \\ \mid\ \sigma_i\ x_i \to \square \\ \mid\ \sigma_j\ x_{j\neq i} \to t\end{array}\right)\left[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P'\right] : N \mid Q_{\mathsf{a}}}$$

the substitution ρ verifies, by inductive hypothesis, that $\sigma_i\ x_i[\rho] = x[\rho]$, so in particular the sum-elimination term, once applied ρ, will reduce to $(D\,[p])[\rho]$ and $(D\,[p'])[\rho]$ respectively, as desired: this means that the distinguishing context D for the premise is also a distinguishing context for the sum-elimination terms.

In the (left) pair case

$$\frac{\Gamma \mid \Xi \vdash_{\mathsf{inv}} D\left[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P\right] : N_1 \mid \emptyset}{\Gamma \mid \Xi \vdash_{\mathsf{inv}} (D, u)\left[\Gamma' \mid \Xi' \vdash_{\mathsf{ne}} p \neq p' : P\right] : N_1 \times N_2 \mid \emptyset}$$

We can obviously keep the same ρ, and build the distinguishing context $C\,[\pi_1\ \square]$, where C is the distinguishing context for the inductive premise.

From this stronger inductive hypothesis (ρ, C), we can build a distinguishing context for $D\,[p] \not\approx_{\mathtt{ctx}} D\,[p']$ as

$$C\,[(\lambda(x_1 \dots x_n \in \Gamma).\,\square)\ (x_1[\rho]) \dots (x_n[\rho])]$$

□

Theorem 13 (Saturated terms are canonical). Let $t_0 \stackrel{\mathsf{def}}{=} t$ and $t'_0 \stackrel{\mathsf{def}}{=} t'$.

By Lemma 6 (Saturated inequivalence) we have $\Sigma_{\mathsf{p}} \mid \emptyset \vdash_{\mathsf{inv}} D_0\left[\Gamma'_{\mathsf{na}} \mid \Xi \vdash_{\mathsf{ne}} p_0 \neq {p'}_0 : P\right] : N \mid Q_{\mathsf{a}}$ with $t_0 \approx_{\mathtt{icc}} D_0\,[p_0]$ and $t'_0 \approx_{\mathtt{icc}} D_0\,[{p'}_0]$, that is, $(t_0 \not\approx_{\mathtt{icc}} t'_0) \rightsquigarrow D_0\,[p_0 \neq {p'}_0]$.

We consider the neutral model $\mathcal{N}_{t_0,t'_0}$ defined in Definition 12 (Neutral model). Let $t_1 \stackrel{\mathsf{def}}{=} \mathcal{N}(t_0)$ and $t'_1 \stackrel{\mathsf{def}}{=} \mathcal{N}(t'_0)$.

By Theorem 11 (Inequivalence in the model) we have $(t_0 \not\approx_{\mathtt{icc}} t'_0) \rightsquigarrow D_1\,[p_1 \neq {p'}_1]$, that is a inequivalence of terms at the atomless judgments $\emptyset; \mathcal{N}(\Sigma_{\mathsf{p}}) \vdash_{\mathsf{inv}} t_0 \not\approx_{\mathtt{icc}} t'_0 : \mathcal{N}(N) \mid \mathcal{N}(Q_{\mathsf{a}})$.

Using the focused function-elimination transformation of Figure 9 (Fun-less focused forms), let us define $t_2 \stackrel{\mathsf{def}}{=} \lfloor t_1 \rfloor$ and $t'_2 \stackrel{\mathsf{def}}{=} \lfloor t'_1 \rfloor$. By Lemma 7 (Function elimination preserves inequivalence) we get $(t_2 \not\approx_{\mathtt{icc}} t'_2) \rightsquigarrow D_2\,[p_2 \neq {p'}_2]$, that is a saturated inequivalence at closed types without functions.

From Corollary 4 (Canonicity), we finally get $D_2\,[p_2] \not\approx_{\mathtt{ctx}} D_2\,[{p'}_2]$, that is $t_2 \not\approx_{\mathtt{ctx}} t'_2$, a contextual inequivalence at closed types without functions. As contextual and semantic equivalence coincide at closed types, this is equivalent to $[\![t_2]\!] \neq [\![t'_2]\!]$, that is, $[\![\lfloor t_1 \rfloor]\!] \neq [\![\lfloor t'_1 \rfloor]\!]$.

By Corollary 3 (Fun-elimination preserves semantic equivalence), we thus have that $[\![t_1]\!] \neq [\![t'_1]\!]$ at closed types, that is, $[\![t_0]\!]_{\mathcal{N}} \neq [\![t'_0]\!]_{\mathcal{N}}$, that is, $t_0 \not\approx_{\mathtt{sem}(\mathcal{N})} t'_0$, that is, $t_0 \not\approx_{\mathtt{sem}} t_0$, or equivalently $t_0 \not\approx_{\mathtt{ctx}} t_0$. □

Corollary 5 (Contextual equivalence implies equality of saturated forms). By contrapositive, we show that $u \not\approx_{\mathtt{icc}} u'$ implies $t \not\approx_{\mathtt{ctx}} t'$. If we had $u \not\approx_{\mathtt{icc}} u'$ we would have $\lfloor u \rfloor_{\mathtt{foc}} \not\approx_{\mathtt{ctx}} \lfloor u' \rfloor_{\mathtt{foc}}$ by Theorem 13 (Saturated terms are canonical). As $\beta\eta$-equivalence implies contextual equivalence, we have $t \approx_{\mathtt{ctx}} \lfloor u \rfloor_{\mathtt{foc}}$ and $t' \approx_{\mathtt{ctx}} \lfloor u' \rfloor_{\mathtt{foc}}$, so $t \not\approx_{\mathtt{ctx}} t'$ by transitivity. □

Corollary 6 (). We know from Section 2 (Equivalences in the full λ-calculus) that $\beta\eta$-equivalence implies contextual equivalence. Conversely, we suppose $t \approx_{\mathtt{ctx}} t'$ and prove $t \approx_{\beta\eta} t'$.

By Theorem 7 (Completeness of focusing), there exists focused forms u, u' that erase to t, t' (respectively) modulo $\beta\eta$; we can even pick them in the subset with positive atoms only.

By Theorem 8 (Completeness of saturation), there exists saturated forms r, r' that have the same erasures as u, u' (respectively) modulo $\beta\eta$, that is they erase to t, t' modulo $\beta\eta$.

By Corollary 5 (Contextual equivalence implies equality of saturated forms), we have $r \approx_{\mathtt{icc}} r'$. As the invertible commuting conversions are sound with respect with $\beta\eta$-equality, we have $\lfloor r \rfloor_{\mathtt{foc}} \approx_{\beta\eta} \lfloor r' \rfloor_{\mathtt{foc}}$, that is, $t \approx_{\beta\eta} t'$. □

Corollary 7 (). To test if two terms are equivalent, we invoke the focusing completeness result then the saturated completeness result, and check if the normal form coincide modulo $(\approx_{\mathtt{icc}})$. Those two transformations are computable and $(\approx_{\mathtt{icc}})$ is decidable, so equivalence is decidable. □

Corollary 8 (). Our notion of semantic equivalence coincides with $\beta\eta$ and contextual equivalence, and only distinguishes terms using sets isomorphic to closed types, which are all finitely inhabited. □