Böhm Trees as Higher-Order Recursive Schemes

P. Clairambault and A. S. Murawski

[1] Computer Laboratory, University of Cambridge
[2] DIMAP and Department of Computer Science, University of Warwick

Abstract. Higher-order recursive schemes (HORS) are schematic representations of functional programs. They generate possibly infinite ranked labelled trees and, in that respect, are known to be equivalent to a restricted fragment of the λY-calculus, consisting of ground-type terms whose free variables have types of the form $o \to \cdots \to o$ (with o being a special case).
In this paper, we show that *any* λY-term (with no restrictions on term type or the types of free variables) can actually be represented by a HORS. More precisely, for any λY-term M, there exists a HORS generating a tree that faithfully represents M's Böhm tree. In particular, the HORS captures higher-order binding information contained in the Böhm tree. An analogous result holds for finitary PCF.
As a consequence, we can reduce a variety of problems related to the λY-calculus or finitary PCF to problems concerning higher-order recursive schemes. For instance, Böhm tree equivalence can be reduced to the equivalence problem for HORS. Our results also enable MSO model-checking of Böhm trees.

1 Introduction

Higher-order recursive schemes (HORS) are a class of programming schemes introduced to account for recursive procedures with higher-order parameters [6]. They can be viewed as grammars describing a potentially infinite tree. As tree generating devices, HORS can be identified with a limited fragment of the λY-calculus [17], consisting of ground-type terms such that the types of their free variables must have order at most 1. The free variables play the role of tree constructors. HORS have recently been investigated intensively in the verification community. Notably, Ong [14] showed that monadic second-order logic (MSO) is decidable over trees generated by HORS and Kobayashi [11] took advantage of the result to propose a novel approach to the verification of higher-order functional programs.

In this paper we reveal a new connection between HORS and the *full* λY-calculus. As mentioned above, HORS are naturally viewed as a fairly small fragment of the calculus. We show that, in fact, they are far more expressive and can be used to represent *arbitrary* λY-terms. Specifically, we prove that, for any λY-term M, there exists a higher-order recursive scheme generating a faithful representation of M's Böhm tree. The main difficulty to overcome is the fact that

λY-calculus terms can contain complicated variable-binding patterns, whereas HORS do not provide native support for binding. We also demonstrate how to obtain an analogous representation theorem for the finitary (finite datatypes) variant PCF_f of PCF [15].

Our results make it possible to recast a variety of problems related to the λY-calculus or finitary PCF as problems concerning higher-order recursive schemes. For example, we obtain a reduction of Böhm tree equivalence in the λY-calculus or PCF_f to the (tree) equivalence problem for HORS. Unfortunately, the latter is currently not known to be decidable, and is closely related to the equivalence problem for *deterministic* collapsible pushdown automata [9]. The Böhm tree equivalence problem for PCF_f also has semantic significance, because Böhm trees characterize contextual equivalence in PCF with respect to contexts featuring state and control effects.

Other consequences, in the form of decidability results, can be derived by applying Ong's decidability result to representations of Böhm trees obtained through our theorems. In this way, one can show that numerous problems for λY or PCF_f are decidable. Examples include normalizability, finiteness, solvability or having a Böhm tree prefixed by a given term. Some of the results are already known, while others appear new.

Thus far, higher-order verification based on HORS focussed on model-checking trees extracted from programs. Our contribution opens the perspective of applying model-checking to terms with binding and arbitrary free variables, such as higher-order components of closed programs.

2 λY-calculus

We work with simple types built from a single atom o using the arrow type constructor. They are defined by the grammar given below.

$$\theta ::= o \mid \theta \to \theta$$

The order of a type is defined as follows.

$$ord(o) = 0 \qquad ord(\theta_1 \to \theta_2) = \max(ord(\theta_1) + 1, ord(\theta_2))$$

Terms are considered up to α-equivalence, and equipped with β-reduction and η-expansion (respectively written $\to_\beta$ and $\to_\eta$). We write $\simeq_{\beta\eta}$ for the symmetric reflexive and transitive closure of $\to_\beta$ and $\to_\eta$. We shall consider several extensions of the simply-typed λ-calculus over the types introduced above.

- The $\lambda\bot$-calculus additionally contains a constant $\bot_o : o$. More generally, we shall write $\bot_\theta$ for terms specified as follows.

$$\bot_\theta = \begin{cases} \bot_o & \theta \equiv o \\ \lambda x^{\theta_1}.\bot_{\theta_2} & \theta \equiv \theta_1 \to \theta_2 \end{cases}$$

We will also consider the extension of the $\lambda\bot$-calculus to infinite terms, which we refer to as the $\lambda\bot^\infty$-calculus. For $\lambda\bot$-terms, let us define a partial order $\sqsubseteq$ (relating only terms with equal types) by

$$\frac{}{M \sqsubseteq M} \qquad \frac{}{\bot_o \sqsubseteq M} \qquad \frac{M_1 \sqsubseteq M_2}{\lambda x.M_1 \sqsubseteq \lambda x.M_2} \qquad \frac{M_1 \sqsubseteq M_1' \qquad M_2 \sqsubseteq M_2'}{M_1\ M_2 \sqsubseteq M_2'\ M_2'}.$$

It can be extended to $\lambda\bot^\infty$-terms to yield an ω-cpo.

– The λY-calculus contains a family of fixed-point operators $Y_\theta : (\theta \to \theta) \to \theta$, where θ ranges over arbitrary types, equipped with the reduction rule $Y_\theta\ M \to_Y M\ (Y_\theta\ M)$.

Definition 1. *Given a λY-term M and $n \in \mathbb{N}$, the nth approximant of M, written $M \upharpoonright n$, is a $\lambda\bot$-term defined by*

$$x \upharpoonright n = x \qquad \lambda x.M \upharpoonright n = \lambda x.(M \upharpoonright n)$$
$$M\ N \upharpoonright n = (M \upharpoonright n)\ (N \upharpoonright n) \qquad Y_\theta\ \upharpoonright n = \lambda f^{\theta\to\theta}.f^n(\bot_\theta).$$

Definition 2. *The Böhm tree of a $\lambda\bot$-term $\Gamma \vdash M : \theta$, written $\mathsf{BT}(M)$, is its β-normal η-long form. For a λY-term $\Gamma \vdash M$, the Böhm tree, also denoted by $\mathsf{BT}(M)$, is defined to be $\bigsqcup_n \mathsf{BT}(M \upharpoonright n)$.*

Definition 3. *λY-terms satisfying $\Gamma \vdash M : o$ are called* ground. *Given $n \geq 0$, we shall say that a ground term $\Gamma \vdash M : o$ is of* level *n if, for all $(x : \theta) \in \Gamma$, we have $ord(\theta) < n$.*

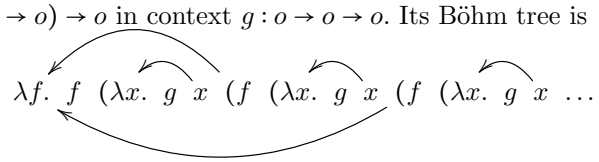
Note that the free variables in a ground term of level 2 can have types of order 0 or 1, i.e. they are of the form $o \to \cdots \to o$, where o has at least one occurrence. Thus, Böhm trees of such terms are (possibly infinite) trees.

The restrictions on types of free variables in ground terms of level 2 are analogous to the restriction concerning types of terminal symbols in higher-order recursion schemes [6]. In fact, it can be shown that the Böhm trees of level-2 ground terms coincide with trees generated by higher-order recursive schemes [16].

In this paper we show that level-2 ground terms are actually expressive enough to represent Böhm trees of *arbitrary* λY-terms. Equivalently, Böhm trees of λY-terms can be captured using higher-order recursive schemes. The remainder of the section is devoted to a formal statement of the result.

Representing binders. Unlike in the case of level-2 ground terms, Böhm trees of arbitrary λY-terms involve *binders*, as illustrated by the following example.

Example 4. Consider the term $M = \lambda f^{(o\to o)\to o}.Y_o(\lambda y^o.f(\lambda x^o.gxy))$, which has type $((o \to o) \to o) \to o$ in context $g : o \to o \to o$. Its Böhm tree is



where the arrows indicate *binding information*: for each variable occurrence, the arrow indicates the location of the associated binder.

The pointers drawn above are the main reason why Böhm trees cannot be directly represented as trees generated by recursion schemes, which do not involve binders. Consequently, we need to consider binder-free representations. For that we rely on *De Bruijn levels* [7]: each bound variable x is given an index i, where i is the sum of a starting index and the number of lambdas enclosing the binding location (De Bruijn levels should not be confused with De Bruijn *indices*, which associate numbers to variable uses and not their points of introduction, and can associate different numbers to different occurrences of the same variable).

Example 5. Using De Bruijn levels, the Böhm tree of M from Example 4 can be represented by $\lambda x_1.x_1\ (\lambda x_2.x_0\ x_2\ (x_1\ (\lambda x_3.x_0\ x_3\ (x_1\ (\lambda x_4.x_0\ x_4 \ldots$, with $g : o \to o \to o$ being given the index 0. Each variable introduction and use is now annotated with a natural number. This representation is unique, in the sense that, for a fixed choice of indices for free variables and a choice of a starting index, two α-equivalent terms have the same representation.

With De Bruijn levels, $\lambda\bot$-terms can be represented as level-2 ground terms, typable in a special context defined below.

Definition 6. *Let Γ_{rep} be the following context.*

$$\{\, z : o,\ succ : o \to o,\ var : o \to o,\ app : o \to o \to o,\ lam : o \to o \to o \,\}$$

Given $n \in \mathbb{N}$, we write $\overline{n}$ for the term $succ^n(z)$.

In particular, the terms $\overline{n}$ will be used to represent binding information as De Bruijn levels.

Definition 7. *Let $\Gamma \vdash M : \theta$ be a $\lambda\bot$-term and let $\nu : \Gamma \to \mathbb{N}$ be an injection. We define another $\lambda\bot$-term $\Gamma_{\mathsf{rep}} \vdash \mathsf{rep}_\nu(M, n) : o$ by*

$$\mathsf{rep}_\nu(\bot, n) = \bot \qquad \mathsf{rep}_\nu(\lambda x.M, n) = lam\ \overline{n+1}\ \mathsf{rep}_{\nu \oplus \{x \mapsto n+1\}}(M, n+1)$$
$$\mathsf{rep}_\nu(x, n) = var\ \overline{\nu(x)} \qquad \mathsf{rep}_\nu(MN, n) = app\ \mathsf{rep}_\nu(M, n)\ \mathsf{rep}_\nu(N, n).$$

Note that, as long as $n > \max_{(x:\theta)\in\Gamma} \nu(x)$, $\mathsf{rep}_\nu(M, n)$ faithfully represents the syntactic structure of M. Given $\Gamma \vdash M : \theta$, where $\Gamma = \{x_1 : \theta_1, \cdots, x_k : \theta_k\}$, let $\nu : \Gamma \to \mathbb{N}$ be defined by $\nu(x_i) = i - 1$. We define $\mathsf{rep}(M)$ to be $\mathsf{rep}_\nu(M, k)$.

For any finite $\lambda\bot$-term M, $\mathsf{rep}(M)$ gives a binder-free representation of M as a ground $\lambda\bot$-term in context Γ_{rep}. By continuity, the Böhm tree of any λY-term can be represented as a ground infinite $\lambda\bot^\infty$-term in context Γ_{rep}. This invites the question whether this infinite $\lambda\bot^\infty$-term can always be obtained as the Böhm tree of a level-2 ground λY-term.

Example 8. In our running example, the binder-free representation of M's Böhm tree can be captured as an open (infinite) term in context Γ_{rep}, namely, $M_\infty = lam\ \overline{1}\ T(2)$, where

$$T(n) = app\ (var\ \overline{1})\ (lam\ \overline{n}\ (app\ (app\ (var\ \overline{0})\ (var\ \overline{n}))\ T(n+1))).$$

More precisely, we have $\bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n)) = M_\infty$. In this case, it is easy to extract a λY-term $\Gamma_{\mathsf{rep}} \vdash M_{rep} : o$ such that $\mathsf{BT}(M_{rep}) = M_\infty$ from the definition of M_∞. This can be done by expressing the coinductive definition of M_∞ inside the λY-calculus as $M_{rep} = lam\ \overline{1}\ (Y_{o\to o}\ M_{step}\ \overline{2})$, where

$$M_{step} = \lambda y^{o\to o}.\lambda v^{o}.app\ (var\ \overline{1})\ (lam\ v\ (app\ (app\ (var\ \overline{0})\ (var\ v))\ (y\ (succ\ v)))).$$

Consequently, the Böhm tree of M_{rep} is a binder-free representation of the Böhm tree of M.

In our paper, we solve the problem of generating representations of Böhm trees of arbitrary λY-terms using level-2 ground terms (equivalently, using higher-order recursive schemes). We prove the following theorem.

Theorem 9. *Let $\Gamma \vdash M : \theta$ by a λY-term. There exists a level-2 λY-term $\Gamma_{\mathsf{rep}} \vdash M_{rep} : o$ such that $\mathsf{BT}(M_{rep}) = \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$.*

Next we describe the construction underlying our proof of Theorem 9. In this section we only provide the necessary definitions, delegating the proof and a methodological discussion to Section 3. The construction is inspired by *normalization by evaluation* [8] and, in particular, the fact that the technique can be internalized within the λY-calculus.

Consider $\Gamma \vdash M : \theta$ with $\Gamma = \{x_1 : \theta_1, \cdots, x_k : \theta_k\}$. First we apply a simple transformation on type annotations inside M and substitute $o \to o$ for o: let M^* be defined as $M[o \to o/o]$. The substitution is meant to be applied to types θ occurring in λ-abstractions $(\lambda x^{\theta}.M)$ and fixed-point combinators (Y_θ). Observe that $x_1 : \theta_1^*, \cdots, x_k : \theta_k^* \vdash M^* : \theta^*$, where

$$o^* = o \to o, \qquad (\theta' \to \theta'')^* = (\theta')^* \to (\theta'')^*.$$

Next we define by mutual recursion two families of λ-terms $\{\downarrow_\theta\}$, $\{\uparrow_\theta\}$, subject to the following conditions: $\Gamma_{\mathsf{rep}} \vdash \downarrow_\theta : \theta^* \to o \to o$ and $\Gamma_{\mathsf{rep}} \vdash \uparrow_\theta : (o \to o) \to \theta^*$. We write $\lambda.M$ for $\lambda z^o.M$, where z does *not* occur in M.

$$\begin{aligned}
\downarrow_o &= \lambda x^{o\to o}.x \\
\downarrow_{\theta'\to\theta''} &= \lambda x^{(\theta'\to\theta'')^*}.\ \lambda v^o.\ lam\ v\ (\downarrow_{\theta''}\ (\ x\ (\uparrow_{\theta'}\ \lambda.(var\ v))\ )\ (succ\, v)\ ) \\
\uparrow_o &= \lambda x^{o\to o}.x \\
\uparrow_{\theta'\to\theta''} &= \lambda e^{o\to o}.\ \lambda a^{(\theta')^*}.\ \uparrow_{\theta''}\ (\ \lambda v^o.\ app\ (e\ v)\ (\downarrow_{\theta'}\ a\ v)\ )
\end{aligned}$$

Now, with all the definitions in place, one can take M_{rep} to be

$$\downarrow_\theta\ \ M^*[\ \uparrow_{\theta_1} \lambda.(var\ \overline{0})/x_1\ ,\ \cdots\ ,\ \uparrow_{\theta_k} \lambda.(var\ \overline{k-1})/x_k\ ]\ \ \overline{k}.$$

In the next section we shall prove that M_{rep} does represent M's Böhm tree in the sense of Theorem 9. Before that, we illustrate the outcome of the construction on our running example.

Example 10. Let us revisit the term M from Example 4. We have

$$M^* = \lambda f^{((o\to o)\to(o\to o))\to(o\to o)}.Y_{o\to o}(\lambda y^{o\to o}.f(\lambda x^{o\to o}.gxy))$$

and, thus, $M_{rep} = \downarrow_{((o\to o)\to o)\to o} \quad M^*[\uparrow_{o\to o\to o} \lambda.(var\,\overline{0})/g] \quad \overline{1}$. By unfolding the definitions of $\downarrow_\theta, \uparrow_\theta$ and applying β-reduction the term can be rewritten. In fact, taking advantage of the β-equivalences

$$\begin{aligned}\uparrow_{o\to o\to o} &= \lambda e_1^{o\to o}.\lambda a_1^{o\to o}.\lambda a_2^{o\to o}.\lambda v_2^o.\ app\ (app\ (e_1\ v_2)\ (a_1\ v_2))\ (a_2\ v_2)\\ \downarrow_{((o\to o)\to o)\to o} &= \lambda x^{(((o\to o)\to o)\to o)^*}.\lambda v_1^o.\ lam\ v_1\ (x\ M_{aux}\ (succ\ v_1)),\end{aligned}$$

where $M_{aux} = \lambda a^{(o\to o)^*}.\lambda v_2^o.\ app\ (var\ v_1)\ (lam\ v_2\ (a\ (\lambda.var\ v_2)\ (succ\ v_2)))$, one can show that M_{rep} is β-equivalent to

$$lam\ \overline{1}\ (Y\ (\lambda y.\lambda v.app\ (var\ \overline{1})\ (lam\ v\ (app\ (var\ \overline{0})\ (var\ v))\ (y\ (succ\ v))))\ \overline{2}),$$

the manually constructed term from Example 8.

3 Internalized normalization by evaluation

In this section, we present the mathematical development that led to the term transformation of the previous section. As already mentioned, the main idea comes from *normalization by evaluation* (NBE). NBE is a general method used to construct normal forms for terms through their denotational semantics; the reader is directed to [8] for an introduction. The basic idea is to interpret terms in a suitable semantic universe. By soundness, the interpretation will be invariant under syntactic reduction. Moreover, if one uses a class of domains expressive enough to encode representations of normal forms, then one can attempt to extract a representation of the normal form of a term from the semantics. Notably, the extraction can be performed by evaluating the interpretation map on well-chosen elements of the domain, which yields an entirely semantic normalization procedure. NBE has been described for a variety of languages, including the simply-typed λ-calculus [5] as well as richer languages with coproducts [4], or type theory [1]. Our NBE presentation follows the standard lines of [8]. The main novelty is the internalization of the process within the λY-calculus itself, which ultimately allows us to present NBE as a term transformation.

Semantic NBE for the λY-calculus We first describe a domain semantics for the λY-calculus that will be exploited to obtain an NBE procedure. For the basic vocabulary of domain theory, the reader is referred to [18]. Let us recall that a ω-cpo is a partial order where each ω-chain (of the form $x_1 \le x_2 \le \dots$) has a supremum. We will interpret types as *pointed* ω-cpos, i.e., ω-cpos with a bottom element written $\bot$. A function $f : A \to B$ is *continuous* if it preserves supremum of ω-chains. The set of continuous functions from A to B, ordered pointwise, forms itself a pointed ω-cpo denoted by $A \to B$.

Our NBE procedure will generate representations of Böhm trees of λY-terms as elements of the pointed ω-cpo of $\lambda\perp^{\infty}$-terms of ground type in context Γ_{rep}, henceforth referred to as E. Representations based on De Bruijn levels lack compositionality: the indices present in a subterm depend on the number of variables abstracted in the context where it appears. In order to overcome the difficulty we follow the approach of [8]: instead of constructing directly De Bruijn terms in E we will construct *term families*, intuitively functions $\mathbb{N} \to E$, which - unlike De Bruijn terms - can be manipulated compositionally.

NBE for λY. We now describe our semantic NBE for λY. As explained above, we will construct De Bruijn terms through the "term families" approach. In our case, E will also be used to represent natural numbers, so term families will be interpreted as continuous functions on the pointed ω-cpo $\widehat{E} = E \to E$, where the first E plays the role of $\mathbb{N}$. For the sake of clarity, we will sometimes write N instead of E to highlight subterms representing natural numbers. This should not create any confusion.

Our NBE relies on the following (standard) interpretation of the λY-calculus. Note we write $\boldsymbol{\lambda}$, rather than λ, to stress that the semantic operation of function abstraction (on continuous functions between pointed ω-cpos) is meant.

$$\begin{array}{ccccc} [\![o]\!] = \widehat{E} & & [\![\lambda x^{\theta}.M]\!]_{\rho} = \boldsymbol{\lambda} a^{[\![\theta]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} & & [\![M\ N]\!]_{\rho} = [\![M]\!]_{\rho}([\![N]\!]_{\rho}) \\ [\![\theta_1 \to \theta_2]\!] = [\![\theta_1]\!] \to [\![\theta_2]\!] & & [\![Y_{\theta}]\!]_{\rho} = \bigsqcup_n \boldsymbol{\lambda} f^{[\![\theta]\!] \to [\![\theta]\!]}.f^n(\perp) & & [\![x]\!]_{\rho} = \rho(x) \end{array}$$

We claimed above that unlike raw De Bruijn terms, term families can be constructed compositionally. This is done with the help of the following continuous functions, which generalize term constructors of E to $\widehat{E}$.

$$\begin{array}{l} \widehat{var} = \boldsymbol{\lambda} v^N.\boldsymbol{\lambda} n^N.var\ v : N \to \widehat{E} \\ \widehat{app} = \boldsymbol{\lambda} e_1^{\widehat{E}}.\boldsymbol{\lambda} e_2^{\widehat{E}}.\boldsymbol{\lambda} n^N.app\ (e_1\ n)\ (e_2\ n) : \widehat{E} \to \widehat{E} \to \widehat{E} \\ \widehat{lam} = \boldsymbol{\lambda} f^{N \to \widehat{E}}.\boldsymbol{\lambda} n^N.lam\ n\ (f\ n\ (succ\ n)) : (N \to \widehat{E}) \to \widehat{E} \end{array}$$

Using these extended constructors, one can define families of continuous functions, traditionally called *reification* ($\Downarrow_{\theta} : [\![\theta]\!] \to \widehat{E}$) and *reflection* ($\Uparrow_{\theta} : \widehat{E} \to [\![\theta]\!]$).

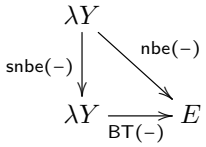
$$\begin{array}{ll} \Downarrow_o x = x & \Downarrow_{\theta_1 \to \theta_2} x = \widehat{lam}\ (\boldsymbol{\lambda} n^N.\Downarrow_{\theta_2} (x\ (\Uparrow_{\theta_1} \widehat{var}\ n))) \\ \Uparrow_o e = e & \Uparrow_{\theta_1 \to \theta_2} e = \boldsymbol{\lambda} x^{[\![\theta_1]\!]}.\Uparrow_{\theta_2} \widehat{app}\ e\ (\Downarrow_{\theta_2} x) \end{array}$$

For closed terms $\vdash M : \theta$, the normalization-by-evaluation function can now be defined by setting $\mathsf{nbe}(M) = \Downarrow_{\theta} [\![M]\!]\ \overline{0}$. In order to apply the same method to open terms $\Gamma \vdash M : \theta$, where $\Gamma = \{x_1 : \theta_1, \ldots, x_k : \theta_k\}$, we need to build a canonical inhabitant of $[\![\Gamma]\!]$ by defining the *canonical* semantic environment $\rho_{\Gamma} : \Pi_{x_i \in \Gamma} [\![\theta_i]\!]$, associating $\Uparrow_{\theta_i} \boldsymbol{\lambda}\_^N.var\ \overline{(i-1)} \in [\![\theta_i]\!]$ to any x_i. This leads to a generalized variant of the normalization function: $\mathsf{nbe}(M) = \Downarrow_{\theta} [\![M]\!]_{\rho_{\Gamma}} \overline{k}$.

Proposition 11. *For any $\lambda\perp$-term $\Gamma \vdash M$ in β-normal η-long form,* $\mathsf{nbe}(M) = \mathsf{rep}(M)$. *For any λY-term $\Gamma \vdash M : \theta$,* $\mathsf{nbe}(M) = \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$.

Proof. The first part is proved by induction on M. For the second part, we reason equationally as follows: $\mathsf{nbe}(M) \overset{(1)}{=} \Downarrow_\theta [\![M]\!]_{\rho_\Gamma}\, \overline{k+1} \overset{(2)}{=} \Downarrow_\theta \bigsqcup_n [\![M \upharpoonright n]\!]_{\rho_\Gamma}\, \overline{k+1} \overset{(3)}{=} \bigsqcup_n \Downarrow_\theta [\![M \upharpoonright n]\!]_{\rho_\Gamma}\, \overline{k+1} \overset{(4)}{=} \bigsqcup_n \Downarrow_\theta [\![\mathsf{BT}(M \upharpoonright n)]\!]_{\rho_\Gamma}\, \overline{k+1} \overset{(5)}{=} \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$ where (1) is by definition of NBE, (2) is by induction on M from the definition of interpretation, (3) is by continuity of $\Downarrow_\theta$ and application. Since $M \upharpoonright n$ is a simply-typed $\lambda\bot$-term, it has a normal form $\mathsf{BT}(M \upharpoonright n)$ such that $M \upharpoonright n \simeq_{\beta\eta} \mathsf{BT}(M \upharpoonright n)$. By soundness of the domain semantics, this implies that $[\![M \upharpoonright n]\!]_\rho = [\![\mathsf{BT}(M \upharpoonright n)]\!]_\rho$ and (4) is true. Finally, (5) follows from the first part. □

Internalization Following Section 2, for any λY-term $\Gamma \vdash M : \theta$, we define the syntactic NBE by $\mathsf{snbe}(M) = \downarrow_\theta M^*[\uparrow_{\theta_i} \lambda.(var\ \overline{i-1})/x_i]\ \overline{k}$ so that $\Gamma_{\mathsf{rep}} \vdash \mathsf{snbe}(M) : o$ ($\downarrow_\theta$ and $\uparrow_\theta$ are the terms defined in Section 2). In the remainder of this section we show the correctness of snbe, namely $\mathsf{BT}(\mathsf{snbe}(M)) = \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$, by noting that snbe amounts to an internalization of the NBE procedure inside the λY-calculus, i.e. that the following square commutes.



To carry out the argument, below we introduce another interpretation $\langle\!-\!\rangle$ of terms $\Gamma_{\mathsf{rep}}, \Gamma \vdash \theta$, which will treat variables from Γ_{rep} as constants. Accordingly, such a sequent is interpreted by a continuous function from $\langle\!\langle\Gamma\rangle\!\rangle$ to $\langle\!\langle\theta\rangle\!\rangle$, with Γ_{rep} is omitted. Similarly, the semantic environment for a context $\Gamma_{\mathsf{rep}}, x_1 : \theta_1, \ldots, x_k : \theta_k$ is simply $\rho : \Pi_{x_i \in \Gamma} \langle\!\langle\theta_i\rangle\!\rangle$. In the same vein, whenever we write $\Gamma \vdash M : \theta$ from now on, we will assume that none of the variables from Γ_{rep} occurs in Γ (this does not affect the generality of our results, as free variables of a λY-term can be renamed to avoid clashes with Γ_{rep}).

$$\begin{array}{rcrcrl} \langle\!\langle o\rangle\!\rangle &=& E & \langle\!\langle x\rangle\!\rangle_\rho &=& \rho(x) & \langle\!\langle var\rangle\!\rangle_\rho &=& \boldsymbol{\lambda} n^E.var\ n \\ \langle\!\langle\theta \to \theta'\rangle\!\rangle &=& \langle\!\langle\theta\rangle\!\rangle \to \langle\!\langle\theta'\rangle\!\rangle & \langle\!\langle M\ N\rangle\!\rangle_\rho &=& \langle\!\langle M\rangle\!\rangle_\rho(\langle\!\langle N\rangle\!\rangle_\rho) & \langle\!\langle app\rangle\!\rangle_\rho &=& \boldsymbol{\lambda} e_1^E.\boldsymbol{\lambda} e_2^E.app\ e_1\ e_2 \\ & & & \langle\!\langle\lambda x^\theta.M\rangle\!\rangle_\rho &=& \boldsymbol{\lambda} a^{\langle\!\langle\theta\rangle\!\rangle}.\langle\!\langle M\rangle\!\rangle_{\rho \oplus \{x \mapsto a\}} & \langle\!\langle lam\rangle\!\rangle_\rho &=& \boldsymbol{\lambda} n^E.\boldsymbol{\lambda} e^E.lam\ n\ e \\ & & & \langle\!\langle z\rangle\!\rangle_\rho &=& z & \langle\!\langle succ\rangle\!\rangle_\rho &=& \boldsymbol{\lambda} n^E.succ\ n \end{array}$$

The interpretations $[\![-]\!]$ and $\langle\!-\!\rangle$ are related by the following lemma, which is easily proved by induction. We apply $\langle\!-\!\rangle$ to $M^\star$ on the understanding that $\Gamma^\star \vdash M : \theta^\star$ implies $\Gamma_{\mathsf{rep}}, \Gamma^\star \vdash M : \theta^\star$ and that $[\![\Gamma]\!] = \langle\!\langle\Gamma_{\mathsf{rep}}, \Gamma^\star\rangle\!\rangle$.

Lemma 12. *For any type θ, $[\![\theta]\!] = \langle\!\langle\theta^\star\rangle\!\rangle$. For any λY-term $\Gamma \vdash M : \theta$ and any semantic environment $\rho \in [\![\Gamma]\!]$, we have $[\![M]\!]_\rho = \langle\!\langle M^\star\rangle\!\rangle_\rho$.*

On the other hand, $\langle\!-\!\rangle$ is related to Böhm trees by the following lemma.

Lemma 13. *For any λY-term $\Gamma_{\mathsf{rep}} \vdash M : o$, we have $\langle\!\langle M\rangle\!\rangle = \mathsf{BT}(M)$.*

Proof. Suppose first that $\Gamma_{\mathsf{rep}} \vdash M : o$ is a $\lambda\bot$-term in β-normal η-long form, i.e. $M = \mathsf{BT}(M)$. Then M must be an applicative term built from $\bot, z, succ, var, lam$ and app. By induction on M it follows that $\langle M \rangle = M$ and, thus, $\langle M \rangle = \mathsf{BT}(M)$.

Now consider arbitrary $\Gamma_{\mathsf{rep}} \vdash M : o$, and $n \in \mathbb{N}$. By construction, $M \upharpoonright n$ is a $\lambda\bot$-term. By normalization of the simply-typed λ-calculus, $M \upharpoonright n$ has a Böhm tree $\mathsf{BT}(M \upharpoonright n)$ such that $M \upharpoonright n \simeq_{\beta\eta} \mathsf{BT}(M \upharpoonright n)$. By soundness of the denotational model, it follows that $\langle M \upharpoonright n \rangle = \langle \mathsf{BT}(M \upharpoonright n) \rangle$. By the first part above, we can deduce $\langle M \upharpoonright n \rangle = \mathsf{BT}(M \upharpoonright n)$. The lemma then follows by continuity. □

Putting all the lemmas together, we arrive at our main result.

Theorem 14. *For any λY-term $\Gamma \vdash M : \theta$,*

$$\mathsf{BT}(\mathsf{snbe}(M)) = \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n)).$$

Proof. $\mathsf{BT}(\mathsf{snbe}(M)) \overset{(1)}{=} \langle \mathsf{snbe}(M) \rangle \overset{(2)}{=} \langle \downarrow_\theta \ M^*[\ \uparrow_{\theta_i} \lambda.(var\ \overline{i-1})/x_i]\ \overline{k} \rangle \overset{(3)}{=} \langle \downarrow_\theta \rangle\ \langle M^* \rangle_{x_i \mapsto \langle \uparrow_{\theta_i} \lambda.(var\ \overline{i-1}) \rangle}\ \langle \overline{k} \rangle \overset{(4)}{=} \Downarrow_\theta \ [\![M]\!]_{x_i \mapsto \Uparrow_{\theta_i}(\boldsymbol{\lambda}.(var\ \overline{i-1}))}\ \overline{k} \overset{(5)}{=} \mathsf{nbe}(M) \overset{(6)}{=} \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$. (1) follows from Lemma 13, (2) and (3) from the definitions of snbe and $\langle - \rangle$ respectively, (4) is a consequence of $\langle \downarrow_\theta \rangle = \Downarrow_\theta$, $\langle \uparrow_\theta \rangle = \Uparrow_\theta$ and Lemma 12. Finally, (5) follows from the definition of nbe and (6) from Proposition 11. □

4 Generalization to PCF

Here we show to the methodology of the previous section can be applied to PCF [15]. This brings us closer to realistic programs that can branch on data values. PCF extends the λY-calculus in that the ground type o is replaced with the type of natural numbers, basic arithmetic and zero testing. It was designed to be Turing-complete, so in order to avoid obvious undecidability we focus on its finitary variants. In what follows we consider boolean PCF_2 in which o is the type of booleans, but it should be clear that the techniques extend to any finite ground type. PCF_2 types are defined by the grammar $\theta ::= \mathtt{B} \mid \theta_1 \to \theta_2$ and terms are given by $M, N ::= \lambda x^\theta.M \mid x \mid M\ N \mid Y_\theta\ M \mid \text{if } M \text{ then } N_1 \text{ else } N_2 \mid tt \mid ff$. The following typing rules apply.

$$\frac{}{\Gamma \vdash tt : \mathtt{B}} \qquad \frac{}{\Gamma \vdash ff : \mathtt{B}} \qquad \frac{\Gamma \vdash M : \mathtt{B} \qquad \Gamma \vdash N_1 : \theta \qquad \Gamma \vdash N_2 : \theta}{\Gamma \vdash \text{if } M \text{ then } N_1 \text{ else } N_2 : \theta}$$

For PCF_2, the canonical forms that we wish to generate are PCF Böhm trees: they are (potentially infinite) trees generated by the rules displayed below.

$$\frac{}{\Gamma \vdash \bot : \mathtt{B}} \qquad \frac{}{\Gamma \vdash tt : \mathtt{B}} \qquad \frac{}{\Gamma \vdash ff : \mathtt{B}} \qquad \frac{\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash M : \mathtt{B}}{\Gamma \vdash \lambda \overrightarrow{x}.M : \overrightarrow{A} \to \mathtt{B}}$$

$$\frac{\Gamma \vdash M_i : \theta_i \ \ (1 \le i \le n) \qquad \Gamma \vdash N_1 : \mathtt{B} \qquad \Gamma \vdash N_2 : \mathtt{B} \qquad (x : \overrightarrow{\theta} \to \mathtt{B}) \in \Gamma}{\Gamma \vdash \text{if } x\ \overrightarrow{M} \text{ then } N_1 \text{ else } N_2 : \mathtt{B}}$$

Such trees were first introduced along with the game semantics for PCF [10, 2], serving as a syntactic counterpart to *strategies*. Later developments in game semantics [3, 12] showed that two terms of PCF have the same strategy, i.e. the same PCF Böhm tree, if and only if they are indistinguishable by evaluation contexts having access to rich computational effects such as state and control operators. This makes PCF Böhm trees a very natural candidate for a representation of higher-order programs to be used for verification purposes, since purely functional programs are eventually executed in runtime environments that have access to such computational effects.

We also consider the variant $\mathrm{PCF}_{2,\perp}$ consisting of Y-free terms of PCF_2, extended with a constant $\perp : \mathsf{B}$. They are partially ordered by the obvious adaptation of the partial order $\sqsubseteq$ on $\lambda\perp$-terms, still written $\sqsubseteq$. As before, this partial order extends to an ω-cpo on infinite terms of $\mathrm{PCF}_{2,\perp}$, referred to as $\mathrm{PCF}^{\infty}_{2,\perp}$.

The operational semantics for PCF_2 for which PCF Böhm trees can be taken to be canonical representatives is obtained by adding the rules listed below to β, η and Y. The rules are restricted to the cases where both sides typecheck.

$$\text{if } \mathit{tt} \text{ then } N_1 \text{ else } N_2 \to_{\beta'_1} N_1 \qquad \text{if } \mathit{ff} \text{ then } N_1 \text{ else } N_2 \to_{\beta'_2} N_2$$

$$M \to_{\eta'} \text{if } M \text{ then } \mathit{tt} \text{ else } \mathit{ff}$$

$$(\text{if } M \text{ then } N_1 \text{ else } N_2)\ N' \to_{\gamma_1} \text{if } M \text{ then } N_1\ N' \text{ else } N_2\ N'$$

$$\text{if } (\text{if } M \text{ then } N_1 \text{ else } N_2) \text{ then } N_3 \text{ else } N_4 \to_{\gamma_2} \text{if } M \text{ then } (\text{if } N_1 \text{ then } N_3 \text{ else } N_4) \text{ else } (\text{if } N_2 \text{ then } N_3 \text{ else } N_4)$$

We write $\simeq_{\mathrm{PCF}}$ for the corresponding equational theory. For $\mathrm{PCF}_{2,\perp}$ we also add $\text{if } \perp \text{ then } N_1 \text{ else } N_2 \to_{\perp} \perp$. The following property can then be established by standard means.

Proposition 15. *For any term $\Gamma \vdash M : \theta$ of $\mathrm{PCF}_{2,\perp}$ there is a* PCF *Böhm tree $\Gamma \vdash \mathsf{BT}(M) : \theta$ such that $M \simeq_{\mathrm{PCF}} \mathsf{BT}(M)$.*

To define Böhm trees of arbitrary PCF_2-terms $\Gamma \vdash M : \theta$, we define the nth approximant of M, written $M \upharpoonright n$, by replacing fixpoint combinators Y_θ with $\lambda f^{\theta}.f^n(\perp_\theta)$, yielding a term of $\mathrm{PCF}_{2,\perp}$. In the general case, the PCF Böhm tree is then defined by $\mathsf{BT}(M) = \bigsqcup_n \mathsf{BT}(M \upharpoonright n)$.

Representability We would like to represent PCF Böhm trees of PCF_2-terms as level-2 ground λY-terms. To that end, we shall use a new context Γ_{pcf}, defined as Γ_{rep} augmented with $\{\, \mathit{tt} : o, \mathit{ff} : o, \mathit{if} : o \to o \to o \to o \,\}$ to take the shape of PCF Böhm trees into account. Just as before, the set of ground $\lambda\perp^{\infty}$-terms typed in context Γ_{pcf} forms a pointed ω-cpo. It will be the target of our transformation and, again, we will denote it by E. The representation function $\mathsf{rep}(-)$ of Section 2 extends easily to a function mapping any $\mathrm{PCF}_{2,\perp}$-term $\Gamma \vdash M : \theta$ to a $\lambda\perp$-term $\Gamma_{\mathsf{pcf}} \vdash \mathsf{rep}(M) : o$ using De Bruijn levels. Our representation result for PCF_2 can then be stated as follows.

Theorem 16. *For any PCF_2 term $\Gamma \vdash M : \theta$, there is a λY-term $\Gamma_{\mathsf{pcf}} \vdash M_{rep} : o$ such that $\mathsf{BT}(M_{rep}) = \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$.*

To construct M_{rep} from M, we first apply the following syntactic transformation translating types and terms of PCF_2 to λY-terms, inlining the operation $(-)^*$ on λY-terms defined in Section 2 with an elimination of booleans through their replacement by the corresponding Church encodings.

$$\begin{aligned}
\mathtt{B}^* &= (o \to o) \to (o \to o) \to (o \to o) & tt^* &= \lambda a^{o\to o}.\lambda b^{o\to o}.a \\
(\theta_1 \to \theta_2)^* &= \theta_1^* \to \theta_2^* & ff^* &= \lambda a^{o\to o}.\lambda b^{o\to o}.b \\
(\text{if } M \text{ then } N_1 \text{ else } N_2)^* &= \lambda \overrightarrow{x}^{\overrightarrow{\theta}}.\lambda a^{o\to o}.\lambda b^{o\to o}.M^*\ (N_1^*\ \overrightarrow{x}\ a\ b)\ (N_2^*\ \overrightarrow{x}\ a\ b)
\end{aligned}$$

where in the last equation, N_1 and N_2 have type $\overrightarrow{\theta} \to \mathtt{B}$. The translation $(-)^*$ can be extended to all the other term constructors in the obvious way. Note that, since this translation gives us a λY-term, we can already apply the transformation of Section 2 and obtain a λY-term generating the Böhm tree of M^*. To get the PCF *Böhm tree of* M instead one can modify the $\downarrow_\theta, \uparrow_\theta$ terms as follows.

$$\begin{aligned}
&\downarrow_{\mathtt{B}} = \lambda x^{\mathtt{B}^*}.x\ (\lambda.tt)\ (\lambda.ff) \qquad \uparrow_{\mathtt{B}} = \lambda e^{o\to o}.\lambda a^{o\to o}.\lambda b^{o\to o}.\lambda n^o.if\ (e\ n)\ (a\ n)\ (b\ n) \\
&\downarrow_{\theta_1\to\theta_2} = \lambda x^{\theta_1^*\to\theta_2^*}.\lambda n^o.lam\ n\ (\downarrow_{\theta_2}\ (x\ (\uparrow_{\theta_1}\ \lambda.var\ n))\ (succ\ n)) \\
&\uparrow_{\theta_1\to\theta_2} = \lambda e^{o\to o}.\lambda a^{\theta_1^*}.\uparrow_{\theta_2}\ \lambda n^o.app\ (e\ n)\ (\downarrow_{\theta_1}\ a\ n)
\end{aligned}$$

Finally, assuming $\Gamma = x_1 : \theta_1, \ldots, x_k : \theta_k$, we set $M_{rep} = \downarrow_\theta M^*[\uparrow_{\theta_i} \lambda.var\ \overline{i-1}/x_i]\ \overline{k}$.

5 Conclusion

We have shown that Böhm trees of arbitrary λY-terms and PCF Böhm trees of terms of PCF_2 can be generated by higher-order recursion schemes. We can use the result to reduce various decision problems related to the λY-calculus or PCF_2 to problems for higher-order recursive schemes.

The first class of problems to which the reduction technique can be applied are equivalence problems.

Corollary 17. *The following problems are equivalent: HORS equivalence, Böhm tree equivalence in the* λY*-calculus,* PCF *Böhm tree equivalence in* PCF_2.

The decidability status of these problems is currently unknown, but they are known to be related to two other interesting problems: the equivalence problem for deterministic collapsible pushdown automata [9] and contextual equivalence for PCF_2-terms with respect to contexts with state and control effects (contextual equivalence for PCF_2-terms with respect to purely functional contexts is undecidable, even without Y [13]). We note that Statman has proved that convertibility ($\simeq_{\beta\eta Y}$) is undecidable for the λY-calculus [17]. The connection with Böhm tree equivalence, if there is one, is not obvious: two λY-terms can have the same Böhm tree without being convertible to each other. For instance, $\lambda z^o.Y_o\ (\lambda x^o.x)$ and $\lambda z^o.Y_{o\to o}\ (\lambda x^{o\to o}.x)\ z$ both have $\lambda z^o.\bot$ as their Böhm tree, but they are not interconvertible (the combinator $Y_{o\to o}$ cannot be created or erased during reduction).

In a different direction, we can exploit the decidability result for MSO on trees generated by HORS [14] to decide a variety of properties of Böhm trees.

Corollary 18. *The following problems are decidable for λY- and PCF_2-terms: normalizability (is a given term equivalent to a Y-free term?), finiteness (is the Böhm tree/PCF Böhm tree finite, i.e. a $\lambda\perp$/$\mathrm{PCF}_{2,\perp}$-term?), solvability (is the Böhm tree/PCF Böhm tree of a term M non-empty after leading abstractions?), prefix (does the Böhm tree/PCF Böhm tree of a term have a given finite prefix?).*

The results follow from the fact that all the relevant properties are easily expressible in the modal μ-calculus, and so in MSO. Normalizability and solvability for the λY-calculus were already known to be decidable [17]. To the best of our knowledge, the other consequences are new.

References

1. A. Abel, T. Coquand, and P. Dybjer. Normalization by evaluation for Martin-Löf type theory with typed equality judgements. In *LICS*, pages 3–12, 2007.
2. S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.
3. S. Abramsky and G. McCusker. Linearity, sharing and state. In P. W. O'Hearn and R. D. Tennent, editors , *Algol-like languages*, pages 297–329. Birkhaüser, 1997.
4. T. Altenkirch, P. Dybjer, M. Hofmann, and P. J. Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *LICS*, pages 303–310, 2001.
5. U. Berger and H. Schwichtenberg. An inverse of the evaluation functional for typed lambda-calculus. In *LICS*, pages 203–211. IEEE Computer Society, 1991.
6. W. Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982.
7. N. G. De Bruijn. Lambda calculus notation with nameless dummies. In *Indagationes Mathematicae (Proceedings)*, volume 75, pages 381–392. Elsevier, 1972.
8. P. Dybjer and A. Filinski. Normalization and partial evaluation. *APPSEM Summer School*, *LNCS*, 2395, pages 137–192, 2000.
9. M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In *LICS*, pages 452–461, 2008.
10. J. M. E. Hyland and C.-H. L. Ong. On Full Abstraction for PCF. *Information and Computation*, 163(2):285–408, 2000.
11. N. Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *Proceedings of POPL*, pages 416–428, 2009.
12. J. Laird. Full abstraction for functional languages with control. In *Proceedings of 12th IEEE Symposium on Logic in Computer Science*, pages 58–67, 1997.
13. R. Loader. Finitary PCF is not decidable. *Theoretical Computer Science*, 266(1-2):341–364, 2001.
14. C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proceedings of LICS*, pages 81–90. Computer Society Press, 2006.
15. G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.
16. S. Salvati and I. Walukiewicz. Recursive schemes, Krivine machines, and collapsible pushdown automata. In *Proceedings of RP*, *LNCS* 7550, pages 6–20. Springer, 2012.
17. R. Statman. On the lambda Y calculus. *Ann. Pure Appl. Logic*, 130(1-3):325–337, 2004.
18. G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, 1993. Foundations of Computing Series.

A Complementary proofs (λY-calculus)

Lemma 19. *For any β-normal, η-long $\lambda\bot$-term $\Gamma \vdash M : \theta$ with $\Gamma = x_1 : \theta_1, \ldots, x_k : \theta_k$,*

$$\Downarrow_\theta [\![M]\!]_{\rho\Gamma}\ \overline{k} = \mathrm{repr}_\nu(M, k)$$

with $\nu : x_i \mapsto i - 1$.

Proof. It is standard that β-normal, η-long $\lambda\bot$-terms can be generated by the following rules.

$$\frac{}{\Gamma \vdash \bot : o} \qquad \frac{\Gamma \vdash M_i : \theta_i \quad (1 \le i \le n) \qquad (x_j : \overrightarrow{\theta} \to o) \in \Gamma}{\Gamma \vdash x_j\ \overrightarrow{M_i} : o}$$

$$\frac{\Gamma, y_1 : \theta'_1, \ldots, y_p : \theta'_p \vdash M : o}{\Gamma \vdash \lambda\overrightarrow{y}.M : \overrightarrow{\theta'} \to o}$$

For any such β-normal η-long $\lambda\bot$-term, we reason by induction on the corresponding derivation.
Bottom. We have:

$$\frac{}{\Gamma \vdash \bot : o}$$

Then $\mathrm{repr}_\nu(\bot, k) = \bot$ by definition. Likewise,

$$\begin{aligned} \Downarrow_o [\![\bot]\!]_{\rho\Gamma}\ \overline{k} &= (\lambda a^E.\bot_o)\ \overline{k} \\ &= \bot_o \end{aligned}$$

Application. The rule for application for normal forms is:

$$\frac{\Gamma \vdash M_i : \theta_i \quad (1 \le i \le n) \qquad (x_j : \overrightarrow{\theta} \to o) \in \Gamma}{\Gamma \vdash x_j\ \overrightarrow{M_i} : o}$$

where $\theta = \theta_1 \to \ldots \to \theta_k \to o$. By definition of repr, we have:

$$\begin{aligned} \mathrm{repr}_\nu(x_j\ \overrightarrow{M_i}, k) &= app\ (var\ \overline{\nu(x_j)})\ \overrightarrow{\mathrm{repr}_\nu(M_i, k)} \\ &= app\ (var\ \overline{j-1})\ \overrightarrow{\mathrm{repr}_\nu(M_i, k)} \end{aligned}$$

where $app\ e\ \overrightarrow{e_i}$ is a shortcut notation for:

$$app\ (app\ (\ldots\ (app\ e\ e_1)\ \ldots)\ e_{n-1})\ e_n$$

Likewise,

$$
\begin{aligned}
&\Downarrow_o [\![x_j\ \overrightarrow{M_i}]\!]_{\rho_\Gamma}\ \overline{k}\\
&= [\![x_j]\!]_{\rho_\Gamma}\ \overrightarrow{[\![M_i]\!]_{\rho_\Gamma}}\ \overline{k}\\
&= (\rho_\Gamma(x_j))\ \overrightarrow{[\![M_i]\!]_{\rho_\Gamma}}\ \overline{k}\\
&= (\Uparrow_\theta \lambda.var\ \overline{j-1})\ \overrightarrow{[\![M_i]\!]_{\rho_\Gamma}}\ \overline{k}\\
&= \Uparrow_{\theta_2\to\ldots\to\theta_k\to o} \widehat{app}\ (\lambda.var\ \overline{j-1})\ (\Downarrow_{\theta_1} [\![M_1]\!]_{\rho_\Gamma})\ [\![M_2]\!]_{\rho_\Gamma}\ \ldots\ [\![M_k]\!]_{\rho_\Gamma}\ \overline{k}\\
&= \Uparrow_{\theta_3\to\ldots\to\theta_k\to o} \widehat{app}\ (\widehat{app}\ (\lambda.var\ \overline{j-1})\ (\Downarrow_{\theta_1} [\![M_1]\!]_{\rho_\Gamma}))\ (\Downarrow_{\theta_2} [\![M_2]\!]_{\rho_\Gamma})\ [\![M_3]\!]_{\rho_\Gamma}\ \ldots\ [\![M_k]\!]_{\rho_\Gamma}\ \overline{k}\\
&= \ldots\\
&= \widehat{app}\ (\lambda.var\ \overline{j-1})\ \overrightarrow{\Downarrow_{\theta_i} [\![M_i]\!]_{\rho_\Gamma}}\ \overline{k}\\
&= app\ (var\ \overline{j-1})\ \overrightarrow{\Downarrow_{\theta_i} [\![M_i]\!]_{\rho_\Gamma}}\ \overline{k}\\
&= app\ (var\ \overline{j-1})\ \overrightarrow{\mathrm{repr}_\nu(M_i,k)}
\end{aligned}
$$

Abstraction. The rule is:

$$\frac{\Gamma, y_1:\theta'_1,\ldots,y_p:\theta'_p \vdash M : o}{\Gamma \vdash \lambda\overrightarrow{y}.M : \overrightarrow{\theta'}\to o}$$

where $\Gamma = x_1:\theta_1,\ldots,x_k:\theta_k$. By definition of repr, we have

$$\mathrm{repr}_\nu(\lambda\overrightarrow{y}.M,k) = lam\ \overline{k}\ (lam\ \overline{k+1}\ \ldots\ (lam\ \overline{k+p-1}\ (\mathrm{repr}_{\nu'}(M,k+p)))\ldots)$$

where $\nu' = \nu \cup \{y_i \mapsto k+i-1\}$. Likewise, we have:

$$
\begin{aligned}
&\Downarrow_{\overrightarrow{\theta'_i}\to o} [\![\lambda\overrightarrow{y}.M]\!]_{\rho_\Gamma}\ \overline{k}\\
&= \widehat{lam}\ (\lambda n^N.\ \Downarrow_{\theta'_2\to\ldots\to\theta'_p\to o} ([\![\lambda\overrightarrow{y}.M]\!]_{\rho_\Gamma}\ (\Downarrow_{\theta'_1} (\widehat{var}\ n))))\ \overline{k}\\
&= lam\ \overline{k}\ (\Downarrow_{\theta'_2\to\ldots\to\theta'_p\to o} ([\![\lambda\overrightarrow{y}.M]\!]_{\rho_\Gamma}\ (\Downarrow_{\theta'_1} (\widehat{var}\ \overline{k})))\ \overline{k+1})\\
&= lam\ \overline{k}\ (\Downarrow_{\theta'_2\to\ldots\to\theta'_p\to o} (\lambda a^{[\![\theta_1]\!]}.[\![\lambda y_2\ldots y_p.M]\!]_{\rho_\Gamma\oplus\{y_1\mapsto a\}})\ (\Downarrow_{\theta'_1} (\widehat{var}\ \overline{k}))\ \overline{k+1})\\
&= lam\ \overline{k}\ (\Downarrow_{\theta'_2\to\ldots\to\theta'_p\to o} [\![\lambda y_2\ldots y_p.M]\!]_{\rho_{\Gamma,y_1:\theta'_1}}\ \overline{k+1})\\
&= \ldots\\
&= lam\ \overline{k}\ (\ldots\ (lam\ \overline{k+p-1}\ (\Downarrow_o [\![M]\!]_{\rho_{\Gamma,y_1:\theta'_1,\ldots,y_p:\theta'_p}}\ \overline{k+p}))\ldots)\\
&= lam\ \overline{k}\ (\ldots\ (lam\ \overline{k+p-1}\ \mathrm{repr}_{\nu'}(M,k+p))\ldots)
\end{aligned}
$$

□

B Complementary proofs for $\mathbf{PCF_2}$

B.1 Each $\mathbf{PCF_{2,\perp}}$ term is convertible to a finite PCF Böhm tree

The fact that each $\mathrm{PCF}_{2,\perp}$ is convertible to a finite PCF Böhm tree is folklore. However we do not know of a published proof, so we include one for completeness.

Proposition 20. *The reduction* $\to_{\beta\cup\beta'_1\cup\beta'_2}$ *terminates on well-typed terms of* $\mathrm{PCF}_{2,\perp}$.

Proof. For each term of $\mathrm{PCF}_{2,\perp}$ the translation $(-)^*$ gives a $\lambda\perp$-term. By induction on M, it is direct to prove that if $M \to_{\beta\cup\beta'_1\cup\beta'_2} N$, then $M^* \to_\beta \leftarrow_\eta^* N^*$, where $\leftarrow_\eta$ is η-contraction. It is well-known that β-reduction together with η-contraction form a normalizing rewriting system on simply typed $\lambda\perp$-terms, so $\to_{\beta\cup\beta'_1\cup\beta'_2}$ terminates on $\mathrm{PCF}_{2,\perp}$ terms. □

Let us write $\to$ for $\to_{\beta\cup\eta\cup\beta'_1\cup\beta'_2\cup\eta'\cup\gamma_1\cup\gamma_2\cup\perp}$ on terms of $\mathrm{PCF}_{2,\perp}$.

Lemma 21. *Let* $\Gamma, x : \mathtt{B} \vdash \text{if } x \text{ then } N_1 \text{ else } N_2 : \mathtt{B}$ *and* $\Gamma \vdash M : \mathtt{B}$ *be two finite* PCF *Böhm trees. Then there exists a* PCF *Böhm tree* $\Gamma \vdash M' : \theta$ *such that* if M then N_1 else $N_2 \to^* M'$.

Proof. By induction on M.

– If $M = \perp$, then:

$$\text{if } \perp \text{ then } N_1 \text{ else } N_2 \to_\perp \perp$$

which is a PCF Böhm tree.

– If $M = tt$, then:

$$\text{if } tt \text{ then } N_1 \text{ else } N_2 \to_{\beta'_1} N_1$$

which is a PCF Böhm tree by construction.

– If $M = ff$, same reasoning.

– If $M = \text{if } M'_1 \text{ then } M'_2 \text{ else } M'_3$, then:

$$\begin{aligned} & \text{if } (\text{if } M'_1 \text{ then } M'_2 \text{ else } M'_3) \text{ then } N_1 \text{ else } N_2 \\ \to_{\gamma_2} & \text{ if } M'_1 \text{ then } (\text{if } M'_2 \text{ then } N_1 \text{ else } N_2) \text{ else } (\text{if } M'_3 \text{ then } N_1 \text{ else } N_2) \\ \to^* & \text{ if } M'_1 \text{ then } M''_1 \text{ else } M'_2 \end{aligned}$$

where, by induction hypothesis, M''_1 and M''_2 are PCF Böhm trees such that:

$$\begin{aligned} &\text{if } M'_2 \text{ then } N_1 \text{ else } N_2 \to^* M''_1 \\ &\text{if } M'_3 \text{ then } N_1 \text{ else } N_2 \to^* M''_2 \end{aligned}$$

It results that if M'_1 then M''_1 else M''_2 is a PCF Böhm tree.

□

Proposition 22. *For any term* $\Gamma \vdash M : \theta$ *of* $\mathrm{PCF}_{2,\perp}$, *there is a* PCF *Böhm tree* $\Gamma \vdash \mathsf{BT}(M) : \theta$ *such that* $M \to^* \mathsf{BT}(M)$.

Proof. By Proposition 20, we can assume M is normal for $\to_{\beta\cup\beta'_1\cup\beta'_2}$. Then, we reason by induction on the lexicographic order (m, t) where t is the length of the type of M and m is the length of M.

– If $M = \bot, tt, ff$, obvious.
– If $M = \lambda x^{\theta}.M'$, with:

$$\Gamma, x : \theta \vdash M' : \theta_1 \to \ldots \to \theta_k \to \mathtt{B}$$

By induction hypothesis, we have $M' \to^* \mathsf{BT}(M')$, which has the form $\lambda \overrightarrow{y}.M''$. Then $N = \lambda x^{\theta}.\lambda \overrightarrow{y}.M''$ is still a PCF Böhm tree, and $M \to^* N$.
– If $M = x\ N_1\ \ldots\ N_k$, where $\Gamma \vdash M : \theta_1 \to \ldots \to \theta_k \to \theta_{k+1} \to \ldots \to \theta_{k+p} \to \mathtt{B}$. Then we consider:

$$\lambda y_1^{\theta_{k+1}} \ldots . \lambda y_p^{\theta_{k+p}}.x\ N_1\ \ldots\ N_k\ y_1\ \ldots\ y_p$$

By induction hypothesis, each of the N_i and y_i reduce to a PCF Böhm tree (each of the N_i has length smaller than M, and each of the y_i has at most the same length of M and a type of length smaller than that of M). So:

$$\begin{aligned} x\ N_1\ \ldots\ N_k &\to_{\eta}^* \lambda y_1^{\theta_{k+1}} \ldots . \lambda y_p^{\theta_{k+p}}.x\ N_1\ \ldots\ N_k\ y_1\ \ldots\ y_p \\ &\to^* \lambda y_1^{\theta_{k+1}} \ldots . \lambda y_p^{\theta_{k+p}}.x\ \mathsf{BT}(N_1)\ \ldots\ \mathsf{BT}(N_k)\ \mathsf{BT}(y_1)\ \ldots\ \mathsf{BT}(y_p) \\ &\to_{\eta'} \lambda y_1^{\theta_{k+1}} \ldots . \lambda y_p^{\theta_{k+p}}.\text{if } x\ \mathsf{BT}(N_1)\ \ldots\ \mathsf{BT}(N_k)\ \mathsf{BT}(y_1)\ \ldots\ \mathsf{BT}(y_p) \text{ then } tt \text{ else } ff \end{aligned}$$

which is a PCF Böhm tree by construction.
– If $M = (\text{if } M_1 \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k$, where

$$\Gamma \vdash (\text{if } M_1 \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k : \theta_1 \to \ldots \to \theta_p \to \mathtt{B}$$

then we consider the term:

$$\text{if } M_1 \text{ then } M_2\ N_1\ \ldots\ N_k \text{ else } M_3\ N_1\ \ldots\ N_k$$

By induction hypothesis, there are PCF Böhm trees $\mathsf{BT}(M_1)$, and:

$$\begin{aligned} \mathsf{BT}(M_2\ N_1\ \ldots\ N_k) &= \lambda y_1^{\theta_1} \ldots . \lambda y_p^{\theta_p}.M_2' \\ \mathsf{BT}(M_3\ N_1\ \ldots\ N_k) &= \lambda y_1^{\theta_1} \ldots . \lambda y_p^{\theta_p}.M_3' \end{aligned}$$

Four cases arise, depending on the shape of $\mathsf{BT}(M_1)$:
- If $\mathsf{BT}(M_1) = \bot$, then:

$$\begin{aligned} (\text{if } M_1 \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k &\to^* (\text{if } \bot \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k \\ &\to_{\bot} \bot\ N_1\ \ldots\ N_k \\ &\to_{\beta^*} \bot \end{aligned}$$

which is a PCF Böhm tree (recall that $\bot$ at higher type is an abbreviation for $\lambda \overrightarrow{y}.\bot$).
- If $\mathsf{BT}(M_1) = tt$, then:

$$\begin{aligned} (\text{if } M_1 \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k &\to^* (\text{if } tt \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k \\ &\to_{\beta_1'} M_2\ N_1\ \ldots\ N_k \\ &\to^* \mathsf{BT}(M_2\ N_1\ \ldots\ N_k) \end{aligned}$$

which is a PCF Böhm tree.

- If $\mathsf{BT}(M_1) = \mathit{ff}$, the proof is symmetric.
- If $\mathsf{BT}(M_1) = \text{if } y\ S_1\ \ldots\ S_q \text{ then } T_1 \text{ else } T_2$, then:

$$
\begin{array}{rl}
 & (\text{if } M_1 \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k \\
\to_{\eta^*} & \lambda\overrightarrow{z}.(\text{if } M_1 \text{ then } M_2 \text{ else } M_3)\ N_1\ \ldots\ N_k\ \overrightarrow{z} \\
\to^*_{\gamma_1} & \lambda\overrightarrow{z}.\text{if } M_1 \text{ then } M_2\ N_1\ \ldots\ N_k\ \overrightarrow{z} \text{ else } M_3\ N_1\ \ldots\ N_k\ \overrightarrow{z} \\
\to^* & \lambda\overrightarrow{z}.\text{if } M_1 \text{ then } (\lambda\overrightarrow{y}.M_2')\ \overrightarrow{z} \text{ else } (\lambda\overrightarrow{y}.M_3')\ \overrightarrow{z} \\
\to^*_{\beta} & \lambda\overrightarrow{z}.\text{if } M_1 \text{ then } M_2'[\overrightarrow{z}/\overrightarrow{y}] \text{ else } M_3'[\overrightarrow{z}/\overrightarrow{y}] \\
\simeq_{\alpha} & \lambda\overrightarrow{y}.\text{if } (\text{if } y\ S_1\ \ldots\ S_q \text{ then } T_1 \text{ else } T_2) \text{ then } M_2' \text{ else } M_3' \\
\to^* & M''
\end{array}
$$

where M'' is the PCF Böhm tree obtained by Lemma 21.

□

B.2 Semantic NBE for $\mathbf{PCF_2}$

Here E is the pointed ω-cpo of β-normal $\lambda\perp^\infty$-terms of ground type in context Γ_{pcf}. We form $\widehat{E} = E \to E$. We first define a domain semantics for PCF_2:

$$
\begin{aligned}
[\![\mathtt{B}]\!] &= \widehat{E} \to \widehat{E} \to \widehat{E} \\
[\![\theta_1 \to \theta_2]\!] &= [\![\theta_1]\!] \to [\![\theta_2]\!]
\end{aligned}
$$

$$
\begin{aligned}
[\![tt]\!]_\rho &= \lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.a \\
[\![\mathit{ff}]\!]_\rho &= \lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.b \\
[\![x]\!]_\rho &= \rho(x) \\
[\![\lambda x^\theta.M]\!]_\rho &= \lambda a^{[\![\theta]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} \\
[\![M\ N]\!]_\rho &= [\![M]\!]_\rho([\![N]\!]_\rho) \\
[\![Y_\theta]\!] &= \bigsqcup_n \lambda f^{[\![\theta]\!] \to [\![\theta]\!]}.f^n(\perp) \\
[\![\text{if } M \text{ then } N_1 \text{ else } N_2]\!]_\rho &= \lambda e_1^{[\![\theta_1]\!]}.\ldots.\lambda e_k^{[\![\theta_k]\!]}.\lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.[\![M]\!]_\rho([\![N_1]\!]_\rho\ e_1\ \ldots\ e_k\ a\ b)([\![N_2]\!]_\rho\ e_1\ \ldots\ e_k\ a\ b)
\end{aligned}
$$

where in the last line, N_1 and N_2 has type $\theta_1 \to \ldots \to \theta_k \to \mathtt{B}$. Soundness of this domain semantics is standard.

We now define the De Bruijn term formers for PCF:

$$\begin{aligned}
\widehat{var} &: E \to \widehat{E} \\
&= \lambda v^{E}.\lambda n^{E}.var(v) \\
\widehat{app} &: \widehat{E} \to \widehat{E} \to \widehat{E} \\
&= \lambda e_1^{\widehat{E}}.\lambda e_2^{\widehat{E}}.\lambda n^{N}.app\ (e_1\ n)\ (e_2\ n) \\
\widehat{lam} &: (E \to \widehat{E}) \to \widehat{E} \\
&= \lambda f^{E\to\widehat{E}}.\lambda n^{E}.lam\ n\ (f\ n\ (succ\ n)) \\
\widehat{if} &: \widehat{E} \to \widehat{E} \to \widehat{E} \to \widehat{E} \\
&= \lambda e_1^{\widehat{E}}.\lambda e_2^{\widehat{E}}.\lambda e_3^{\widehat{E}}.\lambda n^{E}.if\ (e_1\ n)\ (e_2\ n)\ (e_3\ n)) \\
\widehat{tt} &: \widehat{E} \\
&= \lambda n^{E}.tt \\
\widehat{ff} &: \widehat{E} \\
&= \lambda n^{E}.ff
\end{aligned}$$

Now, we describe the NBE construction itself. The reflect and reify functions are:

$$\begin{aligned}
\Downarrow_{\theta} &\in [\![\theta]\!] \to \widehat{E} \\
\Downarrow_{\mathsf{B}} t &= t\ \widehat{tt}\ \widehat{ff} \\
\Downarrow_{\theta_1\to\theta_2} t &= \widehat{lam}\ (\lambda v^{E}.\ \Downarrow_{\theta_2} (t\ (\Uparrow_{\theta_1} \widehat{var}\ v)))
\end{aligned}$$

$$\begin{aligned}
\Uparrow_{\theta} &\in \widehat{E} \to [\![\theta]\!] \\
\Uparrow_{\mathsf{B}} e &= \lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.\widehat{if}\ e\ a\ b \\
\Uparrow_{\theta_1\to\theta_2} e &= \lambda a^{[\![\theta_1]\!]}.\ \Uparrow_{\theta_2} \widehat{app}\ e\ (\Downarrow_{\theta_1} a)
\end{aligned}$$

With that, we can define the NBE function. Let $\Gamma \vdash M : \theta$ be a term of PCF_2, with $\Gamma = x_1 : \theta_1, \ldots, x_k : \theta_k$. Let ρ_Γ be a semantic environment for Γ assoctating to each x_i the element $\Uparrow_{\theta_i} \lambda.var\ \overline{i-1}$. Then we define the NBE function as:

$$\mathsf{nbe}(M) = \Downarrow_{\theta} [\![M]\!]_{\rho_\Gamma}\ \overline{k}$$

B.3 Validity of NBE on finite PCF Böhm trees

Lemma 23. *For any β-normal, η-long $\lambda\bot$-term $\Gamma \vdash M : \theta$ with $\Gamma = x_1 : \theta_1, \ldots, x_k : \theta_k$. Then:*

$$\mathsf{nbe}(M) = \mathrm{repr}_{\nu}(M, k)$$

with $\nu : x_i \mapsto i - 1$.

Proof. By induction on M.

Bot. We have:

$$\frac{}{\Gamma \vdash \bot : \mathtt{B}}$$

Then $\mathrm{repr}_\nu(\bot, k) = \bot$ by definition. Likewise,

$$\begin{aligned}\mathsf{nbe}(\bot) &= \Downarrow_{\mathtt{B}} [\![\bot]\!]_{\rho_\Gamma}\ \overline{k} \\ &= \Downarrow_{\mathtt{B}} (\lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.\bot)\ \overline{k} \\ &= \bot\end{aligned}$$

True/False. The cases of true and false are obviously symmetric, we only treat true. The rule is:

$$\frac{}{\Gamma \vdash tt : \mathtt{B}}$$

We have $\mathrm{repr}_\nu(tt, k) = tt$ by definition. Likewise:

$$\begin{aligned}\mathsf{nbe}(tt) &= \Downarrow_{\mathtt{B}} [\![tt]\!]_{\rho_\Gamma}\ \overline{k} \\ &= \Downarrow_{\mathtt{B}} (\lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.a)\ \overline{k} \\ &= \widehat{tt}\ \overline{k} \\ &= tt\end{aligned}$$

If/application. The rule for if/application is:

$$\frac{\Gamma \vdash M_i : \theta'_i \quad (1 \le i \le n) \qquad \Gamma \vdash N_1 : \mathtt{B} \qquad \Gamma \vdash N_2 : \mathtt{B} \qquad (x : \overrightarrow{\theta'} \to \mathtt{B}) \in \Gamma}{\Gamma \vdash \text{if } x\ \overrightarrow{M} \text{ then } N_1 \text{ else } N_2 : \mathtt{B}}$$

By definition of repr, we have:

$$\begin{aligned}&\mathrm{repr}_\nu(\text{if } x\ \overrightarrow{M_i} \text{ then } N_1 \text{ else } N_2, k) \\ =\ &\mathit{if}\ \mathrm{repr}_\nu(x\ \overrightarrow{M_i}, k)\ \mathrm{repr}_\nu(N_1, k)\ \mathrm{repr}_\nu(N_2, k) \\ =\ &\mathit{if}\ (\mathit{app}\ \mathit{var}(\nu(x))\ \overrightarrow{\mathrm{repr}_\nu(M_i, k)})\ \mathrm{repr}_\nu(N_1, k)\ \mathrm{repr}_\nu(N_2, k)\end{aligned}$$

Likewise,

$$
\begin{aligned}
&\mathsf{nbe}(\text{if } x\ \overrightarrow{M_i} \text{ then } N_1 \text{ else } N_2)\\
&= \Downarrow_{\mathtt{B}} [\![\text{if } x\ \overrightarrow{M_i} \text{ then } N_1 \text{ else } N_2]\!]_{\rho_\Gamma}\ \overline{k}\\
&= [\![\text{if } x\ \overrightarrow{M_i} \text{ then } N_1 \text{ else } N_2]\!]_{\rho_\Gamma}\ \widehat{tt}\ \widehat{ff}\ \overline{k}\\
&= (\lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.[\![x\ \overrightarrow{M_i}]\!]_{\rho_\Gamma}\ ([\![N_1]\!]_{\rho_\Gamma}\ a\ b)\ ([\![N_2]\!]_{\rho_\Gamma}\ a\ b))\ \widehat{tt}\ \widehat{ff}\ \overline{k}\\
&= [\![x\ \overrightarrow{M_i}]\!]_{\rho_\Gamma}\ ([\![N_1]\!]_{\rho_\Gamma}\ \widehat{tt}\ \widehat{ff})\ ([\![N_2]\!]_{\rho_\Gamma}\ \widehat{tt}\ \widehat{ff})\ \overline{k}\\
&= [\![x\ \overrightarrow{M_i}]\!]_{\rho_\Gamma}\ (\Downarrow_{\mathtt{B}} [\![N_1]\!]_{\rho_\Gamma})\ (\Downarrow_{\mathtt{B}} [\![N_2]\!]_{\rho_\Gamma})\ \overline{k}\\
&= \Uparrow_{\overline{\theta'}\to\mathtt{B}} (\lambda.var\ \overline{\nu(x)})\ \overrightarrow{[\![M_i]\!]_{\rho_\Gamma}}\ (\Downarrow_{\mathtt{B}} [\![N_1]\!]_{\rho_\Gamma})\ (\Downarrow_{\mathtt{B}} [\![N_2]\!]_{\rho_\Gamma})\ \overline{k}\\
&= (\lambda \overrightarrow{e_i^{\theta'_i}}.\ \Uparrow_{\mathtt{B}} \widehat{app}\ (\lambda.var\ \overline{\nu(x)})\ \overrightarrow{\Downarrow_{\theta'_i} e_i})\ \overrightarrow{[\![M_i]\!]_{\rho_\Gamma}}\ (\Downarrow_{\mathtt{B}} [\![N_1]\!]_{\rho_\Gamma})\ (\Downarrow_{\mathtt{B}} [\![N_2]\!]_{\rho_\Gamma})\ \overline{k}\\
&= \Uparrow_{\mathtt{B}} \widehat{app}\ (\lambda.var\ \overline{\nu(x)})\ \overrightarrow{\Downarrow_{\theta'_i} [\![M_i]\!]_{\rho_\Gamma}}\ (\Downarrow_{\mathtt{B}} [\![N_1]\!]_{\rho_\Gamma})\ (\Downarrow_{\mathtt{B}} [\![N_2]\!]_{\rho_\Gamma})\ \overline{k}\\
&= (\lambda a^{\widehat{E}}.\lambda b^{\widehat{E}}.\widehat{if}\ (\widehat{app}\ (\lambda.var\ \overline{\nu(x)})\ \overrightarrow{\Downarrow_{\theta'_i} [\![M_i]\!]_{\rho_\Gamma}})\ a\ b)\ (\Downarrow_{\mathtt{B}} [\![N_1]\!]_{\rho_\Gamma})\ (\Downarrow_{\mathtt{B}} [\![N_2]\!]_{\rho_\Gamma})\ \overline{k}\\
&= \widehat{if}\ (\widehat{app}\ (\lambda.var\ \overline{\nu(x)})\ \overrightarrow{\Downarrow_{\theta'_i} [\![M_i]\!]_{\rho_\Gamma}})\ (\Downarrow_{\mathtt{B}} [\![N_1]\!]_{\rho_\Gamma})\ (\Downarrow_{\mathtt{B}} [\![N_2]\!]_{\rho_\Gamma})\ \overline{k}\\
&= if\ (app\ (var\ \overline{\nu(x)})\ \overrightarrow{\Downarrow_{\theta'_i} [\![M_i]\!]_{\rho_\Gamma}\ \overline{k}})\ (\Downarrow_{\mathtt{B}} [\![N_1]\!]_{\rho_\Gamma}\ \overline{k})\ (\Downarrow_{\mathtt{B}} [\![N_2]\!]_{\rho_\Gamma}\ \overline{k})\\
&= if\ (app\ (var\ \overline{\nu(x)})\ \overrightarrow{\mathsf{nbe}(M_i)})\ \mathsf{nbe}(N_1)\ \mathsf{nbe}(N_2)\\
&= if\ (app\ (var\ \overline{\nu(x)})\ \overrightarrow{\mathrm{repr}_\nu(M_i,k)})\ \mathrm{repr}_\nu(N_1,k)\ \mathrm{repr}_\nu(N_2,k)\\
&= \mathrm{repr}_\nu(\text{if } x\ \overrightarrow{M_i} \text{ then } N_1 \text{ else } N_2, k)
\end{aligned}
$$

Abstraction. The rule is:

$$\frac{\Gamma, y_1 : \theta'_1, \ldots, y_n : \theta'_n \vdash M : \mathtt{B}}{\Gamma \vdash \lambda \overrightarrow{y}.M : \overrightarrow{\theta'} \to \mathtt{B}}$$

where $\Gamma = x_1 : \theta_1, \ldots, x_k : \theta_k$. By definition of repr, we have

$$\mathrm{repr}_\nu(\lambda \overrightarrow{y}.M, k) = lam\ k\ (lam\ (k+1)\ (\ldots\ (lam\ (k+n-1)\ \mathrm{repr}_{\nu'}(M, k+n))\ldots))$$

where $\nu' = \nu \cup \{x_i \mapsto k+i-1\}$. Likewise, we have:

$$\begin{aligned}
&\mathsf{nbe}(\lambda\overrightarrow{y}.M)\\
&= \Downarrow_{\overrightarrow{\theta'}\to \mathtt{B}}\ [\![\lambda\overrightarrow{y}.M]\!]_{\rho_\Gamma}\ \overline{k}\\
&= \widehat{lam}\ (\lambda v_1 \ldots \widehat{lam}(\lambda v_n.\ \Downarrow_{\mathtt{B}} [\![\lambda\overrightarrow{y}.M]\!]_{\rho_\Gamma}\ \overrightarrow{\Uparrow_{\theta'_i}\ (\widehat{var}\ v_i)})\ldots)\ \overline{k}\\
&= \widehat{lam}\ (\lambda v_1 \ldots \widehat{lam}\ (\lambda v_n.\ \Downarrow_{\mathtt{B}} (\lambda\overrightarrow{e_i}.[\![M]\!]_{\rho_\Gamma \oplus \{y_i \mapsto e_i\}})\ \overrightarrow{\Uparrow_{\theta'_i}\ (\widehat{var}\ v_i)})\ldots)\ \overline{k}\\
&= \widehat{lam}\ (\lambda v_1 \ldots \widehat{lam}\ (\lambda v_n.\ \Downarrow_{\mathtt{B}} [\![M]\!]_{\rho_\Gamma \oplus \{y_i \mapsto \Uparrow_{\theta'_i}\ (\widehat{var}\ v_i)\}})\ldots)\ \overline{k}\\
&= lam\ \overline{k}\ (\ldots lam\ \overline{k+n-1}\ (\Downarrow_{\mathtt{B}} [\![M]\!]_{\rho_\Gamma \oplus \{y_i \mapsto \Uparrow_{\theta'_i}\ (var\ \overline{k+i-1})\}}\ \overline{k+n})\ldots)\\
&= lam\ \overline{k}\ (\ldots (lam\ \overline{k+n-1}\ (\Downarrow_{\mathtt{B}} [\![M]\!]_{\rho_{\Gamma, \overrightarrow{y_i:\theta'_i}}}\ \overline{k+n}))\ldots)\\
&= lam\ \overline{k}\ (\ldots (lam\ \overline{k+n-1}\ \mathsf{nbe}(M))\ldots)\\
&= lam\ \overline{k}\ (\ldots (lam\ \overline{k+n-1}\ \mathrm{repr}_{\nu'}(M, k+n))\ldots)\\
&= \mathrm{repr}_\nu(\lambda\overrightarrow{y}.M, k)
\end{aligned}$$

□

Proposition 24. *NBE is correct on* PCF_2, i.e. *for any term* $\Gamma \vdash M : \theta$ *of* PCF_2, *we have:*

$$\mathsf{nbe}(M) = \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$$

Proof. Same reasoning as for λY. By Lemma 23, NBE is correct on finite PCF Böhm trees. By Proposition 22 and soundness of the domains model, it is correct on arbitrary terms of $\mathrm{PCF}_{2,\perp}$. By continuity, it is correct on PCF_2. □

B.4 Internalization of NBE for $\mathbf{PCF_2}$

The construction is essentially the same as for λY. Start by defining for any term $\Gamma \vdash M : \theta$ (where $\Gamma = x_1 : \theta_1, \ldots, x_k : \theta_k$) of PCF_2 its syntactic NBE by:

$$\mathsf{snbe}(M) = \downarrow_\theta\ \ M^*[\uparrow_{\theta_i} \lambda.(var\ \overline{i-1})/x_i]\ \ \overline{k}$$

with $\Gamma_{rep} \vdash \mathsf{snbe}(M) : o$, and where the reification and reflection terms $\downarrow_\theta$ and $\uparrow_\theta$ are those defined in Section 4.

We consider the sublanguage λY_{pcf} of λY containing sequents of the form $\Gamma_{pcf}, \Gamma \vdash M : \theta$. We define an interpretation $\langle - \rangle$ of λY_{pcf} into pointed ω-cpos by:

$$\langle o \rangle = E \qquad\qquad \langle \theta \to \theta' \rangle = \langle \theta \rangle \to \langle \theta' \rangle$$

The interpretation $\langle - \rangle$ will interpret a sequent $\Gamma_{pcf}, \Gamma \vdash \theta$ by a continuous function from $\langle \Gamma \rangle$ to $\langle \theta \rangle$. It is defined on terms by:

$$\begin{aligned}
\langle x \rangle_\rho &= \rho(x) & \langle \lambda x^\theta.M \rangle_\rho &= \lambda a^{\langle \theta \rangle}.\langle M \rangle_{\rho \oplus \{x \mapsto a\}} \\
\langle M\ N \rangle_\rho &= \langle M \rangle_\rho(\langle N \rangle_\rho) & \langle z \rangle_\rho &= z \\
\langle succ \rangle_\rho &= \lambda n^E.succ\ n & \langle var \rangle_\rho &= \lambda n^E.var\ n \\
\langle app \rangle_\rho &= \lambda e_1^E.\lambda e_2^E.app\ e_1\ e_2 & \langle lam \rangle_\rho &= \lambda n^E.\lambda e^E.lam\ n\ e \\
\langle tt \rangle_\rho &= tt & \langle ff \rangle_\rho &= ff \\
\langle if \rangle_\rho &= \lambda a^E.\lambda b^E.\lambda c^E.if\ a\ b\ c
\end{aligned}$$

Any λY-term can be regarded as a λY_{pcf}-term by weakening. The interpretations $[\![-]\!]$ and $\langle - \rangle$ are accordingly related by the following lemma.

Lemma 25. *For any type θ, we have $[\![\theta]\!] = \langle \theta^* \rangle$. Moreover for any term $\Gamma \vdash M : \theta$ of* PCF_2*, for any semantic environment $\rho \in [\![\Gamma]\!]$ (which is also $\langle \Gamma^* \rangle$), we have:*

$$[\![M]\!]_\rho = \langle M^* \rangle_\rho$$

Proof. Immediate by induction on M. □

Lemma 26. *For any ground type λY_{pcf}-term $\Gamma_{pcf} \vdash M : o$, we have:*

$$\langle M \rangle = \mathsf{BT}(M)$$

Proof. Same proof as for λY. □

Theorem 27. *For any term $\Gamma \vdash M : \theta$ of* PCF_2*, we have:*

$$\mathsf{BT}(\mathsf{snbe}(M)) = \bigsqcup_n \mathsf{rep}(\mathsf{BT}(M \upharpoonright n))$$

Proof. Same proof as for λY, using the properties proved above and correctness of semantic NBE for PCF_2. □