Operational Algorithmic Game Semantics

Benedict Bunting, University of Oxford, UK
Andrzej S. Murawski, University of Oxford, UK*

Extended paper

Abstract

We consider a simply-typed call-by-push-value calculus with state, and provide a fully abstract trace model via a labelled transition system (LTS) in the spirit of operational game semantics. By examining the shape of configurations and performing a series of natural optimisation steps based on name recycling, we identify a fragment for which the LTS can be recast as a deterministic visibly pushdown automaton. This implies decidability of contextual equivalence for the fragment identified and solvability in exponential time for terms in canonical form. We also identify a fragment for which these automata are finite-state machines.

Further, we use the trace model to prove that translations of prototypical call-by-name (IA) and call-by-value (RML) languages into our call-by-push-value language are fully abstract. This allows our decidability results to be seen as subsuming several results from the literature for IA and RML. We regard our operational approach as a simpler and more intuitive way of deriving such results. The techniques we rely on draw upon simple intuitions from operational semantics and the resultant automata retain operational style, capturing the dynamics of the underlying language.

1 Introduction

The notion of contextual equivalence has been much studied in programming language theory, as it captures an especially natural form of equality between programs, widely applicable in verification. This has spurred efforts to produce denotational models in which equality in the model coincides exactly with contextual equivalence. These 'fully abstract' models represent the 'gold standard'. One particularly successful approach to this problem has arisen through *game semantics* [1, 2, 3], where a term is modelled as a strategy for a game of question and answer played between the term and its context. Over the past 25 years, game semantics has proven to be a powerful tool, providing fully abstract models for languages with a variety of features. A more recent development is *operational game semantics* [4, 5], in which strategies are represented as sets of traces generated by a labelled transition system (LTS) derived from the operational semantics of the language. This approach gives an intuitive and concrete representation of strategies, amenable to the application of operational techniques, as opposed to the abstract properties emphasised in earlier work. We summarise the main contributions of this paper below.

Firstly, we define a Call-By-Push-Value (CBPV) language equipped with dynamically generated first-order store. It can be seen as a canonical language for studying first-order state with arbitrary evaluation order.

Secondly, we present an operational game model for this language, and prove that it is fully abstract with respect to contextual equivalence. This means that terms have the same set of complete traces if and only if they are contextually equivalent.

*This research was funded in whole or in part by EPSRC EP/T006579 and EPSRC Studentship 2742896.

We provide translations from the prototypical languages RML [6] and IA [7] into CBPV, and using our model, prove that these translations are fully abstract. This validates the assertion of CBPV subsuming CBV and CBN in this setting, and justifies studying CBPV as a vehicle for investigating first-order store.

Our final result is to identify two fragments of CBPV where contextual equivalence is decidable, and derive a decision procedure for these fragments. We achieve this by using operational intuitions to refine the corresponding LTS so that it produces traces over a finite alphabet. In particular, we identify a fragment for which the refined LTS can be viewed as a deterministic visibly pushdown automaton (VPA), and a smaller fragment where it is a deterministic finite automaton (DFA). The CBPV language is particularly suited to carrying out this transformation, as it provides a notion of thunks, which turns out to offer the appropriate level of abstraction to discuss when a passage to VPA and DFA should be possible.

The results follow the spirit of algorithmic game semantics by modelling strategies as automata [8], though our automata are more intuitive thanks to their close relation to the operational semantics. Because of the translations into RML and IA, our results unify and provide new proofs for several results from this school [9, 10, 11, 12]. At the technical level, we rely on purely operational techniques, and our work should be accessible to readers without a game semantics background. Overall, our approach represents a simpler way to obtain such decidability results and provides an operational understanding of why they hold.

Other Related Work

Typed Normal Form Bisimulations [13] are arguably closest to our setting. They were developed for Jump-With-Argument, a continuation passing-style version of CBPV without state. Although not made explicit, the transition system these bisimulations are defined on bears similarity to operational game semantics (OGS).

CBPV is already known to accommodate fully abstract translations from CBN and CBV, but their full abstraction in [14] was established in a different setting from ours (with printing as the side-effect). Consequently, we had to develop a separate argument.

We believe ours is the first work seeking to give decision procedures for contextual equivalence by refining operational game models into automata. However, OGS has been used as a foundation for other techniques, particularly bisimulation-based. An example of this strand of work is the equivalence checker SyTeCi [15] for automatically proving contextual equivalences with respect to contexts with general references. It is based on Symbolic Kripke Open Relations, which can be seen as an abstraction of OGS. The associated decidability result is incomparable to ours: it uses more powerful contexts and disallows reference creation inside functions, albeit without a type restriction. HOBBIT is a bounded-complete inequivalence checker, which exploits 'symbolic up-to' techniques to allow it to (semi-) automatically prove many equivalences [16]. Kripke Normal Form Bisimulations [17] are a sound and complete technique for showing contextual equivalence for a family of CBV languages with control and higher-order state, but the cited paper does not discuss decidability. Apart from OGS-inspired research, there has been much related work based on logical relations [18, 19] and normal form bisimulations [20], also without decision procedures.

2 The language

2.1 Syntax

The language being studied in this paper extends Levy's Call-by-Push-Value λ-calculus [21] with mutable state and basic data types. We will deal with the finitary fragment of the language, lacking infinite types and recursion. The types and terms are shown in Figure 1. The types are stratified into *values* (generated by σ) and *computations* (generated by $\underline{\tau}$). The slogan often associated with this division is 'A value is, a computation does'.

Value Type	σ	$\triangleq$	$U\underline{\tau} \mid \mathrm{Unit} \mid \mathrm{Int} \mid \mathrm{Ref}$	Computation Type	$\underline{\tau}$	$\triangleq$	$F\sigma \mid \sigma \to \underline{\tau}$
Value Term	V	$\triangleq$	$\mathrm{thunk}\ M \mid x \mid () \mid \widehat{n} \mid \ell \mid \mathrm{MkVar}\ V\ V$				
Computation Term	M	$\triangleq$	$\mathrm{force}\ V \mid \mathrm{return}\ V \mid \lambda x^{\sigma}.M \mid \mathrm{let}\ x\ \mathrm{be}\ V.M \mid M\ \mathrm{to}\ x.M$ $\mid \mathrm{case}\ V\ \mathrm{of}\ (M_i)_{i\in I} \mid MV \mid \mathrm{ref}\ V \mid !V \mid V := V \mid \mathrm{while}\, M\, \mathrm{do}\, M$				
Evaluation Context	K	$\triangleq$	$\bullet \mid K\ \mathrm{to}\ x.M \mid KV$				
Value Context	V_C	$\triangleq$	$\mathrm{thunk}\ C \mid \mathrm{MkVar}\ V_C\ V \mid \mathrm{MkVar}\ V\ V_C$				
Context	C	$\triangleq$	$\bullet \mid \mathrm{force}\ V_C \mid \mathrm{return}\ V_C \mid \lambda x^{\sigma}.C \mid \mathrm{let}\ x\ \mathrm{be}\ V_C.M \mid \mathrm{let}\ x\ \mathrm{be}\ V.C \mid C\ \mathrm{to}\ x.M$ $\mid M\ \mathrm{to}\ x.C \mid \mathrm{case}\ V\ \mathrm{of}\ (M_i)_{i<j}, C, (M_i)_{j<i} \mid CV \mid MV_C \mid !V_C \mid V_C := V$ $\mid \mathrm{while}\, C\, \mathrm{do}\, M \mid \mathrm{while}\, M\, \mathrm{do}\, C$				

Notational conventions: $x, y \in \mathbf{Var},\ \ell \in \mathbf{Loc},\ I = \{0, \cdots, \max\},\ n \in I$
Syntactic sugar: If x does not occur free in N, we write $M; N$ for M to $x.N$, and Ω for $\mathrm{while}\,(\mathrm{return}\ \widehat{1})\,\mathrm{do}\,(\mathrm{return}\ ())$. We write $V_1 + V_2$ for $\mathrm{case}\ V_1\ \mathrm{of}\ (\mathrm{case}\ V_2\ \mathrm{of}\ (\widehat{i+j})_{j\in I})_{i\in I}$

Figure 1: CBPV syntax

A ***typing judgement*** gives a type to a term given the types of locations and variables. In CBPV, we have separate judgments for value types and computation types. For values we write $\Sigma; \Gamma \vdash^{v} V : \sigma$ and for computations we write $\Sigma; \Gamma \vdash^{v} M : \underline{\tau}$, where Γ is a finite partial function that assigns types to variables, and Σ is a list of locations. We abbreviate $\emptyset; \Gamma \vdash M : \tau$ to $\Gamma \vdash M : \tau$ and $\Sigma; \emptyset \vdash M : \tau$ to $\Sigma \vdash M : \tau$. Typing judgements are derived using the rules in Figure 2.

We treat the type Ref as if it were a pair of thunks: a read thunk of type $UF\mathrm{Int}$ and a write thunk of type $U(\mathrm{Int} \to F\mathrm{Unit})$. This is an old idea, due to Reynolds [22], and is a feature of many game models of languages with references. As a consequence, we need the construct MkVar, which embeds a pair of thunks into the Ref type, allowing 'bad variables', which possibly return different values from those last written. When reading ($!V$) or writing ($V := U$) such a bad variable, we use the appropriate thunk. We include bad variables for comparability with existing results, although they can be avoided by incorporating parts of the heap into the trace semantics [4]. For space reasons, we will sometimes write $\{\!|V_1, V_2|\!\}$ for $\mathrm{MkVar}\ V_1\ V_2$.

2.2 The operational semantics

We present the operational semantics as the reduction relation $\to$ in Figure 3 on configurations, which are pairs of a computation and a heap h (a mapping of locations to values). We write $h : \Sigma$ for $\Sigma \subseteq \mathrm{dom}(h)$. $h[\ell \mapsto V]$ denotes updating ℓ in h.

2.3 Contextual equivalence

Contexts C can be seen as computations with a hole, $\bullet$, into which another computation may be substituted. We can also type a context, by saying $\Sigma; \Gamma \vdash^{k} C : \underline{\tau} \implies \underline{\tau'} :$ if $\Sigma; \Gamma, x : \underline{\tau} \vdash^{c} C[x] : \underline{\tau'} :$ for a fresh x. One key notion of terms being 'equal' is contextual equivalence, which is formalised in terms of termination. A ***terminal*** is a (closed) computation of the form return V or $\lambda x^{\sigma}.M$. Termination means that a term reduces to a terminal: we write $(M, h) \Downarrow_{ter}$ if there exist N, h' such that $(M, h) \to^{*} (N, h')$ and N is a terminal.

Definition 1. Given computations $\Gamma \vdash^{c} M_1, M_2 : \underline{\tau}$, we define $\Gamma \vdash^{c} M_1 \lesssim_{ter}^{\mathrm{CBPV}} M_2$ to hold, when for all contexts $\vdash^{k} C : \underline{\tau} \implies F\sigma$, we have $(C[M_1], \emptyset) \Downarrow_{ter}$ implies $(C[M_2], \emptyset) \Downarrow_{ter}$. We write $\cong_{ter}^{\mathrm{CBPV}}$ for the equivalence induced by $\lesssim_{ter}^{\mathrm{CBPV}}$.

A standard result is that contexts considered for contextual approximation can be restricted to evaluation contexts after instantiating the free variables of computations to closed values (*closed instances of use*, CIU). We write $\Sigma, \Gamma' \vdash \gamma : \Gamma$ for substitutions γ such that, for any $(x, \sigma_x) \in \Gamma$,

$$\frac{}{\Sigma;\Gamma \vdash^v () : \mathrm{Unit}} \qquad \frac{n \in \{0,\cdots,\max\}}{\Sigma;\Gamma \vdash^v \widehat{n} : \mathrm{Int}} \qquad \frac{(x,\sigma) \in \Gamma}{\Sigma;\Gamma \vdash^v x : \sigma} \qquad \frac{\ell \in \Sigma}{\Sigma;\Gamma \vdash^v \ell : \mathrm{Ref}}$$

$$\frac{\Sigma;\Gamma \vdash^c M : \underline{\tau}}{\Sigma;\Gamma \vdash^v \mathrm{thunk}\ M : U\underline{\tau}} \qquad \frac{\Sigma;\Gamma \vdash^v V : \sigma}{\Sigma;\Gamma \vdash^c \mathrm{return}\ V : F\sigma} \qquad \frac{\Sigma;\Gamma \vdash^v V : U\underline{\tau}}{\Sigma;\Gamma \vdash^c \mathrm{force}\ V : \underline{\tau}}$$

$$\frac{\Sigma;\Gamma \vdash^v V : \sigma \qquad \Sigma;\Gamma, x : \sigma \vdash^c M : \underline{\tau}}{\Sigma;\Gamma \vdash^c \mathrm{let}\ x\ \mathrm{be}\ V.M : \underline{\tau}} \qquad \frac{\Sigma;\Gamma \vdash^v V : \mathrm{Int} \qquad \Sigma;\Gamma \vdash^c M_i : \underline{\tau}}{\Sigma;\Gamma \vdash^c \mathrm{case}\ V\ \mathrm{of}\ (M_i)_{i\in I} : \underline{\tau}}$$

$$\frac{\Sigma;\Gamma \vdash^c M : F\sigma \qquad \Sigma;\Gamma, x : \sigma \vdash^c N : \underline{\tau}}{\Sigma;\Gamma \vdash^c M\ \mathrm{to}\ x.N : \underline{\tau}} \qquad \frac{\Sigma;\Gamma, x : \sigma \vdash^c M : \underline{\tau}}{\Sigma;\Gamma \vdash^c \lambda x^\sigma.M : \sigma \to \underline{\tau}}$$

$$\frac{\Sigma;\Gamma \vdash^c M : \sigma \to \underline{\tau} \qquad \Sigma;\Gamma \vdash^v V : \sigma}{\Sigma;\Gamma \vdash^c MV : \underline{\tau}} \qquad \frac{\Sigma;\Gamma \vdash^v V : \mathrm{Int}}{\Sigma;\Gamma \vdash^c \mathrm{ref}\ V : F\mathrm{Ref}}$$

$$\frac{\Sigma;\Gamma \vdash^v V_{\mathrm{read}} : UF\mathrm{Int} \qquad \Sigma;\Gamma \vdash^v V_{\mathrm{write}} : U(\mathrm{Int} \to F\mathrm{Unit})}{\Sigma;\Gamma \vdash^v \mathrm{MkVar}\ V_{\mathrm{read}}\ V_{\mathrm{write}} : \mathrm{Ref}} \qquad \frac{\Sigma;\Gamma \vdash^v V : \mathrm{Ref}}{\Sigma;\Gamma \vdash^c !V : F\mathrm{Int}}$$

$$\frac{\Sigma;\Gamma \vdash^v V : \mathrm{Ref} \qquad \Sigma;\Gamma \vdash^v U : \mathrm{Int}}{\Sigma;\Gamma \vdash^c V := U : F\mathrm{Unit}} \qquad \frac{\Sigma;\Gamma \vdash^c M : F\mathrm{Int} \qquad \Sigma;\Gamma \vdash^c N : F\mathrm{Unit}}{\Sigma;\Gamma \vdash^c \mathrm{while}\ M\ \mathrm{do}\ N : F\mathrm{Unit}}$$

Figure 2: CBPV typing rules

$$\begin{array}{llll}
(K[\mathrm{let}\ x\ \mathrm{be}\ V.M], h) & \to & (K[M\{V/x\}], h) & (K[\mathrm{ref}\ V], h) \to (K[\mathrm{return}\ \ell], h \cdot [\ell \mapsto V]) \\
(K[(\lambda x^\sigma.M)V], h) & \to & (K[M\{V/x\}], h) & (K[!\ell], h) \to (K[\mathrm{return}\ h(\ell)], h) \\
(K[\mathrm{case}\ i\ \mathrm{of}\ (M_i)], h) & \to & (K[M_i], h) & (K[\ell := V], h) \to (K[\mathrm{return}\ ()], h[\ell \mapsto V]) \\
(K[\mathrm{force\ thunk}\ M], h) & \to & (K[M], h) & (K[!(\mathrm{MkVar}\ V_r\ V_w)], h) \to (K[\mathrm{force}\ V_r], h) \\
(K[\mathrm{return}\ V\ \mathrm{to}\ x.M], h) & \to & (K[M\{V/x\}], h) & (K[(\mathrm{MkVar}\ V_r\ V_w) := U], h) \to (K[(\mathrm{force}\ V_w)U], h) \\
\multicolumn{4}{l}{(K[\mathrm{while}\ M\ \mathrm{do}\ N], h) \to (K[M\ \mathrm{to}\ x.\mathrm{case}\ x\ \mathrm{of\ return}\ (), (N\ \mathrm{to}\ y.\mathrm{while}\ M\ \mathrm{do}\ N)_{i>0}], h) \qquad x, y\ \mathrm{are\ fresh}}
\end{array}$$

Figure 3: Operational semantics for CBPV

the value $\gamma(x)$ satisfies $\Sigma;\Gamma' \vdash^v \gamma(x) : \sigma_x$. Then $M\{\gamma\}$ stands for the outcome of applying γ to M.

Definition 2 (CIU Approximation)**.** Let $\Gamma \vdash^c M_1, M_2 : F\sigma$ be CBPV computations. Then $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$, when for all Σ, h, K, γ, such that $h : \Sigma$, $\Sigma \vdash^k K : \underline{\tau} \implies F\sigma$, and $\Sigma \vdash \gamma : \Gamma$, we have $(K[M_1\{\gamma\}], h) \Downarrow_{ter}$ implies $(K[M_2\{\gamma\}], h) \Downarrow_{ter}$.

We obtain a CIU Lemma establishing the sufficiency of CIU testing following the framework of [23].

Lemma 3 (CIU Lemma)**.** *We have* $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}} M_2$ *iff* $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$.

As with other results, the proof is relegated to the Appendix. We make the observation that the only contexts we really need to consider are those of type $F\sigma \implies F\sigma'$.

Lemma 4. *Let* $\Gamma \vdash^c M_1, M_2 : \sigma_1 \to \cdots \to \sigma_k \to F\sigma$. *Then* $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}} M_2$ *iff* $\Gamma, (x_1,\sigma_1), \cdots, (x_k,\sigma_k) \vdash^c M_1\, x_1 \cdots x_k \lesssim_{ter}^{\mathrm{CBPV}} M_2\, x_1 \cdots x_k$.

3 Labelled Transition System

We develop a labelled transition system (LTS) to capture the semantics of terms in the style of [24]. Its traces can be thought of as exchanges of moves between two players, representing the

$$
\begin{array}{lcl@{\qquad}lcl}
\mathbf{AVal}_\sigma(V) & \triangleq & \{(V, \emptyset)\} \quad \text{for } \sigma \in \{\mathrm{Unit}, \mathrm{Int}\} & \mathbf{ASeq}(F\sigma) & \triangleq & \{\epsilon\} \\
\mathbf{AVal}_{U_{\underline{\tau}}}(V) & \triangleq & \{(f, [f \mapsto V]) \mid f \in \mathrm{TNames}_{U_{\underline{\tau}}}\} & \mathbf{ASeq}(\sigma \to \underline{\tau}) & \triangleq & \{A\ s \mid s \in \mathbf{ASeq}(\underline{\tau}), \\
\mathbf{AVal}_{\mathrm{Ref}}\ (\mathrm{MkVar}\ V_1\ V_2) & \triangleq & \{(\mathrm{MkVar}\ f\ g, [f \mapsto V_1, g \mapsto V_2])\} & & & A : \sigma \text{ an abstract value}\} \\
\mathbf{AVal}_{\mathrm{Ref}}\ (\ell) & \triangleq & \{(\mathrm{MkVar}\ f\ g, [f \mapsto \mathrm{thunk}\ (!\ell),\ g \mapsto \mathrm{thunk}\ (\lambda x.\ell := x)])\} & & &
\end{array}
$$

Figure 4: Value decomposition into abstract values and substitutions, and generation of abstract value sequences

context and the term respectively. This way of modelling contextual interactions is often called *operational game semantics*.

3.1 Names and Abstract Values

In actions of this game, players pass (fresh) names to represent thunks passed between the two players. As these represent thunks, the names have a type $U_{\underline{\tau}}$. In keeping with the Reynolds approach of embedding references as a pair of thunks, we will 'decompose' references into separate thunk names for reading and writing respectively.

We will also use continuation names, to identify the question move being answered in an answer move. This is simply a technical convenience, as in the setting without control operators, all continuation names can be reconstructed due to the bracketing condition.

Definition 5. Let $\mathrm{TNames} = \biguplus_{\underline{\tau}} \mathrm{TNames}_{U_{\underline{\tau}}}$ be the set of ***thunk names***, partitioned into mutually disjoint countably infinite sets $\mathrm{TNames}_{U_{\underline{\tau}}}$. We use f, g to range over TNames, and write $f : U_{\underline{\tau}}$ for $f \in \mathrm{TNames}_{U_{\underline{\tau}}}$. Analogously, let $\mathrm{CNames} = \biguplus_{\sigma} \mathrm{CNames}_\sigma$ be the set of ***continuation names***. c ranges over CNames, and $c : \sigma$ denotes $c \in \mathrm{CNames}_\sigma$. We assume CNames, TNames are disjoint and let $\mathrm{Names} = \mathrm{TNames} \uplus \mathrm{CNames}$. Elements of Names will appear in structures throughout this work, and so $\nu(X)$ refers to the set of names used in some entity X.

Players will take actions which consist of a name applied to some (sequence of) values. To handle passing thunks, we will use ***abstract values***. These are values with occurrences of thunks replaced by names, so are generated by the grammar:

$$A \triangleq f \mid () \mid \widehat{n} \mid \mathrm{MkVar}\ A\ A$$

As names are intrinsically typed, abstract values can be typed in the obvious way, denoted $A : \sigma$. Given a value $V : \sigma$, $\mathbf{AVal}_\sigma(V)$ is the set of pairs (A, γ) such that A is an abstract value and $\gamma : \nu(A) \to \mathrm{Vals}$ is a substitution (defined in Figure 4). It is not quite the case that $A\{\gamma\} = V$ due to the treatment of locations. However, it is the case that $M\{V/x\} \Downarrow_{ter}$ iff $M\{A\{\gamma\}/x\} \Downarrow_{ter}$.

Remark 6. Note that $\cdot$ implicitly requires that function domains be disjoint, and $\uplus$ means the argument sets are disjoint.

Unlike in CBV, it does not suffice to consider passing only single abstract values. In CBPV, question actions occur when a thunk is forced (i.e. as we need a computation from the environment to occur), and answer actions correspond to a thunk returning a value (i.e reducing to return V). Operationally, a CBPV thunk, when forced, will consume all of its arguments (the sequence of values it is applied to) before returning a value to a consumer (a context of form $\bullet$ to $x.M$). As these arguments are values, they cannot change during the reduction of the computation before they are consumed. Thus, from the view of contextual equivalence, a forced thunk behaves as if all arguments are consumed immediately when the thunk is forced.

To handle this, actions include sequences of abstract values, denoted by $\overrightarrow{A}$, and referred to as abstract argument sequences. Given a sequence of values (an argument sequence, written $\overrightarrow{V}$), we can find a decomposition into an abstract argument sequence and a substitution by applying

AVal() to the values and combining the substitutions (ensuring names are disjoint). We abuse notation to write $\mathbf{AVal}(\overrightarrow{V})$ for the result.

For a given computation type, we can construct a sequence of abstract values using the function in Figure 4 (assuming name choices are fresh). We define the return type of a computation by $\mathbf{RType}(\sigma_1 \to \cdots \to \sigma_k \to F\sigma) \triangleq \sigma$.

3.2 Play

Our traces will consist of actions that have a polarity, either P (Player) or O (Opponent), depending on whether they are made by the term being tested (P) or the context (O). Names will be ***introduced*** either in a set N_O of names already introduced by O, a set N_P of names already introduced by P, or in values appearing in actions. Names are owned by the player whose action introduced the name (with names in N_O owned by O, N_P by P), and are referred to as O-names or P-names. The players take alternating actions, applying a name introduced by the other player to an abstract argument sequence. The types of actions are:

- ***Player Answer*** (PA) $\bar{c}(A)$, where $c : \sigma$ and $A : \sigma$. This corresponds to the term returning a value A to the consumer corresponding to continuation name c.
- ***Player Question*** (PQ) $\bar{f}(\overrightarrow{A}, c)$, where $f : U\underline{\tau}$, $\overrightarrow{A} \in \mathbf{ASeq}(\underline{\tau})$, $c : \mathbf{RType}(\underline{\tau})$. This corresponds to the term forcing the thunk named by f, passing $\overrightarrow{A}$ as arguments, and expecting the result with the consumer corresponding to continuation name c.
- ***Opponent Answer*** (OA) $c(A)$, $c : \sigma$ then $A : \sigma$. In this case, the environment is producing a value A to the term, which is acting as a consumer with continuation name c.
- ***Opponent Question*** (OQ) $f(\overrightarrow{A}, c)$, where $f : U\underline{\tau}$, $\overrightarrow{A} \in \mathbf{ASeq}(\underline{\tau})$, $c : \mathbf{RType}(\underline{\tau})$. This action corresponds to the environment forcing the thunk named by f from the term, passing $\overrightarrow{A}$ as arguments, and expecting the result with the consumer corresponding to continuation name c.

In what follows, $\mathbf{a}$ is used to range over actions. We refer to f in $\bar{f}(\overrightarrow{A}, c)$ and $f(\overrightarrow{A}, c)$, and to c in $\bar{c}(A)$ and $c(A)$ as the *head names* of $\mathbf{a}$.

Definition 7. Let $N_O, N_P \subseteq$ Names. An (N_O, N_P)-***trace*** is a sequence t of actions such that: the actions alternate between P and O actions; no name is introduced twice; names from N_O, N_P need no introduction; any action $\mathbf{a}$ must have the form $\bar{f}(\overrightarrow{A}, c)$, $f(\overrightarrow{A}, c)$, $\bar{c}(A)$ or $c(A)$, where the head name of $\mathbf{a}$ has been introduced by an earlier action $\mathbf{a}'$ of opposite polarity or (respectively) $f \in N_O$, $f \in N_P$, $c \in N_O$, $c \in N_P$.

Note that the first action in a $(N_O, \emptyset)$-trace must be by P.

Example 8. Let $N_O = \{f : U(\text{Ref} \to \text{Int} \to F\text{Unit}), c : U(UF\text{Int} \to F\text{Int})\}$. Then $\mathtt{t} = \bar{f}([r, w]\widehat{4}, c_1)\ w(\widehat{1}, c_2)\ \bar{c_2}(())\ c_1(())\ \bar{c}(g)\ g(h, c_3)\ \bar{h}(\epsilon, c_4)\ c_4(\widehat{2})\ \bar{c_3}(\widehat{3})$. is an $(N_O, \emptyset)$-trace.

3.3 Visibility, bracketing, and completeness

It will turn out that the traces required to capture our CBPV language will have special properties, which correspond to the fact that our language lacks higher-order store (visibility) and control-flow operations (bracketing). The identification of these properties occurred in game semantics [3, 25, 26], and they are enforced by our transition system in the style of [24].

Definition 9. Given a prefix t of a (N_O, N_P)-trace:

- an OQ-action $f(\overrightarrow{A}, c)$ occurring in t is said to be ***unanswered*** in t if there does not exist a PA-move of the form $\bar{c}(A')$ in t;

$$
\begin{aligned}
\mathsf{Vis}_O(\epsilon) &= N_P^T \\
\mathsf{Vis}_O(t\ \bar{c}(A)) &= (N_P^T) \cup \nu(A) && c \in N_O \\
\mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A'}, c)\ t'\ \bar{c}(A)) &= \mathsf{Vis}_O(t) \cup \nu(A) \\
\mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A}, c)) &= \nu(\overrightarrow{A}) \cup \{c\} && f \in N_O \\
\mathsf{Vis}_O(t\ f'(\overrightarrow{A'}, c')\ t'\ \bar{f}(\overrightarrow{A}, c)) &= \mathsf{Vis}_O(t) \cup \nu(\overrightarrow{A}) \cup \{c\} && f \in \nu(\overrightarrow{A'}) \\
\mathsf{Vis}_O(t\ c'(A')\ t'\ \bar{f}(\overrightarrow{A}, c)) &= \mathsf{Vis}_O(t) \cup \nu(\overrightarrow{A}) \cup \{c\} && f \in \nu(A'))
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Top}_O(\epsilon) &= N_P^C \\
\mathsf{Top}_O(t\ \bar{c}(A)) &= N_P^C && c \in N_O \\
\mathsf{Top}_O(t\ f(\overrightarrow{A'}, c)\ t'\ \bar{c}(A)) &= \mathsf{Top}_O(t) \\
\mathsf{Top}_O(t\ \bar{f}(\overrightarrow{A}, c)) &= \{c\}
\end{aligned}
$$

Figure 5: O-visible names $\mathsf{Vis}_O(t)$ and top continuation name $\mathsf{Top}_O(t)$ for (N_O, N_P)-trace t, where $N_P^T = N_P \cap \mathrm{TNames}$ and $N_P^C = N_P \cap \mathrm{CNames}$.

- a PQ-action $\bar{f}(\overrightarrow{A}, c)$ occurring in t is said to be ***unanswered*** in t if there does not exist an OA-move of the form $c(A')$ in t.

To define the O-visibility and O-bracketing constraints, we will define a set $\mathsf{Vis}_O(t)$ of ***O-visible names*** and the ***top continuation name*** $\mathsf{Top}_O(t)$ in Figure 5. The function $\mathsf{Top}_O(t)$ essentially identifies the latest unanswered PQ-action, and returns the continuation name introduced then. $\mathsf{Vis}_O(t)$ is defined by tracing the head name of a move to its introduction, analogously to the game semantic notion of *view* [3].

Definition 10. Let t be a (N_O, N_P)-trace. t is ***O-visible*** if, for any prefix $t'\ f'(\overrightarrow{A'}, c')$ of t, we have $f' \in \mathsf{Vis}_O(t')$. t is ***O-bracketed*** if, for any prefix $t'\ c'(A)$ of t (i.e. any prefix ending with an O-answer), we have $c' \in \mathsf{Top}_O(t')$.

We now introduce some useful notation. Given a (N_O, N_P)-trace t, we write $t^\perp$ for the (N_P, N_O)-trace obtained by changing the polarity of each name: $f(\overrightarrow{A}, c')$ becomes $\bar{f}(\overrightarrow{A}, c')$ (and vice versa) and $c(A)$ becomes $\bar{c}(A)$ (and vice versa). Using this we can provide the dual notions to O-visibility and O-bracketing. We say an (N_O, N_P)-trace t is P-visible if $t^\perp$ is O-visible (and define $\mathsf{Vis}_P(t) = \mathsf{Vis}_O(t^\perp)$), and t is P-bracketed if $t^\perp$ is O-bracketed (and define $\mathsf{Top}_P(t) = \mathsf{Top}_O(t^\perp)$).

To capture termination, it will suffice to reason about complete traces [24].

Definition 11. An O-bracketed $(N_O, \emptyset)$-trace t starting with a P action is ***complete*** if $\mathsf{Top}_O(t) = \emptyset$. A P-bracketed $(\{\circ\}, N_P)$-trace t starting with an O action is ***complete*** if $\mathsf{Top}_P(t) = \{\circ\}$.

Example 12. The trace `t` from Example 8 is an O-bracketed, P-bracketed, O-visible, P-visible, complete trace.

3.4 Transition System

Using the above, we can define an LTS, called $\mathcal{L}_{\mathrm{CBPV}}$, which will generate the set of traces corresponding to a term. $\mathcal{L}_{\mathrm{CBPV}}$ will contain terms built from CBPV syntax, extended with all thunk names as values (with the obvious typing rule), with $\to$ behaving accordingly. $\mathcal{L}_{\mathrm{CBPV}}$ will contain configurations of the form $(\mathbf{S}, S)$, where $\mathbf{S}$ is a *state* and S a *stack*. There are two types of states in $\mathcal{L}_{\mathrm{CBPV}}$ (and so two kinds of configurations): $\langle \gamma, \phi, h, H, Fn \rangle$ (*passive*, O to play) and $\langle M, c, \gamma, \phi, h, H \rangle$ (*active*, internal or P to play). In both, ϕ contains all names introduced so far by both players and h is the current heap. γ is an *environment* mapping thunk names introduced by P to thunks. In an active configuration, M is the *term* component, which captures the current behaviour of P, and c is the continuation name to produce the result to. The stack is used to enforce O-bracketing. It consists of elements of the form $(c_P, (K, c_O))$ where c_P is a continuation P-name, K is an evaluation context in which to use the value produced to c_P, and c_O is an continuation O-name to return the result of K to. Fn represents the set of thunk P-names currently available to O (so after generating a prefix t, Fn contains thunk names from $\mathsf{Vis}_O(t)$), and H contains historical information about availability. This is used to enforce O-visibility, by

maintaining the property that for a thunk O-name f, $H(f)$ contains the thunk P-names available to O when f was introduced. P-visibility is a consequence of the behaviour of the term.

We can now present the $\mathcal{L}_{\text{CBPV}}$ transition rules in Figure 6. These are presented as labelled transitions between states, with $\mathbf{a}/m$ denoting pushing m to the stack when producing $\mathbf{a}$, and $\mathbf{a}, m$ denoting popping m when producing $\mathbf{a}$.

$$
\begin{array}{ll|lll}
(P\tau) & \langle M, c, \gamma, \phi, h, H\rangle & \xrightarrow{\tau} & \langle N, c, \gamma, \phi, h', H\rangle \\
 & \text{when } (M,h) \to (N,h') & & \\
(PA) & \langle \text{return } V, c, \gamma, \phi, h, H\rangle & \xrightarrow{\bar{c}(A)} & \langle \gamma\cdot\gamma', \phi \uplus \nu(A), h, H, H(c) \uplus \nu(A)\rangle \\
 & \text{when } c:\sigma,\ (A,\gamma') \in \mathbf{AVal}_\sigma(V) & & \\
(PQ) & \langle K[(\text{force } f)\overrightarrow{V}], c, \gamma, \phi, h, H\rangle & \xrightarrow{\bar{f}(\overrightarrow{A},c')/(c',(K,c))} & \langle \gamma\cdot\gamma', \phi \uplus \phi', h, H, H(f) \uplus \nu(\overrightarrow{A})\rangle \\
 & \text{when } f : U\underline{\tau},\ (\overrightarrow{A},\gamma') \in \mathbf{AVal}(\overrightarrow{V}),\ \sigma = \mathbf{RType}(\underline{\tau}),\ c':\sigma \text{ and } \phi' = \nu(\overrightarrow{A}) \uplus \{c'\} & & \\
(OA) & \langle \gamma, \phi, h, H, Fn\rangle & \xrightarrow{c(A),(c,(K,c'))} & \langle K[\text{return } A], c', \gamma, \phi \uplus \nu(A), h, H\cdot[\nu(A) \mapsto Fn]\rangle \\
 & \text{when } c:\sigma,\ A:\sigma & & \\
(OQ) & \langle \gamma, \phi, h, H, Fn\rangle & \xrightarrow{f(\overrightarrow{A},c)} & \langle (\text{force } V)\overrightarrow{A}, c, \gamma, \phi \uplus \phi', h, H\cdot[\phi' \mapsto Fn]\rangle \\
 & \text{when } f \in Fn,\ f : U\underline{\tau},\ \overrightarrow{A} \in \mathbf{ASeq}(\underline{\tau}),\ \sigma = \mathbf{RType}(\underline{\tau}),\ c:\sigma,\ \gamma(f) = V \text{ and } \phi' = \nu(\overrightarrow{A}) \uplus \{c\} & &
\end{array}
$$

Given $N \subseteq \text{Names}$, $[N \mapsto \mathcal{V}]$ stands for the map $[n \mapsto \mathcal{V} \mid n \in N]$.

Figure 6: $\mathcal{L}_{\text{CBPV}}$ transition rules

Let $\Gamma \vdash^c M : F\sigma$ be a CBPV computation such that $\Gamma = \{x_1 : \sigma_1, \cdots, x_k : \sigma_k\}$. A Γ-***assignment*** ρ is a map from $\{x_1, \cdots, x_k\}$ to the set of abstract values such that, for all $1 \leq i \neq j \leq k$, we have $\rho(x_i) : \sigma_i$ and $\nu(\rho(x_i)) \cap \nu(\rho(x_j)) = \emptyset$. ρ simply creates a supply of names corresponding to the context. Let $c : \sigma$ and $N_O = \nu(\rho) \cup \{c\}$. Then the active initial configuration $\mathsf{C}_M^{\rho,c}$ is defined to be

$$(\langle M\{\rho\}, c, \emptyset, N_O, \emptyset, [N_O \mapsto \emptyset]\rangle, \bot)$$

Definition 13. Given two configurations $\mathbf{C}, \mathbf{C}'$, we write $\mathbf{C} \overset{\mathbf{a}}{\Rightarrow} \mathbf{C}'$ if $\mathbf{C} \xrightarrow{\tau}{}^* \mathbf{C}'' \xrightarrow{\mathbf{a}} \mathbf{C}'$, with $\xrightarrow{\tau}{}^*$ representing multiple (possibly none) τ-actions. This notation is extended to sequences of actions: given $\mathtt{t} = \mathbf{a}_1 \ldots \mathbf{a}_n$, we write $\mathbf{C} \overset{\mathtt{t}}{\Rightarrow} \mathbf{C}'$, if there exist $\mathbf{C}_1, \ldots, \mathbf{C}_{n-1}$ such that $\mathbf{C} \overset{\mathbf{a}_1}{\Longrightarrow} \mathbf{C}_1 \cdots \mathbf{C}_{n-1} \overset{\mathbf{a}_n}{\Longrightarrow} \mathbf{C}'$. We define $\mathbf{Tr}_{\text{CBPV}}(\mathbf{C}) = \{\mathtt{t} \mid \text{there exists } \mathbf{C}' \text{ such that } \mathbf{C} \overset{\mathtt{t}}{\Rightarrow} \mathbf{C}'\}$.

Remark 14. Due to the freedom of name choice, note that $\mathbf{Tr}_{\text{CBPV}}(\mathbf{C})$ is closed under type-preserving renamings that preserve names from $\mathbf{C}$.

Definition 15. A ***path*** in an LTS from a configuration $\mathbf{C}$ to $\mathbf{C}'$ is a sequence of transitions and intermediate configurations by which $\mathbf{C}'$ can be reached from $\mathbf{C}$ when viewing the LTS as a directed graph.

Traces of $\mathcal{L}_{\text{CBPV}}$ satisfy the following property.

Lemma 16. *Given $\Gamma \vdash^c M : F\sigma$ and a Γ-assignment ρ, for any $t \in \mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c})$, we have t is O-visible, O-bracketed, P-visible and P-bracketed.*

Definition 17. The ***trace semantics*** of a CBPV computation $\Gamma \vdash^c M : F\sigma$ is defined to be $\mathbf{Tr}_{\text{CBPV}}(\Gamma \vdash^c M : F\sigma) \triangleq \{((\rho, c), t) \mid \rho \text{ is a } \Gamma\text{-assignment}, c : \sigma, t \in \mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c}), t \text{ is complete}\}$.

Example 18. Let $\Gamma = \{f : U(\text{Ref} \to \text{Int} \to F\text{Unit})\}$, $\sigma = U(UF\text{Int} \to F\text{Int})$, and

$$M \quad = \quad \text{ref}\,\widehat{0} \quad \text{to} \quad x.(\text{force} \quad f)\,x\,\widehat{4}; \text{return} \quad \text{thunk} \quad (\lambda h.\text{force} \quad h \quad \text{to} \quad y.!x \quad \text{to} \quad z.y \; + \; z)$$

If $\rho = [f \mapsto f]$ and $\mathtt{t}$ is the trace in Example 8, then $((\rho, c), \mathtt{t}) \in \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M : F\sigma)$. The full derivation of this trace is presented below. Let $N_O = \nu(\rho) \cup \{c\}$ and $H_0 = [N_O \mapsto \emptyset]$

$\mathsf{C}_M^{\rho,c}$ $\xrightarrow{\tau}{}^*$ $(\langle M_0, c, \emptyset, N_O, [\ell \mapsto \widehat{0}], H_0\rangle, \bot)$
where $M_0 = (\text{force } f)\, \ell\, \widehat{4}; \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!\ell \text{ to } z.y + z)$

$\xrightarrow{\bar{f}([r,w]\,\widehat{4}, c_1)}$ $(\langle \gamma_0, \phi_0, \{r, w\}, [\ell \mapsto \widehat{0}], H_0\rangle, (c_1, (K_1, c)) : \bot)$ where $\phi_0 = N_O \cup \{r, w, c_1\}$
$\gamma_0 = [r \mapsto \text{thunk } (!\ell),\ w \mapsto \text{thunk } (\lambda x.\ell := x)]$
and $K_1 = \bullet; \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!\ell \text{ to } z.y + z)$

$\xrightarrow{w(\widehat{1}, c_2)}$ $(\langle(\text{force thunk } (\lambda x.\ell := x))\widehat{1}, c_2, \gamma_0, \phi_1, [\ell \mapsto \widehat{0}], H_1\rangle, (c_1, (K_1, c)) : \bot)$
where $H_1 = H_0 \cdot [c_2 \mapsto \{r, w\}]$ and $\phi_1 = \phi_0 \cup \{c_2\}$

$\xrightarrow{\tau}{}^*$ $(\langle \text{return } (), c_2, \gamma_0, \phi_1, h_1, H_1\rangle, (c_1, (K, c)) : \bot)$ where $h_1 = [\ell \mapsto \widehat{1}]$

$\xrightarrow{\bar{c_2}(())}$ $(\langle \gamma_0, \phi_1, h_1, H_1, \{r, w\}\rangle, (c_1, (K_1, c)) : \bot)$

$\xrightarrow{c_1(())}$ $(\langle K_1[\text{return } ()], c, \gamma_0, \phi_1, h_1, H_1\rangle, \bot)$

$\xrightarrow{\tau}{}^*$ $(\langle \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!x \text{ to } z.y + z), c, \gamma_0, \phi_1, h_1, H_1\rangle, \bot)$

$\xrightarrow{\bar{c}(g)}$ $(\langle \gamma_1, \phi_2, h_1, H_1, \{g\}\rangle, \bot)$
where $\gamma_1 = \gamma_0 \cdot [g \mapsto \text{thunk } (\lambda h.\text{force } h \text{ to } y.!\ell \text{ to } z.y + z)]$ and $\phi_2 = \phi_1 \cup \{g\}$

$\xrightarrow{g(h, c_3)}$ $(\langle(\text{force thunk } (\lambda h.\text{force } h \text{ to } y.!x \text{ to } z.y + z))h, c_3, \gamma_1, \phi_3, h_1, H_2\rangle, \bot)$
where $H_2 = H_1 \cdot [h, c_3 \mapsto \{g\}]$ and $\phi_3 = \phi_2 \cup \{h, c_3\}$

$\xrightarrow{\tau}{}^*$ $(\langle \text{force } h \text{ to } y.!x \text{ to } z.y + z, c_3, \gamma_1, \phi_3, h_1, H_2\rangle, \bot)$

$\xrightarrow{\bar{h}(\epsilon, c_4)}$ $(\langle \gamma_1, \phi_4, h_1, H_2, \{g\}\rangle, (c_4, (K_2, c_3)) : \bot)$
where $K_2 = \bullet \text{ to } y.!\ell \text{ to } z.y + z$ and $\phi_4 = \phi_3 \cup \{c_4\}$

$\xrightarrow{c_4(\widehat{2})}$ $(\langle K_2[\text{return } \widehat{2}], c_3, \gamma_1, \phi_4, h_1, H_2\rangle, \bot)$

$\xrightarrow{\tau}{}^*$ $(\langle \text{return } \widehat{3}, c_3, \gamma_1, \phi_4, h_1, H_2\rangle, \bot)$

$\xrightarrow{\bar{c_3}(\widehat{3})}$ $(\langle \gamma_1, \phi_4, h_1, H_2, \{g\}\rangle, \bot)$

4 Full Abstraction

To establish the soundness of this model (trace inclusion implies contextual inclusion), we will wish to reason about a computation in a specific context. Let $\Gamma \vdash^c M : F\sigma$. Using the CIU lemma, we will consider testing using a heap $h : \Sigma$, evaluation context $\vdash^k K : F\sigma \implies F\sigma'$ and a substitution $\gamma : \Gamma$. Let us fix a continuation name $\circ : \sigma'$, which we use to determine when a context has returned.

Next we define the set $\mathbf{AVal}_\Gamma(\gamma)$ of all disjoint decompositions of values from γ into abstract values and the corresponding matchings by

$$\mathbf{AVal}_\Gamma(\gamma) = \{(\vec{A_i}, \vec{\gamma_i}) \mid 1 \le i \le k, (A_i, \gamma_i) \in \mathbf{AVal}_{\sigma_i}(\gamma(x_i)), \\ \nu(A_1), \cdots, \nu(A_k) \text{ mutually disjoint}\}$$

where $\Gamma = \{x_1 : \sigma_1, \cdots, x_k : \sigma_k\}$, $\vec{A_i}$ stands for $(A_1, \cdots, A_k)$, and $\vec{\gamma_i}$ for $(\gamma_1, \cdots, \gamma_k)$.

Definition 19 (Context configuration)**.** Given Σ, $h : \Sigma$, $\Sigma \vdash^c K : F\sigma \implies F\sigma'$, $\Sigma \vdash^c \gamma : \Gamma$, $(\vec{A_i}, \vec{\gamma_i}) \in \mathbf{AVal}_\Gamma(\gamma)$ and $c : \sigma$ $(c \notin \circ)$, the corresponding configuration $\mathsf{C}^{\vec{\gamma_i}, c}_{h,K,\gamma}$ is defined by

$$\mathsf{C}^{\vec{\gamma_i}, c}_{h,K,\gamma} = (\langle \biguplus_{i=1}^{k} \gamma_i,\ \phi' \uplus \{c\},\ h,\ [\circ \mapsto \emptyset],\ \phi'\rangle,\ (c, (K, \circ)) : \bot)$$

where $\phi' = \biguplus_{i=1}^{k} \nu(A_i)$

Intuitively, the names $\nu(A_i)$ correspond to thunks extracted from γ, whereas c corresponds to K. Note that traces in $\mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\vec{\gamma_i}, c}_{h,K,\gamma})$ will be $(\{\circ\}, \biguplus_{i=1}^{k} \nu(A_i) \uplus \{c\})$-traces.

For the next result, we introduce the following notation. Given $(\vec{A_i}, \vec{\gamma_i}) \in \mathbf{AVal}_\Gamma(\gamma)$, we define a Γ-assignment $\rho_{\vec{A_i}}$ by $\rho_{\vec{A_i}}(x_i) = A_i$. Note that $\nu(\rho_{\vec{A_i}}) = \biguplus_{i=1}^{k} \mathrm{dom}(\gamma_i)$. The key lemma we now need to prove will relate traces of a term to its ability to converge in a given evaluation context. From correctness, we can then obtain soundness.

Lemma 20 (Correctness)**.** *Let $\Gamma \vdash^c M : F\sigma$ be a* CBPV *computation, let Σ, h, K, γ be as above, $(\vec{A_i}, \vec{\gamma_i}) \in \mathbf{AVal}_\Gamma(\gamma)$, and $c : \sigma$ ($c \neq \circ$). Then $(K[M\{\gamma\}], h) \Downarrow_{ter}$ iff there exist t, A such that $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_M^{\rho_{\vec{A_i}}, c})$ and $t^\perp \, \bar{\circ}(A) \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}, c})$. Moreover, t satisfies $\nu(t) \cap \{\circ\} = \emptyset$.*

Theorem 21 (Soundness)**.** *For any* CBPV *computations $\Gamma \vdash^c M_1, M_2 : F\sigma$, $\mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_1) \subseteq \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_2)$ then $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$.*

For the opposite direction, we establish that any trace of a suitable shape corresponds to a context.

Lemma 22 (Definability)**.** *Suppose $\phi \subseteq$ TNames and t is an even-length O,P-visible, O,P-bracketed $(\{\circ\}, \phi \uplus \{c\})$-trace starting with an O-action, such that $t = t' \, \bar{\circ}(A)$ and t' is complete. There exists a passive configuration $\mathbf{C}$ such that $\mathbf{Tr}^{\mathrm{even}}(\mathbf{C})$ is the even-length prefixes of t (along with their renamings via permutations on* Names *that fix $\phi \uplus \{\circ\}$). Moreover, $\mathbf{C} = \langle \gamma, \phi \uplus \{c\}, h, [\circ \mapsto \emptyset], \phi \rangle, (c, (K, \circ)) : \perp)$ for some h, K, γ.*

Completeness follows from definability and correctness.

Theorem 23 (Completeness)**.** *For any* CBPV *computations $\Gamma \vdash^c M_1, M_2 : F\sigma$, if $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$ then $\mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_1) \subseteq \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_2)$.*

Using soundness (Theorem 21), completeness (Theorem 23), and CIU lemma (Lemma 3), we have the following corollary.

Corollary 24 (Full Abstraction)**.** *For any* CBPV *computations $\Gamma \vdash^c M_1, M_2 : F\sigma$, then $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}} M_2$ iff $\mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_1) \subseteq \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_2)$.*

5 From LTS to Automata (alphabet)

In this section, we demonstrate how $\mathcal{L}_{\mathrm{CBPV}}$ gives rise directly to an automaton, when considering terms drawn from a particular fragment of CBPV. Initially, we will look for a fragment of CBPV for which (a faithful representation of) a term's traces can be captured using a (deterministic) ***Visibly Pushdown Automaton*** (VPA) [27]. A VPA is a type of pushdown automata in which the action on the stack is determined by the input symbol. The alphabet is partitioned into call (push), return (pop), and internal (noop) symbols. This ensures that inclusion (and so equivalence) of deterministic VPA is decidable in polynomial time.

Formally, a ***pushdown alphabet*** is a triple $\langle \Sigma_c, \Sigma_r, \Sigma_{int} \rangle$ of disjoint, finite alphabets, and we write $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_{int}$. Informally, we push on symbols in Σ_c, pop on symbols in Σ_r, and ignore the stack on symbols in Σ_{int}.

Definition 25. A ***Visibly Pushdown Automaton (VPA)*** over $\langle \Sigma_c, \Sigma_r, \Sigma_{int} \rangle$ is a tuple $\mathcal{A} = (Q, Q_0, \Gamma, \delta, Q_F)$, where:

- Q is a finite set of states,
- $Q_0 \subseteq Q$ is a set of initial states,
- Γ is a finite stack alphabet with distinguished bottom-of-stack symbol $\perp$,
- $\delta \subseteq (Q \times \Sigma_c \times Q \times (\Gamma \backslash \{\perp\})) \cup (Q \times \Sigma_r \times \Gamma \times Q) \cup (Q \times \Sigma_{int} \times Q)$, and,
- $Q_F \subseteq Q$ is the set of final states.

A VPA is ***deterministic*** if Q_0 is a singleton and for any $q \in Q$:

- for any $a \in \Sigma_c$ there is at most one transition of the form $(q, a, q', \gamma) \in \delta$,
- for any $a \in \Sigma_{int}$ there is at most one transition of the form $(q, a, q') \in \delta$,
- for any (a, γ) with $a \in \Sigma_r$, $\gamma \in \Gamma$ there is at most one transition of the form $(q, a, \gamma, q') \in \delta$.

For a word $w = a_1\, a_2 \ldots a_k \in \Sigma^*$, a run of $\mathcal{A}$ on w is a sequence $\rho = (q_0, \sigma_0), \ldots (q_k, \sigma_k)$, where each $q_i \in Q$, each σ_i is a stack (from $(\Gamma \backslash \{\bot\})^* \bot$), $q_0 \in Q_0$, and $\sigma_0 = \bot$ satisfying for every $1 \leq i \leq k$:

- if $a_i \in \Sigma_c$, then for some $\gamma \in \Gamma$, $(q_i, a_i, q_{i+1}, \gamma) \in \delta$ and $\sigma_{i+1} = \gamma : \sigma_i$.
- if $a_i \in \Sigma_{int}$, then $(q_i, a_i, q_{i+1}) \in \delta$ and $\sigma_{i+1} = \sigma_i$.
- if $a_i \in \Sigma_r$, then for some $\gamma \in \Gamma$, $(q_i, a_i, \gamma, q_{i+1}) \in \delta$ and either $\gamma \neq \bot$ and $\sigma_i = \gamma : \sigma_{i+1}$ or $\gamma = \bot$ and $\sigma_i = \sigma_{i+1} = \bot$.

A run $(q_0, \sigma_0), \ldots (q_k, \sigma_k)$ is accepting if $q_k \in Q_F$, a word is accepted if it has an accepting run, and the ***language*** of a VPA $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A})$ of words it accepts.

VPAs are closed under intersection, complementation, concatenation and star [27], and have the desirable property that equivalence and inclusion are decidable in EXPTIME in general, and polynomial time for deterministic VPAs.

The VPA which we construct is deterministic, but it is convenient to consider silent ϵ-transitions (which do not appear in the resulting word), which can be treated like transitions on Σ_{int}. As long as an ϵ-transition $q \xrightarrow{\epsilon} q'$ is the only transition from q, we can eliminate it by 'merging' q and q' into a single state. In this sense ϵ-transitions can be 'squeezed' out. We can also use a weaker notion of acceptance, where we require that the last stack in an accepting run is $\bot$ (this is weaker in the sense that automata with such acceptance conditions can accept fewer languages than general VPA). We can easily convert a deterministic automaton over alphabet Σ with such an acceptance condition into a standard one by intersecting it with the two state automaton accepting every string over Σ which has at least as many symbols from Σ_r as from Σ_c.

In seeking to identify a fragment for which VPA's suffice, we need to ensure that the space of states and the alphabet of actions are finite. Primarily, this is an issue when it is not possible to bound the set of names visible to O, $\mathsf{Vis}_O(t)$ for a trace t generated by the LTS. As the configuration will require a map from (at least the visible) names to the corresponding thunks from P, not having a bound on visible names will also mean we cannot bound the size of this map. We will see how this consideration restricts the type of terms through examples.

Example 26. Let $N_O^1 = \{c : UFUF\mathrm{Int}\}$. Consider $(N_O^1, \emptyset)$-traces of the form $\bar{c}(g)\ g(\epsilon, c_1)\ \bar{c_1}(f_1)$ $\cdots\ g(\epsilon, c_n)\ \bar{c_n}(f_n)\ f_i(\epsilon, c')$. To capture them, one needs to generate arbitrarily many fresh names, because the last action could refer to any f_i. This issue arises whenever we permit P to provide to O a thunk which returns a thunk (g in this case), as O can then obtain arbitrarily many, (potentially) distinct thunks.

Let $N_O^2 = \{c : U(U(UF\mathrm{Int} \to F\mathrm{Unit}) \to F\mathrm{Unit})\}$. Consider $(N_O^2, \emptyset)$-traces of the form $\bar{c}(g)$ $g(f_1, c_1)\ \bar{f_1}(h_1, c_1')\ \cdots\ g(f_n, c_n)\ \bar{f_n}(h_n, c_n')\ h_i(\epsilon, c')$. Here we can see that the same issue arises when we allow an argument passed by O (f_i) to itself receive an argument thunk (h_i) from P.

The fragment defined below is designed precisely to circumvent the problems identified above.

Definition 27. A CBPV computation $\Gamma \vdash^c M : F\sigma^P$ is in the ***P-thunk-restricted*** (PTR) fragment when all types in Γ can be generated by σ^2 in the grammar below.

$$
\begin{array}{llll}
\sigma^2 & \triangleq & \sigma^1 \mid U\underline{\tau}^2 & \qquad \sigma^P \triangleq \sigma^0 \mid \mathrm{Ref} \mid U\underline{\tau}^P \\
\underline{\tau}^2 & \triangleq & F\sigma^2 \mid \sigma^P \to \underline{\tau}^2 & \qquad \underline{\tau}^P \triangleq F\sigma^0 \mid \sigma^1 \to \underline{\tau}^P \\
\sigma^1 & \triangleq & \sigma^0 \mid \mathrm{Ref} \mid U\underline{\tau}^1 & \qquad \underline{\tau}^1 \triangleq F\sigma^1 \mid \sigma^0 \to \underline{\tau}^1 \\
\sigma^0 & \triangleq & \mathrm{Int} \mid \mathrm{Unit} &
\end{array}
$$

$$\begin{array}{llll}
\mathbf{BVals}^{\Delta}_{\sigma}(d) & \triangleq & \{V \mid V:\sigma\} & \text{where } \sigma \in \{\mathrm{Int}, \mathrm{Unit}\} \\
\mathbf{BVals}^{\Delta}_{U\underline{\tau}}(d) & \triangleq & \{f\} & \text{where } \mathrm{Suc_T}(d) = f \\
\mathbf{BVals}^{\Delta}_{\mathrm{Ref}}(d) & \triangleq & \{\{\!|f, g|\!\}\} & \text{where } \mathrm{Suc_T}(d) = (f, g)
\end{array}$$

$$\begin{array}{llll}
\mathbf{BValSeq}^{\Delta}(f) & \triangleq & \{\epsilon\} & \text{where } f : F\sigma \\
\mathbf{BValSeq}^{\Delta}(f) & \triangleq & \{B_1 \cdots B_k \mid B_i \in \mathbf{BVals}^{\Delta}_{\sigma_i}((f,i))\} & \\
 & & & \text{where } f : \sigma_1 \to \cdots \to \sigma_k \to F\sigma
\end{array}$$

Figure 7: Definition of base abstract values for given base head names and $\Delta = (\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$

$$\begin{array}{llll}
\mathbf{Base}^{\Delta}_{t}(n) & \triangleq & n & \text{where } n \in N_O \\
\mathbf{Base}^{\Delta}_{t}(c) & \triangleq & \mathrm{Suc_C}(g) & \text{where } c \text{ is introduced in } f(A,c) \text{ or } \bar{f}(A,c) \text{ and } g = \mathbf{Base}^{\Delta}_{t}(f) \\
\mathbf{Base}^{\Delta}_{t}(f) & \triangleq & g & \text{where } f \text{ is introduced in } c(A) \text{ or } \bar{c}(A) \text{ with } c:\sigma,\ c' = \mathbf{Base}^{\Delta}_{t}(c) \\
 & & & \{B\} = \mathbf{BVals}^{\Delta}_{\sigma}(c) \text{ and } g = \mathbf{Match}(A, B, f) \\
\mathbf{Base}^{\Delta}_{t}(f) & \triangleq & g & \text{where } f \text{ is introduced in } f'(\overrightarrow{A}, c) \text{ or } \bar{f}'(\overrightarrow{A}, c) \text{ with } g' = \mathbf{Base}^{\Delta}_{t}(f'), \\
 & & & \overrightarrow{B} \in \mathbf{BValSeq}^{\Delta}(g') \text{ and } g = \mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, f) \\
\mathbf{Marked}(t) & \triangleq & \{t' \mid \mathbf{a} \text{ is an O-action in } t,\ f \text{ is introduced in } \mathbf{a}, \text{ and } t' \text{ is } t \text{ with } f \text{ replaced by } \hat{f}\} & \\
\mathbf{Rename}^{\Delta}(t) & \triangleq & \mathbf{Base}^{\Delta}(t) \cup \bigcup_{t' \in \mathbf{Marked}(t)} \mathbf{Base}^{\Delta}(t') &
\end{array}$$

Figure 8: The function $\mathbf{Base}^{\Delta}_{t}()$ which converting names appearing in t to base names from $\Delta = (\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$

Remark 28. Note that all thunk P-names in a trace generated by a computation in the PTR-fragment have the type $U\underline{\tau}^P$.

Remark 29. An alternative way to characterise the the PTR-fragment is by polarising the occurrences of U, which correspond to question actions. If one writes U^+ for occurrences of U that produce O-questions, and U^- for those producing P-questions, the problematic types in Example 26 are then $U^+FU^+F\mathrm{Int}$ and $U^+(U^-(U^+F\mathrm{Int} \to F\mathrm{Unit}) \to F\mathrm{Unit})$, both of which contain nested occurrences of U^+. The PTR-fragment is then obtained by forbidding nested occurrences of U^+, while allowing nested occurrences of U^-.

Definition 30. A $(N_O, \emptyset)$-trace is a PTR-***trace*** when it is O- and P-bracketed, O- and P-visible, and it starts with a P-action with $\{c\} = N_O \cap \mathrm{CNames}$ where $c : \sigma^P$, and for $f \in N_O \cap \mathrm{TNames}$, $f : \sigma^2$, where σ^P, σ^2 are as defined in Definition 27.

Observe that these are exactly the traces which $\mathcal{L}_{\mathrm{CBPV}}$ generates on PTR computations.

To provide a representation of traces without arbitrarily many fresh names, we develop the notion of a ***name scheme***. The idea is to associate each thunk type appearing in a typing judgment with a fixed name. Readers familiar with game semantics will find the definition similar to that of an arena, where thunk names and continuation names indicate the position of questions and answers respectively, and $\mathrm{Suc_T}, \mathrm{Suc_C}$ correspond to the enabling relation.

Definition 31. A $(\Gamma, F\sigma)$-***name scheme*** is a tuple $(\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$ such that ρ is a Γ-assignment, $c_0 : \sigma$, and $\mathrm{TB} \subseteq \mathrm{TNames}$ and $\mathrm{CB} \subseteq \mathrm{CNames}$ are the smallest sets such that $\nu(\rho) \subseteq \mathrm{TB}$, $c_0 \in \mathrm{CB}$ and the conditions listed below are satisfied. We set $\mathrm{TB}_{U\underline{\tau}} \triangleq \mathrm{TB} \cap \mathrm{TNames}_{U\underline{\tau}}$ and $\mathrm{CB}_{\sigma} \triangleq \mathrm{CB} \cap \mathrm{CNames}_{\sigma}$.

- $\mathrm{Suc_T}$ is the least partial function from $(\mathrm{TB} \times \mathbb{N}) \uplus \mathrm{CB}$ to $\mathrm{TB} \cup (\mathrm{TB} \times \mathrm{TB})$ such that: if $c \in \mathrm{CB}_{U\underline{\tau}}$ then $\mathrm{Suc_T}(c) \in \mathrm{TB}_{U\underline{\tau}}$; if $c \in \mathrm{CB}_{\mathrm{Ref}}$ then $\mathrm{Suc_T}(c) \in \mathrm{TB}_{UF\mathrm{Int}} \times \mathrm{TB}_{U(\mathrm{Int} \to F\mathrm{Unit})}$; if $f \in \mathrm{TB}_{U(\sigma_1 \to \cdots \to \sigma_k \to FU\sigma')}$ and $1 \le i \le k$ then $\mathrm{Suc_T}(f,i) \in \mathrm{TB}_{U\underline{\tau_i}}$ for $\sigma_i = U\underline{\tau_i}$ and $\mathrm{Suc_T}(f,i) \in \mathrm{TB}_{UF\mathrm{Int}} \times \mathrm{TB}_{U(\mathrm{Int} \to F\mathrm{Unit})}$ for $\sigma_i = \mathrm{Ref}$.
- $\mathrm{Suc_C} : \mathrm{TB} \to \mathrm{CB}$ is a function such that if $f \in \mathrm{TB}_{U\underline{\tau}}$ then $\mathrm{Suc_C}(f) \in \mathrm{CB}_{\mathbf{RType}(\underline{\tau})}$.

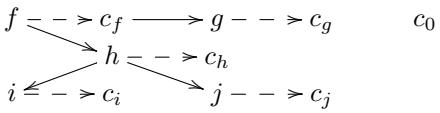
- $\nu(\mathrm{Suc_X}(d)) \cap \nu(\mathrm{Suc_X}(d')) = \emptyset$ for $d \neq d'$ and $X \in \{\mathrm{T}, \mathrm{C}\}$ (which implies injectivity) and $(\mathrm{img}(\mathrm{Suc_T}) \cup \mathrm{img}(\mathrm{Suc_C})) \cap (\nu(\rho) \cup \{c_0\}) = \emptyset$.

Elements of TB and CB will be referred to as ***base thunk names*** and ***base continuation names*** respectively. Abstract values containing base names only will be called ***base abstract values***. We shall write $\Delta_{\Gamma,F\sigma}$ for a $(\Gamma, F\sigma)$-name scheme, and Δ when we leave $(\Gamma, F\sigma)$ implicit.

Example 32. Consider $\underline{\tau} = U(\underline{\tau'}) \to FUF\mathrm{Unit}$, where $\underline{\tau'} = UF\mathrm{Int} \to UF\mathrm{Unit} \to F\mathrm{Int}$, $\Gamma = \{f : U\underline{\tau}\}$, $c_0 : \mathrm{Unit}$. For simplicity, assume $f \in \mathrm{TB}_{U\underline{\tau}}$ and $\rho(f) = f$. Then $\Delta_{\Gamma,F\mathrm{Unit}} = (\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$ is a name scheme, where

- $\mathrm{TB} = \{f : U\underline{\tau}, g : UF\mathrm{Unit}, h : U\underline{\tau'}, i : UF\mathrm{Int}, j : UF\mathrm{Unit}\}$;
- $\mathrm{CB} = \{c_0 : \mathrm{Unit}, c_f : UF\mathrm{Unit}, c_g : \mathrm{Unit}, c_h : \mathrm{Int}, c_i : \mathrm{Int}, c_j : \mathrm{Unit}\}$;
- $\mathrm{Suc_T}(f, 1) = h$, $\mathrm{Suc_T}(h, 1) = i$, $\mathrm{Suc_T}(h, 2) = j$, $\mathrm{Suc_T}(c_f) = g$; and
- $\mathrm{Suc_C}(f) = c_f$, $\mathrm{Suc_C}(g) = c_g$, $\mathrm{Suc_C}(h) = c_h$, $\mathrm{Suc_C}(i) = c_i$, $\mathrm{Suc_C}(j) = c_j$.

Showing $\mathrm{Suc_T}$ with solid arrows, and $\mathrm{Suc_C}$ with dashed, this can be visualised as a forest.



We can recast many definitions to use name schemes. For a start, we will redefine the notion of a trace so that it relies on base head names only, and base abstract values depend upon the name they are passed to via $\mathrm{Suc_T}$ or $\mathrm{Suc_C}$. To this end, in Figure 7 we define $\mathbf{BVals}^{\Delta}_{\sigma}(d)$ (for $d \in (\mathrm{TB} \times \mathbb{N}) \uplus \mathrm{CB}$) and $\mathbf{BValSeq}^{\Delta}(f)$ (for $f \in \mathrm{TB}$) to indicate the associated base abstract values and sequences thereof respectively. Note that they are determined uniquely up to numerical constants.

Definition 33. Let $\Delta = (\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$ be a name scheme. A Δ-***trace*** is a sequence t of actions such that: the actions alternate between P and O actions; names from $\nu(\rho) \cup c_0$ need no introduction; and the possible actions are:

- $\bar{f}(\overrightarrow{A}, c)$ where $f \in \mathrm{TB}$, $c = \mathrm{Suc_C}(f)$, $\overrightarrow{A} \in \mathbf{BValSeq}^{\Delta}(f)$ and f was introduced by an earlier O-action or $f \in \nu(\rho)$,
- $\bar{c}(A)$ where $c : \sigma \in \mathrm{CB}$, $A \in \mathbf{BVals}^{\Delta}_{\sigma}(c)$ and c was introduced by an earlier O-action or $c = c_0$,
- $f(\overrightarrow{A}, c)$ where $f \in \mathrm{TB}$, $c = \mathrm{Suc_C}(f)$, $\overrightarrow{A} \in \mathbf{BValSeq}^{\Delta}(f)$ and f was introduced by an earlier P-action.
- $c(A)$ where $c : \sigma \in \mathrm{CB}$, $A \in \mathbf{BVals}^{\Delta}_{\sigma}(c)$ and c was introduced by an earlier P-action.

Given the structure on base names, we can now introduce some terminology to distinguish the different classes of names.

Definition 34. Let $\Delta = (\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$. The names in $\nu(\rho)$ and c_0 are said to be ***initial***. A name $f \in \mathrm{TB}$ is said to be a ***level-n*** name if $n = 0$ and the name is initial, or $n = 1$ and $f \in \nu(\mathrm{Suc_T}(c_0))$, or there exists a level-$(n-1)$ name g and $j \in \mathbb{N}$ such that $f \in \nu(\mathrm{Suc_T}(g, j))$, or

- there exists a level-n name g and $c \in \mathrm{CB}$ such that $f \in \nu(\mathrm{Suc_T}(c))$ and $c = \mathrm{Suc_C}(g)$.

For level-n name f, the sequence of thunk names induced by the repeated use of the last rule will be called an ***introduction chain***, and the first name in the chain (i.e. introduced by the earlier rules) is called the ***originator***.

Example 35. In $\Delta_{\Gamma,F\mathrm{Unit}}$ from Example 32, f, g are level 0, h is level 1, and i, j are level 2. f is the originator of g. These align with the position of names in the visualisation.

Remark 36. Observe that, when used in a trace, the level 0 names will be O-names, so the level 1 names will be P-names, and the level 2 names will be O-names. In the PTR-fragment, all thunk names have level at most 2.

We shall say that a name scheme $\Delta = (\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$ ***agrees with*** $N_O \subseteq \mathrm{Names}$ if $N_O = \nu(\rho) \cup c_0$. Observe that, given a $(N_O, \emptyset)$-trace t and a Δ agreeing with N_O, we can construct a function $\mathbf{Base}^{\Delta}_t$ that maps names in $\nu(t)$ to base names by recursing on the introduction of names until we reach names in N_0. This is given formally in Figure 8, where $\mathbf{Match}(A, B, f)$ finds the base name in B in the same position as f in A.We can extend $\mathbf{Base}^{\Delta}_t$ to sets of names, abstract values, actions, and also to an entire trace, which we shall write simply as $\mathbf{Base}^{\Delta}(t)$. This is analogous to erasing justification pointers in game-semantic plays. We can now state a first useful result about base names. Intuitively, it means that, in PTR traces, base names suffice to distinguish O-visible thunk names, i.e. in such traces we can use base names to represent OQ actions faithfully.

Lemma 37. *Let t be a* PTR *$(N_O, \emptyset)$-trace ending in a P-action. Then $f, f' \in \mathsf{Vis}_O(t)$, we have $\mathbf{Base}^{\Delta}_t(f) \neq \mathbf{Base}^{\Delta}_t(f')$.*

Proof. This is a proof by induction on the length of t. The base case is when $t = t'\ \bar{c_0}(A)$ with $c_0 \in N_O$ or $t = t'\ \bar{g}(\overrightarrow{A}, c)$ with $g \in N_O$. Then $\mathsf{Vis}_O(t)$ is $\nu(A)$ or $\nu(\overrightarrow{A})$, and so all the base name are distinct. The inductive case $t = t'\ \bar{c}(A)$ is trivial, as $c : F\mathrm{Int}$ or $c : F\mathrm{Unit}$ so this reduces to an application of the I.H. The other inductive case is $t = t'\ \bar{g}(\overrightarrow{A}, c)$. Assume that there is some $f, f' \in \mathsf{Vis}_O(t)$ s.t. $\mathbf{Base}^{\Delta}_t(f) = \mathbf{Base}^{\Delta}_t(f')$. Then by the I.H, one of f, f' is introduced in $\overrightarrow{A}$. W.L.O.G, f was introduced in $\overrightarrow{A}$, and f' in some earlier move $\bar{g'}(\overrightarrow{A'}, c')$ with $\mathbf{Base}^{\Delta}_t(g) = \mathbf{Base}^{\Delta}_t(g')$. Observe that $\mathbf{Base}^{\Delta}_t(g)$ must be a level 0 name. Recall that the definition of $\mathsf{Vis}_O(t)$ involves a process of chasing names. We can therefore consider the sequence of O-names used as the head names for moves which introduce P-names in the definition of $\mathsf{Vis}_O(t)$. We will show that the base names in this sequence are exactly the introduction chain for $\mathbf{Base}^{\Delta}_t(g)$, from which it will follow that g' cannot be in this sequence (and so $f' \notin \mathsf{Vis}_O(t)$).

We prove this by induction on the length of the introduction chain. In the base case, $\mathbf{Base}^{\Delta}_t(g)$ is initial, so $g \in N_O$, and $\mathsf{Vis}_O(t) = \nu(\overrightarrow{A})$, so no other names in the sequence. In the inductive case, let $g \in \nu(A)$ s.t. $\mathsf{Vis}_O(t) = \mathsf{Vis}_O(s\ \mathbf{a}\ d(A)\ s'\ \bar{g}(\overrightarrow{A}, c)) = \nu(\overrightarrow{A}) \cup \mathsf{Vis}_O(s\ \mathbf{a})$. Consider what $\mathbf{a}$ is. If it is a PQ-action, then it must be on continuation name $d' \neq c_0$, so we have $\mathsf{Vis}_O(s\ \mathbf{a}) = \mathsf{Vis}_O(s_1\ h(\overrightarrow{A''}, d')\ \bar{d'}()) = \mathsf{Vis}_O(s_1)$. By repeating this, it suffices to consider only the case that $\mathbf{a}$ is a PQ-action, and so by the bracketing condition, $\mathbf{a} = \bar{h}(\overrightarrow{A''}, d)$. Then $\mathsf{Vis}_O(s\ \mathbf{a}) = \nu(\overrightarrow{A''}) \cup \mathsf{Vis}_O(s)$. We must have $\mathbf{Base}^{\Delta}_t(d) = \mathrm{Suc_C}(\mathbf{Base}^{\Delta}_t(h))$ and $\mathbf{Base}^{\Delta}_t(g) = \mathrm{Suc_T}(\mathbf{Base}^{\Delta}_t(d))$, so $\mathbf{Base}^{\Delta}_t(h)$ is earlier in the introduction chain of $\mathbf{Base}^{\Delta}_t(g)$, as required. Applying the I.H. to the introduction chain of $\mathbf{Base}^{\Delta}_t(h)$ completes this proof. □

It would be desirable if we could represent the traces of a PTR-computation using Δ-traces. However, it turns out that simply applying $\mathbf{Base}^{\Delta}$ to a trace loses information.

Example 38. Let $\Gamma = \{f : UFUF\mathrm{Unit}\}$ and $\Delta = (\{\ f : UFUF\mathrm{Unit},\ g : UF\mathrm{Unit}\ \}, \{c_0 : \mathrm{Unit},\ d : UF\mathrm{Unit},\ e : \mathrm{Unit}\}, \rho, c_0, [d \mapsto g], [f \mapsto d, g \mapsto e])$, where $\rho = [f \mapsto f]$. Consider the two computations $\Gamma \vdash^c M_1, M_2 : F\mathrm{Unit}$, where $M_i = \mathrm{force}\ f\ \mathrm{to}\ g_1.\mathrm{force}\ f\ \mathrm{to}\ g_2.\mathrm{force}\ g_i$. We have that the complete traces in $\mathbf{Tr}(\mathsf{C}^{\rho,c_0}_{M_i})$ all have the form

$$\mathtt{t}_i = \bar{f}(\epsilon, c_1)\ c_1(g_1)\ \bar{f}(\epsilon, c_2)\ c_2(g_2)\ \bar{g_i}(\epsilon, c_3)\ c_3(())\ \bar{c_0}(())$$

That is, the traces are distinguished by the use of either g_1 or g_2. However, we have $\mathbf{Base}^{\Delta}\mathbf{Tr}(\mathsf{C}^{\rho,c_0}_{M_i})$ as given in Figure 9. That is, we lose the distinction between g_1 and g_2, and so if we simply used Δ-traces to model terms, we would equate M_1 with M_2, which would mean losing the soundness property.

$$
\begin{array}{rcl}
\mathbf{Base}^{\Delta}(\mathbf{Tr}(\mathsf{C}_{M_i}^{\rho,c_0})) & = & \{\mathtt{t}\} \text{ where } \mathtt{t} = \bar{f}(\epsilon,d)\ d(g)\ \bar{f}(\epsilon,d)\ d(g)\ \bar{g}(\epsilon,e)\ e(())\ \bar{c_0}(()) \\
\mathbf{Rename}^{\Delta}(\mathbf{Tr}(\mathsf{C}_{M_1}^{\rho,c_0})) & = & \{\mathtt{t}, \bar{f}(\epsilon,d)\ d(\hat{g})\ \bar{f}(\epsilon,d)\ d(g)\ \bar{\hat{g}}(\epsilon,e)\ e(())\ \bar{c_0}(()), \bar{f}(\epsilon,d)\ d(g)\ \bar{f}(\epsilon,d)\ d(\hat{g})\ \bar{g}(\epsilon,e)\ e(())\ \bar{c_0}(())\} \\
\mathbf{Rename}^{\Delta}(\mathbf{Tr}(\mathsf{C}_{M_2}^{\rho,c_0})) & = & \{\mathtt{t}, \bar{f}(\epsilon,d)\ d(\hat{g})\ \bar{f}(\epsilon,d)\ d(g)\ \bar{g}(\epsilon,e)\ e(())\ \bar{c_0}(()), \bar{f}(\epsilon,d)\ d(g)\ \bar{f}(\epsilon,d)\ d(\hat{g})\ \bar{\hat{g}}(\epsilon,e)\ e(())\ \bar{c_0}(())\}
\end{array}
$$

Figure 9: Translation of traces for Examples 38 and 39

A natural solution to this would be to take inspiration from the way that $\mathtt{t}_i$ is presented in this example, and use traces in which base names can appear scripted by when they are introduced. We could also exploit Lemma 43 to allow us to reset these indices after a PA-action. However, the presence of while loops means that this is not viable. A while loop might cause a PQ-action to occur arbitrarily many times, seemingly requiring an unbounded number of indices. We could attempt to exploit the fact that the scopes in the language ensure that any name introduced during an iteration of a loop cannot escape that loop to reset the indices at the end of a loop. However, the end of a loop cannot be apparent in a trace. This makes it difficult to see how to equate the terms

$$
\begin{array}{rcl}
N_1 & = & (\text{force } f \text{ to } g.\text{force } g); \text{force } f \text{ to } g.\text{force } g \\
N_2 & = & \text{ref } 2 \text{ to } x.\text{while } !x \text{ do } ((\text{force } f \text{ to } g.\text{force } g); \\
 & & \qquad\qquad !x \text{ to } v.v-1 \text{ to } w.x := w\ )
\end{array}
$$

This is because, for the first, we would want to have traces like $\bar{f}(\epsilon,d_1)\ d_1(g_1)\ \bar{g_1}(\epsilon,e_1)\ e_1(())$ $\bar{f}(\epsilon,d_2)\ d_2(g_2)\ \bar{g_2}(\epsilon,e_2)\ e_2(())\ \bar{c_0}(())$, as it is not 'safe' to reset the index counter after the $e_1(())$, whereas for the second we would need to have traces like $\bar{f}(\epsilon,d_1)\ d_1(g_1)\ \bar{g_1}(\epsilon,e_1)\ e_1(())\ \bar{f}(\epsilon,d_1)$ $d_1(g_1)\ \bar{g_1}(\epsilon,e_1)\ e_1(())\ \bar{c_0}(())$, as we would be resetting the index at the end of every iteration.

This issue also appears in the work of Hopkins et al. [12], which they resolve by encoding a single P-pointer in each word their automata generate, and then use the fact that a set of words, each with one pointer, can be used to uniquely represent a full play. We adopt the same approach, and adapt it to our name-based setting.

The key to this is the notion of a ***marked name***, which we shall write as $\hat{f}$, where f is said to be the underlying name, which can be either from TNames or TBNames for some Δ, depending on context. We introduce f into our structures (traces, abstract values, etc.) by permitting a marked name wherever the underlying name can occur, and will refer to these also as marked (e.g. marked trace). In particular, for base abstract values, a marked name can appear in place of its underlying name in $\mathbf{BVals}^{\Delta}_{\sigma}(d)$. We also extend the functions $\mathbf{Base}^{\Delta}$ and $\mathbf{Base}^{\Delta}_{t}$ so that they preserve marks.

With this notion, we can now define an appropriate mapping of PTR-traces to a set of marked, Δ-traces as $\mathbf{Rename}^{\Delta}(t)$ in Figure 8. Observe that $\mathbf{Rename}^{\Delta}(t)$ consists of traces whose underlying names are all the same, and for each trace, at most one O-base name introduced during t has been marked. We lift $\mathbf{Rename}()$ to sets in the obvious way.

Example 39. Let Δ, M_1, M_2 be as in Example 38. Then we have $\mathbf{Rename}^{\Delta}(\mathbf{Tr}(\mathsf{C}_{M_i}^{\rho,c_0}))$ as shown in Figure 9.

$\mathbf{Rename}^{\Delta}()$ turns out to be sufficiently informative to provide a faithful representation of PTR-traces.

Lemma 40. *Suppose t_1, t_2 are* PTR *$(N_O, \emptyset)$-traces, Δ agrees with N_O and $\mathbf{Rename}^{\Delta}(t_1) = \mathbf{Rename}^{\Delta}(t_2)$. Then t_1 and t_2 are equal up to a permutation of names that preserves N_O.*

Proof. We will prove this by contradiction. Let t_1, t_2 be s.t. $\mathbf{Rename}^{\Delta}(t_1) = \mathbf{Rename}^{\Delta}(t_2)$ but are not equal up to permutations of names which preserve N_O. Observe that t_1 and t_2 must have the same sequence of moves, differing only in the head names. Consider the shortest (equal length) prefixes s_1 and s_2 of t_1 and t_2 which are not equal up to permutation of names. Apply a permutation to s_1 and s_2 so that they are equal, save for the last action. Let s be the common

$$
\begin{array}{lcl}
\mathbf{IVal}^{\Delta}_{\sigma}(d,V,\eta) & \triangleq & (V,\emptyset,\eta) \quad \text{for } \sigma \in \{\mathrm{Unit},\mathrm{Int}\} \\
\mathbf{IVal}^{\Delta}_{U_{\mathcal{T}}}(d,V,\eta) & \triangleq & (f^i,[f^i \mapsto V],\eta[f \mapsto i+1]) \text{ where } \mathrm{Suc}_{\mathrm{T}}(d)=f, \eta(f)=i \\
\mathbf{IVal}^{\Delta}_{\mathrm{Ref}}(d,\{\!|V_1,V_2|\!\},\eta) & \triangleq & (\{\!|f^{\eta(f)},g^{\eta(j)}|\!\},[f^{\eta(f)} \mapsto V_1,\, g^{\eta(g)} \mapsto V_2],\eta[f,g \mapsto \eta(f)+1,\eta(g)+1]) \text{ where } \mathrm{Suc}_{\mathrm{T}}(d)=(f,g) \\
\mathbf{IVal}^{\Delta}_{\mathrm{Ref}}(d,\ell,\eta) & \triangleq & (\{\!|f^i,g^j|\!\},[f^i \mapsto \text{thunk } (!\ell),\, g^j \mapsto \text{thunk } (\lambda x.\ell := x)],\eta[f,g \mapsto i+1,j+1]) \\
 & & \text{where } \mathrm{Suc}_{\mathrm{T}}(d)=(f,g), \eta(f)=i, \eta(g)=j \\
\mathbf{IVal}^{\Delta}(f,\overrightarrow{V},\eta_0) & \triangleq & (A_1 \cdots A_k, \gamma_1 \cdot \gamma_2 \cdots \gamma_k, \eta_k) \text{ where } f : U(\sigma_1 \to \cdots \to \sigma_k \to F\sigma), \overrightarrow{V} = V_1 \cdots V_k \\
 & & \text{and for } 1 \le i \le k, (A_i,\gamma_i,\eta_i) = \mathbf{IVal}^{\Delta}_{\sigma_i}((f,i),V_i,\eta_{i-1})
\end{array}
$$

$$
\begin{array}{lcll}
\mathbf{IVals}^{\Delta}_{\sigma}(d,\eta) & \triangleq & \{(V,\eta) \mid V : \sigma\} & \text{where } \sigma \in \{\mathrm{Int},\mathrm{Unit}\} \\
\mathbf{IVals}^{\Delta}_{U_{\mathcal{T}}}(d,\eta) & \triangleq & \{(f^i,\eta[f \mapsto i+1])\} & \text{where } \mathrm{Suc}_{\mathrm{T}}(d)=f,\ \eta(f)=i \\
\mathbf{IVals}^{\Delta}_{\mathrm{Ref}}(d,\eta) & \triangleq & \{(\mathrm{MkVar}\ f^i\ g^j,\eta[f,g \mapsto i+1,j+1])\} & \text{where } \mathrm{Suc}_{\mathrm{T}}(d)=(f,g), \eta(f)=i, \eta(g)=j \\
\mathbf{IValSeq}^{\Delta}(f,\eta_0) & \triangleq & \{(B_1 \cdots B_k,\eta_k) \mid (B_i,\eta_i) \in \mathbf{IVals}^{\Delta}_{\sigma_i}((f,i),\eta_{i-1})\} & \text{where } f : U(\sigma_1 \to \cdots \to \sigma_k \to F\sigma)
\end{array}
$$

Figure 10: Functions for decomposing values in indexed abstract values and maps, for $\Delta = (\mathrm{TB},\mathrm{CB},\rho,c_0,\mathrm{Suc}_{\mathrm{T}},\mathrm{Suc}_{\mathrm{C}})$

prefix, and observe the action they disagree on must have been a question, as the head name used in answer actions is determined by the bracketing conditions.

Now, by Lemma 37, this was not an OQ-action, as there are not distinct $f, f' \in \mathsf{Vis}_O(s)$ with $\mathbf{Base}^{\Delta}_s(f) = \mathbf{Base}^{\Delta}_s(f')$. So this last move was a PQ-action $\mathbf{a}_q$, so let f, f' be the two distinct head names with $\mathbf{Base}^{\Delta}_s(f) = g = \mathbf{Base}^{\Delta}_s(f')$. Let $\mathbf{a}_a$ be the action f is introduced in. Now, $\mathbf{Rename}^{\Delta}(t_1)$ will contain a marked Δ-trace with g marked at the action corresponding to $\mathbf{a}_a$, and in the head of the action corresponding to $\mathbf{a}_q$. But $\mathbf{Rename}^{\Delta}(t_2) = \mathbf{Rename}^{\Delta}(s_2)$ cannot contain this trace, as $\mathbf{Marked}(s_2)$ cannot contain a trace with both f in $\mathbf{a}_a$ marked and f' in $\mathbf{a}_q$ marked. Thus, we have a contradiction. □

Corollary 41. *For* PTR*-computations* $\Gamma \vdash^c M_1, M_2 : F\sigma$*, continuation name* $c : \sigma$*, a* Γ*-assignment* ρ *and* $\Delta_{\Gamma,c} = (\mathrm{TB},\mathrm{CB},\rho,c,\mathrm{Suc}_{\mathrm{T}},\mathrm{Suc}_{\mathrm{C}})$*, we have* $\mathbf{Tr}(\mathsf{C}^{M_1}_{\rho,c}) = \mathbf{Tr}(\mathsf{C}^{M_2}_{\rho,c})$ *iff* $\mathbf{Rename}^{\Delta_{\Gamma,c}}(\mathbf{Tr}(\mathsf{C}^{M_1}_{\rho,c})) = \mathbf{Rename}^{\Delta_{\Gamma,c}}(\mathbf{Tr}(\mathsf{C}^{M_2}_{\rho,c}))$.

6 From LTS to Automata (transitions)

Let $\Delta = (\mathrm{TB},\mathrm{CB},\rho,c_0,\mathrm{Suc}_{\mathrm{T}},\mathrm{Suc}_{\mathrm{C}})$ be a name scheme. We wish to arrive at an LTS $\mathcal{L}^{\Delta}_{\mathrm{PTR}}$ generating marked Δ-traces. In particular, the traces should be those arising from an application of $\mathbf{Rename}^{\Delta}(t)$, i.e. they need to include at most one introduction of a marked name, which must be an O-name. Although traces of $\mathcal{L}^{\Delta}_{\mathrm{PTR}}$ will rely on base names, the configurations will distinguish their occurrences via indexing. An ***indexed name*** has the form b^i, where $b \in \mathrm{TB} \cup \mathrm{CB}$, and $i \in \mathbb{N}$ is an ***index***. Indexed names will appear in the domain of components like γ and in terms, but never in traces. Indexed names can also be marked in the same way as others.

To handle the generation of new indices, our configurations will contain a function $\eta : \mathrm{TB} \cup \mathrm{CB} \to \mathbb{N}$ mapping each base name to the next available index. In Figure 10 we define new versions of functions analogous to $\mathbf{AVal}_{\sigma}(V)$, $\mathbf{BVals}^{\Delta}_{\sigma}(d)$, and $\mathbf{BValSeq}^{\Delta}(f)$ but taking an additional argument η. They generate abstract values with indexed names and an updated η'. Given an abstract value or sequence with indexed names, we write $\beta(A)$ to denote the same abstract value with indices removed (but preserving marks).

Note that a typical update to η will make the values grow. In order to keep them bounded, we will implement a recycling scheme for indices. In order to formulate it, we need to transfer the notion of level to thunk names in a trace. Let t be an N_O-trace such that Δ agrees with N_O. Then f in t is a level-n name if $\mathbf{Base}^{\Delta}_t(f)$ is a level-n name, and g is the originator of f if g can be reached from f by following the introduction of head names, and $\mathbf{Base}^{\Delta}_t(g)$ is the originator of $\mathbf{Base}^{\Delta}_t(f)$.

Example 42. In trace `t` in Example 8, f is a level-0 name, r, w, g are level 1, and h is level 2.

Our recycling scheme is inspired by the Lemmata below.

Lemma 43. *Let $c : \sigma^0$ be a continuation name (one which corresponds to returning a value of a basic type). Then, for any O/P-visible, and O/P-bracketed trace $s = t\ f(\overrightarrow{A}, c)\ t'\ \bar{c}(A')\ t''$, no names introduced in $f(\overrightarrow{A}, c)\ t'$ appear in $\mathsf{Vis}_O(s)$ (if s ends in a P-action) or $\mathsf{Vis}_P(s)$ (if s ends in an O-action).*

Lemma 44. *Let $s = t\ f(\overrightarrow{A}, c)\ t'\ \bar{g}(\overrightarrow{A'}, d)$ and $s' = s\ t''\ d(A)$ be* PTR *$(N_O, \emptyset)$-traces, where g is a level-2 name whose originator is introduced in $\overrightarrow{A}$. Let X be the names introduced in $f(\overrightarrow{A}, c)\ t'\ \bar{g}(\overrightarrow{A'}, d)$. Then if s'' is a proper prefix of s' at least as long as s, $\mathsf{Vis}_O(s'') \cap X = \emptyset$ (if s'' ends in a P-action) and $\mathsf{Vis}_P(s'') \cap X = \emptyset$ (if s'' ends in an O-action).*

To prove Lemma 44, we will need a helper Lemma.

Lemma 45. *Let $t\ \ g(\overrightarrow{A}, c)\ \ t'\ \ \bar{f}(\overrightarrow{A'}, c')$ be a* PTR*-trace, with f a level 2 name, and f' its originator, introduced in $\overrightarrow{A}$. Then $\mathsf{Vis}_O(t\ \ g(\overrightarrow{A}, c)\ \ t'\ \ \bar{f}(\overrightarrow{A'}, c')) = \mathsf{Vis}_O(t)$*

Proof of Lemma 44. We proceed by induction on the length on the length of s''.

- In the base case, the shortest s'' is simply s, which has odd-length. We can then appeal to Lemma 102 to obtain that $\mathsf{Vis}_O(s'') = \mathsf{Vis}_O(t)$, so cannot contain any names in X.
- In the inductive case, we proceed by cases on the last action in s''.
  - $s'' = s'''\ f'(\overrightarrow{A''}, c'')$. s''' has odd length, so the I.H gives that $f' \notin X$. Thus, f' is introduced in either t or in t''. In the first case, we have that $\mathsf{Vis}_P(s'') = \mathsf{Vis}_P(t_1) \cup \nu(\overrightarrow{A''})$ for some t_1 a prefix of t, and so contains no names in X. In the second case, we get that $\mathsf{Vis}_P(s'') = \mathsf{Vis}_P(s_1) \cup \nu(\overrightarrow{A''})$ where s_1 prefix of s'' at least as long as s, so the result hold by the I.H.
  - $s'' = s'''\ \ c''(A'')$. This is an even length trace. Then, due to the bracketing condition, we have that $c''(A'')$ is answering a question in t''. Thus we get that $\mathsf{Vis}_O(s'') = \mathsf{Vis}_O(s_1) \cup \nu(A'')$ where s_1 prefix of s'' at least as long as s, so the result hold by the I.H.
  - $s'' = s'''\ \bar{f}'(\overrightarrow{A''}, c'')$. s''' has even length, so the I.H gives that $f' \notin X$. Thus, f' is introduced in either t or in t''. In the first case, we have that $\mathsf{Vis}_O(s'') = \mathsf{Vis}_O(t_1) \cup \nu(\overrightarrow{A''})$ for some t_1 a prefix of t, and so contains no names in X. In the second case, we get that $\mathsf{Vis}_O(s'') = \mathsf{Vis}_O(s_1) \cup \nu(\overrightarrow{A''})$ where s_1 prefix of s'' at least as long as s, so the result hold by the I.H.
  - $s'' = s'''\ \ \bar{c''}(A'')$. This is an odd length trace. Then, due to the bracketing condition, we have that $\bar{c''}(A'')$ is answering a question in t''. Thus we get that $\mathsf{Vis}_P(s'') = \mathsf{Vis}_P(s_1) \cup \nu(A'')$ where s_1 prefix of s'' at least as long as s, so the result hold by the I.H.

□

Note that both Lemmata 43 and 44 state that certain names become unavailable. In the first case this deactivation is permanent after $\bar{c}(A')$, whereas in the second case it is temporary: it starts after a level-2 name is used in $\bar{g}(\overrightarrow{A'}, d)$ and ends after the corresponding $d(A)$. We will take advantage of the deactivation period to reuse the deactivated indices. In the first case, this will be done simply by resetting the relevant bounds. In the PTR fragment, all non-initial continuation names have type σ^0, so this recycling is actually widely applicable. In the second case, we will reset the parameters temporarily and, to be able to restore them, will push the information related to deactivated names on the stack (PQ). It can then be restored during the matching pop (OA).

$(K[\text{while } M \text{ do } N], h, i_h, \eta) \to_e (K[M \text{ to } x.\text{case } x \text{ of return } (), (N \text{ to } y.\text{end}(i_h, \eta).\text{while } M \text{ do } N)_{j>0}], h, i_h, \eta)$
$(K[\text{end}(i_h, \eta').\text{while } M \text{ do } N], h, j_h, \eta) \to_e (K[M \text{ to } x.\text{case } x \text{ of return } (), (N \text{ to } y.\text{end}(i_h, \eta).\text{while } M \text{ do } N)_j], h_{<i_h}, i_h, \eta')$

Figure 11: The modifications needed to produce reduction relation $\to_e$

Example 46. To better explain why this second scheme is necessary, recall Γ and name scheme $\Delta_{\Gamma,F\text{Unit}}$ from Example 32, and consider the computation $\Gamma \vdash^c (\text{force } f)(\text{thunk } \lambda i.\lambda j.(\text{force } i))$ to $g.\text{force } g : F\text{Unit}$. Some of the associated traces, written with base names and specific indices, have the following shape:

$$\bar{f^0}(h^0, c_f^0)\ h^0(i^0\, j^0, c_h^0)\ \bar{i^0}(\epsilon, c_i^0)\ h^0(i^1\, j^1, c_h^1)\ \bar{i^1}(\epsilon, c_i^1)\ \ldots.$$

What happens here is that, when P calls i^n, h^0 becomes visible to O. This allows O to call h^0 again with i^{n+1}. As this can repeat unboundedly many times, we must recycle the indices on i and j, which is what the second recycling scheme permits.

Before we can present $\mathcal{L}^{\Delta}_{\text{PTR}}$ we will need one final element, modifications to the operational semantics given in Figure 3. Their purpose is to replace the generation of arbitrary new locations with locations drawn sequentially from $\mathbb{N}$, similarly to how we intend to use indexed names. This will enable us to exploit the fact that Lemmata 43 and 44 mean that available locations are also restricted. Instead of having configurations of the form (M, h), we have ones of the form (M, h, i_h, η), where i_h is the next available location (and η will be a function as above). The previous operational rules $\to$ (save those for $\text{ref } V$ and $\text{while } M \text{ do } N$) are embedded into the new reduction $\to_e$ using the rule

$$\frac{(M, h) \to (M', h')}{(M, h, i_h, \eta) \to_e (M', h', i_h, \eta)}.$$

The reduction rule for handling new references is replaced by $(K[\text{ref } V], h, i_h, \eta) \to_e (K[i_h], h \cdot [i_h \mapsto V], i_h + 1, \eta)$. It uses i_h as the location for the new reference, and then sets the next location to be $i_h + 1$. This gives an operational semantics which is behaviorally the same as generating a fresh location, so long as i_h is larger than any name appearing in h.

We also make changes to handle the while do construct. The idea is to reset both i_h and η back to the value before the loop once we reach the end of the loop. This is due to the fact that, by the way the scopes work in the language, any name or location generated in the loop cannot be used outside of (that iteration of) the loop. In particular, we introduce a new construct, $\text{end}(i_h, \eta).M$, to indicate the end of an iteration of a loop. We provide rules for while and end in Figure 11, where $h_{<i_h}$ denotes the heap h restricted to domain of location smaller than i_h. Similarly, if ζ is a partial map from indexed names, and η maps base names to indices, we write ζ_η to mean ζ restricted to indexed names f^i for which $i < \eta(f)$. We will use $h_{\geq i_h}$ and $\zeta_{\geq \eta}$ analogously.

Finally, we present the LTS $\mathcal{L}^{\Delta}_{\text{PTR}}$ in Figure 12. Active configurations of $\mathcal{L}^{\Delta}_{\text{PTR}}$ have the form $\langle M, c, \gamma, h, H, i_h, \eta, \mu, l\rangle$ and passive ones $\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle$. As described above, η is a function from base names to the next available index, which we call the ***(next) index component***. i_h is the ***(next) location component***, the next available location. μ is the ***reset*** component, a partial map from (indexed) level-2 thunk names and O-continuation names to the value of (i_h, η) prior to the move that introduced the name. l is a binary flag used to indicate whether a marked name has been produced in the trace so far.

We now need to define initial configurations. Let $\Gamma \vdash^c M : F\sigma$ be a PTR computation and $\Delta = (\text{TB}, \text{CB}, \rho, c_0, \text{Suc}_\text{T}, \text{Suc}_\text{C})$ be a $(\Gamma, F\sigma)$-name scheme. Let $\rho^0 = [x_i \mapsto \rho(x_i)^0]$, $N_O = \nu(\rho) \cup \{c_0\}$ and $N_O^0 = \{n^0 \mid n \in N_0\}$. Then the active initial configuration $\mathsf{C}_M^{\text{PTR},\Delta}$ is defined to be

$$(\langle M\{\rho^0\}, c_0^0, \emptyset, \emptyset, [N_O^0 \mapsto \emptyset], 0, \eta, \emptyset, 0\rangle, \bot)$$

where $\eta = [N_O, c_0 \mapsto 1] \cdot [(\text{TB} \cup \text{CB}) \setminus (N_O \cup \{c_0\}) \mapsto 0]$.

The main change to the LTS, is to 'recycle' the indices, so as to keep the space of reachable configurations finite. This is the role of the μ component, based on the properties identified in

$(P\tau)$ $\langle M, c^j, \gamma, h, H, i_h, \eta, \mu, l\rangle \xrightarrow{\tau} \langle N, c^j, \gamma_{<\eta'}, h', H_{<\eta'}, i'_h, \eta', \mu_{<\eta'}, l\rangle$
when $(M, h, i_h, \eta) \to_e (N, h', i'_h, \eta')$

(PA) $\langle \text{return } V, c_0^0, \gamma, h, H, i_h, \eta, \mu, l\rangle \xrightarrow{\bar{c_0}(\beta(A))} \langle \gamma \cdot \gamma', h, H, H(c_0) \uplus \nu(A), i_h, \eta', \mu, l\rangle$
when $c_0 : \sigma$, $(A, \gamma', \eta') = \mathbf{IVal}^{\Delta}_{\sigma}(c_0, V, \eta)$

(PA) $\langle \text{return } V, c^i, \gamma, h, H, i_h, \eta, \mu, l\rangle \xrightarrow{\bar{c}(V)} \langle \gamma_{<\eta'}, h_{<i'_h}, H_{<\eta'}, H(c^i), i'_h, \eta', \mu_{<\eta'}, l\rangle$
when $c \neq c_0$ and $(i'_h, \eta') = \mu(c^i)$

(PQ) $\langle K[(\text{force } f^i)\overrightarrow{V}], c'^j, \gamma, h, H, i_h, \eta, \mu, l\rangle \xrightarrow{\bar{f}(\beta(\overrightarrow{A}),c)/(c^0,(K,c'^j))} \langle \gamma \cdot \gamma', h, H, H(f^i) \uplus \nu(\overrightarrow{A}), i_h, \eta', \mu, l\rangle$
when f is not a level 2 name, $(\overrightarrow{A}, \gamma', \eta') \in \mathbf{IVal}^{\Delta}(f, \overrightarrow{V}, \eta)$, and $\text{Suc}_{\text{C}}(f) = c$

(PQ) $\langle K[(\text{force } f^i)\overrightarrow{V}], c'^j, \gamma, h, H, i_h, \eta, \mu, l\rangle \xrightarrow{\bar{f}(\overrightarrow{V},c)/(c^0,(K,c'^j),P)} \langle \gamma_{<\eta'}, h_{<i'_h}, H_{<\eta'}, H(f^i), i'_h, \eta', \mu_{<\eta'}, l\rangle$
when f is a level 2 name, and $(i'_h, \eta') = \mu(f^i)$, $\text{Suc}_{\text{C}}(f) = c$, and $P = (i_h, \eta, \gamma_{\geq\eta'}, h_{\geq i'_h}, H_{\geq\eta'}, \mu_{\geq\eta'})$

(OA) $\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle \xrightarrow{c(\beta(A)),(c^0,(K,c'^j))} \langle K[\text{return } A], c'^j, \gamma, h, H \cdot [\nu(A) \mapsto Fn], i_h, \eta', \mu, l'\rangle$
when $c : \sigma$, $(A', \eta') \in \mathbf{IVals}^{\Delta}_{\sigma}(c, \eta)$ and if $l = 1$ then $A = A', l' = 1$ else $A \in \mathbf{Select}(A')$, and $l' = \mathbf{IsMark}(A)$

(OA) $\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle \xrightarrow{c(\beta(A)),(c^0,(K,c'^j),P)} \langle K[\text{return } A], c'^j, \gamma \cdot \gamma', h, H', i'_h, \eta', \mu', l'\rangle$
when $c : \sigma$, $P = (i'_h, \eta'', \gamma', h', H'', \mu'')$, $(A', \eta') \in \mathbf{IVals}^{\Delta}_{\sigma}(c, \eta'')$ and if $l = 1$ then $A = A', l' = 1$
else $A \in \mathbf{Select}(A')$, and $l' = \mathbf{IsMark}(A)$; and $H' = H \cdot H'' \cdot [\nu(A) \mapsto Fn]$, and $\mu' = \mu \cdot \mu'' \cdot [\nu(A) \mapsto (i_h, \eta)]$

(OQ) $\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle \xrightarrow{f(\beta(\overrightarrow{A}),c)} \langle \text{force } V\overrightarrow{A}, c^j, \gamma, h, H \cdot [\nu(\overrightarrow{A}), c^j \mapsto Fn], i_h, \eta', \mu', l\rangle$
when $f^i \in Fn$, $(\overrightarrow{A'}, \eta'') \in \mathbf{IValSeq}^{\Delta}(f, \eta)$, $\text{Suc}_{\text{C}}(f) = c$, $\eta(c) = j, \eta' = \eta''[c \mapsto j+1], \gamma(f^i) = V$, and
if $l = 1$ then $A = A', l' = 1$ else $A \in \mathbf{Select}(A')$, and $l' = \mathbf{IsMark}(A)$; and $\mu' = \mu \cdot [\nu(\overrightarrow{A}), c^j \mapsto (i_h, \eta)]$

In the PQ rules, the name f can be either marked or unmarked. In the second PA (PQ), V ($\overrightarrow{V}$) does not contain thunks, so is an abstract value. The second OA rule is sound as γ', h', H'', μ'' are disjoint from γ, h, H, μ. $\mathbf{Select}(A)$ is the set of marked indexed abstract values obtained by marking at most one name in A. $\mathbf{IsMark}(A) = 1$ if a name in A is marked, 0 otherwise.

Figure 12: $\mathcal{L}^{\Delta}_{\text{PTR}}$ transition rules for name scheme $\Delta = (\text{TBNames}, \text{CBNames}, \rho, c_0, \text{Suc}_{\text{T}}, \text{Suc}_{\text{C}})$

Lemmata 43 and 44. In particular, after a PA-action (other than on the initial continuation name), we 'prune' the domains of the components to the index names and locations introduced before the OQ-action being answered. Similarly, after a PQ-action on a level-2 name f, we split the components between the index names and locations introduced before the OQ-action introducing the originator of f, and those after. Those from before the OQ-action become part of the next configuration, whereas those from after are stored on the stack until they can be restored after the matching OA-action. Let $\mathbf{Tr}^{\Delta}_{\text{PTR}}(\mathbf{C})$ be the set of base traces generated from $\mathbf{C}$ in $\mathcal{L}^{\Delta}_{\text{PTR}}$.

Definition 47. The PTR-***trace semantics*** of a PTR-computation $\Gamma \vdash^c M : F\sigma$ is defined to be $\mathbf{Tr}_{\text{PTR}}(\Gamma \vdash^c M : F\sigma) \triangleq \{\, (\Delta, t) \mid \Delta \text{ is a } (\Gamma, F\sigma)\text{-name scheme}, t \in \mathbf{Tr}^{\Delta}_{\text{PTR}}(\mathsf{C}_M^{\text{PTR},\Delta}), t \text{ is complete}\}$.

Example 48. Recall that in Example 18 we gave a derivation that the trace

$$\mathtt{t} = \bar{f}([r, w]\,\widehat{4}, c_1)\ w(\widehat{1}, c_2)\ \bar{c_2}(())\ c_1(())\ \bar{c}(g)\ g(h, c_3)\ \bar{h}(\epsilon, c_4)\ c_4(\widehat{2})\ \bar{c_3}(\widehat{3})$$

is in $\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c})$ where $\rho = [x \mapsto f]$, and

$$M = \text{ref}\,\widehat{0} \text{ to } x.(\text{force } f)\,x\,\widehat{4}; \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!x \text{ to } z.y + z)$$

Let $\Delta = (\{f, w, r, g, h\}, \{c, c_f, c_w, c_r, c_g, c_h\}, \rho, c, \text{Suc}_{\text{T}}, \text{Suc}_{\text{C}})$ with $\text{Suc}_{\text{C}}(f) = c_f$, $\text{Suc}_{\text{C}}(w) = c_w$, $\text{Suc}_{\text{C}}(r) = c_r$, $\text{Suc}_{\text{C}}(g) = c_g$, and $\text{Suc}_{\text{C}}(h) = c_h$, and $\text{Suc}_{\text{T}}(f, 1) = (w, r)$, $\text{Suc}_{\text{T}}(c) = g$, and $\text{Suc}_{\text{T}}(g, 1) = h$. We will now show how the base trace

$$\mathbf{Base}^{\Delta}_{\mathtt{t}}(\mathtt{t}) = \bar{f}([r, w]\,\widehat{4}, c_f)\ w(\widehat{1}, c_w)\ \bar{c_w}(())\ c_f(())\ \bar{c}(g)\ g(h, c_g)\ \bar{h}(\epsilon, c_h)\ c_h(\widehat{2})\ \bar{c_g}(\widehat{3})$$

is in $\mathbf{Tr}^{\Delta}_{\mathrm{PTR}}(\mathsf{C}^{\mathrm{PTR},\Delta}_{M})$. Let $N_O = \{f, c\}$ and $\eta_0 = [f, c \mapsto 1] \cdot [w, r, g, h, c_f, c_w, c_r, c_g, c_h \mapsto 0]$ and $H_0 = [f^0, c^0 \mapsto \emptyset]$.

$$
\begin{array}{lll}
\mathsf{C}^{\rho,c}_{M} & \xrightarrow{\tau}{}^{*} & (\langle M_0, c^0, \emptyset, [0 \mapsto \widehat{0}], H_0, 1, \eta_0, \emptyset, 0\rangle, \bot) \\
 & & \text{where } M_0 = (\text{force } f^0)\, \ell\, \widehat{4}; \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z) \\
 & \xrightarrow{\bar{f}([r,w]\,\widehat{4}, c_f)} & (\langle \gamma_0, [0 \mapsto \widehat{0}], H_0, \{r^0, w^0\}, 1, \eta_1, \emptyset, 0\rangle, (c^0_f, (K_1, c^0)) : \bot) \\
 & & \text{where } \gamma_0 = [r^0 \mapsto \text{thunk } (!0),\ w^0 \mapsto \text{thunk } (\lambda x.0 := x)], \eta_1 = \eta_0[w, r \mapsto 1] \\
 & & \text{and } K_1 = \bullet; \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z) \\
 & \xrightarrow{w(\widehat{1}, c_w)} & (\langle (\text{force thunk } (\lambda x.0 := x))\widehat{1}, c^0_w, \gamma_0, [0 \mapsto \widehat{0}], H_1, 1, \eta_2, [c^0_w \mapsto (1, \eta_1)], 0\rangle, (c^0_f, (K_1, c^0)) : \bot) \\
 & & \text{where } H_1 = H_0 \cdot [c^0_w \mapsto \{r^0, w^0\}] \text{ and } \eta_2 = \eta_1[c_w \mapsto 1] \\
 & \xrightarrow{\tau}{}^{*} & (\langle \text{return } (), c^0_w, \gamma_0, [0 \mapsto \widehat{1}], H_1, 1, \eta_2, [c^0_w \mapsto (1, \eta_1)], 0\rangle, (c^0_f, (K_1, c^0)) : \bot) \\
 & \xrightarrow{\bar{c_w}(())} & (\langle \gamma_0, [0 \mapsto \widehat{1}], H_0, \{r^0, w^0\}, 1, \eta_1, \emptyset, 0\rangle, (c^0_f, (K_1, c^0)) : \bot) \\
 & \xrightarrow{c_f(())} & (\langle K_1[\text{return } ()], c^0, \gamma_0, [0 \mapsto \widehat{1}], H_0, 1, \eta_1, \emptyset, 0\rangle, \bot) \\
 & \xrightarrow{\tau}{}^{*} & (\langle \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z), c^0, \gamma_0, [0 \mapsto \widehat{1}], H_0, 1, \eta_1, \emptyset, 0\rangle, \bot) \\
 & \xrightarrow{\bar{c}(g)} & (\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_2, \emptyset, 0\rangle, \bot) \\
 & & \text{where } \gamma_1 = \gamma_0 \cdot [g^0 \mapsto \text{thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z)] \text{ and } \eta_2 = \eta_1[g \mapsto 1] \\
 & \xrightarrow{g(h, c_g)} & (\langle (\text{force thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z))h^0, c^0_g, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c^0_g \mapsto (1, \eta_2)], 0\rangle, \bot) \\
 & & \text{where } H_2 = H_0 \cdot [h^0, c^0_g \mapsto \{g^0\}] \text{ and } \eta_3 = \eta_2[h, c_g \mapsto 1] \\
 & \xrightarrow{\tau}{}^{*} & (\langle \text{force } h^0 \text{ to } y.!0 \text{ to } z.y + z, c^0_g, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c^0_g \mapsto (1, \eta_2)], 0\rangle, \bot) \\
 & \xrightarrow{\bar{h}(\epsilon, c_h)} & (\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_0, \emptyset, 0\rangle, (c^0_h, (K_2, c^0_g), P) : \bot) \\
 & & \text{where } K_2 = \bullet \text{ to } y.!0 \text{ to } z.y + z, \text{ and } P = (1, \eta_3, \emptyset, \emptyset, [h^0, c^0_g \mapsto \{g^0\}], [h^0, c^0_g \mapsto (1, \eta_2)]) \\
 & \xrightarrow{c_h(\widehat{2})} & (\langle K_2[\text{return } \widehat{2}], c^0_g, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c^0_g \mapsto (1, \eta_2)], 0\rangle, \bot) \\
 & \xrightarrow{\tau}{}^{*} & (\langle \text{return } \widehat{3}, c^0_g, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c^0_g \mapsto (1, \eta_2)], 0\rangle, \bot) \\
 & \xrightarrow{\bar{c_g}(\widehat{3})} & (\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_2, \emptyset, 0\rangle, \bot)
\end{array}
$$

To gain a further understanding of the operation $\mathcal{L}^{\Delta_{\mathrm{r}},\sigma}_{\mathrm{PTR}}$, we also see how the trace

$$\mathtt{t}' = \bar{f}([r,w]\,\widehat{4}, c_f)\ w(\widehat{1}, c_w)\ \bar{c_w}(())\ c_f(())\ \bar{c}(g)\ g(h, c_g)\ \bar{h}(\epsilon, c_h)\ g(\hat{h}, c_g)\ \bar{\hat{h}}(\epsilon, c_h)\ c_h(\widehat{2})\ \bar{c_g}(\widehat{3})\ c_h(\widehat{2})\ \bar{c_g}(\widehat{3})$$

in $\mathbf{Tr}^{\Delta}_{\mathrm{PTR}}(\mathsf{C}^{\mathrm{PTR},\Delta}_{M})$ is derived.

$\mathsf{C}_M^{\rho,c}$ $\xrightarrow{\tau}{}^*$ $(\langle M_0, c^0, \emptyset, [0 \mapsto \widehat{0}], H_0, 1, \eta_0, \emptyset, 0\rangle, \bot)$
where $M_0 = (\text{force } f^0)\, \ell\, \widehat{4}; \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z)$

$\xrightarrow{\bar{f}([r,w]\widehat{4}, c_f)}$ $(\langle \gamma_0, [0 \mapsto \widehat{0}], H_0, \{r^0, w^0\}, 1, \eta_1, \emptyset, 0\rangle, (c_f^0, (K_1, c^0)) : \bot)$
where $\gamma_0 = [r^0 \mapsto \text{thunk } (!0),\ w^0 \mapsto \text{thunk } (\lambda x.0 := x)], \eta_1 = \eta_0[w, r \mapsto 1]$
and $K_1 = \bullet; \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z)$

$\xrightarrow{w(\widehat{1}, c_w)}$ $(\langle (\text{force thunk } (\lambda x.0 := x))\widehat{1}, c_w^0, \gamma_0, [0 \mapsto \widehat{0}], H_1, 1, \eta_2, [c_w^0 \mapsto (1, \eta_1)], 0\rangle, (c_f^0, (K_1, c^0)) : \bot)$
where $H_1 = H_0 \cdot [c_w^0 \mapsto \{r^0, w^0\}]$ and $\eta_2 = \eta_1[c_w \mapsto 1]$

$\xrightarrow{\tau}{}^*$ $(\langle \text{return } (), c_w^0, \gamma_0, [0 \mapsto \widehat{1}], H_1, 1, \eta_2, [c_w^0 \mapsto (1, \eta_1)], 0\rangle, (c_f^0, (K_1, c^0)) : \bot)$

$\xrightarrow{\bar{c_w}(())}$ $(\langle \gamma_0, [0 \mapsto \widehat{1}], H_0, \{r^0, w^0\}, 1, \eta_1, \emptyset, 0\rangle, (c_f^0, (K_1, c^0)) : \bot)$

$\xrightarrow{c_f(())}$ $(\langle K_1[\text{return } ()], c^0, \gamma_0, [0 \mapsto \widehat{1}], H_0, 1, \eta_1, \emptyset, 0\rangle, \bot)$

$\xrightarrow{\tau}{}^*$ $(\langle \text{return thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z), c^0, \gamma_0, [0 \mapsto \widehat{1}], H_0, 1, \eta_1, \emptyset, 0\rangle, \bot)$

$\xrightarrow{\bar{c}(g)}$ $(\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_2, \emptyset, 0\rangle, \bot)$
where $\gamma_1 = \gamma_0 \cdot [g^0 \mapsto \text{thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z)]$ and $\eta_2 = \eta_1[g \mapsto 1]$

$\xrightarrow{g(h, c_g)}$ $(\langle (\text{force thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z))h^0, c_g^0, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c_g^0 \mapsto (1, \eta_2)], 0\rangle, \bot)$
where $H_2 = H_0 \cdot [h^0, c_g^0 \mapsto \{g^0\}]$ and $\eta_3 = \eta_2[h, c_g \mapsto 1]$

$\xrightarrow{\tau}{}^*$ $(\langle \text{force } h^0 \text{ to } y.!0 \text{ to } z.y + z, c_g^0, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c_g^0 \mapsto (1, \eta_2)], 0\rangle, \bot)$

$\xrightarrow{\bar{h}(\epsilon, c_h)}$ $(\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_0, \emptyset, 0\rangle, (c_h^0, (K_2, c_g^0), P) : \bot)$
where $K_2 = \bullet \text{ to } y.!0 \text{ to } z.y + z,$ and $P = (1, \eta_3, \emptyset, \emptyset, [h^0, c_g^0 \mapsto \{g^0\}], [h^0, c_g^0 \mapsto (1, \eta_2)])$

$\xrightarrow{g(\hat{h}, c_g)}$ $(\langle (\text{force thunk } (\lambda h.\text{force } h \text{ to } y.!0 \text{ to } z.y + z))\hat{h}^0, c_g^0, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [\hat{h}^0, c_g^0 \mapsto (1, \eta_2)], 1\rangle,$
$(c_h^0, (K_2, c_g^0), P) : \bot)$
where $\hat{H}_2 = H_0 \cdot [\hat{h}^0, c_g^0 \mapsto \{g^0\}]$

$\xrightarrow{\tau}{}^*$ $(\langle \text{force } \hat{h}^0 \text{ to } y.!0 \text{ to } z.y + z, c_g^0, \gamma_1, [0 \mapsto \widehat{1}], \hat{H}_2, 1, \eta_3, [h^0, c_g^0 \mapsto (1, \eta_2)], 1\rangle, (c_h^0, (K_2, c_g^0), P) : \bot)$

$\xrightarrow{\bar{\hat{h}}(\epsilon, c_h)}$ $(\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_0, \emptyset, 1\rangle, (c_h^0, (K_2, c_g^0), \hat{P}) : (c_h^0, (K_2, c_g^0), P) : \bot)$
where $\hat{P} = (1, \eta_3, \emptyset, \emptyset, [\hat{h}^0, c_g^0 \mapsto \{g^0\}], [\hat{h}^0, c_g^0 \mapsto (1, \eta_2)])$

$\xrightarrow{c_h(\widehat{3})}$ $(\langle K_2[\text{return } \widehat{3}], c_g^0, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \hat{\eta}_3, [\hat{h}^0, c_g^0 \mapsto (1, \eta_2)], 1\rangle, (c_h^0, (K_2, c_g^0), P) : \bot)$

$\xrightarrow{\tau}{}^*$ $(\langle \text{return } \widehat{4}, c_g^0, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \hat{\eta}_3, [\hat{h}^0, c_g^0 \mapsto (1, \eta_2)], 1\rangle, (c_h^0, (K_2, c_g^0), P) : \bot)$

$\xrightarrow{\bar{c_g}(\widehat{4})}$ $(\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_2, \emptyset, 1\rangle, (c_h^0, (K_2, c_g^0), P) : \bot)$

$\xrightarrow{c_h(\widehat{2})}$ $(\langle K_2[\text{return } \widehat{2}], c_g^0, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c_g^0 \mapsto (1, \eta_2)], 1\rangle, \bot)$

$\xrightarrow{\tau}{}^*$ $(\langle \text{return } \widehat{3}, c_g^0, \gamma_1, [0 \mapsto \widehat{1}], H_2, 1, \eta_3, [h^0, c_g^0 \mapsto (1, \eta_2)], 1\rangle, \bot)$

$\xrightarrow{\bar{c_g}(\widehat{3})}$ $(\langle \gamma_1, [0 \mapsto \widehat{1}], H_0, \{g^0\}, 1, \eta_2, \emptyset, 1\rangle, \bot)$

6.1 Full Abstraction of $\mathcal{L}_{\text{PTR}}^{\Delta}$

We can show that the new semantics agree with the full trace semantics on PTR-computations. This aim can be expressed as the following Lemma.

Lemma 49. *For any* PTR*-computation* $\Gamma \vdash M : F\sigma$*, a* $(\Gamma, F\sigma)$*-name scheme* $\Delta = (\text{TB}, \text{CB}, \rho, c_0, \text{Suc}_\text{T}, \text{Suc}_\text{C})$*,* $\mathbf{Tr}_{\text{PTR}}^{\Delta}(\mathsf{C}_M^{\text{PTR},\Delta}) = \mathbf{Rename}^{\Delta}(\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$.

To simplify the statement of results in this sections, we will fix a PTR-computation $\Gamma \vdash^c M : F\sigma_0$, (Γ, σ)-name scheme $\Delta = (\text{TB}, \text{CB}, \rho, c_0, \text{Suc}_\text{T}, \text{Suc}_\text{C})$, and let $N_O = \nu(\rho) \cup \{c_0\}$.

We will prove Lemma 49 using a bisimulation technique, though we will not be able to give a bisimulation directly between $\mathcal{L}_{\text{CBPV}}$ and $\mathcal{L}_{\text{PTR}}^{\Delta_{\Gamma,\sigma}}$. Instead, we will introduce a LTS $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$, the traces of which will be exactly $\mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$. Given a path p, let $\text{Tr}(p)$ be the trace induced by p.

Definition 50. The configurations of $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ have the form (p, F) where p is a path in $\mathcal{L}_{\text{CBPV}}$ starting in C_M^{ρ,c_0} and F is either empty, or a set containing a single name f s.t. f is introduced in an O-action in $\text{Tr}(p)$. Let $\text{Mark}(f, X)$ for some structure X (an action etc.) be obtained by marking every occurrence of f. The transitions are then (where $\mathbf{a}$ includes τ actions)

- $(p, \{f\}) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \{f\})$ where p ends with $\mathbf{C}$, $\mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$, $\mathbf{a}'' = \text{Mark}(f, \mathbf{a})$ and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a}'')$.
- $(p, \emptyset) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \emptyset)$ where p ends with $\mathbf{C}$, $\mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$, and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a})$.
- $(p, \emptyset) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \{g\})$ where $\mathbf{a} = c(A)$, $g \in \nu(A)$, $\mathbf{a}'' = \text{Mark}(g, \mathbf{a})$ and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a}'')$.
- $(p, \emptyset) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \{g\})$ where $\mathbf{a} = f(\overrightarrow{A}, c)$, $g \in \nu(\overrightarrow{A})$, $\mathbf{a}'' = \text{Mark}(g, \mathbf{a})$ and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a}'')$.

Observe that $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ is deterministic in the sense that if for configuration (p, F), there is only one t s.t. $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t}{\Rightarrow} (p, F)$.

We can then establish the correctness of $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ with respect to $\mathcal{L}_{\text{CBPV}}$ and the tranlation function $\mathbf{Rename}^{\Delta_{\Gamma,\sigma}}()$.

Lemma 51. *The traces of* $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ *starting from* $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$ *are exactly* $\mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$.

Proof. We prove using the following interemediate result. Let p is a path in $\mathcal{L}_{\text{CBPV}}$ starting from C_M^{ρ,c_0}, and $t = \text{Tr}(p)$, then

1. $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t'}{\Longrightarrow} (p, \emptyset)$ where $t' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(t)$; and
2. if t a thunk O-name introduced in t, then $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t''}{\Longrightarrow} (p, \{f\})$, where $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(\text{Mark}(f, t))$.

This proof proceeds by induction on the length of the path p.

Using this result, we can proceed to prove the Lemma as follows. For the first direction, let $t'' \in \mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$. Then there exists $t \in \mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0})$ and $t' \in \mathbf{Marked}(t)$ s.t. $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(t')$. Let p be the path in generating t. If $t = t'$, we have by the above property that $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t''}{\Longrightarrow} (p, \emptyset)$, so t'' is generated by $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ starting from $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$. Otherwise, let f be the name marked in t'. By the above property, $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t''}{\Longrightarrow} (p, \{f\})$, so t'' is generated by $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ starting from $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$.

For the other direction, let t'' be a trace generated by $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ starting from $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$. Then there must be a path p s.t. $\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t''}{\Longrightarrow} (p, \emptyset)$ or $\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t''}{\Longrightarrow} (p, \{f\})$ for some f introduced in an O-action of p. Let $t = \text{Tr}(p)$, so $t \in \mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0})$. Then it follows from the above result, and the determinism of $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$, that $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(t)$ or $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(\text{Mark}(f, t))$. Thus, $t'' \in \mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$. □

Now, to prove Lemma 49, we construct a bisimulation between $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ and $\mathcal{L}_{\text{PTR}}^{\Delta_{\Gamma,\sigma}}$. To do this, we will use the function $\mathbf{Conv}(p)$, which will take path p to a tuple $(\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$. κ maps names introduced in p to indexed base names, and locations to numerical locations. We lift κ to any structure containing names and/or locations in the obvious way. ψ maps indexed base names to names introduced in p (a partial inverse of κ), η maps base names to the next index and μ maps level 2 and O-continuation indexed base names to the value of (i_h, η) before their

introduction. ψ^h, i_h play an analogous role to ψ, η but for locations. Finally, N is simply the term components of the final state in p (or $\emptyset$ if the last configuration is passive), but with the result of expanding while loops annotated with $\text{end}(i_h, \eta)..T$ is an extended stack, which relates to the stack in the same way as N does to term, but with added elements at some levels of the stack. We also define the reduction $(N, h) \to_m (N', h')$ on terms of the extended syntax (with $\text{end}(i_h, \eta).$), as having the same rules as $\to$ (and so getting stuck if it encounters $\text{end}(i_h, \eta).$). Let $\tilde{N}$ be the function which removes occurrences of $\text{end}(i_h, \eta).$ from a term/context.

We write $p'\ \mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$ when $\mathbf{C}, \mathbf{C}'$ are the last two configurations in a path (where p' can be empty). Recall also that we have $\mathbf{Match}(A, B, f)$ transfers any mark on f in A to its result.

Definition 52. We define $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$ with the following cases. If $p = \mathsf{C}_M^{\rho, c_0}$ then $\mu = \psi^h = \emptyset$, $i_h = 0$, $N = M$, $K = \bot$, $\kappa = [N_O \mapsto N_O^0]$, $\psi = [N_O^0 \mapsto N_O]$, η is 1 on N_O, 0 otherwise. Otherwise, $p = p'\ \mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$. Let $\mathbf{Conv}_F(p'\ \mathbf{C}) = (\kappa', \psi', \eta', \mu', \psi'^h, i'_h, N', T')$. We then proceed by the follwoing cases:

- $\mathbf{a} = \tau$, $\mathbf{C} = (\langle K[\text{ref}\, V], \cdots \rangle, S)$ and $\mathbf{C}' = (\langle K[\ell], \cdots \rangle, S)$. If $N' = K'[\text{ref}\, V]$ then $N = K'[i'_h]$, $i_h = i'_h + 1$, $\kappa = \kappa' \cdot [\ell \mapsto i'_h], \psi^h = \psi'^h \cdot [i'_h \mapsto \ell]$, $\psi = \psi'$, $\eta = \eta'$, $\mu = \mu'$, and $T = T'$;
- $\mathbf{a} = \tau$ where $\mathbf{C} = (\langle K[\text{while}\, N_1\, \text{do}\, N_2], \cdots \rangle, S)$.
  - If $N' = K'[\text{while}\, N_1\, \text{do}\, N_2]$, then
    $$N = K'[N_1\ \text{to}\ x.\text{case}\ x\ \text{of return}\ (), (N_2\ \text{to}\ y.\text{end}(i_h, \eta).\text{while}\, N_1\, \text{do}\, N_2)_{j>0}]$$
    and $\kappa = \kappa'$, $\psi = \psi'$, $\eta = \eta'$, $\mu = \mu'$, $\psi^h = \psi'^h$, $i_h = i'_h$, and $T = T'$;
  - If $N' = K'[\text{end}(i_h, \eta).\text{while}\, N_1\, \text{do}\, N_2]$, then
    $$N = K'[\text{end}M\ \text{to}\ x.\text{case}\ x\ \text{of return}\ (), (N\ \text{to}\ y.\text{end}(i_h, \eta).\text{while}\, M\, \text{do}\, N)_{j>0}.]$$
    $\psi = \psi'_{<\eta}$, $\psi^h = \psi'^h_{<i_h}$, $\kappa = \kappa'$, $\mu = \mu'$, and $T = T'$;
- $\mathbf{a} = \tau$ otherwise, where $\mathbf{C} = (\langle M', c, \gamma, \phi, h, H \rangle, S)$, $\mathbf{C}' = (\langle M'', c, \gamma, \phi, h', H \rangle, S))$. Then N is s.t $(N', h) \to_m (N, h')$, $\kappa = \kappa'$, $\psi = \psi'$, $\eta = \eta'$, $\mu = \mu'$, $\psi^h = \psi'^h$, $i_h = i'_h$, and $T = T'$;
- $\mathbf{a} = \bar{f}(\overrightarrow{A}, c)$ where f is not a level 2 name, and $\mathbf{C} = (\langle K[\text{force}\ f\ \overrightarrow{V}], c', \cdots \rangle, S')$. If $N' = K'[\text{force}\ f\ \overrightarrow{V}]$, then $T = (c, (K', c')) : T'$, $N = \emptyset$ $\mu = \mu'$, $\psi^h = \psi'^h$, and $i_h = i'_h$. If $g^i = \kappa'(f)$,$d = \text{Suc}_{\mathrm{C}}(g)$, and $(\overrightarrow{B}, \gamma, \eta'') = \mathbf{IVal}^{\Delta}(g, \overrightarrow{V}, \eta')$, then $\kappa = \kappa' \cdot [c \mapsto d^0] \cdot [f' \mapsto \mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, f')]_{f' \in \nu(\overrightarrow{A})}$, and $\psi = \psi' \cdot [\mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, f') \mapsto f']_{f' \in \nu(\overrightarrow{A})}$;
- $\mathbf{a} = \bar{f}(\overrightarrow{A}, c)$ where f is a level 2 name and $\mathbf{C} = (\langle K[\text{force}\ f\ \overrightarrow{V}], c', \cdots \rangle, S')$. If $N' = K'[\text{force}\ f\ \overrightarrow{V}]$ and $(i_h, \eta) = \mu(f)$, then $\psi = \psi'_{<\eta}$, $\psi^h = \psi'^h_{<i_h}$, $T = (c, (K', c'), (i'_h, \eta', \psi'^h_{\geq i_h}, \psi'_{\geq \eta})) : T'$, $N = \emptyset$, $\kappa = \kappa' \cdot [c \mapsto d^0]$, and $\mu = \mu'$;
- $\mathbf{a} = f(\overrightarrow{A}, c)$ where $\mathbf{C} = (\langle \gamma, \phi, h, H, Fn \rangle, S)$. If $N' = \emptyset$ then $N = \gamma(f)$, $\psi^h = \psi'^h$, $i_h = i'_h$, and $T = T'$. If $g^i = \kappa'(f)$, $d = \text{Suc}_{\mathrm{C}}(g)$, $j = \eta(d)$, $(\overrightarrow{B}, \eta'') = \mathbf{IValSeq}^{\Delta}(g, \eta')$ and $\overrightarrow{A'} = \text{Mark}(F, \overrightarrow{A})$, then $\kappa = \kappa' \cdot [c \mapsto d^j] \cdot [f \mapsto \mathbf{Match}(\overrightarrow{A'}, \overrightarrow{B}, f)]_{f \in \nu(\overrightarrow{A})}$, $\psi = \psi' \cdot [d^j \mapsto c] \cdot [\mathbf{Match}(\overrightarrow{A'}, \overrightarrow{B}, f) \mapsto f]_{f \in \nu(\overrightarrow{A})}$, $\mu = \mu' \cdot [\nu(\overrightarrow{A}), c \mapsto (i_h, \eta)]$, and $\eta = \eta''[d \mapsto j + 1]$;
- $\mathbf{a} = \bar{c_0}(A)$ where $\mathbf{C} = (\langle \text{return}\ V, \cdots \rangle, S)$. If $(B, \gamma, \eta) = \mathbf{IVal}_{\sigma}^{\Delta}(c_0, V, \eta')$, then $N = \emptyset$, $\kappa = \kappa' \cdot [f \mapsto \mathbf{Match}(A, B, f)]_{f \in \nu(A)}$, $\psi' = \psi \cdot [\mathbf{Match}(A, B, f) \mapsto f]_{f \in \nu(A)}$, $\mu = \mu'$, $\psi^h = \psi'^h$, $i_h = i'_h$,and $T = T'$;
- $\mathbf{a} = \bar{c}(A)$. If $(i_h, \eta) = \mu(c)$, then $\psi = \psi'_{<\eta}$, $\psi^h = \psi'^h_{<i_h}$, $N = \emptyset$, $\kappa = \kappa'$, $\mu = \mu'$, and $T = T'$;
- $\mathbf{a} = c(A)$ with $A : \sigma$.

– If $T' = (c, (K, c')) : T''$ and $d = \kappa'(c)$, then $T = T''$, $N = K[\text{return } A]$, and for $A' = \text{Mark}(F, A)$, for any $(B', \eta) \in \mathbf{IVals}^{\Delta}_{\sigma}(d, \eta')$, $\kappa = \kappa' \cdot [f \mapsto \mathbf{Match}(A', B, f)]_{f \in \nu(A)}$, $\psi = \psi' \cdot [\mathbf{Match}(A', B, f) \mapsto f]_{f \in \nu(A)}$, $\mu = \mu'$, $\psi^h = \psi'^h$, and $i_h = i'_h$;

– If $T' = (c, (K, c'), (i_h, \eta'', \psi''^h, \psi'')) : T''$, $d = \kappa'(c)$, and ψ''^h, ψ'' disjoint from ψ'^h, ψ' then $T = T''$, $N = K[\text{return } A]$, and for $A' = \text{Mark}(F, A)$, for any $(B', \eta) \in \mathbf{IVals}^{\Delta}_{\sigma}(d, \eta'')$, $\psi^h = \psi'^h \cdot \psi''^h$ $\kappa = \kappa' \cdot [f \mapsto \mathbf{Match}(A', B, f)]_{f \in \nu(A)}$, $\psi = \psi' \cdot \psi'' \cdot [\mathbf{Match}(A', B, f) \mapsto f]_{f \in \nu(A)}$ and $\mu = \mu' \cdot [\nu(A) \mapsto (i'_h, \eta')]$.

We can prove via simple inductions, that for any path p, $\mathbf{Conv}_F(p)$ is defined (that is, the conditions assume in the inductive construction do hold).

An important yet unsurprising property of $\mathbf{Conv}_F(p)$ is that for names and locations in the last configuration of p, κ and ψ (ψ^h) are inverses.

Lemma 53. *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, and $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$. Then:*

- *if p ends in a active state $(\langle N, c, \gamma, \phi, h, H\rangle, S)$, then for $f \in \nu(N)$, we have $(\psi \circ \kappa)(f) = f$, $(\psi \circ \kappa)(c) = c$, and for location ℓ in N, $(\psi^h \circ \kappa)(\ell) = \ell$;*
- *if p ends in a passive state $(\langle \gamma, \phi, h, H, Fn\rangle, S)$, then for $f \in Fn$, we have $(\psi \circ \kappa)(f) = f$.*

We define the following function mapping extended stacks to stacks with the same structure as $\mathcal{L}^{\Delta}_{\text{PTR}}$.

$$\begin{array}{rcl}
\mathbf{Stack}_{\gamma,h,H,\mu}(\bot) & \triangleq & \bot \\
\mathbf{Stack}_{\gamma,h,H,\mu}((c, (K, c')) : T) & \triangleq & (c, (K, c')) : \mathbf{Stack}_{\gamma,h,H,\mu}(T) \\
\mathbf{Stack}_{\gamma,h,H,\mu}((c, (K, c'), (i_h, \eta, \psi^h, \psi)) : T) & \triangleq & (c, (K, c'), (i_h, \eta, \gamma \circ \psi, h \circ \psi^h, H \circ \psi, \mu \circ \psi)) : \mathbf{Stack}_{\gamma,h,H,\mu}(T)
\end{array}$$

Finally, we can define a functional bisimulation Θ from $\mathcal{L}_{\text{Path}}$ to $\mathcal{L}^{\Delta_{\Gamma,\sigma}}_{\text{PTR}}$:

$$\begin{array}{rcl}
\Theta((p\ \mathbf{C}, F)) & \triangleq & (\langle \kappa(N), \kappa(c), \kappa \circ \gamma \circ \psi, h \circ \psi^h, \kappa \circ H \circ \psi, i_h, \eta, \mu \circ \psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T))), \\
& & \text{if } \mathbf{C} = (\langle N', c, \gamma, \phi, h, H\rangle, S) \\
\Theta((p\ \mathbf{C}, F)) & \triangleq & (\langle \kappa \circ \gamma \circ \psi, h \circ \psi^h, \kappa \circ H \circ \psi, \kappa(Fn), i_h, \eta, \mu \circ \psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T))), \\
& & \text{if } \mathbf{C} = (\langle \gamma, \phi, h, H, Fn\rangle, S)
\end{array}$$

where $(\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T) = \mathbf{Conv}_F(p\ \mathbf{C})$ and if $F = \emptyset$ then $l = 0$, otherwise $l = 1$.

Lemma 54. *Θ is a functional bisimulation between $\mathcal{L}^{\Delta}_{\text{Path}}$ and $\mathcal{L}^{\Delta}_{\text{PTR}}$*

Using this bisimulation, we can complete our proof of correctness.

Proof of Lemma 49. This follows from Lemma 51 and Lemma 54, as bisimulation implies trace equivalence for deterministic LTS. □

Lemma 49 with Corollaries 24 and 41 imply the following.

Theorem 55 (PTR Full Abstraction)**.** *For any* PTR *computations $\Gamma \vdash M_1, M_2 : F\sigma$, then $\Gamma \vdash M_1 \lesssim^{\text{CBPV}}_{ter} M_2$ iff $\mathbf{Tr}_{\text{PTR}}(\Gamma \vdash M_1) \subseteq \mathbf{Tr}_{\text{PTR}}(\Gamma \vdash M_2)$.*

6.2 $\mathcal{L}^{\Delta}_{\text{PTR}}$ is an automaton

$\mathcal{L}^{\Delta}_{\text{PTR}}$ turns out to be a VPA for any PTR-computation. In general, it inherits the non-elementary bounds from the λ-calculus but, for terms in canonical form (see Figure 13), we obtain an exponential bound. The main result for these canonical forms are that every term has a contextually equivalent cannonical form.

Ground Types	β	$\triangleq$	Unit | Int
Restricted Values	V_0	$\triangleq$	$x \mid () \mid \widehat{n} \mid \ell \mid \text{MkVar } V_0\ V_0$
Values	V	$\triangleq$	$V_0 \mid \text{thunk } M \mid \text{MkVar (thunk } M\text{) (thunk } M\text{)}$
Restricted Computations	M_0	$\triangleq$	force $V_0 \mid$ return $V_0 \mid M_0 V \mid$ ref $V \mid !V_0 \mid V_0 := V_0$
Computations	M	$\triangleq$	$M_0 \mid$ return $V \mid \lambda x^\sigma.M \mid$ let x^β be $V.M$
			$\mid M$ to $x^\beta.M \mid M_0$ to $x.M \mid$ case V of $(M_i)_{i\in I} \mid$ while M do M

Figure 13: The grammar for terms in canonical form

Lemma 56. *Given a CBPV computation* $\Gamma \vdash^c M : \underline{\tau}$*, there exists a computation* $\Gamma \vdash^c \mathbf{Canon}(M) : \underline{\tau}$ *in canonical form such that* $\Gamma \vdash^c M \simeq_{ctx}^{\text{CBPV}} \mathbf{Canon}(M) : \underline{\tau}$

We can then establish the finiteness of the reachable state space of $\mathcal{L}_{\text{PTR}}^{\Delta}$ for any computation.

Lemma 57. *For* PTR*-computation* $\Gamma \vdash^c M : F\sigma$*,* (Γ, σ)*-name scheme* Δ*, the set of of states reachable from* $\mathsf{C}_M^{\text{PTR},\Delta}$ *in* $\mathcal{L}_{\text{PTR}}^{\Delta}$ *is finite. If* M *is in canonical form, it is exponential in the size of* M*.*

The overview of our technique is to consider that state space of $\mathcal{L}_{\text{PTR}}^{\Delta}$ reachable from the inital configuration, but ignoring potential differences in the heap. We will obtain a bound on this state space by looking at the maximal length of paths before we reach a duplicated state. We can then account for the heap seperately, by considering how many locations can be generated along such paths.

In line with this, we say two states differ only in the heap when all components other than the heap (h) and tag (l) are the same, and that they are distinct beyond the heap when some component other than the heap or tag is different. We extend these notions analogously to configurations.

To prove Lemma 57, we will need a series of intermediate results. To simplify the statement of results in this section, we will fix a PTR-computation $\Gamma \vdash^c M : F\sigma$, (Γ, σ)-name scheme $\Delta = (\text{TBNames}, \text{CBNames}, \rho, c_0, \text{Suc}_\text{T}, \text{Suc}_\text{C})$, and let $\mathbf{C}_0 = \mathsf{C}_M^{\text{PTR},\Delta}$ and $N_O = \nu(\rho) \cup \{c_0\}$.

The first tranche of result establish states in a path which must differ only in the heap from some earlier state in the path.

Lemma 58. *Let* $\mathbf{C}$ *be a passive configuration reachable from* $\mathbf{C}_0$ *in* $\mathcal{L}_{\text{PTR}}^{\Delta}$*. Let* $\mathbf{C}'$ *be a passive state reachable from* $\mathbf{C}$ *with the same stack component* S *as* $\mathbf{C}$*, and with the stack never being shorter than* S *in an intermediate configuration. Then* $\mathbf{C}$ *and* $\mathbf{C}'$ *differ only in the heap.*

Lemma 59. *Let* $M_1 = N_1$ to x.case x of return $(), (N_2$ to y.end(i_h, η).while M do $N)_{j>0}$*. Let* $\mathbf{C} = (\langle K[M_1], c, \gamma, h, H, i_h, \eta, \mu, l\rangle, S)$ *be reachable from* $\mathbf{C}_0$*. Let* $\mathbf{C}' = (\langle K[M_1], c, \gamma', h', H', i_h', \eta', \mu', l'\rangle, S)$ *be a configuration reached from* $\mathbf{C}$ *by a path* p *which does not include a PA-action on* c*. Then* $\mathbf{C}$ *and* $\mathbf{C}'$ *differ only in the heap.*

Lemma 60. *Let* $\mathbf{C} = (\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle, S)$ *be reachable from* $\mathbf{C}_0$*, which occurs immediately before OQ-action* $g(\overrightarrow{A}, c)$ *introducing a level 2 name* f'*. Let* f *be a name which has* f' *as its originator. Let* p *be a path from* $\mathbf{C}$ *such that*

- *no answer action in* p *answers a question not occurring in* p*;*
- *no* τ*-action is produced involving reducing an occurrence of* end. *found in* $\mathbf{C}$*;*
- p *includes no unanswered PQ-actions on a level 2 name whose;*
- p *ends in active* $\mathbf{C}' = (\langle K[\text{force } f^i \overrightarrow{V}], c', \gamma', h', H', i_h', \eta', \mu', l'\rangle, S')$*.*

Then we have if $\mathbf{C}' \xrightarrow{\bar{f}(\overrightarrow{A}, d)} \mathbf{C}''$*, the states of* $\mathbf{C}$ *and* $\mathbf{C}''$ *differ only in the heap.*

We then wish to determine how long the paths reaching these repeated states can be. For this, we have results regarding the number of unanswered questions may occur before an PQ-action using a level 2 head name must occur, and the number of distinct beyond the heap states which may occur between unanswered questions.

Lemma 61. *Let $\Gamma \vdash^c M : F\sigma$ be a* PTR*-computation in canonical form, Γ-assignment ρ and continuation name c_0. Let $t \in \mathbf{Tr}(\mathsf{C}_M^{\rho,c_0})$ be a trace which does not have an unanswered PQ-action using a level 2 head name, then the number of unanswered PQ-actions in t is bounded by the size of M.*

Lemma 62. *Let M be in canonical form. Let $\mathbf{C}$ be an active configuration reachable from $\mathbf{C}_0$, which occurs immediately after a OQ-action, or is $\mathbf{C}_0$. Let $\mathbf{C}'$ be an active configuration reachable from $\mathbf{C}$ by a path p in which no OQ or PA-actions occur, and each occurrence of the while construct is reduced at most once. Then the number of intermediate configurations which are distinct beyond the heap is polynomial in M.*

We are finally ready to prove Lemma 57.

Proof of Lemma 57. To conduct this proof, we are going to exploit the fact that a state can be decomposed into the heap and tag, and 'everything else'. Formally, we decompose as follows:

$$\begin{aligned}\mathbf{Decomp}((\langle N, c, \gamma, h, H, i_h, \eta, \mu, l\rangle, S)) = \\ ((\langle N, c, \gamma, H, i_h, \eta, \mu\rangle, S), (h, l))\\ \mathbf{Decomp}((\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle, S)) = \\ ((\langle \gamma, h, H, Fn, i_h, \eta, \mu\rangle, S), (h, l))\end{aligned}$$

We will write configurations without the heap and tag as $\bar{\mathbf{C}}$. Then we can write that $\bar{\mathbf{C}} \to \bar{\mathbf{C}}'$ if there is some $\mathbf{C}, h, l, \mathbf{C}', h, l'$ and action $\mathbf{a}$ (inc. τ) s.t. $\mathbf{Decomp}(\mathbf{C}) = (\bar{\mathbf{C}}, (h, l))$, $\mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$, and $\mathbf{Decomp}(\mathbf{C}') = (\bar{\mathbf{C}}', (h', l'))$.

Let $\mathbf{Decomp}(\mathbf{C}_0) = (\bar{\mathbf{C}}_0, (h_0, l_0))$. Then define the set **Reach** inductively as follows:

- $\bar{\mathbf{C}}_0 \in \mathbf{Reach}$
- if $\bar{\mathbf{C}}$ is in reach, and $\bar{\mathbf{C}} \to \bar{\mathbf{C}}'$, then $\bar{\mathbf{C}}' \in \mathbf{Reach}$

Now, let us write $|\mathbf{Reach}|$ for the number of (distinct) states (not configurations) appearing in **Reach**. Then the set of states reachable from $\mathbf{C}_0$ in $\mathcal{L}_{SO}$ has size bounded by the product of $|\mathbf{Reach}|$, the size of the set of all possible heaps, and the number of possible values of tags. Thus, we seek to obtain a bound on $|\mathbf{Reach}|$.

First, we will consider the case where M is canonical, so we are seeking an exponetial bound.

We do this by observing that the inductive definition gives rise to a tree. The branching factor is the maximum number of successors a configuration can have, which is fixed by the size of the Int-type and the maximum arity of functions in the Σ, ρ. The maximum length of a branch in the tree (that is of a derivation that $\bar{\mathbf{C}} \in \mathbf{Reach}$), is polynomial in the size of M, as we now show. By Lemma 58, we only need to consider derivations where all OQ-actions are unanswered. Further, by Lemma 60, we only need to consider derivations in which all PQ-actions using level 2 names are answered immediately. This is as the state a PQ on a level 2 name is the same as that before the move introducing the originator of that name, so we can only reach states already on the path until the PQ is answered, as the stack plays no role until then. Lemma 61 places a bound on the number of such unanswered OQ-actions which can occur in a derivation where these PQ-actions are answered. By Lemma 59 we need only consider derivations which include a single iteration of while loops. Lemma 62 then gives a polynomial bound on the number of configurations appearing between two OQ-actions, which overall gives a polynomial bound on the length of a branch. Thus, we have an exponential bound on $|\mathbf{Reach}|$.

Recall that for any state, the size of the heap is bounded by the next location component i_h. Due to the construction above, it follows that the i_h is bounded by a polynomial in the size of M

(as, in each step of a derivation that $\mathbf{C}'_h$ is in **Reach**, the next location component can increase at most once). Thus, as the heap stores integers, the set of possible heaps is exponential in the size of M. Similarly, the tag can only be 0 or 1, which simply doubles the state space. Putting all this together yields an exponential bound on the state space. We can generate this state space in exponential time by observing that the states in **Reach** can be constructed in exponential time. This is as, when we compute configurations $\bar{\mathbf{C}}'$ s.t. $\bar{\mathbf{C}} \stackrel{\cdot}{\to} \bar{\mathbf{C}}'$, it suffices to observe that at most 1 element of the heap can be read, and so this reduces considering all possible heaps to considering the values in Int.

Now, consider what must change if M is not in canonical form. The first issue is that Lemma 61 provides a bound on the unanswered OQ-actions only if M is in canonical form. But by Lemma 56, we have that $\mathbf{Canon}(M)$ is contextually equivalent to M, thus $\mathsf{C}^{\mathrm{PTR},\Delta}_{\mathbf{Canon}(M)}$ generates the same traces as M, and so can obtain a finite bound on unanswered OQ-actions (as the blow up in going to canonical forms is finite, although non-elementary). The second issue is that Lemma 62 only applies to canonical terms. However, we can easily see that any term component in the LTS is finite, and a reduction sequence from such a finite term reaches finitely many states. Thus, it follows in the same way as above that the state space is finite. □

Lemma 63. *For* PTR*-computation* $\Gamma \vdash^c M : F\sigma$ *and* (Γ, σ)*-name scheme* Δ*, one can effectively construct a deterministic VPA accepting* $\mathbf{Tr}^{\Delta}_{\mathrm{PTR}}(\mathsf{C}^{\mathrm{PTR},\Delta}_{M})$*. If* M *is in canonical form, the construction can be carried out in exponential time.*

Proof. We can construct VPA $\mathcal{A} = (Q, \{q_0\}, \Pi, \delta, Q_F)$ with epsilon transitions, and accepting on the condition of reaching both a state and an empty stack as follows (which we recall can be converted into a standard VPA by appealing to the construction given at the begining of Section 5). Thus, from $\mathcal{A}$ one can obtain a standard VPA. The alphabet are the actions which can appear in any Δ-trace, and is partitioned so that OQ, PA-actions are internal, PQ-actions are calls, and OA-actions are returns.

We take the states (Q) to be the states of $\mathcal{L}^{\Delta_{\Gamma,\sigma}}_{\mathrm{PTR}}$ reachable from $\mathsf{C}^{\mathrm{PTR},\Delta_{\Gamma,\sigma}}_{M}$, and the initial state q_0 is the one in configuration $\mathsf{C}^{\mathrm{PTR},\Delta_{\Gamma,\sigma}}_{M}$. We take the transitions (δ) to be those transitions in $\mathcal{L}^{\Delta_{\Gamma,\sigma}}_{\mathrm{PTR}}$ from states in Q (so $(q, \mathbf{a}, q') \in \delta$ iff $q \xrightarrow{\mathbf{a}} q'$, $(q, \epsilon, q') \in \delta$ iff $q \xrightarrow{\tau} q'$, $(q, \mathbf{a}, q', \pi) \in \delta$ iff $q \xrightarrow{\mathbf{a}/\pi} q'$ and $(q, \mathbf{a}, \pi, q') \in \delta$ iff $q \xrightarrow{\mathbf{a},\pi} q'$). The stack symbols ($\Pi$) are those pushed onto the stack in the transitions in δ (so pushed in the transitions obtained from $\mathcal{L}^{\Delta_{\Gamma,\sigma}}_{\mathrm{PTR}}$) and $\bot$. As a τ-transition is the unique transition from a state, so the corresponding ϵ-transtion is the sole transition from that state, so by the discussion given at the begining of Section 5, we can safely 'squeeze' them out.

We obtain an automaton accepting exactly the complete traces by making passive states final (Q_F), as the acceptance condition ensures that the stack is empty, and so the trace has no unanswered PQ-actions (and as only passive are accepting, no unanswered OQ-actions).

It follows from Lemma 57 that the state space is finite (exponential in the size of M for canonical forms), and so the outbound transitions must also be finite (exponential for canonical forms), as there can be at most one transtion from one state to another (so quadratic in the state space). Further, we can see that the alphabet is bound by Γ, σ and the size of Int, due to the shape of possible actions. The stack alphabet is bounded by the state space, as each symbol is determined entirely by the state before it is pushed. For terms in canonical from, the construction can be done in exponential time, by first constructing the state space (as in Lemma 57), and then for each state generating the possible transitions.

The VPA is deterministic as $\mathcal{L}^{\Delta_{\Gamma,\sigma}}_{\mathrm{PTR}}$ is. In particular, for each state, action, and stack symbol, there is exactly one sucessor. □

7 Decidability, complexity and translations

Theorem 64. *Contextual approximation for the* PTR*-fragment of* CBPV *is decidable. For computations in canonical form, it is decidable in exponential time.*

Proof. From Theorem 55, testing two computations $\Gamma \vdash^c M_1, M_2 : F\sigma$ for contextual approximation can be done by comparing the complete traces generated by $\mathcal{L}^{\Delta}_{\mathrm{PTR}}$ for every possible name scheme Δ. As choice of base names in Γ, σ is arbitrary, we need only care about the Ints occurring in Γ, which gives exponentially many Δ. By Lemma 63, each comparison reduces to a language equivalence test. For canonical forms, the two VPA's are constructible in exponential time. In particular, they will be of exponential size. Because language equivalence is in P for deterministic VPA, the lemma follows. □

One can show that it is the use of level-2 names that forces us to make use of an unbounded stack. The computations that omit level-2 names are of the form $\Gamma \vdash^c M : F\sigma^1$, where each type in Γ is a σ^2 type according to the grammar given below.

$$\begin{array}{lcllcl} \sigma^2 & \triangleq & \sigma^1 \mid U\underline{\tau}^1 & \sigma^1 & \triangleq & \sigma^0 \mid \mathrm{Ref} \mid U\underline{\tau}^0 \\ \underline{\tau}^1 & \triangleq & F\sigma^2 \mid \sigma^1 \to \underline{\tau}^1 & \underline{\tau}^0 & \triangleq & F\sigma^0 \mid \sigma^0 \to \underline{\tau}^0 \\ \sigma^0 & \triangleq & \mathrm{Int} \mid \mathrm{Unit} & & & \end{array}$$

In this case one can show that the stack height is bounded and, for canonical forms, the bound is linear. Consequently, we can treat the (bounded) stack as part of the state space and convert the VPA to a finite-state machine.

The fact that our results are stated for CBPV makes it possible to specialise them to the CBN- and CBV-variants of the language, known in the literature as Idealised Algol [7] and RML [6] respectively. This can be done by translation provided it is fully abstract (preserves and reflects contextual equivalence). Our translations extend the standard translations from the CBN and CBV λ-calculus respectively [21]. The translations of types are given in the table below. For RML, a term $M : \sigma$ is translated into a computation $M^{\mathrm{RML}} : F\sigma^{\mathrm{RML}}$.

RML type	CBPV value types
$\mathrm{Int}, \mathrm{Unit}, \mathrm{Ref}$	$\mathrm{Int}, \mathrm{Unit}, \mathrm{Ref}$
$\sigma_1 \to \sigma_2$	$U({\sigma_1}^{\mathrm{RML}} \to F{\sigma_2}^{\mathrm{RML}})$

IA type	CBPV computation types
$\underline{\mathrm{expr}}, \underline{\mathrm{com}}$	$F\mathrm{Int}, F\mathrm{Unit}$
$\underline{\mathrm{var}}$	$\mathrm{Int} \to \mathrm{Int} \to F\mathrm{Int}$
$\underline{\tau_1} \to \underline{\tau_2}$	$U{\underline{\tau_1}}^{\mathrm{IA}} \to {\underline{\tau_2}}^{\mathrm{IA}}$

Remark 65. The CBN translation of $\underline{\mathrm{var}}$ into $\mathrm{Int} \to \mathrm{Int} \to F\mathrm{Int}$ uses the first argument as a boolean flag to indicate whether reading or writing will take place. The term translation ensures that, during reading, the second parameter will be ignored. For writing, the translated term will always return 0. If the first argument is different from 0 or 1, the translated term will diverge.

That the translations turn out fully abstract is not completely surprising: there are several similar results in the literature, though none of them applies to the framework we are considering, e.g. the results from [21] are phrased for higher-order references and observing output instead of termination. Our fully abstract model $\mathcal{L}_{\mathrm{CBPV}}$ plays a crucial role in establishing the full abstraction of our translations.

Theorem 66. *Let* $\lesssim^{\mathrm{IA}}_{ter}$, $\lesssim^{\mathrm{RML}}_{ter}$ *be the notions of contextual approximation in* IA *and* RML *respectively. For IA terms* $\Gamma \vdash M_1 \lesssim^{\mathrm{IA}}_{ter} M_2$ *iff* ${\Gamma^{\mathrm{IA}}} \vdash^c {M_1}^{\mathrm{IA}} \lesssim^{\mathrm{CBPV}}_{ter} {M_2}^{\mathrm{IA}}$, *and for RML terms* $\Gamma \vdash M_1 \lesssim^{\mathrm{RML}}_{ter} M_2$ *iff* ${\Gamma^{\mathrm{RML}}} \vdash^c {M_1}^{\mathrm{RML}} \lesssim^{\mathrm{CBPV}}_{ter} {M_2}^{\mathrm{RML}}$.

The above result means that Theorem 64 subsumes existing decidability results for IA and RML from [10, 12], as the translations of third-order IA types [10] and O-strict RML types [12] belong to the PTR fragment. Consequently, the present results can be seen as an operational explanation of the earlier results for CBN and CBV.

8 Conclusion

We demonstrated an approach to proving decidability results for contextual equivalence by deriving decidable automata models from labelled transitions systems through a series of relatively easy optimisations. The configurations of these automata retain operational character, which makes them suitable for specification of further program analysis tasks. Since operational game models are in general easier to construct and understand, we believe the approach is likely to turn out fruitful when it comes to analyzing more complicated frameworks in the future.

References

[1] H. Nickau, "Hereditarily Sequential Functionals," in *Proceedings of LFCS*, ser. LNCS, vol. 813. Springer, 1994, pp. 253–264.

[2] S. Abramsky, R. Jagadeesan, and P. Malacaria, "Full Abstraction for PCF," *Inf. Comput.*, vol. 163, no. 2, pp. 409–470, 2000.

[3] J. M. E. Hyland and C. L. Ong, "On Full Abstraction for PCF: I, II, and III," *Inf. Comput.*, vol. 163, no. 2, pp. 285–408, 2000.

[4] J. Laird, "A Fully Abstract Trace Semantics for General References," in *Proceedings of ICALP*, ser. LNCS, vol. 4596. Springer, 2007, pp. 667–679.

[5] G. Jaber, "Operational Nominal Game Semantics," in *Proceedings of FoSSaCS*, ser. LNCS, vol. 9034. Springer, 2015, pp. 264–278.

[6] S. Abramsky and G. McCusker, "Call-by-Value Games," in *Proceedings of CSL*, ser. LNCS, vol. 1414. Springer, 1997, pp. 1–17.

[7] ——, "Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions," in *Algol-like languages*, P. W. O'Hearn and R. D. Tennent, Eds. Birkhaüser, 1997, pp. 297–329.

[8] S. Abramsky, D. R. Ghica, A. S. Murawski, and C.-H. L. Ong, "Applying Game Semantics to Compositional Software Modelling and Verification," in *Proceedings of TACAS*, ser. LNCS, vol. 2988. Springer, 2004, pp. 421–435.

[9] D. R. Ghica and G. McCusker, "The regular language semantics of second-order Idealized Algol," *Theor. Comput. Sci.*, vol. 309, pp. 469–502, 2003.

[10] A. S. Murawski and I. Walukiewicz, "Third-order Idealized Algol with iteration is decidable," *Theor. Comput. Sci.*, vol. 390, no. 2-3, pp. 214–229, Jan. 2008.

[11] A. S. Murawski, "Functions with local state: Regularity and Undecidability," *Theor. Comput. Sci.*, vol. 338, no. 1-3, pp. 315–349, 2005.

[12] D. Hopkins, A. S. Murawski, and C.-H. L. Ong, "A Fragment of ML Decidable by Visibly Pushdown Automata," in *Proceedings of ICALP*, ser. LNCS, vol. 6756. Springer, 2011, pp. 149–161.

[13] S. B. Lassen and P. B. Levy, "Typed Normal Form Bisimulation," in *Proceedings of CSL*, ser. LNCS, vol. 4646. Springer, 2007, pp. 283–297.

[14] P. B. Levy, "Call-by-push-value: Decomposing call-by-value and call-by-name," *High. Order Symb. Comput.*, vol. 19, no. 4, pp. 377–414, 2006.

[15] G. Jaber, "SyTeCi: automating contextual equivalence for higher-order programs with references," *Proc. ACM Program. Lang.*, vol. 4, no. POPL, pp. 59:1–59:28, 2019.

[16] V. Koutavas, Y. Y. Lin, and N. Tzevelekos, "From Bounded Checking to Verification of Equivalence via Symbolic Up-to Techniques," in *Proceedings of TACAS*, ser. LNCS, vol. 13244. Springer, 2022, pp. 178–195.

[17] G. Jaber and A. S. Murawski, "Compositional relational reasoning via operational game semantics," in *Proceedings of LICS*. IEEE, 2021, pp. 1–13.

[18] A. M. Pitts and I. D. B. Stark, "Operational Reasoning for Functions with Local State," in *Higher-Order Operational Techniques in Semantics*, A. D. Gordon and A. M. Pitts, Eds. CUP, 1998, pp. 227–273.

[19] D. Dreyer, G. Neis, and L. Birkedal, "The impact of higher-order state and control effects on local relational reasoning," *J. Funct. Program.*, vol. 22, no. 4-5, pp. 477–528, 2012.

[20] D. Biernacki, S. Lenglet, and P. Polesiuk, "A Complete Normal-Form Bisimilarity for State," in *Proceedings of FoSSaCS*, ser. LNCS, vol. 11425. Springer, 2019, pp. 98–114.

[21] P. B. Levy, *Call-By-Push-Value. A Functional/Imperative Synthesis*, ser. Semantics Structures in Computation. Springer, 2004, vol. 2.

[22] J. C. Reynolds, "The Essence of Algol," in *Algol-like languages*, P. W. O'Hearn and R. D. Tennent, Eds. Birkhaüser, 1997, pp. 67–88.

[23] C. Talcott, "Reasoning about Programs With Effects," *Electron. Notes Theor. Comput. Sci.*, vol. 14, pp. 301–314, 1998.

[24] G. Jaber and A. S. Murawski, "Complete trace models of state and control," in *Proceedings of ESOP*, ser. LNCS, vol. 12648. Springer, 2021, pp. 348–374.

[25] S. Abramsky, K. Honda, and G. McCusker, "A Fully Abstract Game Semantics for General References," in *Proceedings of LICS*. IEEE, 1998, pp. 334–344.

[26] J. Laird, "Full Abstraction for Functional Languages with Control," in *Proceedings of LICS*. IEEE, 1997, pp. 58–67.

[27] R. Alur and P. Madhusudan, "Visibly Pushdown Languages," in *Proceedings of STOC*. ACM, 2004, pp. 202–211.

[28] D. Hopkins, "Game Semantics Based Equivalence Checking of Higher-Order Programs," Ph.D. dissertation, Oxford University, UK, 2016.

[29] P. B. Levy, "Call-by-push-value," Ph.D. dissertation, Queen Mary University of London, UK, 2001. [Online]. Available: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.369233

A CIU lemma

A.1 Proof of CIU lemma

To prove this result, we will show that $\lesssim_{ter}^{\mathrm{CBPV}(ciu)}$ is a precongruence (meaning that the relation respects the inductive construction of terms). We will prove a couple of helper lemmata to do this. The first handles the case of a computation appearing in a value context.

Lemma 67. *Suppose* $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$. *Let* V_C *be a value context s.t.* $\Gamma' \vdash^v V_C[M_1] : V_C[M_2]$. *Define*

$$C_V \triangleq \text{force } \bullet \mid \text{return } \bullet \mid \text{let } x \text{ be } \bullet.M \mid M \bullet \mid !\bullet \mid \bullet := V$$

Then we have (where types match) $\Gamma \vdash^c C_V[V_C[M_1]] \lesssim_{ter}^{\mathrm{CBPV}(ciu)} C_V[V_C[M_2]]$

Proof. Take K, γ, h such that $(K[C_V[V_C[M_1]]\{\gamma\}], h) \Downarrow_{ter}$. Write M_i^γ for $M_1\{\gamma\}$. We need to show $(K[C_V[V_C[M_2]]\{\gamma\}], h) \Downarrow_{ter}$.

We shall prove the more general result that for any M, V_C', h', we have for fresh z that $(M\{V_C'[M_1^\gamma]/z\}, h') \Downarrow_{ter}$ implies $(M\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$. We get the desired result by setting $M = K[C_V[z]]$.

We use induction on the number of steps k in $(M\{V_C'[M_1^\gamma]/z\}, h') \Downarrow_{ter}$.

- If $k = 0$ then $M = \text{return } z$, in which case $M\{V_C'[M_2^\gamma]/z\}$ is terminal, so $(M\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$.
- If $k > 0$, we have the following case:
  - $(M = K'[N]$ and $(K'[N], h') \to (K'[N'], h''))$
    Then $(K'[N']\{V_C'[M_1^\gamma]/z\}, h') \Downarrow_{ter}$ in $(k-1)$ steps. So by IH $(K'[N']\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$ in $(k-1)$ steps, and as $(M\{V_C'[M_2^\gamma]/z\}, h') \to K'[N']\{V_C'[M_2^\gamma]/z\}, h')$ we are done.
  - $(M = K'[\text{force } z])$ Then $V_C' = \text{thunk } \bullet$.
    We have that $(K'[M_1^\gamma]\{V_C'[M_1^\gamma]/z\}, h') \Downarrow_{ter}$ in $(k-1)$ steps. By the IH $(K'[M_1^\gamma]\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$. Because $M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$, this implies $(K'[M_2^\gamma]\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$, and as $(M\{V_C'[M_2^\gamma]/z\}, h') \to (K'[M_2^\gamma]\{V_C'[M_2^\gamma]/z\}, h')$ we are done.
  - $(M = K'[!z])$ Then $V_C' = \text{MkVar thunk } \bullet\ V$.
    We have that $(K'[\text{force thunk } M_1^\gamma]\{V_C'[M_1^\gamma]/z\}, h') \Downarrow_{ter}$ in $(k-1)$ steps. By the IH $(K'[\text{force thunk } M_1^\gamma]\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$. Because $M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$, this implies $(K'[\text{force thunk } M_2^\gamma]\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$, and as $(M\{V_C'[M_2^\gamma]/z\}, h') \to (K'[\text{force thunk } M_2^\gamma]\{V_C'[M_2^\gamma]/z\}, h')$ we are done.
  - $(M = K'[z := V'])$ Then $V_C' = \text{MkVar } V \text{ thunk } \bullet$.
    We have that $(K'[(\text{force thunk } M_1^\gamma)V']\{V_C'[M_1^\gamma]/z\}, h') \Downarrow_{ter}$ in $(k-1)$ steps. By the IH $(K'[(\text{force thunk } M_1^\gamma)V']\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$. Because $M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$, this implies $(K'[(\text{force thunk } M_2^\gamma)V']\{V_C'[M_2^\gamma]/z\}, h') \Downarrow_{ter}$, and as $(M\{V_C'[M_2^\gamma]/z\}, h') \to (K'[(\text{force thunk } M_2^\gamma)V']\{V_C'[M_2^\gamma]/z\}, h')$ we are done.

□

Lemma 68 (while CIU)**.** *Suppose* $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$. *Then* $\Gamma \vdash^c \text{while } M_1 \text{ do } M \lesssim_{ter}^{\mathrm{CBPV}(ciu)} \text{while } M_2 \text{ do } M$ *and* $\Gamma \vdash^c \text{while } M \text{ do } M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} \text{while } M \text{ do } M_2$.

Proof. We will show the first case only, as the other is analogous. Let the number of iterations of a loop be the number of times we reach a configuration of the form $(K'[(\text{while } M_1 \text{ do } M)\{\gamma'\}], h')$ during reduction. We prove by induction on k that for any γ, K, h if $(K[(\text{while } M_1 \text{ do } M)\{\gamma\}], h) \Downarrow_{ter}$ in k iterations, then $(K[(\text{while } M_2 \text{ do } M)\{\gamma\}], h) \Downarrow_{ter}$.

- $k = 1$ The base case occurs when $(M_1\{\gamma\}, h) \to^* (\text{return } \widehat{0}, h')$. This means

$$\begin{array}{lll}(K[(\text{while } M_1 \text{ do } M)\{\gamma\}], h) & \to & (K[(M_1 \text{ to } x.\text{case } x \text{ of return } (), (M \text{ to } y.\text{while } M_1 \text{ do } M)_{i>0})\{\gamma\}], h) \\ & \to^* & (K[(\text{return } \widehat{0} \text{ to } x.\text{case } x \text{ of return } (), (M \text{ to } y.\text{while } M_1 \text{ do } M)_{i>0})\{\gamma\}], h') \\ & \to & (K[(\text{case } \widehat{0} \text{ of return } (), (M \text{ to } y.\text{while } M_1 \text{ do } M)_{i>0})\{\gamma\}], h') \\ & \to & (K[\text{return } ()], h')\end{array}$$

  So $(K[\text{return } ()], h') \Downarrow_{ter}$. Taking $K' = K[\bullet \text{ to } x.(\text{case } x \text{ of return } (), (M \text{ to } y.\text{while } M_2 \text{ do } M)_{i>0})\{\gamma\}]$, by following the above reduction, we have $(K'[M_1\{\gamma\}], h) \Downarrow_{ter}$, and so as $\Gamma \vdash^c M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} M_2$, we have $(K'[M_2\{\gamma\}], h)$. As $(K[(\text{while } M_2 \text{ do } M)\{\gamma\}], h) \to (K'[M_2\{\gamma\}], h)$ we are done.

- $k > 1$ As we are not on the last iteration, $(M_1\{\gamma\}, h) \to^* (\text{return } \widehat{n}, h')$, with $n > 0$. This means

$$\begin{array}{lll}(K[(\text{while } M_1 \text{ do } M)\{\gamma\}], h) & \to & (K[(M_1 \text{ to } x.\text{case } x \text{ of return } (), (M \text{ to } y.\text{while } M_1 \text{ do } M)_{i>0})\{\gamma\}], h) \\ & \to^* & (K[(\text{return } \widehat{n} \text{ to } x.\text{case } x \text{ of return } (), (M \text{ to } y.\text{while } M_1 \text{ do } M)_{i>0})\{\gamma\}], h'') \\ & \to & (K[(\text{case } \widehat{n} \text{ of return } (), (M \text{ to } y.\text{while } M_1 \text{ do } M)_{i>0})\{\gamma\}], h'') \\ & \to & (K[(M \text{ to } y.\text{while } M_1 \text{ do } M)\{\gamma\}], h'') \\ & \to^* & (K[(\text{while } M_1 \text{ do } M)\{\gamma\}], h')\end{array}$$

  Thus, $(K[(\text{while } M_1 \text{ do } M)\{\gamma\}], h') \Downarrow_{ter}$ in $k' < k$ iterations, so by the I.H., $(K[(\text{while } M_2 \text{ do } M)\{\gamma\}], h') \Downarrow_{ter}$. Taking $K' = K[\bullet \text{ to } x.(\text{case } x \text{ of return } (),$
  $(M \text{ to } y.\text{while } M_2 \text{ do } M)_{i>0})\{\gamma\}]$, by following the above reduction, we have $(K'[M_1\{\gamma\}], h) \to^*$ $(K[(\text{while } M_2 \text{ do } M)\{\gamma\}], h')$, and so $(K'[M_1\{\gamma\}], h) \Downarrow_{ter}$. Thus as $\Gamma \vdash^c M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} M_2$, we have $(K'[M_2\{\gamma\}], h) \Downarrow_{ter}$, and as $(K[(\text{while } M_2 \text{ do } M)\{\gamma\}], h) \to (K'[M_2\{\gamma\}], h)$ we are done.

In one step □

Lemma 69. *Suppose* $\Gamma \vdash^c M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} M_2$. *Then, as long as the types work, we have:*

- $\Gamma \vdash^c M_1 \text{ to } x.M \lesssim_{ter}^{\text{CBPV}(ciu)} M_2 \text{ to } x.M$, $\Gamma \vdash^c M_1 V \lesssim_{ter}^{\text{CBPV}(ciu)} M_2 V$
- $\Gamma \vdash^c \lambda x.M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} \lambda x.M_2$,
  $\Gamma \vdash^c \text{case } V \text{ of } (M^i)_{i<j}, M_1, (M^i)_{j<i} \lesssim_{ter}^{\text{CBPV}(ciu)} \text{case } V \text{ of } (M^i)_{i<j}, M_2, (M^i)_{j<i}$,
  $\Gamma \vdash^c \text{let } x \text{ be } V.M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} \text{let } x \text{ be } V.M_2$,
  $\Gamma \vdash^c M \text{ to } x.M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} M \text{ to } x.M_2$.

Proof. We handle the classes separately, as the proofs for the cases within each class are similar.

- These cases are trivial, as these constructs are those used in evaluation contexts.
- We will show the first case only, as the others follow analogously. Let γ, K, h be s.t. $(K[(\lambda x.M_1)\{\gamma\}], h) \Downarrow_{ter}$. Observe that $(\lambda x.M_i)\{\gamma\} = \lambda x.(M_i\{\gamma\})$. Due to the types, we must have $K = K'[\bullet V]$. Thus, we have $(K'[M_1\{\gamma \cdot [x \mapsto V]\}], h) \Downarrow_{ter}$, and so as $\Gamma \vdash^c M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} M_2$, we have $(K'[M_2\{\gamma \cdot [x \mapsto V]\}], h) \Downarrow_{ter}$. Thus $(K[(\lambda x.M_2)\{\gamma\}], h) \Downarrow_{ter}$.

□

Proof of Lemma 3. The left-to-right implication follows directly from the fact that testing with γ, K, h is a special case of testing with an arbitrary context. The right-to-left direction follows from the fact that if $\Gamma \vdash^c M_1 \lesssim_{ter}^{\text{CBPV}(ciu)} M_2$ and $\Gamma' \vdash^c C[M_1], C[M_2] : F\sigma$, then $\Gamma' \vdash^c C[M_1] \lesssim_{ter}^{\text{CBPV}(ciu)} C[M_2]$. This result can be proved using structural induction on contexts, using the above three lemmata. □

A.2 Proof of Lemma 4

Proof of Lemma 4. The left-to-right implication is trivial. For the right-to-left implication, we can use Lemma 3, so we need only show that $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$. Let h, K, γ be s.t. $(K[M_1\{\gamma\}], h) \Downarrow_{ter}$. Now, observe that $K = K'[\bullet\, V_1 \cdots V_k]$. Let $C = K'[\text{let } x_1 \text{ be } V_1. \cdots \text{let } x_k \text{ be } V_k.\bullet]$. Observe that for any M, $(K[M\{\gamma\}], h) \Downarrow_{ter}$ iff $(C[M], h) \Downarrow_{ter}$. From the fact that $\Gamma, (x_1, \sigma_1), \cdots, (x_k, \sigma_k) \vdash^c M_1\, x_1 \cdots x_k \lesssim_{ter}^{\mathrm{CBPV}} M_2\, x_1 \cdots x_k$, it follows $(K[M_2\{\gamma\}], h) \Downarrow_{ter}$, and so $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$. □

A.3 CBPV equational theory

It will be useful to be aware of the equational theory of CBPV, and its relation to contextual equivalence, to facilitate some of our proofs regarding canonical forms and translations. Our equational theory is that presented by Levy [14], specialised to the constructions we include, and extended with the appropriate rules for state. We present three relation $\rightsquigarrow_\beta$, $\rightsquigarrow_\eta$ and $\rightsquigarrow_\varsigma$ below, implicitly requiring the two sides to have the same type when typed under appropriate $\Sigma; \Gamma$.

		β-rules
$(\lambda x.M)V$	$\rightsquigarrow_\beta$	$M\{V/x\}$
let x be V.M	$\rightsquigarrow_\beta$	$M\{V/x\}$
return V to x.M	$\rightsquigarrow_\beta$	$M\{V/x\}$
force thunk M	$\rightsquigarrow_\beta$	M
case $\widehat{n}$ of $(M_i)_{i\in I}$	$\rightsquigarrow_\beta$	M_n
!MkVar V_1 V_2	$\rightsquigarrow_\beta$	force V_1
MkVar V_1 $V_2 := V$	$\rightsquigarrow_\beta$	(force V_1)V
while M do N	$\rightsquigarrow_\beta$	M to x.case x of return $(), (N$ to y.while M do $N)_i$
		x, y are fresh
		η-rules
M to x.return x	$\rightsquigarrow_\eta$	M
M to x.return $()$	$\rightsquigarrow_\eta$	M where x : Unit
$\lambda x.M\, x$	$\rightsquigarrow_\eta$	M x not free in M
thunk force V	$\rightsquigarrow_\eta$	V
MkVar (thunk !V) (thunk $\lambda y.V := y$)	$\rightsquigarrow_\eta$	V y not free in V
		ς-rules
$(N_1$ to $x.N_2)$ to $y.N_3$	$\rightsquigarrow_\varsigma$	N_1 to $x.(N_2$ to $y.N_3)$ x not free in N_3
M to $x.\lambda y.N$	$\rightsquigarrow_\varsigma$	$\lambda y.(M$ to $x.N)$ y not free in M
(case V of $(M_i)_i$) to $x.M$	$\rightsquigarrow_\varsigma$	case V of $(M_i$ to $x.M)_i$
case V of $(\lambda x.M_i)_i$	$\rightsquigarrow_\varsigma$	λx.case V of $(M_i)_i$ x not free in V
Ω to $x.M$	$\rightsquigarrow_\varsigma$	Ω

We can obtain notions of equality from $\rightsquigarrow_\beta$, $\rightsquigarrow_\eta$, and $\rightsquigarrow_\varsigma$, namely $=_\beta$ (β-equality), $=_\eta$ (η-equality), and $=_\varsigma$ (sequence or ς-equality) by making each a symmetric, transitive, congruence (closing under the constructions of terms). We can also obtain the equality from their union, $=_{\beta\eta\varsigma}$ ($\beta\eta\varsigma$-equality).

The key property we are interested in is that $\beta\eta\varsigma$-equality implies contextual equivalence.

Lemma 70. *If* $\Gamma \vdash^c M_1, M_2 : \underline{\tau}$ *are* CBPV *terms and* $M_1 =_{\beta\eta\varsigma} M_2$*, then* $\Gamma \vdash^c M_1 \cong_{ter}^{\mathrm{CBPV}} M_2$.

Proof. For this, it suffice to check that $\rightsquigarrow_\beta$, $\rightsquigarrow_\eta$, and $\rightsquigarrow_\varsigma$ imply contextual equivalence, as the construction for a congruence and equality relation preserve this. More specifically, we will do this when $\rightsquigarrow_\beta$, $\rightsquigarrow_\eta$, and $\rightsquigarrow_\varsigma$ relate computations. When we have rules relating values, we have to show if $V_1 \rightsquigarrow V_2$, that $\Gamma \vdash^c C_V[V_1] \cong_{ter}^{\mathrm{CBPV}} C_V[V_2]$ for any C_V defined as in Lemma 67. We will do this by appealing to the CIU Lemma (3), and showing that $M_1 \rightsquigarrow_\beta M_2$ (or $M_1 \rightsquigarrow_\eta M_2$ or $M_1 \rightsquigarrow_\varsigma M_2$) imply $\Gamma \vdash^c M_1 \cong_{ter}^{\mathrm{CBPV}(ciu)} M_2$, as this reduces the contexts we must consider. As

these are similar in each case, we will show a few enlightening cases only. Let Σ, h, K, γ be s.t. $h : \Sigma$, $\Sigma \vdash^k K : \underline{\tau} \implies F\sigma$, and $\Sigma \vdash \gamma : \Gamma$.

- Now, suppose $M_1 \rightsquigarrow_\beta M_2$ by the first rule, so $M_1 = (\lambda x.M)V$ and $M_2 = M\{V/x\}$, where x is not free in γ. Then it is the case that

$$\begin{aligned}(K[M_1\{\gamma\}], h) = (K[((\lambda x.M)V)\{\gamma\}], h) &= (K[(\lambda x.M\{\gamma\})V\{\gamma\}], h) \\ &\to (K[(M\{\gamma\})\{V\{\gamma\}/x\}], h) = (K[(M\{V/x\})\{\gamma\}], h) = (K[M_2\{\gamma\}], h)\end{aligned}$$

  Thus, we obtain that $(K[M_1\{\gamma\}], h) \Downarrow_{ter}$ iff $(K[M_2\{\gamma\}], h) \Downarrow_{ter}$. Thus, $\Gamma \vdash^c M_1 \cong_{ter}^{\mathrm{CBPV}(ciu)} M_2$.

- Instead suppose that $M_1 \rightsquigarrow_\eta M_2$ by the final η-rule. Then it must be the case that $M_1 = C_V[\mathrm{MkVar}\ (\mathrm{thunk}\ !V)\ (\mathrm{thunk}\ \lambda y.V := y)]$ and $M_2 = C_V[V]$. Let $V' = \mathrm{MkVar}\ (\mathrm{thunk}\ !V)\ (\mathrm{thunk}\ \lambda y.V := y)$. We will proceed to show that for any M,h', $(M\{V'\{\gamma\}/z\}, h') \Downarrow_{ter}$ iff $(M\{V\{\gamma\}/z\}, h') \Downarrow_{ter}$, from which our result follows by taking $M = C_V[z]$. We first show that $(M\{V'\{\gamma\}/z\}, h') \Downarrow_{ter}$ implies $M\{V/z\}\{\gamma\}$ by induction on the number of steps k in $(M\{V'\{//\gamma\}z\}, h') \Downarrow_{ter}$.

  If $k = 0$, then $M = \mathrm{return}\ z$, so both $(M\{V'\{\gamma\}/z\}, h')$ and $(M\{V\{\gamma\}/z\}, h')$ are terminal. If $k > 0$ we proceed by case.

  - $M = K'[N]$ and $(K'[N], h') \to (K'[N'], h')$. Thus $(K'[N']\{V'\{\gamma\}/z\}, h') \Downarrow_{ter}$ in $(k-1)$ steps, so by the I.H. $(K'[N']\{V\{\gamma\}/z\}, h') \Downarrow_{ter}$ in $(k-1)$ steps, so $(M\{V\{\gamma\}/z\}, h') \Downarrow_{ter}$.
  - $M = K'[!z]$. Then

$$\begin{aligned}(M\{V'\{\gamma\}/z\}, h') &\to (K'[!V'\{\gamma\}]\{V'\{\gamma\}/z\}, h') \\ &= (K'[!\mathrm{MkVar}\ (\mathrm{thunk}\ !V\{\gamma\})\ (\mathrm{thunk}\ \lambda y.V\{\gamma\} := y)]\{V'\{\gamma\}/z\}, h') \\ &\to (K'[\mathrm{force}\ \mathrm{thunk}\ !V\{\gamma\}]\{V'\{\gamma\}/z\}, h') \\ &\to (K'[!V\{\gamma\}]\{V'\{\gamma\}/z\}, h')\end{aligned}$$

    Now, by the I.H, we have $(K'[!V\{\gamma\}]\{V\{\gamma\}/z\}, h') \Downarrow_{ter}$, so $(M\{V\{\gamma\}/z\}, h') \Downarrow_{ter}$.
  - $M = K'[z := W]$ Follows same pattern as above.

  This proves one direction. The other is by a similar induction on the number of steps in $(M\{V\{//\gamma\}z\}, h') \Downarrow_{ter}$.

- Instead suppose the $M_1 \rightsquigarrow_\varsigma M_2$ by the first ς-rule. Then $M_1 = (N_1\ \mathrm{to}\ x.N_2)\ \mathrm{to}\ y.N_3$ and $M_2 = N_1\ \mathrm{to}\ x.(N_2\ \mathrm{to}\ y.N_3)$ where x is not free in γ. Now, if $(N_1\{\gamma\}, h) \not\Downarrow_{ter}$ then clearly $(K[M_1\{\gamma\}], h) \not\Downarrow_{ter}$ and $(K[M_2\{\gamma\}], h) \not\Downarrow_{ter}$. In the case $(N_1\{\gamma\}, h) \Downarrow_{ter}$, assume that $(N_1\{\gamma\}, h) \to^* (\mathrm{return}\ V_1, h')$. Then

$$\begin{aligned}(K[M_1\{\gamma\}], h) &= (K[(N_1\{\gamma\}\ \mathrm{to}\ x.N_2\{\gamma\})\ \mathrm{to}\ y.(N_3\{\gamma\})], h) \\ &\to^* (K[(\mathrm{return}\ V_1\{\gamma\}\ \mathrm{to}\ x.N_2\{\gamma\})\ \mathrm{to}\ y.(N_3\{\gamma\})], h') \\ &\to (K[N_2\{\gamma\}\{V_1\{\gamma\}/x\}\ \mathrm{to}\ y.(N_3\{\gamma\})], h') \\ &= (K[N_2\{V_1/x\}\{\gamma\}\ \mathrm{to}\ y.(N_3\{\gamma\})], h')\end{aligned}$$

  and

$$\begin{aligned}(K[M_2\{\gamma\}], h) &= (K[N_1\{\gamma\}\ \mathrm{to}\ x.(N_2\{\gamma\}\ \mathrm{to}\ y.(N_3\{\gamma\}))], h) \\ &\to^* (K[\mathrm{return}\ V_1\{\gamma\}\ \mathrm{to}\ x.(N_2\{\gamma\}\ \mathrm{to}\ y.(N_3\{\gamma\}))], h') \\ &\to (K[N_2\{\gamma\}\{V_1\{\gamma\}/x\}\ \mathrm{to}\ y.(N_3\{\gamma\}\{V_1\{\gamma\}/x\})], h') \\ &= (K[N_2\{\gamma\}\{V_1\{\gamma\}/x\}\ \mathrm{to}\ y.(N_3\{\gamma\})], h') \\ &= (K[N_2\{V_1/x\}\{\gamma\}\ \mathrm{to}\ y.(N_3\{\gamma\})], h')\end{aligned}$$

  Thus, we obtain that $(K[M_1\{\gamma\}], h) \Downarrow_{ter}$ iff $(K[M_2\{\gamma\}], h) \Downarrow_{ter}$. Thus, $\Gamma \vdash^c M_1 \cong_{ter}^{\mathrm{CBPV}(ciu)} M_2$.

□

From this it follows that β- η- and ς-equality imply contextual equivalence.

Lemma 71. *If* $\Gamma \vdash^c M_1, M_2 : \underline{\tau}$ *are* CBPV *terms and* $M_1 =_\beta M_2$*, then* $\Gamma \vdash^c M_1 \cong_{ter}^{\text{CBPV}} M_2$.

Lemma 72. *If* $\Gamma \vdash^c M_1, M_2 : \underline{\tau}$ *are* CBPV *terms and* $M_1 =_\eta M_2$*, then* $\Gamma \vdash^c M_1 \cong_{ter}^{\text{CBPV}} M_2$.

Lemma 73. *If* $\Gamma \vdash^c M_1, M_2 : \underline{\tau}$ *are* CBPV *terms and* $M_1 =_\varsigma M_2$*, then* $\Gamma \vdash^c M_1 \cong_{ter}^{\text{CBPV}} M_2$.

B Correctness and Soundness

B.1 Proof of Correctness (Lemma 20)

To prove this we will consider the 'composite interaction' of a 'compatible' term configuration and context configuration.

Definition 74. A *composite configuration* $\mathbf{D}$ is $(\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ with M a term, c a continuation name, γ_P, γ_O two environments, ϕ a set of names, h_P, h_O two heaps H_P, H_O two histories, Fn a set of available thunk names and S_P, S_O two stacks (with element of the form $(c, (K, c'))$).

We write $\circ$ for the final continuation name, used by Opponent to answer the resulting value of the whole interaction.

Definition 75. A *valid composite configuration* $\mathbf{D}$ is $(\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ with:

- $\text{dom}(\gamma_P) \cap \text{dom}(\gamma_O) = \varnothing$ and $\circ \notin \text{dom}(\gamma_P) \cup \text{dom}(\gamma_O)$;
- $\text{dom}(\gamma_O) = \text{dom}(H_P) \cap \text{TNames}$ and $\text{dom}(\gamma_P) = \text{dom}(H_O) \cap \text{TNames}$.
- $\circ \in \text{dom}(H_O)$
- If $(c, (K, c'))$ in S_P then $c \in \text{dom}(H_O) \cap \phi$ and if $(c, (K, c'))$ in S_O then $c \in \text{dom}(H_P) \cap \phi$
- $\text{codom}(H_P) = \mathcal{P}(\text{dom}(\gamma_P))$ and $\text{codom}(H_O) = \mathcal{P}(\text{dom}(\gamma_O))$
- $\text{dom}(\gamma_P) \cup \text{dom}(\gamma_O) = \phi$;
- $\gamma_P \cdot \gamma_O$ is well-typed;
- $c \in \phi \cup \{\circ\}$ with $c : \sigma$ and $\vdash^c M : F\sigma$;
- For $c \neq \circ$ either $S_P = (c, (K, c')) : S'_P$ or $S_O = (c, (K, c')) : S'_O$
- $\text{dom}(h_P) \cap \text{dom}(h_O) = \varnothing$;
- No continuation name appears twice in S_P or S_O
- $S_P \; \mathcal{R}_\mathcal{S} \; S_O$ where $\mathcal{R}_\mathcal{S}$ is the least relation s.t.:

$$
\begin{array}{rcl}
\bot & \mathcal{R}_\mathcal{S} & \bot \\
\bot & \mathcal{R}_\mathcal{S} & (c, (K, \circ)) : \bot \\
(c, (K, c')) : S & \mathcal{R}_\mathcal{S} & (c', (K', c''))S' \\
 & & \text{if } S \; \mathcal{R}_\mathcal{S} \; (c', (K', c'')) : S' \\
(c', (K', c'')) : S & \mathcal{R}_\mathcal{S} & (c, (K, c')) : S' \\
 & & \text{if } (c', (K', c'')) : S \; \mathcal{R}_\mathcal{S} \; S'
\end{array}
$$

$(P\tau)$ $(\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O) \xrightarrow{\tau} (\langle N, c, \gamma_P, \gamma_O, \phi, h'_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$
when $(M, h_P) \to (N, h'_P)$, and $S_O = (c, (K, c')) : S'_O$

(PA) $(\langle \text{return } V, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, (c, (K, c')) : S_O) \xrightarrow{\bar{c}(A)}$
$(\langle K[\text{return } A], c', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \nu(A), h_P, h_O, H_P, H_O \cdot [\nu(A) \mapsto Fn], H_P(c) \uplus \nu(A)\rangle, S_P, S_O)$
when $c : \sigma$, $(A, \gamma') \in \mathbf{AVal}_\sigma(V)$

(PQ) $\langle K[(\text{force } f)\overrightarrow{V}], c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, (c, (K, c')) : S_O) \xrightarrow{\bar{f}(\overrightarrow{A}, c')}$
$(\langle \text{force } U\overrightarrow{A}, c'', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \phi', h_P, h_O, H_P, H_O \cdot [\phi' \mapsto Fn], H_P(f) \uplus \nu(\overrightarrow{A})\rangle, (c'', (K, c)) : S_P, (c, (K, c')) : S_O)$
when $\overrightarrow{V}$ is maximal, $f : \underline{\tau}, f \in Fn$, $U = \gamma_O(f)$, $(\overrightarrow{A}, \gamma') \in \mathbf{AVal}(\overrightarrow{V})$, $\sigma = \mathbf{RType}(\underline{\tau})$, $c'' : \sigma$ and $\phi' = \nu(\overrightarrow{A}) \uplus \{c'\}$

$(O\tau)$ $(\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O) \xrightarrow{\tau}$
$(\langle N, c, \gamma_P, \gamma_O, \phi, h_P, h'_O, H_P, H_O, Fn\rangle, S_P, S_O)$
when $(M, h_O) \to (N, h'_O)$, and $S_P = (c, (K, c')) : S'_P$ or $c = \circ$

(OA) $(\langle \text{return } V, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, (c, (K, c')) : S_P, S_O) \xrightarrow{c(A)}$
$(\langle K[\text{return } A], c', \gamma_P, \gamma_O \cdot \gamma', \phi \uplus \nu(A), h_P, h_O, H_P \cdot [\nu(A) \mapsto Fn], H_O, H_O(c) \uplus \nu(A)\rangle, S_P, S_O)$
when $c : \sigma$, $(A, \gamma') \in \mathbf{AVal}_\sigma(V)$

(OQ) $\langle K[(\text{force } f)\overrightarrow{V}], c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, (c, (K, c')) : S_P, S_O) \xrightarrow{f(\overrightarrow{A}, c')}$
$(\langle \text{force } U\overrightarrow{A}, c'', \gamma_P, \gamma_O \cdot \gamma', \phi \uplus \phi', h_P, h_O, H_P \cdot [\phi' \mapsto Fn], H_O, H_O(f) \uplus \nu(\overrightarrow{A})\rangle, (c, (K, c')) : S_P, (c'', (K, c)) : S_O)$
when $\overrightarrow{V}$ is maximal, $f : \underline{\tau}, f \in Fn$, $U = \gamma_P(f)$, $(\overrightarrow{A}, \gamma') \in \mathbf{AVal}(\overrightarrow{V})$, $\sigma = \mathbf{RType}(\underline{\tau})$, $c'' : \sigma$ and $\phi' = \nu(\overrightarrow{A}) \uplus \{c'\}$

Figure 14: The composite $\mathcal{L}_{\text{CBPV}}$ transition rules

The composite LTS, defined on such composite configurations, is given in Figure 14. Up to choice of name, it is deterministic.

Definition 76. Two valid configurations $\mathbf{C}_P, \mathbf{C}_O$ are said to be *compatible* if one of the two is active and the other one is passive, and, without loss of generality, supposing that $\mathbf{C}_P$ is the active configuration $(\langle M, c, \gamma_P, \phi_P, h_P, H_P\rangle, S_P)$ and $\mathbf{C}_O$ the passive configuration $(\langle \gamma_O, \phi_O, h_O, H_O, Fn\rangle, S_O)$, then $\phi_O = \phi_P = \phi$ and the composite configuration $(\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$, written $\mathbf{C}_P \between \mathbf{C}_O$, is valid.

The first half of the proof builds towards Lemma 83, which relates the behavior of the composite LTS on $\mathbf{C}_P \between \mathbf{C}_O$ to the traces generated by $\mathbf{C}_P$ and $\mathbf{C}_O$ independently, in the manner needed by Lemma 20. Intuitively, we wish to consider cases where $\mathbf{C}_P$ corresponds to a term, and $\mathbf{C}_O$ a context.

Lemma 77. *Taking* $\mathbf{D}$ *a valid composite configuration and* $\mathbf{D}'$ *a composite configuration s.t.* $\mathbf{D} \overset{\mathbf{a}}{\Rightarrow} \mathbf{D}'$*, then* $\mathbf{D}'$ *is valid.*

Proof. Simple case analysis and induction on the number of τ transitions. □

Lemma 78. *Taking* $\mathbf{C}_P, \mathbf{C}_O$ *two compatible configurations, for all composite configuration* $\mathbf{D}'$*, if* $(\mathbf{C}_P \between \mathbf{C}_O) \overset{\mathbf{a}}{\Rightarrow} \mathbf{C}'$ *then there exists two compatible configurations* $\mathbf{C}'_P, \mathbf{C}'_O$ *s.t.:*

- $\mathbf{D}' = \mathbf{C}'_P \between \mathbf{C}'_O$*;*
- $\mathbf{C}_P \overset{\mathbf{a}}{\Rightarrow} \mathbf{C}'_P$ *and* $\mathbf{C}_O \overset{\mathbf{a}^\perp}{\Longrightarrow} \mathbf{C}'_O$*.*

Proof. W.L.O.G, we suppose that $\mathbf{C}_P$ is active and $\mathbf{C}_O$ passive. So let $\mathbf{C}_P = (\langle M, c, \gamma_P, \phi, h_P, H_P\rangle, S_P)$ and $\mathbf{C}_O = (\langle \gamma_O, \phi, h_O, H_O, Fn\rangle, S_O)$. We then proceed by cases.

- If $\mathbf{a}$ is $PA\ \bar{c}(A)$, then there exists V, h'_P s.t.

$$(\mathbf{C}_P \between \mathbf{C}_O) \xrightarrow{\tau} (\langle \text{return } V, c, \gamma_P, \gamma_O, \phi, h'_P, h_O, H_P, H_O, Fn\rangle, S_P, (c, (K, c')) : S_O)$$

  s.t. $(M, h_P) \to (\text{return } V, h'_P)$. We have $c : \sigma$ and $(A, \gamma') \in \mathbf{AVal}_\sigma(V)$, giving

$$\mathbf{D}' = (\langle K[\text{return } A], c', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \nu(A), h'_P, h_O, H_P, H_O \cdot [\nu(A) \mapsto Fn], H_P(c) \uplus \nu(A)\rangle, S_P, S'_O)$$

  Let $\mathbf{C}'_P = (\langle \gamma_P \cdot \gamma', \phi \uplus \nu(A), h'_P, H_P, H_P(c) \uplus \nu(A)\rangle, S_P)$ and $\mathbf{C}'_O = (\langle K[\text{return } A], c', \gamma, \phi \uplus \nu(A), h_O, H_O \cdot [\nu(A) \mapsto Fn]\rangle, S'_O)$. It is easy to verify that:

  - $\mathbf{C}'_P, \mathbf{C}'_O$ are two compatible configurations;
  - $\mathbf{D}' = \mathbf{C}'_P \between \mathbf{C}'_O$;
  - $\mathbf{C}_P \xrightarrow{\tau} (\langle \text{return } V, c, \gamma_P, \phi, h'_p, H_P\rangle, S_P) \xrightarrow{\bar{c}(A)} \mathbf{C}'_P$;
  - $\mathbf{C}_O \xrightarrow{c(A)} \mathbf{C}'_O$.

- If $\mathbf{a}$ is a $PQ\ \bar{f}(\overrightarrow{A}, c')$, then there exists K, h'_P and maximal $\overrightarrow{V}$, s.t.

$$(\mathbf{C}_P \between \mathbf{C}_O) \xrightarrow{\tau} (\langle K[(\text{force } f)\overrightarrow{V}], c, \gamma_P, \gamma_O, \phi, h'_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$$

  With $(M, h_P) \to (K[(\text{force } f)\overrightarrow{V}], h'_P)$. Then $f : \underline{\tau}, f \in Fn$, $U = \gamma_O(f)$, $(\overrightarrow{A}, \gamma') \in \mathbf{AVal}(\overrightarrow{V})$, $\sigma = \mathbf{RType}(\underline{\tau})$, $c' : \sigma$ and $\phi' = \nu(\overrightarrow{A}) \uplus \{c'\}$ so that

$$\mathbf{D}' = (\langle \text{force } U\overrightarrow{A}, c', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \phi', h'_P, h_O, H_P, H_O \cdot [\phi' \mapsto Fn], H_P(f) \uplus \nu(\overrightarrow{A}))\rangle, (c', (K, c)) : S_P, S_O)$$

  Let $\mathbf{C}'_P = (\langle \gamma_P \cdot \gamma', \phi \uplus \phi', h'_P, H_P, H_P(c) \uplus \nu(\overrightarrow{A})\rangle, (c', (K, c)) : S_P)$ and $\mathbf{C}'_O = (\langle K[\text{force } U\overrightarrow{A}], c', \gamma, \phi \uplus \phi', h_O, H_O \cdot [\phi' \mapsto Fn]\rangle, S_O)$. It is easy to verify that:

– $\mathbf{C}'_P, \mathbf{C}'_O$ are two compatible configurations;
– $\mathbf{D}' = \mathbf{C}'_P \between \mathbf{C}'_O$;
– $\mathbf{C}_P \xrightarrow{\tau} (\langle K[(\text{force } f)\overrightarrow{V}], c, \gamma_P, \phi, h'_p, H_P\rangle, S_P) \xrightarrow{\bar{f}(\overrightarrow{A},c')} \mathbf{C}'_P$;
– $\mathbf{C}_O \xrightarrow{f(\overrightarrow{A},c')} \mathbf{C}'_O$.

□

Lemma 79. *Let $\mathbf{C}_P, \mathbf{C}_O$ be compatible configurations. If*

- $\mathbf{C}_P \overset{\mathbf{a}}{\Rightarrow} \mathbf{C}'_P$;
- $\mathbf{C}_O \overset{\mathbf{a}^\perp}{\Longrightarrow} \mathbf{C}'_O$;

then $\mathbf{C}'_P, \mathbf{C}'_O$ are two compatible configurations and $(\mathbf{C}_P \between \mathbf{C}_O) \overset{\mathbf{a}}{\Rightarrow} (\mathbf{C}'_P \between \mathbf{C}'_O)$.

Proof. W.L.O.G, we suppose that $\mathbf{C}_P$ is active and $\mathbf{C}_O$ passive. So let $\mathbf{C}_P = (\langle M, c, \gamma_P, \phi, h_P, H_P\rangle, S_P)$ and $\mathbf{C}_O = (\langle \gamma_O, \phi, h_O, H_O, Fn\rangle, S_O)$. We then proceed by cases.

- If $\mathbf{a}$ is a *PA* $\bar{c}(A)$, then there exists V, h'_P s.t. $\mathbf{C}_P \xrightarrow{\tau} (\langle \text{return } V, c, \gamma_P, \phi, h'_P, H_P\rangle, S_P)$ so that $(M, h_P) \to (V, h'_P)$. Then:
  – there exists σ s.t. $c : \sigma$, and γ', s.t. $(A, \gamma') \in \mathbf{AVal}_\sigma(V)$ so that $\mathbf{C}'_P = (\langle \gamma_P \cdot \gamma', \phi \uplus \nu(A), h'_P, H_P, H_P(c) \uplus \nu(A)\rangle, S_P)$;
  – $S_O = (c, (K, c')) : S'_O$ and $\mathbf{C}'_O = (\langle K[\text{return } A], c', \gamma_O, \phi \uplus \nu(A), h_O, H_O \cdot [\nu(A) \mapsto Fn]\rangle, S'_O)$.

  Then it is easy to verify $\mathbf{C}'_P, \mathbf{C}'_O$ are compatible, and:

$$\begin{array}{rll} (\mathbf{C}_P \between \mathbf{C}_O) & \xrightarrow{\tau} & (\langle \text{return } V, c, \gamma_P, \gamma_O, \phi, h'_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O) \\ & \xrightarrow{\bar{c}(A)} & (\langle K[\text{return } A], c', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \nu(A), h'_P, h_O, H_P, \\ & & \quad H_O \cdot [\nu(A) \mapsto Fn], H_P(c) \uplus \nu(A)\rangle, S_P, S'_O) \\ & = & \mathbf{C}'_P \between \mathbf{C}'_O \end{array}$$

- If $\mathbf{a}$ is a *PQ* $\bar{f}(\overrightarrow{A}, c')$, there exists K, h'_P and maximal $\overrightarrow{V}$ s.t. $\mathbf{C}_P \xrightarrow{\tau} (\langle K[(\text{force } f)\overrightarrow{V}], c, \gamma_P, \phi, h'_P, H_P\rangle, S_P)$ so that $(M, h_P) \to (K[(\text{force } f)\overrightarrow{V}], h'_P)$. Then:
  – there exist τ, σ, γ' s.t. $f : \underline{\tau}$, $(\overrightarrow{A}, \gamma') \in \mathbf{AVal}(\overrightarrow{V})$, $\sigma = \mathbf{RType}(\underline{\tau})\, c' : \sigma$ and let $\phi' = \nu(\overrightarrow{A}) \uplus \{c'\}$ so that $\mathbf{C}'_P = \langle \gamma_P \cdot \gamma', \phi \uplus \phi', h'_P, H_P, H_P(f) \uplus \nu(\overrightarrow{A}), (c', (K, c)) : S_P$;
  – $f \in Fn$, $\overrightarrow{A} \in \mathbf{ASeq}(\underline{\tau})$ and there exists U s.t. $\gamma_O(f) = U$ and $\mathbf{C}'_O = (\langle (\text{force } U)\overrightarrow{A}, c', \gamma_O, \phi \uplus \phi', h_O, H_O \cdot [\phi' \mapsto Fn]\rangle, S_O)$.

  Then one easily checks that $\mathbf{C}'_P, \mathbf{C}'_O$ are two compatible configurations, and:

$$\begin{array}{rll} (\mathbf{C}_P \between \mathbf{C}_O) & \xrightarrow{\tau} & (\langle (\text{force } f)\overrightarrow{V}, c, \gamma_P, \gamma_O, \phi, h'_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O) \\ & \xrightarrow{\bar{f}(A,c')} & (\langle (\text{force } f)\overrightarrow{A}, c', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \phi', h'_P, h_O, H_P, H_O \cdot [\phi' \mapsto Fn], \\ & & \quad H_P(c) \uplus \nu(A)\rangle, (c', (K, c)) : S_P, S_O) \\ & = & \mathbf{C}'_P \between \mathbf{C}'_O \end{array}$$

□

Definition 80. A composite configuration $\mathbf{D}$ *terminates* following a trace $\mathtt{t}$, written $\mathbf{D} \Downarrow^{\mathtt{t}}_{ter}$, when there exists a *final* composite configuration $\mathbf{D}_f = (\langle \text{return } V, \circ, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ s.t. $\mathbf{D} \stackrel{\mathtt{t}}{\Rightarrow} \mathbf{D}_f$. We often omit the trace $\mathtt{t}$ and simply write $\mathbf{D} \Downarrow_{ter}$.

Remark 81. Note that if we begin with a valid composite configuration, we have that S_P, S_O will both be $\perp$.

Definition 82. Taking $\mathbf{C}_P, \mathbf{C}_O$ two compatible configurations, write $(\mathbf{C}_P|\mathbf{C}_O) \downarrow^{\mathtt{t}}_{ter}$, when $\mathtt{t} \in \mathbf{Tr}(\mathbf{C}_P)$ and $\mathtt{t}^{\perp} \cdot \bar{\circ}(A) \in \mathbf{Tr}(\mathbf{C}_O)$ for some A.

Lemma 83. *Taking $\mathbf{C}_P, \mathbf{C}_O$ two compatible configurations and $\mathtt{t}$ a trace, then $(\mathbf{C}_P|\mathbf{C}_O) \downarrow^{\mathtt{t}}_{ter}$ iff $(\mathbf{C}_P \between \mathbf{C}_O) \Downarrow^{\mathtt{t}}_{ter}$.*

Proof. We first prove that if $(\mathbf{C}_P|\mathbf{C}_O) \downarrow^{\mathtt{t}}_{ter}$ then $(\mathbf{C}_P \between \mathbf{C}_O) \Downarrow^{\mathtt{t}}_{ter}$ by induction on the length of $\mathtt{t}$:

- if $\mathtt{t}$ is empty, then $\bar{\circ}(A) \in \mathbf{Tr}(\mathbf{C}_O)$, so there exists $V, \gamma_O, \phi, h_O, H_O, \sigma$ s.t. $(A, -) \in \mathbf{AVal}_{\sigma'}(V)$ and $\mathbf{C}_O \stackrel{\tau}{\rightarrow} (\langle \text{return } V, \circ, \gamma_O, \phi, h_O, H_O\rangle, S_O) = \mathbf{C}'_O$. Since $\mathbf{C}'_O$ is an active configuration, $\mathbf{C}_P$ must be a passive configuration, $(\langle \gamma_P, \phi, h_P, H_P, Fn\rangle, S_P)$. Then $\mathbf{C}_P \between \mathbf{C}_O \stackrel{\tau}{\rightarrow} \mathbf{C}_P \between \mathbf{C}'_O = (\langle \text{return } V, \circ, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$, so $(\mathbf{C}_P \between \mathbf{C}_O) \Downarrow^{\epsilon}_{ter}$.
- if $\mathtt{t} = \mathbf{a} \cdot \mathtt{t}'$, then there exists two configurations $\mathbf{C}'_P, \mathbf{C}'_O$ s.t.:
  - $\mathbf{C}_P \stackrel{\mathbf{a}}{\Rightarrow} \mathbf{C}'_P$;
  - $\mathbf{C}_O \stackrel{\mathbf{a}^{\perp}}{\Longrightarrow} \mathbf{C}'_O$;
  - $(\mathbf{C}'_P|\mathbf{C}'_O) \downarrow^{\mathtt{t}'}_{ter}$.

  From Lemma 79, we get that $\mathbf{C}'_P, \mathbf{C}'_O$ are two compatible configurations and $(\mathbf{C}_P \between \mathbf{C}_O) \stackrel{\mathbf{a}}{\Rightarrow} (\mathbf{C}'_P \between \mathbf{C}'_O)$. By the induction hypothesis, $(\mathbf{C}'_P \between \mathbf{C}'_O) \Downarrow^{\mathtt{t}'}_{ter}$, and so $(\mathbf{C}_P \between \mathbf{C}_O) \Downarrow^{\mathtt{t}}_{ter}$.

We now prove that if $(\mathbf{C}_P \between \mathbf{C}_O) \Downarrow^{\mathtt{t}}_{ter}$ then $(\mathbf{C}_P|\mathbf{C}_O) \downarrow^{\mathtt{t}}_{ter}$, by induction on the length of $\mathtt{t}$:

- if $\mathtt{t}$ is empty, then $(\mathbf{C}_P \between \mathbf{C}_O) \stackrel{\tau}{\rightarrow} (\langle \text{return } V, \circ, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$. So $\mathbf{C}_O \stackrel{\tau}{\rightarrow} (\langle \text{return } V, \circ, \gamma_O, \phi, h_O, H_O\rangle, S_O)$ and $\mathbf{C}_P = (\langle \gamma_P, \phi, h_P, H_P, Fn\rangle, S_P)$. Thus $\mathbf{C}_O \stackrel{\bar{\circ}(A)}{\Longrightarrow} (\langle \gamma \cdot \gamma', \phi \uplus \nu(A), h, H, \nu(A)\rangle, S_O)$, where $(A, \gamma') \in \mathbf{AVal}_{\sigma'}(V)$ so $(\mathbf{C}_P|\mathbf{C}_O) \downarrow^{\epsilon}_{ter}$.
- if $\mathtt{t} = \mathbf{a} \cdot \mathtt{t}'$, then there exists $\mathbf{D}'$ s.t. $(\mathbf{C}_P \between \mathbf{C}_O) \stackrel{\mathbf{a}}{\Rightarrow} \mathbf{D}'$ and $\mathbf{D}' \Downarrow^{\mathtt{t}'}_{ter}$. By Lemma 78, there exist two compatible configurations $\mathbf{C}'_P, \mathbf{C}'_O$ s.t.:
  - $\mathbf{D}' = \mathbf{C}'_P \between \mathbf{C}'_O$;
  - $\mathbf{C}_P \stackrel{\mathbf{a}}{\Rightarrow} \mathbf{C}'_P$;
  - $\mathbf{C}_O \stackrel{\mathbf{a}^{\perp}}{\Longrightarrow} \mathbf{C}'_O$.

  Thus we have $(\mathbf{C}'_P \between \mathbf{C}'_O) \Downarrow^{\mathtt{t}'}_{ter}$, so by the induction hypothesis $(\mathbf{C}'_P|\mathbf{C}'_O) \downarrow^{\mathtt{t}'}_{ter}$ and so $(\mathbf{C}_P|\mathbf{C}_O) \downarrow^{\mathtt{t}}_{ter}$.

□

To complete the proof of Lemma 20, we will need to show that for any M, K, γ etc. we have $(\mathsf{C}^{\rho_{\vec{A_i}}, c}_{M} \between \mathsf{C}^{\vec{\gamma_i}, c}_{h,K,\gamma}) \Downarrow_{ter}$ coincides with $(K[M\{\gamma\}], h) \Downarrow_{ter}$ which we shall do by constructing a bisimulation. To do this, we will first need a way of constructing term and heap from a composite configuration, for which we need a few auxiliary notions.

The first is a subtle modification to operational semantics. This is to account for the fact that the decomposition of a value V into an abstract value A and substitution γ does not satisfy $A\{\gamma\} = V$, due to the treatment of references. It has the effect of η-expanding occurrences of locations passed a arguments between the term and the context. We handle this by considering terms which never have an occurrence of MkVar thunk $!\ell$ thunk $\lambda x.\ell := x$, by allowing η-contractions whenever such sub-terms form.

Definition 84.

- For a term M, we define $\eta(M)$ to be the term obtained by replacing all sub-terms of the form MkVar (thunk $!\ell$) (thunk $\lambda x.\ell := x$) by ℓ, iteratively.
- We define the reduction relation $\to_\eta$ by

$$(M, h) \to_\eta (\eta(M'), h') \text{ where } (M, h) \to (M', h')$$

- We write $(M, h) \Downarrow^\eta_{ter}$ if there exists N, h' such that $(M, h) \to_\eta (N, h')$ and N is terminal.

We can then easily prove the following result relating termination under $\to$ to termination under $\to_\eta$.

Lemma 85. *For a* CBPV *term* $\Sigma; \Gamma \vdash^c M : \tau$ *and* $h : \Sigma$*, then* $(M, h) \Downarrow_{ter}$ *iff* $(\eta(M), h) \Downarrow^\eta_{ter}$.

We will now proceed to show a bisimulation result between reduction of composite configurations, and $\to_\eta$. Next, we need a way to construct a continuation from the pair of stacks.

Definition 86. Taking two stacks S_P and S_O, with c, c' two continuation names, we define the evaluation context $K_{c,c'}$ (if possible) by $K_{c,c'}(S_P, S_O)$ where:

- $K_{c,c}(S_P, S_O) \triangleq \bullet$
- $K_{c,c'}((c, (K, c'') : S_P, S_O) \triangleq K_{c'',c'}(S_P, S_O)[K]$
- $K_{c,c'}(S_P, (c, (K, c'') : S_O) \triangleq K_{c'',c'}(S_P, S_O)[K]$

We write K_c for $K_{c,\circ}$.

Definition 87. To an environment γ, we associate an idempotent substitution δ defined as the relation:

- $\delta^0 \triangleq \{(f, V) \mid f \in \mathrm{dom}(\gamma) \wedge \gamma(f) = V\}\}$
- $\delta^{i+1} \triangleq \{(f, V\{\delta^i\}) \mid (f, V) \in \delta^i\}\}$ with $V\{\delta^i\}$ denoting substitution.

then there exists $n \in \mathbb{N}$ s.t. $\delta^{n+1} = \delta^n$, and δ is then defined as δ^n.

The iterative construction is to give idempotency. The reason we give this construction is that in an environment, the image of some thunk names will be a thunk containing other thunk names, and so by doing this iteration, we fill in those names with closed thunks. This is possible because there is no cycles between names. The reason we do this is to handle environments formed of $\gamma_P \cdot \gamma_O$, where names in γ_P are defined using names in γ_O and vice versa, corresponding to the interleaving of questions between P and O.

With this in hand, we can now define a way to construct a term and heap from a composite configuration, which will allow us to establish the desired bisimulation result.

Definition 88. One define the configuration transformation θ from valid composite configurations to pair formed by a term and a heap, defined as

$$\theta : (\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O) \mapsto (\eta((K_c[M])\{\delta\}), h_P \cdot h_O)$$

with δ the idempotent substitution associated to $\gamma_P \cdot \gamma_O$.

Notice how in the definition of θ, we see that H_P, H_O, Fn play no role. This is because they are redundant in the composite LTS. These components enforce the visibility constraints on the original LTS, and so a present in the composite one to simplify the proof of Lemma 83. We can now show that they are functionally redundant by proving the following lemma.

Definition 89. A composite configuration $\mathbf{D} = (\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ is ***visibly valid*** if it is valid and also satisfies the properties

- $\nu(M) \subseteq Fn$
- for $f \in \mathrm{dom}(\gamma_P)$, $\nu(\gamma_P(f)) \subseteq H_O(\gamma_P(f))$
- for $f \in \mathrm{dom}(\gamma_O)$, $\nu(\gamma_O(f)) \subseteq H_P(\gamma_O(f))$
- for $(c, (K, c')) \in S_O$, $\nu(K) \subseteq H_P(c)$
- for $(c, (K, c')) \in S_P$, $\nu(K) \subseteq H_O(c)$

Lemma 90. *Given a visibly valid composite configuration* $\mathbf{D}$*, then if* $\mathbf{D} \xrightarrow{\mathbf{a}} \mathbf{D}'$*, the* $\mathbf{D}'$ *is a visibly valid composite configuration.*

Proof. The proof of this is by case analysis on $\mathbf{a}$. We omit the O cases, as they are symmetric with the P cases. Let $\mathbf{D} = (\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$.

- $\mathbf{D} \xrightarrow{\tau} (\langle M', c, \gamma_P, \gamma_O, \phi, h'_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$

  In this case, $\gamma_P, \gamma_O, S_P, S_O, H_P$, and H_O remain unchanged, and so the conditions upon them hold. We need to check that $\nu(M') \subseteq Fn$, but this follows from the fact that the rules defining $\rightarrow$ do not introduce new names.

- $\mathbf{D} \xrightarrow{\bar{c}(A)} (\langle K[\text{return } A], c', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \nu(A), h_P, h_O, H_P, H_O \cdot [\nu(A) \mapsto Fn], H_P(c) \uplus \nu(A)\rangle, S_P, S'_O)$ where $S_O = (c, (K, c')) : S'_O, M = \text{return } V$ and $(A, \gamma') \in \mathbf{AVal}_\sigma(V)$.

  γ_O, H_P are unchanged, so we do not need to check. As S_P is unchanged, S'_O smaller than S_O, and the mapping H_O enlarged, we don't need to check the condition on stacks. So we are left to check that

  - For $f \in \mathrm{dom}(\gamma')$, we have $\nu(\gamma'(f)) \subseteq H_O \cdot [\nu(A) \mapsto Fn](f)$.
    If $f \in \mathrm{dom}(\gamma')$, then $f \in \nu(A)$ and $\gamma'(f)$ occurs in V. Thus, as $\mathbf{D}$ is visibly valid, $\nu(\gamma'(f)) \subseteq \nu(\text{return } V) \subseteq Fn = H_O \cdot [\nu(A) \mapsto Fn](f)$.
  - $\nu(K[\text{return } A]) \subseteq H_P(c) \uplus \nu(A)$
    As $\mathbf{D}$ is visibly valid, $\nu(K) \subseteq H_P(c)$, and as $\nu(K[\text{return } A]) = \nu(K) \cup \nu(A)$ we are done.

- $\mathbf{D} \xrightarrow{\bar{f}(\overrightarrow{A}, c')} (\langle \text{force } U \overrightarrow{A}, c', \gamma_P \cdot \gamma', \gamma_O, \phi \uplus \phi', h'_P, h_O, H_P, H_O \cdot [\phi' \mapsto Fn], H_P(f) \uplus \nu(\overrightarrow{A})\rangle, (c', (K, c)) : S_P, S_O)$ where $U = \gamma_O(f)\ (\overrightarrow{A}, \gamma') \in \mathbf{AVal}(\overrightarrow{V}), \phi' = \nu(\overrightarrow{A}) \uplus \{c'\}$ and $M = K[\text{force } f \overrightarrow{V}]$.

  γ_O, H_P are unchanged, so we do not need to check. S_O is unchanged and the mapping H_O is expanded, so we don't need to check the condition on S_O. So we are left to check that

  - For $f \in \mathrm{dom}(\gamma')$, we have $\nu(\gamma'(f)) \subseteq H_O \cdot [\phi' \mapsto Fn](f)$.
    If $f \in \mathrm{dom}(\gamma')$, then $f \in \nu(\overrightarrow{A}) \subseteq \phi'$ and $\gamma'(f)$ occurs in $\overrightarrow{V}$. Thus, as $\mathbf{D}$ is visibly valid, $\nu(\gamma'(f)) \subseteq \nu(\overrightarrow{V}) \subseteq Fn = H_O \cdot [\phi' \mapsto Fn](f)$.
  - $\nu(K) \subseteq H_O \cdot [\phi' \mapsto Fn](f)(c')$
    As $\mathbf{D}$ is visibly valid, $\nu(K) \subseteq Fn$, and as $c' \in \phi'$, we are done.
  - $\nu(\text{force } U \overrightarrow{A}) \subseteq H_P(f) \uplus \nu(\overrightarrow{A})$
    As $\mathbf{D}$ is visibly valid, $\nu(U) \subseteq H_P(f)$, and as $\nu(\text{force } U \overrightarrow{A}) = \nu(U) \cup \nu(\overrightarrow{A})$ we are done.

□

Due to this lemma, we can ignore the condition that $f \in Fn$ in question moves. Before proving a bisimulation, we will need a few extra results to bring the behavior of the composite LTS closer to reduction. The following lemma shows that, in essence, we can safely ignore situations in which $\gamma(f) = g$, where we simply have a name mapped to another name, as these are artifacts of the LTS, and not of the underlying reduction taking place on $\theta(\mathbf{D})$.

Lemma 91. *Taking* $\mathbf{D} = (\langle K[(\text{force } f)\overrightarrow{V}], c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ *a valid composite configuration that is going to perform a question, with* $f \in \text{dom}(\gamma)$*, where* $\gamma = \gamma_P \cdot \gamma_O$*, there exists a functional name* g*, an abstract value sequence* $\overrightarrow{A}$*, a composite configuration* $\mathbf{D}'$ *and a trace* $\mathtt{t}$ *formed by questions s.t.:*

- $\gamma(g)$ *has the form* thunk M*,* (MkVar M_r M_w).read*,* (MkVar M_r M_w).write,ℓ.read*, or* ℓ.write*;*
- $\delta(f) = \delta(g)$*, writing* δ *for the idempotent substitution associated to* γ *(that is, alternate substitutions from* γ_P *and* γ_O *make no difference);*
- $\mathbf{D} \xrightarrow{\mathtt{t}} \mathbf{D}'$*;*
- $\mathbf{D}'$ *can be written as* $(\langle K'[(\text{force } g)\overrightarrow{V'}], c', \gamma_P \cdot \gamma'_P, \gamma_O \cdot \gamma'_O, \phi \uplus \text{dom}(\gamma'_P) \uplus \text{dom}(\gamma'_O), h_P, h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$*;*
- $\eta(\overrightarrow{V'}\{\delta'\}) = \eta(\overrightarrow{V})$*, with* δ' *the idempotent substitution associated to* $\gamma'_P \cdot \gamma'_O$*;*
- *if* $c' = c$*, then* $K_{c',c}(S'_P, S'_O) = \bullet$ *and* $K' = K$*, otherwise* $K_{c',c}(S'_P, S'_O) = K$ *and* $K' = \bullet$

The next lemma shows that when we have continuations on the stacks which are empty, we can safely ignore the interleaving of answer moves between P and O as these are again merely artifacts of the LTS, and don't correspond to any underlying reduction.

Lemma 92. *Let* $\mathbf{D} = (\langle \text{return } V, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ *be a valid composite configuration that is going to perform an answer. Suppose that there exists* c' *s.t.* $K_{c,c'} = \bullet$*. Then there exists a composite configuration* $\mathbf{D}' = (\langle \text{return } V', c', \gamma_P \cdot \gamma'_P, \gamma_O \cdot \gamma'_O, \phi \uplus \text{dom}(\gamma'_P) \uplus \text{dom}(\gamma'_O), h_P, h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$ *and a trace* $\mathtt{t}$ *formed only by answers s.t.* $\mathbf{D} \xrightarrow{\mathtt{t}} \mathbf{D}'$ *and* $\eta(V'\{\delta'\}) = \eta(V)$*, with* δ' *the idempotent substitution associated to* $\gamma'_P \cdot \gamma'_O$*.*

Lemma 93. *Taking* $\mathbf{D}, \mathbf{D}'$ *two valid composite configuration and* $\mathbf{a}$ *an action (different of* τ*) s.t.* $\mathbf{D} \xrightarrow{\mathbf{a}} \mathbf{D}'$ *then* $\theta(\mathbf{D}) = \theta(\mathbf{D}')$*.*

Proof. Let $\mathbf{D} = (\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ and $\mathbf{D}' = (\langle M', c', \gamma'_P, \gamma'_O, \phi', h_P, h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$. W.L.O.G assume $\mathbf{D}$ is P-active, so $S_O = (c, (K, c')) : S''_O$. Let δ be the idempotent substitution associated to $\gamma_P \cdot \gamma_O$ and δ' be the idempotent substitution associated to $\gamma'_P \cdot \gamma'_O$.

We reason by case analysis over α:

- If $\alpha = \bar{c}(A)$, so that $M = \text{return } V$. Then we have:
  - $c : \sigma$;
  - $\gamma'_O = \gamma_O$, $\gamma'_P = \gamma_P \cdot \gamma_A$ and $\phi' = \phi \uplus \text{dom}(\gamma_A)$; with $(A, \gamma_A) \in \mathbf{AVal}_\sigma(V)$;
  - $S'_P = S_P, S'_O = S''_O$
  - $M' = K[\text{return } A]$.

  We conclude using these and the fact that:
  - $K_c(T_P, (c, (K, c')) : T_O) = K_{c'}(T_P, T_O)[K]$;
  - $\eta(A\{\gamma_A\}) = \eta(V)$;

– if $f \in \mathrm{dom}(\delta), \delta'(f) = \delta(f)$;

that $(\eta(K_c(S_P, S_O)[\text{return } V]))\{\delta\} = (\eta(K_{c'}(S_P, S'_O)[K[\text{return } A]]))\{\delta'\}$. So $\theta(\mathbf{D}) = \theta(\mathbf{D}')$.

- If $\alpha = \bar{f}(\overrightarrow{A}, c')$, so that $M = K[(\text{force } f)\overrightarrow{V}]$ for some context K, value sequence $\overrightarrow{V}$, and thunk name f. Then we have:
  - $\gamma_O(f) = U$;
  - $\gamma'_O = \gamma_O$, $\gamma'_P = \gamma_P \cdot \gamma_{\overrightarrow{A}}$, with $(\overrightarrow{A}, \gamma_{\overrightarrow{A}}) \in \mathbf{AVal}(\overrightarrow{V})$;
  - $S'_P = (c', (K, c)) : S_P, S'_O = S_O$
  - $M' = (\text{force } U)\overrightarrow{A}$.

  We conclude using these and the fact that:
  - $K_{c'}((c', (K, c)) : T_P, T_O) = K_c(T_P, T_O)[K]$;
  - $\eta(\overrightarrow{A}\{\gamma_{\overrightarrow{A}}\}) = \eta(\overrightarrow{V})$;
  - if $f \in \mathrm{dom}(\delta), \delta'(f) = \delta(f)$;

  that $(K_c(S_P, S_O)[K[(\text{force } f)\overrightarrow{V}]])\{\delta\} = (K_{c'}(S'_P, S_O)[(\text{force } U)\overrightarrow{A})\{\delta'\}$. So $\theta(\mathbf{D}) = \theta(\mathbf{D}')$.

□

We need a way to capture those transition that a composite configuration make simply to carry out η-contraction.

Definition 94. Taking $\mathbf{D} = (\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O), \mathbf{D}'$ two composite configuration, we write $\mathbf{D} \xrightarrow{\eta} \mathbf{D}'$ if one of the following hold:

- $M =$!MkVar (thunk !ℓ) (thunk $\lambda x.\ell := x$) and

$$\mathbf{D} \xrightarrow{\tau} (\langle \text{force thunk } !\ell, \cdots\rangle, S_P, S_O) \xrightarrow{\tau} (\langle !\ell, \cdots\rangle, S_P, S_O)$$

- $M =$ MkVar (thunk !ℓ) (thunk $\lambda x.\ell := x$) $:= V$ and

$$\mathbf{D} \xrightarrow{\tau} (\langle(\text{force thunk } \lambda x.\ell := x)V, \cdots\rangle, S_P, S_O) \xrightarrow{\tau} (\langle(\lambda x.\ell := x)V, \cdots\rangle, S_P, S_O) \xrightarrow{\tau} (\langle(\ell := V, \cdots\rangle, S_P, S_O)$$

- $M =$!MkVar f g where $f, g \in \phi$ and $f\{\delta\} =$ thunk ℓ, $g\{\delta\} =$ thunk $\lambda x.\ell := x$ where δ is the idempotent substitution associated with $\gamma_P \cdot \gamma_O$ and for some $\mathtt{t}$ consisting entirely of question actions,

$$\mathbf{D} \xrightarrow{\tau} (\langle \text{force } f, \cdots\rangle, S_P, S_O) \xrightarrow{\mathtt{t}} (\langle \text{force thunk } !\ell, \cdots\rangle, S'_P, S'_O) \xrightarrow{\tau} (\langle !\ell, \cdots\rangle, S'_P, S'_O)$$

- $M =$ MkVar f $g := V$ where $f, g \in \phi$ and $f\{\delta\} =$ thunk ℓ, $g\{\delta\} =$ thunk $\lambda x.\ell := x$ where δ is the idempotent substitution associated with $\gamma_P \cdot \gamma_O$ and for some $\mathtt{t}$ consisting entirely of question actions,

$$\begin{array}{rcl} \mathbf{D} & \xrightarrow{\tau} & (\langle(\text{force } g)V, \cdots\rangle, S_P, S_O) \\ & \xrightarrow{\mathtt{t}} & (\langle(\text{force thunk } \lambda x.\ell := x)V, \cdots\rangle, S'_P, S'_O) \\ & \xrightarrow{\tau} & (\langle(\lambda x.\ell := x)V, \cdots\rangle, S'_P, S'_O) \\ & \xrightarrow{\tau} & (\langle(\ell := V, \cdots\rangle, S'_P, S'_O) \end{array}$$

Lemma 95. *Taking* $\mathbf{D} = (\langle K[M], c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ *a valid composite configuration where* $M = !\text{MkVar } V_{\text{read}}\ V_{\text{write}}$ *or* $M = (\text{MkVar } V_{\text{read}}\ V_{\text{write}}) := U$*, with* $V_{\text{read}}\{\delta\} = \text{thunk } !\ell$ *and* $V_{\text{write}}\{\delta\} = \text{thunk } \lambda x.\ell := x$ *where* δ *is the idempotent substitution associated with* $\gamma_p \cdot \gamma_O$*, there exists* $\mathbf{D}'$ *s.t.* $\mathbf{D} \xrightarrow{\eta} \mathbf{D}'$*. Furthermore,* $\theta(\mathbf{D}) = \theta(\mathbf{D}')$*.*

Definition 96. Taking $\mathbf{D}, \mathbf{D}'$ two composite configuration, we write $\mathbf{D} \rightsquigarrow \mathbf{D}'$ when $\mathbf{D} \xrightarrow{\eta} \mathbf{D}'' \xrightarrow{\tau} \mathbf{D}'$, or if there exist no such $\mathbf{D}''$, there exists a trace $\mathtt{t}$ of actions (without any τ-actions) s.t. $\mathbf{D} \xrightarrow{\mathtt{t}\cdot\tau} \mathbf{D}'$ ($\mathtt{t}$ can be ϵ only if there is no possible $\rightarrow_\eta$ reduction).

Lemma 97. *The configuration transformation* θ *is a functional bisimulation between the transition system over visibly valid composite configurations* $(CompConf, \rightsquigarrow)$ *and the operational transition system* $(\Lambda \times \text{Heap}, \rightarrow_\eta)$*, that is, for all visibly valid composite configuration* $\mathbf{D}$*:*

- *for all composite configuration* $\mathbf{D}'$*, if* $\mathbf{D} \rightsquigarrow \mathbf{D}'$ *then* $\theta(\mathbf{D}) \rightarrow_\eta \theta(\mathbf{D}')$*;*
- *for all pairs* (N, h) *formed by a term an a heap* h'*, if* $\theta(\mathbf{D}) \rightarrow_\eta (N, h')$ *then there exists a valid composite configuration* $\mathbf{D}'$ *s.t.* $\mathbf{D} \rightsquigarrow \mathbf{D}'$ *and* $(N, h') = \theta(\mathbf{D}')$

Proof. To simplify the proof, we let:

- $\mathbf{D} = (\langle M, c, \gamma_P, \gamma_O, \phi, h_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$;
- $\gamma = \gamma_P \cdot \gamma_O$;
- δ be the idempotent substitution associated to $\gamma_P \cdot \gamma_O$;
- $\theta(\mathbf{D}) = ((K_c[M])\{\delta\}, h)$ with $h = (h_P \cdot h_O)$.

We first suppose that $\mathbf{D} \rightsquigarrow \mathbf{D}'$. We first consider the case where there exists a trace $\mathtt{t}$ of actions (without any τ) and a composite configurations $\mathbf{D}_1$ s.t. $\mathbf{D} \xrightarrow{\mathtt{t}} \mathbf{D}_1 \xrightarrow{\tau} \mathbf{D}'$. From Lemma 93, we get that $\theta(\mathbf{D}) = \theta(\mathbf{D}_1)$.

Instead suppose that there exists $\mathbf{D}_1$ s.t $\mathbf{D} \xrightarrow{\eta} \mathbf{D}_1 \xrightarrow{\tau} \mathbf{D}'$. Due to the fact that the construction of θ does η-contraction, we can easily verify that $\theta(\mathbf{D}) = \theta(\mathbf{D}_1)$.

Without loss of generality, we suppose the composite configuration $\mathbf{D}_1$ is P-active. We write $\mathbf{D}'$ as $(\langle M', c', \gamma'_P, \gamma'_O, \phi', h'_P, h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$ and $\mathbf{D}_1$ as $(\langle M_1, c', \gamma'_P, \gamma'_O, \phi', h_P, h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$, so that we have $(M_1, h_P) \rightarrow (M', h'_P)$. If $M = K[!V]$ or $M = K[V := U]$, we must have that V is a location ℓ due to the definition of $\rightsquigarrow$. Therefore, $(\eta(M_1), h_P) \rightarrow_\eta (\eta(M'), h'_P)$. It follows that $\theta(\mathbf{D}_1) = (\eta((K_{c'}(S'_P, S'_O)[M_1])\{\delta'\}), h_P \cdot h_O) \rightarrow_\eta (\eta((K_{c'}(S'_P, S'_O)[M'])\{\delta'\}), h'_P \cdot h_O) = \theta(\mathbf{D}')$ where δ' is the idempotent substitution associated to $\gamma'_P \cdot \gamma'_O$.

Now, we suppose that there exists a term N and a heap h' s.t. $\theta(\mathbf{D}) \rightarrow_\eta (N, h')$. We now proceed by cases on the possible ways that $\theta(\mathbf{D}) \rightarrow_\eta (N, h')$:

- $\theta(\mathbf{D}) = (K[!\ell], h)$ or $(K[\ell := V], h)$, and $M = K'[!\text{MkVar } V_{\text{read}}\ V_{\text{write}}]$ or $K'[\text{MkVar } V_{\text{read}}\ V_{\text{write}} := V]$. By Lemma 95, there exists $\mathbf{D}_1$ s.t $\mathbf{D} \xrightarrow{\eta} \mathbf{D}_1$ and $\theta(\mathbf{D}) = \theta(\mathbf{D}_1)$. We write $\mathbf{D}_1$ as $(\langle K_1[M_1], c', \gamma'_P, \gamma'_O, \phi', h_P, h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$, where $M_1 = !\ell$ or $\ell := V$ as appropriate. W.L.O.G, we assume $\ell \in h_P$. So either $(N, h') = (\eta(K_c(S'_P, S'_O)[K_1[\text{return } h_P(\ell)]]\{\delta\}), h_P \cdot h_O)$ or $(\eta(K_c(S'_P, S'_O)[K_1[\text{return } ()]]\{\delta\}), h_P \cdot [\ell \mapsto V] \cdot h_O)$ as appropriate. So we take $\mathbf{D} = (\langle K_1[\text{return } h_P(\ell)], c', \gamma'_P, \gamma'_O, \phi', h_P, h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$ or $(\langle K_1[\text{return } ()], c', \gamma'_P, \gamma'_O, \phi', h_P \cdot [\ell \mapsto V], h_O, H'_P, H'_O, Fn'\rangle, S'_P, S'_O)$ as appropriate. The $\mathbf{D} \xrightarrow{\eta} \mathbf{D}_1 \xrightarrow{\tau} \mathbf{D}'$ and $\theta(\mathbf{D}') = (N, h')$ as required.
- Either $(\eta(M), h_P \cdot h_O)$ is reducible. W.L.O.G, we suppose that $\mathbf{D}$ is P-active, so that $(\eta(M), h_P)$ is reducible. Then there exists (M', h'_P) s.t.:
  - $(\eta(M), h_P) \rightarrow (M', h'_P)$;
  - $N = \eta(K_c(S_P, S_O)[M']\{\delta\})$;
  - $h' = h'_P \cdot h_O$.

So we take $\mathbf{D}' = (\langle M', c, \gamma_P, \gamma_O, \phi, h'_P, h_O, H_P, H_O, Fn\rangle, S_P, S_O)$ so that $\mathbf{D} \xrightarrow{\tau} \mathbf{D}'$.

- Or M is forcing a thunk name:
  - $M = K[(\text{force } f)\overrightarrow{V}]$ for some context K, value sequence V, and thunk name f;
  - $\delta(f)$ is thunk M' for some computation M';
  - $N = \eta((K_c(S_P, S_O)[K[M'\overrightarrow{V}]])\{\delta\})$;
  - $h' = h$;

  From Lemma 91, there exists a functional name g, an abstract value A_1, a composite configuration $\mathbf{D}_1$ and a trace $\mathtt{t}$ formed by questions s.t.:
  - $\gamma(g)$ is thunk $\hat{M}'$ for some computation $\hat{M}'$;
  - $\delta(f) = \delta(g)$;
  - $\mathbf{D} \xrightarrow{\mathtt{t}} \mathbf{D}_1$;
  - $\mathbf{D}_1$ can be written as $(\langle K_1[(\text{force } g)\overrightarrow{V_1}], c_1, \gamma_P{\cdot}\gamma_{1,P}, \gamma_O{\cdot}\gamma_{1,O}, \phi \uplus \text{dom}(\gamma_{1,P}) \uplus \text{dom}(\gamma_{1,O}), h_P, h_O, H_{1,P}, H_{1,O}, Fn_1\rangle, S_{1,P}, S_{1,O})$;
  - $\eta(\overrightarrow{V_1}\{\delta_1\}) = \eta(\overrightarrow{V})$, with δ_1 the idempotent substitution associated to $\gamma_{1,P} \cdot \gamma_{1,O}$;
  - if $c_1 = c$, then $K_{c_1,c}(S_{1,P}, S_{1,O}) = \bullet$ and $K' = K$, otherwise $K_{c_1,c}(S_{1,P}, S_{1,O}) = K$ and $K' = \bullet$.

  Without loss of generality, we suppose the composite configuration $\mathbf{D}_1$ is P-active. By Lemma 90, $g \in Fn_1$, so we have:

$$\mathbf{D} \xrightarrow{\mathtt{t}} \mathbf{D}_1 \xrightarrow{\bar{g}(A_2,c_2)} \overbrace{(\langle(\text{force thunk } \hat{M}')\overrightarrow{A_2}, c_2, \gamma_{2,P}, \gamma_O \cdot \gamma_{1,O}, \phi_2, h_P, h_O, H_{1,P}, H_{2,O}, Fn_2\rangle, (c_2, (K_1, c_1)) : S_{1,P}, S_{1,O})}^{\mathbf{D}_2}$$
$$\xrightarrow{\tau} \underbrace{(\langle \hat{M}'\overrightarrow{A_2}, c_2, \gamma_{2,P}, \gamma_O \cdot \gamma_{1,O}, \phi_2, h_P, h_O, H_{1,P}, H_{2,O}, Fn_2\rangle, (c_2, (K_1, c_1)) : S_{1,P}, S_{1,O})}_{\mathbf{D}'}$$

  with $\gamma_{2,P} = \gamma_P{\cdot}\gamma_{1,P}{\cdot}\gamma_{\overrightarrow{A_2}}$ and $\eta(\overrightarrow{A_2}\{\gamma_{\overrightarrow{A_2}}\}) = \eta(\overrightarrow{V_1})$. From Lemma 93, we have $\theta(\mathbf{D}) = \theta(\mathbf{D}_2)$.

  We have that $\eta((M'\overrightarrow{V})\{\delta\}) = \eta((\hat{M}'\overrightarrow{A_2})\{\delta_2\})$ from:
  - $\eta(\overrightarrow{A_2}\{\delta_2\}) = \eta(\overrightarrow{V})$ since $\eta(\overrightarrow{V_1}\{\delta_1\}) = \eta(\overrightarrow{V})$ and $\eta(\overrightarrow{V_1}) = \eta(A_2\{\gamma_{A_2}\})$
  - $\hat{M}'\{\delta\} = M'\{\delta\}$ since $\delta(f) = \delta(g)$ and $\gamma(g) = \text{thunk } \hat{M}'$.

  As $K_{c_2}((c_2, (K_1, c_1)) : S_{1,P}, S_{1,O}) = K_{c_1}(S_{1,P}, S_{1,O})[K_1] = K_c(S_P, S_O)[K]$ we have $\theta(\mathbf{D}') = (N.h)$.

- Otherwise, $M = \text{return } V$ for a value V and K_c an evaluation context larger than $\bullet$. Then there exists a continuation name c_1 s.t.:
  - $K_{c,c_1}(S_P, S_O) = \bullet$;
  - $(c_1, (K, c_2))$ is in S_P or S_O with K an evaluation context larger than $\bullet$.

  From Lemma 92, there exists a value V_1, a composite configuration $\mathbf{D}_1$ and a trace $\mathtt{t}$ formed by answers s.t.:
  - $\mathbf{D} \xrightarrow{\mathtt{t}} \mathbf{D}_1$;
  - $\mathbf{D}_1$ can be written as $(\langle \text{return } V_1, c_1, \gamma_P{\cdot}\gamma_{1,P}, \gamma_O{\cdot}\gamma_{1,O}, \phi \uplus \text{dom}(\gamma_{1,P}) \uplus \text{dom}(\gamma_{1,O}), h_P, h_O, H_{1,P}, H_{1,O}, Fn_1\rangle, S_{1,P}, S_{1,O})$;

– $\eta(V_1\{\delta_1\}) = \eta(V)$, with δ_1 the idempotent substitution associated to $\gamma_{1,P} \cdot \gamma_{1,O}$;

Since K is larger than $\bullet$, we must have for some K' that $K = K'[\bullet \text{ to } y.M']$. W.L.O.G suppose the composite configuration $\mathbf{D}_1$ is P-active. Then as $\mathbf{D}_1$ is valid $S_{1,O} = (c_1, (K, c_2)) : S_{2,O}$. Then we have:

$$\mathbf{D} \xrightarrow{\mathtt{t}} \mathbf{D}_1 \xrightarrow{\bar{c_1}(A_2)} \overbrace{(\langle K[\text{return } A_2], c_2, \gamma_{2,P}, \gamma_O \cdot \gamma_{1,O}, \phi_2, h_P, h_O, H_{1,P}, H_{2,O}, Fn_2\rangle, S_{1,P}, S_{2,O})}^{\mathbf{D}_2}$$

with $\gamma_{2,P} = \gamma_{1,P} \cdot \gamma_{A_2}$ and $\eta(A_2\{\gamma_{A_2}\}) = \eta(V_1)$.

From Lemma 93, we have that $\theta(\mathbf{D}) = \theta(\mathbf{D}_2)$. From $K_{c,c_1}(S_P, S_O) = \bullet$, we get that $K_c(S_P, S_O) = K_{c_1}(S_{1,P}, S_{1,O})$, so $K_{c_2}(S_{1,P}, S_{2,O})[K] = K_c(S_P, S_O)$.

Thus, we have $\theta(\mathbf{D}) = \eta((K_{c_2}(S_{1,P}, S_{2,O})[K'[\text{return } A_2 \text{ to } y.M']]\{\delta_2\}), h) \to_\eta \eta((K_{c_2}(S_{1,P}, S_{2,O})[K'[M'\{A_2/y\}]]\{\delta_2\}), h) = (N, h')$. So, if we let $\mathbf{D}' = (\langle K'[M'\{A_2/y\}], c_2, \gamma_{2,P}, \gamma_O \cdot \gamma_{1,O}, \phi_2, h_P, h_O, H_{1,P}, H_{2,O}, Fn_2\rangle, S_{1,P}, S_{2,O})$, and we get $\mathbf{D} \rightsquigarrow \mathbf{D}'$ and $\theta(\mathbf{D}') = (N, h')$, as required.

□

Using this bisimulation, we can easily prove this corollary using induction:

Corollary 98. *Taking* $\mathbf{D}$ *a visibly valid composite configurations, then* $\mathbf{D} \Downarrow_{ter}$ *iff* $\theta(\mathbf{D}) \Downarrow^{\eta}_{ter}$.

We are finally in a position to prove Lemma 20.

Proof of Lemma 20. Note that $(K[M\{\gamma\}], h) \Downarrow_{ter}$ iff $\theta(\mathsf{C}_M^{\rho_{\vec{A_i}}, c} \between \mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}}) \Downarrow^{\eta}_{ter}$. Furthermore, we can check that $\mathsf{C}_M^{\rho_{\vec{A_i}}, c} \between \mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}}$ is visibly valid. From Corollary 98, this is equivalent to the existence of a trace $\mathtt{t}$ such that $(\mathsf{C}_M^{\rho_{\vec{A_i}}, c} \between \mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}}) \Downarrow^{\mathtt{t}}_{ter}$. By Lemma 83, this is the same as $(\mathsf{C}_M^{\rho_{\vec{A_i}}, c} | \mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}}) \downarrow^{\mathtt{t}}_{ter}$, which implies the Lemma. □

Remark 99. Observe that if $t^\perp \, \bar{\circ}(A) \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}, c})$, then as these trace satisfies P-bracketing, $\mathsf{Top}_P(t^\perp) = \{\circ\}$ and so the $(N_O, \emptyset)$-trace $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_M^{\rho_{\vec{A_i}}, c})$ satisfies $\mathsf{Top}_O(t) = \emptyset$, and so must be a complete trace.

B.2 Proof of Soundness

Proof of Theorem 21. Suppose $\mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_1) \subseteq \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_2)$.

Let Σ, h, K, γ be such that $(K[M_1\{\gamma\}], h) \Downarrow_{ter}$. Suppose $(\vec{A_i}, \vec{\gamma_i}) \in \mathbf{AVal}_\Gamma(\gamma)$. By Lemma 20 (left-to-right), there exist t, c' such that $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_1}^{\rho_{\vec{A_i}}, c})$ and $t^\perp \, \bar{\circ}(A) \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}, c})$. Further more, by Remark 99, we have that t is complete. By $\mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_1) \subseteq_c \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma \vdash^c M_2)$, we have $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_2}^{\rho_{\vec{A_i}}, c})$. Because $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_2}^{\rho_{\vec{A_i}}, c})$ and $t^\perp \, \bar{\circ}(A) \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{h,K,\gamma}^{\vec{\gamma_i}, c})$, by Lemma 20 (right-to-left) we can conclude $(K[M_2\{\gamma\}], h) \Downarrow_{ter}$. Thus, $\Gamma \vdash^c M_1 \lesssim^{\mathrm{CBPV}(ciu)}_{ter} M_2$. □

C Definability and Completeness

C.1 Proof of Definability (Lemma 22)

To prove Lemma 22, we will use backwards induction on the traces. To do this, we will need a more general lemma, from which Lemma 22 can be recovered by taking $i = 0$.

Lemma 100. *Suppose* $\phi \subseteq \mathrm{TNames}$, $c \in \mathrm{CNames}$ *and* $t = o_1 p_1 \cdots o_n p_n$ *is a P-visible, P-bracketed, O-visible, P-visible* $(\{\circ\}, \phi \uplus \{c\})$*-trace starting with an O-action, so that* $o_1 \cdots o_n$ *is complete. Given* $0 \leq i \leq n$, *let* $t_i = o_{i+1} p_{i+1} \cdots o_n p_n$. *There exist passive configurations* $\mathbf{C}_i$ *such that* $\mathbf{Tr}^{\mathrm{even}}(\mathbf{C}_i)$ *is the even-length prefixes of* t_i *(along with their renamings via permutations on* Names *that fix* ϕ_i*).*

Moreover, $\mathbf{C}_i = \langle \gamma_i, \phi_i, h_i, H_i, Fn_i\rangle, S_i)$ *, where*

- $\mathrm{dom}(\gamma_i)$ *consists of* ϕ *and all names introduced by* P *in* $o_1p_1 \cdots o_ip_i$*;*
- $\nu(\gamma_i(x)) \subseteq \mathsf{Vis}_P(o_1p_1 \cdots o_j)$ *if* x *has been introduced in* p_j *(*$\phi \uplus \{c\}$ *are deemed to have been introduced in* p_0 *and* $\mathsf{Vis}_P(o_1 \cdots o_0) = \emptyset$*);*
- *if* p_j *is the* k*-th most recent unanswered PQ-action in* $o_1p_1 \cdots o_ip_i$*, and* c' *is the continuation name introduced by* p_j*, then the* k*-th item from the top of* S_i *has the form* $(c', (K, c''))$*;*
- *if* $(c', (K, c'')) \in S_i$*, and* $c' : \sigma'$ *is introduced in action* p_j*, then* $\{c'' : \sigma''\} \in \mathsf{Top}_P(o_1 \cdots o_j)$*,* $\nu(K) \in \mathsf{Vis}_P(o_1p_1 \cdots o_j)$ *and* $K : \sigma' \implies \sigma''$*;*
- *the bottom-most element of* S_i *is* $(c, (K, \circ))$ *if no OA-action using* c *has occurred, and the stack is empty if a move using* $\circ$ *has occurred;*
- $Fn = \mathsf{Vis}_O(o_1p_1 \cdots o_ip_i)$*;*
- *if* $f \in \phi_i$*, and was introduced in* o_j*, then* $H(f) = \mathsf{Vis}_O(o_1p_1 \cdots o_{j-1}p_{j-1})$*;*
- ϕ_i *consists of* $\phi \uplus \{c\}$ *and all names introduced in* $o_1p_1 \cdots o_ip_i$*;*
- $h_i = \{\mathit{time} \mapsto i\}$*, where* $\mathit{time} : \mathrm{Ref}$ *.*

Proof. This construction works by using the single reference, *time* to count the steps in the trace. This will then be used to allow the correct action to be enabled and all others disabled.

(Note that if Int is set to be smaller than n, we can use multiple references, and encode the time step across them. For simplicity, we assume Int can hold a large enough value.) In defining terms using the language, we will need some shorthand.

- We will make use of operations such as + and equality, even though they are not in the language, as a shorthand for the appropriate construction using case.
- We abuse notation to write Ω for a non-terminating computation of arbitrary type (which can easily be constructed from the $\Omega : F\mathrm{Unit}$ in the language by doing Ω to $x.M$, where M is any computation of the correct type).
- We write inc *time* for the computation $!\mathit{time}$ to $t.(t+1)$ to $t'.\mathit{time} := t'$.
- Given an abstract value sequence $\overrightarrow{A}$ introduced in p_i, we write $\overrightarrow{A}\{\gamma_{i+1}\}$ for the substitution of the names introduce in A by thunks from γ_{i+1}.
- We will write $\lambda.\overrightarrow{x}$ to indicate a sequence of lambda abstractions, which we shall use to match the sequence of arguments of a computation type.
- If $\overrightarrow{x}$ is a sequence of variables, and $\overrightarrow{A}$ is a sequence of abstract values with matching types, we write $M\{\overrightarrow{x}/\overrightarrow{A}\}$ for the act of substituting names found in $\overrightarrow{A}$ with the corresponding variable from $\overrightarrow{x}$ in M (ignoring values occurring in $\overrightarrow{A}$). In particular, if we need to make a substitution of x : Ref for MkVar f g, we do the following: occurrences of force f are replaced by $!x$, and occurrences of (force g)V by $x := V$.
- Finally, we define $\mathit{assert}(x \sim A)$ to be a test comparing a variable with an abstract value. If the type of A is not Int, this is empty (i.e. return (), otherwise it is defined as case x of $(\Omega)_{i<A}$, return $(), (\Omega)_{A<i}$. We extend this to sequences as $\mathit{assert}(\overrightarrow{x} \sim \overrightarrow{A})$

The above description already specifies $\phi_i, \mathrm{dom}(\gamma_i), H_i, Fn_i$, and h_i. It also specifies the continuation names appearing in S_i, but not the continuations K appearing in the stack. To complete the definition of C_i, we will need to specify the environment γ_i and the stack S_i.

Recall that, we need to define $\gamma_0(f)$ for $f \in \phi$ and, in other cases, $\gamma_j(f)$ ($f \in \mathrm{TNames}$) will be defined for all $j \geq i$ if x was introduced by P in p_i. Recall also that once $\gamma_j(f)$ is defined, it never

changes. Hence if f was introduced by in p_i, we will only specify $\gamma_i(f)$ on the understanding that $\gamma_{i'}(f) = \gamma_i(f)$ for all $i' > i$.

We will provide these definitions using a backwards induction on i, meaning we define S_i and $\gamma_i(f)$ for f introduced in p_i with reference to S'_i and $\gamma_{i'}(f')$ for $i' > i$ and f' introduced in p'_i. In particular, the names ϕ are deemed to be introduced in a (fictional) p_o. Once $\gamma_i(x)$ is defined, we will argue that $\nu(\gamma_i(x)) \subseteq \mathsf{Vis}_P(o_1 p_1 \cdots o_i)$. Similarly, once S_i is defined, we will argue that if $(c', (K, c'') \in S_i$ with c' introduced in p_j, then $\nu(K) \in \mathsf{Vis}_P(o_1 p_1 \cdots o_j)$.

For the base case, where $i = n$, things are straight forward. The stack in this case must be empty (as we have just done a $\bar{\mathsf{o}}(A)$ action, so popped the last thing from the stack). The names introduced in p_n are never used, so we can define $\gamma_n(f) = \Omega$.

For the inductive case, will consider $\gamma_i(f)$ and S_i separately. We first define $\gamma_i(f)$ for $f \in$ TNames introduced in p_i, or ϕ if $i = 0$. Let I_f be the set of occurrences of f as head name in t_i. That is, $I_f = \{i < u \leq n \mid o_u = f(\overrightarrow{A_u}, c_u)\}$. We will now define

$$\gamma_i(f) \quad = \quad \text{thunk } (\lambda \overrightarrow{x}.(\text{inc } \mathit{time}; !\mathit{time} \text{ to } t.\text{case } t \text{ of } (M_j)_{0 \leq j \leq n})$$

This code simply increments the time, and dispatches to the appropriate M_j based on the time, which we now define by cases, analysis the type of move p_j is.

- $j \notin I_f$. In this case, we wish simply to diverge, so $M_j = \Omega$.
- $j \in I_f$ and $p_j = \bar{c_j}(A'_j)$. c_j is the continuation name introduced by o_j due to P-bracketing. Thus, the call to f at time j needs to simply return the value corresponding to A'_j. As $i < j$, we can use the inductive hypothesis to get that names introduced in A'_j are defined in γ_{j+1}. So we have
$$M_j = \mathit{assert}(\overrightarrow{x} \sim \overrightarrow{A_j}); \text{return } A'_j\{\gamma_{j+1}\}\{\overrightarrow{x}/\overrightarrow{A_j}\}$$
M_j uses only names in $A'_j\{\gamma_{j+1}\}$ which are not introduced in $\overrightarrow{A_j}$. We have by the IH that $\nu(\overrightarrow{A'_j}\{\gamma_{j+1}\}) \subseteq \mathsf{Vis}_P(o_1 \cdots o_j) = \mathsf{Vis}_P(o_1 \cdots o_i) \cup \nu(\overrightarrow{A_j})$. Thus, we have $M_j \subseteq \mathsf{Vis}_P(o_1 \cdots o_i)$.
- $j \in I_f$ and $p_j = \bar{g}(\overrightarrow{A'_j}, c'_j)$. In this case, by the fact p_j is an unanswered question, we have $S_{j+1} = (c'_j, (K, c_j)) : S'_{j+1}$. So what the call to f at time j needs to do is call g with argument sequence corresponding to $\overrightarrow{A'_j}$, and then have the result of this returned to K. As $i < j$, we can use the inductive hypothesis to get that names introduced in A'_j are defined in γ_{j+1}. So we have
$$M_j = \mathit{assert}(\overrightarrow{x} \sim \overrightarrow{A_j}); K[(\text{force } g)(\overrightarrow{A'_j}\{\gamma_{j+1}\})]\{\overrightarrow{x}/\overrightarrow{A_j}\}$$
This uses only names in $\overrightarrow{A'_j}\{\gamma_{j+1}\}$ and K which are not introduced in $\overrightarrow{A_j}$. By the IH, we have that $\nu(\overrightarrow{A'_j}\{\gamma_{j+1}\}) \in \mathsf{Vis}_P(o_1 \cdots o_j) = \mathsf{Vis}_P(o_1 \cdots o_i) \cup \nu(\overrightarrow{A_j})$. Similarly, we have that $\nu(K_{i+1}) \subseteq \mathsf{Vis}_P(o_1 \cdots o_j)$. Thus, we have $\nu(M_j) \subseteq \mathsf{Vis}_P(o_1 \cdots o_i)$.

Now we turn to constructing S_i, for which we proceed by a case analysis on o_i.

- o_i is a *OQ* action. Then this action does not change the stack. Consider then p_i. If this is a *PA* action, then we simply take $S_i = S_{i+1}$ as neither action changes the stack. If it is a *PQ* action $\bar{f}(\overrightarrow{A}, c')$, then as this is the most recent unanswered question, we have that $S_{i+1} = (c', (K, c'')) : S'_{i+1}$. We can therefore take $S_i = S'_{i+1}$. In either case, the inductive construction means this satisfies the visibility condition on S_i.
- o_i is a *OA* action, $c'(A)$. We therefore need to arrange that the topmost item on the stack is of the form $(c', (K, c''))$. As mentioned above, c'' is fixed. The appropriate definition for K depends on p_i.

– If p_i is a *PA* action, it has the form $\bar{c'}(A')$. We therefore construct K so that is returns the value corresponding to A' if *time* is i and diverges otherwise. By the IH, names introduced in A' are defined in γ_{i+1}. So

$$K = \bullet \text{ to } x.(\mathit{assert}(x \sim A); \text{inc } \mathit{time}; !\mathit{time} \text{ to } t. \\ \text{case } t \text{ of } ((\Omega)_{i<t}, (\text{return } (A')\{\gamma_{i+1}\}\{x/A\}), (\Omega)_{t<i}))$$

This uses only names in $A'\{\gamma_{i+1}\}$ which are not introduced in A. Assume c' is introduced in p_j. By the IH, we have that $\nu(A'\{\gamma_{i+1}\}) \subseteq \mathsf{Vis}_P(o_1 \cdots o_i) = \mathsf{Vis}_P(o_1 \cdots o_j) \cup \nu(A)$. We then have that $\nu(K) \subseteq \mathsf{Vis}_P(o_1 \cdots o_j)$, as required. Thus we let $S_i = (c', (K, c'')) : S_{i+1}$

– If p_i is a *PQ* action, it has form $\bar{f}(\overrightarrow{A'}, c''')$. In this case, as p_i is the most recent unanswered question, it must be that $S_{i+1} = (c''', (K_{i+1}, c'') : S'_{i+1}$. This means that we wish to construct K which calls f with argument sequence corresponding to $\overrightarrow{A'}$ time i, then uses the result of this in K_{i+1}. By the IH, names introduced in $\overrightarrow{A'}$ are defined in γ_{i+1}.

$$K = \bullet \text{ to } x.(\mathit{assert}(x \sim A); \text{inc } \mathit{time}; !\mathit{time} \text{ to } t. \\ \text{case } t \text{ of } ((\Omega)_{i<t}, (K_{i+1}[(\text{force } f)\,(\overrightarrow{A'}\{\gamma_{i+1}\})]\{x/A\}), (\Omega)_{t<i}))$$

This uses only names in $\overrightarrow{A'}\{\gamma_{i+1}\}$ and K_{i+1} which are not introduced in A. Let c' be introduced in p_j. By the IH, we have that $\nu(\overrightarrow{A'}\{\gamma_{i+1}\}) \in \mathsf{Vis}_P(o_1 \cdots o_i) = \mathsf{Vis}_P(o_1 \cdots o_j) \cup \nu(A)$. Similarly, we have that $\nu(K_{i+1}) \subseteq \mathsf{Vis}_P(o_1 \cdots o_i)$. Thus, we have $\nu(K) \subseteq \mathsf{Vis}_P(o_1 \cdots o_j)$. Thus we let $S_i = (c', (K, c'')) : S'_{i+1}$.

It is easy to verify that this definition of $\mathbf{C}_i$ does indeed generate the required traces. In particular, we can see that if O takes an action not corresponding to the one taken at o_i (an odd position), then the subsequent term component will reduce to Ω, and so produce no action. □

Proof of Theorem 23. Suppose $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$. Let ρ be a Γ-assignment, $A_i = \rho(x_i)$, $c : \sigma$ and $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_1}^{\rho_{\vec{A_i}}, c})$ s.t. t is complete. Then t is a $(\nu(\rho) \uplus \{c\}, \emptyset)$-trace. W.L.O.G, due to the closure of $\mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_1}^{\rho_{\vec{A_i}}, c})$ under renaming, we can ensure that $\circ$ does not appear in t. Let $\circ : \mathrm{Unit}$ and $t_1 = t^{\perp}\, \bar{\circ}(())$, a $(\{\circ\}, \nu(\rho) \uplus \{c\})$-trace. Then, as t is O-visible, O-bracketed, P-visible, P-bracketed and complete, so is $t^{\perp}$ and so t_1 is. Thus, we can appeal to Lemma 22 to get a passive configuration $\mathbf{C}_O = \langle \gamma_O, \phi \uplus \{c\}, h, [\circ \mapsto \emptyset], \phi \rangle, (c, (K, \circ)) : \perp)$ for some h, K, γ, such that $\mathbf{Tr}^{\mathrm{even}}_{\mathrm{CBPV}}(\mathbf{C}_O)$ consists of all even-length prefixes of t_1, up to renamings which preserve $\nu(\rho) \uplus \{c, \circ\}$.

Observe that $\mathbf{C}_O = \mathsf{C}^{\vec{\gamma_i}, c}_{h,K,\gamma}$ where $\gamma(x_i) = A_i\{\gamma_O\}$ and $\gamma_i = \gamma_O \upharpoonright \nu(A_i)$. Then we have the $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_1}^{\rho_{\vec{A_i}}, c})$ and $t_1 \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\vec{\gamma_i}, c}_{h,K,\gamma})$. We can then apply Lemma 20 (right-to-left) to obtain $(K[M_1]\{\gamma\}, h) \Downarrow_{ter}$, and as $\Gamma \vdash^c M_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2$, we have $(K[M_2]\{\gamma\}, h) \Downarrow_{ter}$. By Lemma 20 (left-to-right), we have a complete trace $t' \in\in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_2}^{\rho_{\vec{A_i}}, c})$ such that $t'^{\perp}\, \bar{\circ}(()) \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\vec{\gamma_i}, c}_{h,K,\gamma})$. By the definition of $\mathbf{C}_O$, all complete traces in $\mathbf{Tr}_{\mathrm{CBPV}}(\mathbf{C}_O)$ are equal to t_1 up to renaming of names preserving $\nu(\rho) \uplus \{c, \circ\}$, so t' is equal to t (up to a renaming of names preserving $\nu(\rho) \uplus \{c\}$). Therefore, by the closure property, we have $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{M_2}^{\rho_{\vec{A_i}}, c})$, as required. □

C.2 Additional material on Name Schemes

In Figure 15 we present the full definition of $\mathbf{Base}^{\Delta}_t()$, including the function $\mathbf{Match}(A, B, f)$.

Proof of Lemma 37. This is a proof by induction on the length of t. The base case is when $t = t'\ \bar{c_0}(A)$ with $c_0 \in N_O$ or $t = t'\ \bar{g}(\overrightarrow{A}, c)$ with $g \in N_O$. Then $\mathsf{Vis}_O(t)$ is $\nu(A)$ or $\nu(\overrightarrow{A})$, and so all the base name are distinct. The inductive case $t = t'\ \bar{c}(A)$ is trivial, as $c : F\mathrm{Int}$ or $c : F\mathrm{Unit}$ so this reduces to an application of the I.H. The other inductive case is $t = t'\ \bar{g}(\overrightarrow{A}, c)$. Assume that there is some $f, f' \in \mathsf{Vis}_O(t)$ s.t. $\mathbf{Base}^{\Delta}_t(f) = \mathbf{Base}^{\Delta}_t(f')$. Then by the I.H, one of f, f'

$$
\begin{array}{lcll}
\mathbf{Base}^{\Delta}_t(n) & \triangleq & n & \text{where } n \in N_O \\
\mathbf{Base}^{\Delta}_t(c) & \triangleq & \mathrm{Suc}_{\mathrm{C}}(g) & \text{where } c \text{ is introduced in } f(A,c) \text{ or } \bar{f}(A,c) \text{ and } g = \mathbf{Base}^{\Delta}_t(f) \\
\mathbf{Base}^{\Delta}_t(f) & \triangleq & g & \text{where } f \text{ is introduced in } c(A) \text{ or } \bar{c}(A) \text{ with } c : \sigma,\ c' = \mathbf{Base}^{\Delta}_t(c) \\
 & & & \{B\} = \mathbf{BVals}^{\Delta}_{\sigma}(c) \text{ and } g = \mathbf{Match}(A, B, f) \\
\mathbf{Base}^{\Delta}_t(f) & \triangleq & g & \text{where } f \text{ is introduced in } f'(\overrightarrow{A}, c) \text{ or } \bar{f}'(\overrightarrow{A}, c) \text{ with } g' = \mathbf{Base}^{\Delta}_t(f'), \\
 & & & \overrightarrow{B} \in \mathbf{BValSeq}^{\Delta}(g') \text{ and } g = \mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, f)
\end{array}
$$

$$
\begin{array}{rcll}
\mathbf{Match}(f, g, f) & \triangleq & g & \\
\mathbf{Match}(\{\!|f, f'|\!\}, \{\!|g, g'|\!\}, f) & \triangleq & g & \\
\mathbf{Match}(\{\!|f', f|\!\}, \{\!|g', g|\!\}, f) & \triangleq & g & \\
\mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, f) & \triangleq & \mathbf{Match}(\overrightarrow{A}_i, \overrightarrow{B}_i, f) & \text{where } f \in \nu(\overrightarrow{A}_i)
\end{array}
$$

Figure 15: The function $\mathbf{Base}^{\Delta}_t()$ which converting names appearing in t to base names from $\Delta = (\mathrm{TBNames}, \mathrm{CBNames}, \rho, c_0, \mathrm{Suc}_{\mathrm{T}}, \mathrm{Suc}_{\mathrm{C}})$

is introduced in $\overrightarrow{A}$. W.L.O.G, f was introduced in $\overrightarrow{A}$, and f' in some earlier move $\bar{g'}(\overrightarrow{A'}, c')$ with $\mathbf{Base}^{\Delta}_t(g) = \mathbf{Base}^{\Delta}_t(g')$. Observe that $\mathbf{Base}^{\Delta}_t(g)$ must be a level 0 name. Recall that the definition of $\mathsf{Vis}_O(t)$ involves a process of chasing names. We can therefore consider the sequence of O-names used as the head names for moves which introduce P-names in the definition of $\mathsf{Vis}_O(t)$. We will show that the base names in this sequence are exactly the introduction chain for $\mathbf{Base}^{\Delta}_t(g)$, from which it will follow that g' cannot be in this sequence (and so $f' \notin \mathsf{Vis}_O(t)$).

We prove this by induction on the length of the introduction chain. In the base case, $\mathbf{Base}^{\Delta}_t(g)$ is initial, so $g \in N_O$, and $\mathsf{Vis}_O(t) = \nu(\overrightarrow{A})$, so no other names in the sequence. In the inductive case, let $g \in \nu(A)$ s.t. $\mathsf{Vis}_O(t) = \mathsf{Vis}_O(s\ \mathbf{a}\ d(A)\ s'\ \bar{g}(\overrightarrow{A}, c)) = \nu(\overrightarrow{A}) \cup \mathsf{Vis}_O(s\ \mathbf{a})$. Consider what $\mathbf{a}$ is. If it is a PQ-action, then it must be on continuation name $d' \neq c_0$, so we have $\mathsf{Vis}_O(s\ \mathbf{a}) = \mathsf{Vis}_O(s_1\ h(\overrightarrow{A''}, d')\ \bar{d'}()) = \mathsf{Vis}_O(s_1)$. By repeating this, it suffices to consider only the case that $\mathbf{a}$ is a PQ-action, and so by the bracketing condition, $\mathbf{a} = \bar{h}(\overrightarrow{A''}, d)$. Then $\mathsf{Vis}_O(s\ \mathbf{a}) = \nu(\overrightarrow{A''}) \cup \mathsf{Vis}_O(s)$. We must have $\mathbf{Base}^{\Delta}_t(d) = \mathrm{Suc}_{\mathrm{C}}(\mathbf{Base}^{\Delta}_t(h))$ and $\mathbf{Base}^{\Delta}_t(g) = \mathrm{Suc}_{\mathrm{T}}(\mathbf{Base}^{\Delta}_t(d))$, so $\mathbf{Base}^{\Delta}_t(h)$ is earlier in the introduction chain of $\mathbf{Base}^{\Delta}_t(g)$, as required. Applying the I.H. to the introduction chain of $\mathbf{Base}^{\Delta}_t(h)$ completes this proof. □

Proof of Lemma 40. We will prove this by contradiction. Let t_1, t_2 be s.t. $\mathbf{Rename}^{\Delta}(t_1) = \mathbf{Rename}^{\Delta}(t_2)$ but are not equal up to permutations of names which preserve N_O. Observe that t_1 and t_2 must have the same sequence of moves, differing only in the head names. Consider the shortest (equal length) prefixes s_1 and s_2 of t_1 and t_2 which are not equal up to permutation of names. Apply a permutation to s_1 and s_2 so that they are equal, save for the last action. Let s be the common prefix, and observe the action they disagree on must have been a question, as the head name used in answer actions is determined by the bracketing conditions.

Now, by Lemma 37, this was not an OQ-action, as there are not distinct $f, f' \in \mathsf{Vis}_O(s)$ with $\mathbf{Base}^{\Delta}_s(f) = \mathbf{Base}^{\Delta}_s(f')$. So this last move was a PQ-action $\mathbf{a}_q$, so let f, f' be the two distinct head names with $\mathbf{Base}^{\Delta}_s(f) = g = \mathbf{Base}^{\Delta}_s(f')$. Let $\mathbf{a}_a$ be the action f is introduced in. Now, $\mathbf{Rename}^{\Delta}(t_1)$ will contain a marked Δ-trace with g marked at the action corresponding to $\mathbf{a}_a$, and in the head of the action corresponding to $\mathbf{a}_q$. But $\mathbf{Rename}^{\Delta}(t_2) = \mathbf{Rename}^{\Delta}(s_2)$ cannot contain this trace, as $\mathbf{Marked}(s_2)$ cannot contain a trace with both f in $\mathbf{a}_a$ marked and f' in $\mathbf{a}_q$ marked. Thus, we have a contradiction. □

C.3 Proof of Lemma 44

To complete the proof of Lemma 44, we need to give a proof of Lemma 45. To do this, we need the following result.

Lemma 101. *Let* $t\ \bar{f}(\overrightarrow{A}, c)\ t'\ c(A)$ *be a* PTR*-trace, with* f *a level 2 name. Then* $\mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A}, c)\ t') = \mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A}, c))$

Proof. We prove this with the following induction hypothesis: for s a prefix of t' with no unanswered OQ-action, $\mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A}, c)\ s) = \mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A}, c))$. As t' is such a prefix, this shows the results.

- *Base case* In the case s has length 0, then the result is immediate.
- *Inductive case* Observe that the bracketing condition implies that $s = s'\ f'(\overrightarrow{A'}, c')\ s''\ \bar{c'}(A')$, and so $\mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A}, c)\ s) = \mathsf{Vis}_O(t\ \bar{f}(\overrightarrow{A}, c)\ s') \cup \nu(A') = \mathsf{Vis}_O(t\ \bar{f}(A', c')\ s')$ as in PTR, A' contains no thunk names. We can then appeal to the I.H. on s'' to get the result.

□

Lemma 102 (Lemma 45)**.** *Let* $t\ g(\overrightarrow{A}, c)\ t'\ \bar{f}(\overrightarrow{A'}, c')$ *be a* PTR*-trace, with* f *a level 2 name, and* f' *its originator, introduced in* $\overrightarrow{A}$*. Then* $\mathsf{Vis}_O(t\ g(\overrightarrow{A}, c)\ t'\ \bar{f}(\overrightarrow{A'}, c')) = \mathsf{Vis}_O(t)$

Proof. This proof is by induction on the length of the introduction chain of f.

- *Base case* If $f = f'$ (f is its own originator), we have $\mathsf{Vis}_O(t\ g(\overrightarrow{A}, c)\ t'\ \bar{f}(\overrightarrow{A'}, c')) = \mathsf{Vis}_O(t)$.
- *Inductive case* We have that $t' = s\ \bar{f''}(\overrightarrow{A''}, c'')\ s'\ c''(A'')\ s''$ where f' is the originator of f'', and f is introduced in A''. We have that $\mathsf{Vis}_O(t\ g(\overrightarrow{A}, c)\ t'\ \bar{f}(\overrightarrow{A'}, c')) = \mathsf{Vis}_O(t\ g(\overrightarrow{A}, c)\ s\ \bar{f''}(\overrightarrow{A''}, c'')\ s')$. Now, by the I.H. $\mathsf{Vis}_O(t\ g(\overrightarrow{A}, c)\ s\ \bar{f''}(\overrightarrow{A''}, c'')) = \mathsf{Vis}_O(t)$. By Lemma 101 we have $\mathsf{Vis}_O(t\ g(\overrightarrow{A}, c)\ s\ \bar{f''}(\overrightarrow{A''}, c'')\ s') = \mathsf{Vis}_O(t)$. Thus $\mathsf{Vis}_O(t\ g(\overrightarrow{A}, c)\ t'\ \bar{f}(\overrightarrow{A'}, c')) = \mathsf{Vis}_O(t)$, as required.

□

C.4 Proof of Correctness for $\mathcal{L}_{\mathrm{PTR}}$

In this appendix, we give more fully the proof of correctness for $\mathcal{L}_{\mathrm{PTR}}$. For clarity, we will state again some of the definitions and results given in the main body of the paper. We will identify when we have repeated definitions or results. To simplify the statement of results in this sections, we will fix a PTR-computation $\Gamma \vdash^c M : F\sigma_0$, (Γ, σ)-name scheme $\Delta = (\mathrm{TB}, \mathrm{CB}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$, and let $N_O = \nu(\rho) \cup \{c_0\}$.

We start with a helpful result relating $\mathbf{BVals}^{\Delta}_{\sigma}(d)$ and $\mathbf{IVal}^{\Delta}_{\sigma}(d, V, \eta)$. We write $\circ$ for composition of (partial) functions.

Lemma 103. *Let* κ *be a mapping from names to (indexed) base names,* $V : \sigma$ *and* $(A, \gamma) \in \mathbf{AVal}_{\sigma}(V)$*. Let* t *be a* $(N_O, \emptyset)$*-trace in which* A *occurs in an action answer* $\mathbf{a}$ *with head name* c*. If* $(B, \gamma', \eta') = \mathbf{IVal}^{\Delta}_{\sigma}(\mathbf{Base}^{\Delta}_{t}(c), \kappa(V), \eta)$*, then* $\beta(B) = \mathbf{Base}^{\Delta}_{t}(A)$*, and* $\gamma' = \kappa \circ \gamma \circ [\mathbf{Match}(A, B, f) \mapsto f]_{f \in \nu(A)}$.

Lemma 104. *Let* κ *be a mapping from names to (indexed) base names,* $V : \sigma$ *and* $(\overrightarrow{A}, \gamma) \in \mathbf{AVal}(\overrightarrow{V})$*. Let* t *be a* $(N_O, \emptyset)$*-trace in which* $\overrightarrow{A}$ *occurs in an question answer* $\mathbf{a}$ *with with head name* f*. If* $(\overrightarrow{B}, \gamma', \eta') = \mathbf{IVal}^{\Delta}_{\sigma}(\mathbf{Base}^{\Delta}_{t}(f), \kappa(V), \eta)$*, then* $\beta(\overrightarrow{B}) = \mathbf{Base}^{\Delta}_{t}(\overrightarrow{A})$*, and* $\gamma' = \kappa \circ \gamma \circ [\mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, g) \mapsto g]_{g \in \nu(\overrightarrow{A})}$.

Lemma 105. *Let* $c : \sigma \in \mathrm{CB}$*,* η *a map from* $\mathrm{TB} \cup \mathrm{CB}$ *to indexes, and* $(B, \eta') \in \mathbf{IVals}^{\Delta}_{\sigma}(c, \eta)$*. Let* t *be an* $(N_O, \emptyset)$*-trace and* c' *a P-name in* t *s.t.* $c = \mathbf{Base}^{\Delta}_{t}(c')$*. Then there exists* A *s.t.* $\beta(B) = \mathbf{Base}^{\Delta}_{t\ c(A)}(A)$.

Lemma 106. *Let* $f \in \mathrm{TB}$*,* η *a map from* $\mathrm{TB} \cup \mathrm{CB}$ *to indexes, and* $(\overrightarrow{B}, \eta') \in \mathbf{IValSeq}^{\Delta}(f, \eta)$*. Let* t *be an* $(N_O, \emptyset)$*-trace and* f' *a P-name in* t *s.t.* $f = \mathbf{Base}^{\Delta}_{t}(f')$*. Then there exists* $\overrightarrow{A}$ *s.t.* $\beta(\overrightarrow{B}) = \mathbf{Base}^{\Delta}_{t\ f'(\overrightarrow{A}, c)}(\overrightarrow{A})$.

We will prove the correctness using a bisimulation technique, though we will not be able to give a bisimulation directly between $\mathcal{L}_{\text{CBPV}}$ and $\mathcal{L}_{\text{PTR}}^{\Delta_{\Gamma,\sigma}}$. Instead, we will introduce an LTS $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$, the traces of which will be exactly $\mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$. Given a path p, let $\text{Tr}(p)$ be the trace induced by p.

Definition 107 (Definition 50)**.** The configurations of $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ have the form (p, F) where p is a path in $\mathcal{L}_{\text{CBPV}}$ starting in C_M^{ρ,c_0} and F is either empty, or a set containing a single name f s.t. f is introduced in an O-action in $\text{Tr}(p)$. Let $\text{Mark}(f, X)$ for some structure X (an action etc.) be obtained by marking every occurrence of f. The transitions are then (where $\mathbf{a}$ includes τ actions)

- $(p, \{f\}) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \{f\})$ where p ends with $\mathbf{C}$, $\mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$, $\mathbf{a}'' = \text{Mark}(f, \mathbf{a})$ and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a}'')$.
- $(p, \emptyset) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \emptyset)$ where p ends with $\mathbf{C}$, $\mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$, and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a})$.
- $(p, \emptyset) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \{g\})$ where $\mathbf{a} = c(A)$, $g \in \nu(A)$, $\mathbf{a}'' = \text{Mark}(g, \mathbf{a})$ and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a}'')$.
- $(p, \emptyset) \xrightarrow{\mathbf{a}'} (p \xrightarrow{\mathbf{a}} \mathbf{C}', \{g\})$ where $\mathbf{a} = f(\overrightarrow{A}, c)$, $g \in \nu(\overrightarrow{A})$, $\mathbf{a}'' = \text{Mark}(g, \mathbf{a})$ and $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{\text{Tr}(p \xrightarrow{\mathbf{a}} \mathbf{C}')}(\mathbf{a}'')$.

Observe that $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ is deterministic in the sense that if for configuration (p, F), there is only one t s.t. $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t}{\Rightarrow} (p, F)$.

Lemma 108 (Lemma 51)**.** *The traces of* $\mathcal{L}_{\text{Path}}^{\Delta_{\Gamma,\sigma}}$ *starting from* $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$ *are exactly* $\mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\text{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$.

Proof. We prove this by first proving the follwing result. Let p is a path in $\mathcal{L}_{\text{CBPV}}$ starting from C_M^{ρ,c_0}, and $t = \text{Tr}(p)$. Then

1. Then $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t'}{\Rightarrow} (p, \emptyset)$ where $t' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_t(t)$.
2. If t a thunk O-name introduced in t, then $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t''}{\Rightarrow} (p, \{f\})$, where $t'' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_t(\text{Mark}(f, t))$.

This proceeds by induction on the length of the path p. The base case is when p is simply C_M^{ρ,c_0}, which is trivial. The inductive cases are as follows. Let $p = p_1 \xrightarrow{\mathbf{a}} \mathbf{C}'$ and $t_1 = \text{Tr}(p_1)$. If $\mathbf{a} \neq \tau$, then $t = t_1\ \mathbf{a}$.

1. Let $t_1' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{t_1}(t_1)$. By the I.H.(1), we have that $\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t_1'}{\Rightarrow} (p_1, \emptyset)$. Now, by definiton of the transition, $(p_1, \emptyset) \xrightarrow{\mathbf{a}'} (p_1 \xrightarrow{\mathbf{a}} \mathbf{C}', \emptyset)$ where $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_t(\mathbf{a})$. If $\mathbf{a} = \tau$, $t' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_t(t) = t_1'$, otherwise $t' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_t(t) = t_1'\ \mathbf{a}'$. Thus, we obtain $\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t'}{\Rightarrow} (p, \emptyset)$, as required.
2. For this case, do a case analysis on whether f is introduced in $\mathbf{a}$.

   If it is, then let $t_1' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{t_1}(t_1)$. By the I.H.(1), we have that $\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t_1'}{\Rightarrow} (p_1, \emptyset)$. Now, by definiton of the transition, $(p_1, \emptyset) \xrightarrow{\mathbf{a}'} (p_1 \xrightarrow{\mathbf{a}} \mathbf{C}', \{f\})$ where $\mathbf{a}' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_t(\mathbf{a}'')$ and $\mathbf{a}'' = \text{Mark}(f, \mathbf{a})$. Observe that $\text{Mark}(f, t) = t_1\ \mathbf{a}''$, so $t'' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_t(\text{Mark}(f, t)) = t_1'\ \mathbf{a}'$. Thus we obtain $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t''}{\Rightarrow} (p, \{f\})$, as required.

   If it is not, then f must be intoduced in t_1. Let $t_1'' = \mathbf{Base}^{\Delta_{\Gamma,\sigma}}_{t_1}(\text{Mark}(f, t_1))$. By the I.H(2), we have $\mathsf{C}_M^{\rho,c_0}, \emptyset) \overset{t_1''}{\Rightarrow} (p_1, \{f\})$. Now, by definiton of the transition, $(p_1, \{f\}) \xrightarrow{\mathbf{a}'}$

$(p_1 \xrightarrow{\mathbf{a}} \mathbf{C}', \{f\})$ where $\mathbf{a}' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(\mathbf{a}'')$ and $\mathbf{a}'' = \mathrm{Mark}(f, \mathbf{a})$. If $\mathbf{a} = \tau$, then $\mathrm{Mark}(f,t) = \mathrm{Mark}(f, t_1)$ so $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(\mathrm{Mark}(f,t)) = t_1''$. Otherwise, $\mathrm{Mark}(f,t) = \mathrm{Mark}(f,t_1)\ \mathbf{a}''$, so $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(\mathrm{Mark}(f,t)) = t_1''\ \mathbf{a}'$. Thus we obtain $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \xRightarrow{t''} (p, \{f\})$, as required.

From this we can prove the result as follows.

For the first direction, let $t'' \in \mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$. Then there exists $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_M^{\rho,c_0})$ and $t' \in \mathbf{Marked}(t)$ s.t. $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(t')$. Let p be the path in generating t. If $t = t'$, we have by the above property that $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \xRightarrow{t''} (p, \emptyset)$, so t'' is generated by $\mathcal{L}_{\mathrm{Path}}^{\Delta_{\Gamma,\sigma}}$ starting from $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$. Otherwise, let f be the name marked in t'. By the above property, $(\mathsf{C}_M^{\rho,c_0}, \emptyset) \xRightarrow{t''} (p, \{f\})$, so t'' is generated by $\mathcal{L}_{\mathrm{Path}}^{\Delta_{\Gamma,\sigma}}$ starting from $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$.

For the other direction, let t'' be a trace generated by $\mathcal{L}_{\mathrm{Path}}^{\Delta_{\Gamma,\sigma}}$ starting from $(\mathsf{C}_M^{\rho,c_0}, \emptyset)$. Then there must be a path p s.t. $\mathsf{C}_M^{\rho,c_0}, \emptyset) \xRightarrow{t''} (p, \emptyset)$ or $\mathsf{C}_M^{\rho,c_0}, \emptyset) \xRightarrow{t''} (p, \{f\})$ for some f introduced in an O-action of p. Let $t = \mathrm{Tr}(p)$, so $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_M^{\rho,c_0})$. Then it follows from the above result, and the determinstm of $\mathcal{L}_{\mathrm{Path}}^{\Delta_{\Gamma,\sigma}}$, that $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(t)$ or $t'' = \mathbf{Base}_t^{\Delta_{\Gamma,\sigma}}(\mathrm{Mark}(f,t))$. Thus, $t'' \in \mathbf{Rename}^{\Delta_{\Gamma,\sigma}}(\mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_M^{\rho,c_0}))$. □

We will now establish a bisimulation between $\mathcal{L}_{\mathrm{Path}}$ and $\mathcal{L}_{\mathrm{PTR}}^{\Delta_{\Gamma,\sigma}}$. To construct this, we will use the function $\mathbf{Conv}(p)$, which will take path p to a tuple $(\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$. κ maps names introduced in p to indexed base names, and locations to numerical locations. We lift κ to any structure containing names and/or locations in the obvious way. ψ maps indexed base names to names introduced in p (a partial inverse of κ), η maps base names to the next index and μ maps level 2 and O-continuation indexed base names to the value of (i_h, η) before their introduction. ψ^h, i_h play an analogous role to ψ, η but for locations. Finally, N is simply the term components of the final state in p (or $\emptyset$ if the last configuration is passive), but with the result of expanding while loops annotated with $\mathrm{end}(i_h, \eta)$.. T is an extended stack, which relates to the stack in the same way as N does to term, but with added elements in some levels of the stack. We also define the reduction $(N, h) \to_m (N', h')$ on terms of the extended syntax (with $\mathrm{end}(i_h, \eta).$), as having the same rules as $\to$ (and so getting stuck if it encounters $\mathrm{end}(i_h, \eta).$). Let $\tilde{N}$ be the function which removes occurrences of $\mathrm{end}(i_h, \eta).$ from a term/context.

We write $p'\ \mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$ when $\mathbf{C}, \mathbf{C}'$ are the last two configurations in a path (where p' can be empty). Recall also that we have $\mathbf{Match}(A, B, f)$ transfers any mark on f in A to its result.

We recall Definition 52. We defined $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$ with the following cases.

If $p = \mathsf{C}_M^{\rho,c_0}$ then $\mu = \psi^h = \emptyset$, $i_h = 0$, $N = M$, $K = \bot$, $\kappa = [N_O \mapsto N_O^0]$, $\psi = [N_O^0 \mapsto N_O]$, η is 1 on N_O, 0 otherwise. Otherwise, $p = p'\ \mathbf{C} \xrightarrow{\mathbf{a}} \mathbf{C}'$. Let $\mathbf{Conv}_F(p'\ \mathbf{C}) = (\kappa', \psi', \eta', \mu', \psi'^h, i_h', N', T')$. We then proceed by the follwoing cases:

- $\mathbf{a} = \tau$, $\mathbf{C} = (\langle K[\mathrm{ref}\, V], \cdots \rangle, S)$ and $\mathbf{C}' = (\langle K[\ell], \cdots \rangle, S)$. If $N' = K'[\mathrm{ref}\, V]$ then $N = K'[i_h']$, $i_h = i_h' + 1$, $\kappa = \kappa' \cdot [\ell \mapsto i_h']$, $\psi^h = \psi'^h \cdot [i_h' \mapsto \ell]$, $\psi = \psi'$, $\eta = \eta'$, $\mu = \mu'$, and $T = T'$;
- $\mathbf{a} = \tau$ where $\mathbf{C} = (\langle K[\mathrm{while}\, N_1\, \mathrm{do}\, N_2], \cdots \rangle, S)$.
  - If $N' = K'[\mathrm{while}\, N_1\, \mathrm{do}\, N_2]$, then

    $$N = K'[N_1 \text{ to } x.\mathrm{case}\ x \text{ of return } (), (N_2 \text{ to } y.\mathrm{end}(i_h, \eta).\mathrm{while}\, N_1\, \mathrm{do}\, N_2)_{j>0}]$$

    and $\kappa = \kappa'$, $\psi = \psi'$, $\eta = \eta'$, $\mu = \mu'$, $\psi^h = \psi'^h$, $i_h = i_h'$, and $T = T'$;
  - If $N' = K'[\mathrm{end}(i_h, \eta).\mathrm{while}\, N_1\, \mathrm{do}\, N_2]$, then

    $$N = K'[\mathrm{end}M \text{ to } x.\mathrm{case}\ x \text{ of return } (), (N \text{ to } y.\mathrm{end}(i_h, \eta).\mathrm{while}\, M\, \mathrm{do}\, N)_{j>0}.]$$

    $\psi = \psi'_{<\eta}$, $\psi^h = \psi'^h_{<i_h}$, $\kappa = \kappa'$, $\mu = \mu'$, and $T = T'$;

- $\mathbf{a} = \tau$ otherwise, where $\mathbf{C} = (\langle M', c, \gamma, \phi, h, H\rangle, S)$, $\mathbf{C}' = (\langle M'', c, \gamma, \phi, h', H\rangle, S))$. Then N is s.t $(N', h) \rightarrow_m (N, h')$, $\kappa = \kappa'$, $\psi = \psi'$, $\eta = \eta'$, $\mu = \mu'$, $\psi^h = \psi'^h$, $i_h = i'_h$, and $T = T'$;
- $\mathbf{a} = \bar{f}(\overrightarrow{A}, c)$ where f is not a level 2 name, and $\mathbf{C} = (\langle K[\text{force } f\ \overrightarrow{V}], c', \cdots\rangle, S')$. If $N' = K'[\text{force } f\ \overrightarrow{V}]$, then $T = (c, (K', c')) : T'$, $N = \emptyset$ $\mu = \mu'$, $\psi^h = \psi'^h$, and $i_h = i'_h$. If $g^i = \kappa'(f)$,$d = \text{Suc}_{\text{C}}(g)$, and $(\overrightarrow{B}, \gamma, \eta'') = \mathbf{IVal}^{\Delta}(g, \overrightarrow{V}, \eta')$, then $\kappa = \kappa' \cdot [c \mapsto d^0] \cdot [f' \mapsto \mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, f')]_{f' \in \nu(\overrightarrow{A})}$, and $\psi = \psi' \cdot [\mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, f') \mapsto f']_{f' \in \nu(\overrightarrow{A})}$;
- $\mathbf{a} = \bar{f}(\overrightarrow{A}, c)$ where f is a level 2 name and $\mathbf{C} = (\langle K[\text{force } f\ \overrightarrow{V}], c', \cdots\rangle, S')$. If $N' = K'[\text{force } f\ \overrightarrow{V}]$ and $(i_h, \eta) = \mu(f)$, then $\psi = \psi'_{<\eta}$, $\psi^h = \psi'^h_{<i_h}$, $T = (c, (K', c'), (i'_h, \eta', \psi'^h_{\geq i_h}, \psi'_{\geq \eta})) : T'$, $N = \emptyset$, $\kappa = \kappa' \cdot [c \mapsto d^0]$, and $\mu = \mu'$;
- $\mathbf{a} = f(\overrightarrow{A}, c)$ where $\mathbf{C} = (\langle \gamma, \phi, h, H, Fn\rangle, S)$. If $N' = \emptyset$ then $N = \gamma(f)$, $\psi^h = \psi'^h$, $i_h = i'_h$, and $T = T'$. If $g^i = \kappa'(f)$, $d = \text{Suc}_{\text{C}}(g)$, $j = \eta(d)$, $(\overrightarrow{B}, \eta'') = \mathbf{IValSeq}^{\Delta}(g, \eta')$ and $\overrightarrow{A'} = \text{Mark}(F, \overrightarrow{A})$, then $\kappa = \kappa' \cdot [c \mapsto d^j] \cdot [f \mapsto \mathbf{Match}(\overrightarrow{A'}, \overrightarrow{B}, f)]_{f \in \nu(\overrightarrow{A})}$, $\psi = \psi' \cdot [d^j \mapsto c] \cdot [\mathbf{Match}(\overrightarrow{A'}, \overrightarrow{B}, f) \mapsto f]_{f \in \nu(\overrightarrow{A})}$, $\mu = \mu' \cdot [\nu(\overrightarrow{A}), c \mapsto (i_h, \eta)]$, and $\eta = \eta''[d \mapsto j + 1]$;
- $\mathbf{a} = \bar{c_0}(A)$ where $\mathbf{C} = (\langle \text{return } V, \cdots\rangle, S)$. If $(B, \gamma, \eta) = \mathbf{IVal}^{\Delta}_{\sigma}(c_0, V, \eta')$, then $N = \emptyset$, $\kappa = \kappa' \cdot [f \mapsto \mathbf{Match}(A, B, f)]_{f \in \nu(A)}$, $\psi' = \psi \cdot [\mathbf{Match}(A, B, f) \mapsto f]_{f \in \nu(A)}$, $\mu = \mu'$, $\psi^h = \psi'^h$, $i_h = i'_h$,and $T = T'$;
- $\mathbf{a} = \bar{c}(A)$. If $(i_h, \eta) = \mu(c)$, then $\psi = \psi'_{<\eta}$, $\psi^h = \psi'^h_{<i_h}$, $N = \emptyset$, $\kappa = \kappa'$, $\mu = \mu'$, and $T = T'$;
- $\mathbf{a} = c(A)$ with $A : \sigma$.
  - If $T' = (c, (K, c')) : T''$ and $d = \kappa'(c)$, then $T = T''$, $N = K[\text{return } A]$, and for $A' = \text{Mark}(F, A)$, for any $(B', \eta) \in \mathbf{IVals}^{\Delta}_{\sigma}(d, \eta')$, $\kappa = \kappa' \cdot [f \mapsto \mathbf{Match}(A', B, f)]_{f \in \nu(A)}$, $\psi = \psi' \cdot [\mathbf{Match}(A', B, f) \mapsto f]_{f \in \nu(A)}$, $\mu = \mu'$, $\psi^h = \psi'^h$, and $i_h = i'_h$;
  - If $T' = (c, (K, c'), (i_h, \eta'', \psi''^h, \psi'')) : T''$, $d = \kappa'(c)$, and ψ''^h, ψ'' disjoint from ψ'^h, ψ' then $T = T''$, $N = K[\text{return } A]$, and for $A' = \text{Mark}(F, A)$, for any $(B', \eta) \in \mathbf{IVals}^{\Delta}_{\sigma}(d, \eta'')$, $\psi^h = \psi'^h \cdot \psi''^h$ $\kappa = \kappa' \cdot [f \mapsto \mathbf{Match}(A', B, f)]_{f \in \nu(A)}$, $\psi = \psi' \cdot \psi'' \cdot [\mathbf{Match}(A', B, f) \mapsto f]_{f \in \nu(A)}$ and $\mu = \mu' \cdot [\nu(A) \mapsto (i'_h, \eta')]$.

The following Lemmata follow from simple inductions, and ensure that for any path p, $\mathbf{Conv}_F(p)$ is defined (that is, the conditions assume in the inductive construction do hold).

Lemma 109. *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, and $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$. Then for name $d \in \text{dom}(\kappa)$, $\kappa(d) = b^i$ for some i where $b = \mathbf{Base}^{\Delta}_{\text{Tr}(p)}(d)$.*

Lemma 110. *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$ with p ending in active configuration $(\langle M', c, \cdots\rangle, S)$, and $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$. Then $M' = \tilde{N}$ and replacing entries of the form $(c_1, (K, c_2), P)$ by $(c_1, (\tilde{K}, c_2))$ in T produces S.*

Lemma 111. *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, and $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$. Then for name $b^i \in \text{dom}(\psi)$, $i < \eta(b)$, and for $i \in \text{dom}(\psi^h)$, $i < i_h$.*

Lemma 112. *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$ where p ends with a passive state and $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, (c, (K, c'), (i'_h, \eta', \psi'^h, \psi')) : T)$. Then the domains of ψ, ψ', and ψ^h, ψ'^h are disjoint. Moreover, ψ' is defined on d^i where $\eta(d) \leq i < \eta'(d)$, and ψ'^h on $i_h \leq i < i'_h$.*

The following two Lemmata express the intuitive property that extending a path does not remove mappings from κ, μ in $\mathbf{Conv}_F(p)$, and that the stack T follows a stack discipline.

Lemma 113. *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, and $(p'F')$ a configuration s.t $F' = F$ or $F = \emptyset$, p' is a path extending p. Let $\mathbf{Conv}_F(p\ \mathbf{C}) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$ and $\mathbf{Conv}_F(p'\ \mathbf{C}') = (\kappa', \psi', \eta', \mu', \psi'^h, i'_h, N', T')$. Then κ', μ' are extensions of κ, μ.*

Lemma 114. *Let $(p\ \mathbf{C}, F)$ be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, and $(p'\ \mathbf{C}', F')$ a configuration s.t $F' = F$ or $F = \emptyset$, p' is a path extending $p\ \mathbf{C}$, no answer to a question before $\mathbf{C}$ occurs between $\mathbf{C}$ and $\mathbf{C}'$, and $\mathbf{C}'$ has the same stack as $\mathbf{C}$. Let $\mathbf{Conv}_F(p\ \mathbf{C}) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$ and $\mathbf{Conv}_F(p'\ \mathbf{C}') = (\kappa', \psi', \eta', \mu', \psi'^h, i'_h, N', T')$. Then $T = T'$.*

We now establish a useful Lemma regarding the preservation of values of ψ along a path.

Lemma 115. *Let $(p\ \mathbf{C}, F)$ be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, and $(p'\ \mathbf{C}', F')$ a configuration s.t $F' = F$ or $F = \emptyset$, p' is a path extending $p\ \mathbf{C}$, no answer to a question before $\mathbf{C}$ occurs between $\mathbf{C}$ and $\mathbf{C}'$, and $\mathbf{C}'$ has the same stack as $\mathbf{C}$. Let $\mathbf{Conv}_F(p\ \mathbf{C}) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$ and $\mathbf{Conv}_F(p'\ \mathbf{C}') = (\kappa', \psi', \eta', \mu', \psi'^h, i'_h, N', T')$.*

Then if $\mathbf{C}$ and $\mathbf{C}'$ are both active (with N containing no occurrences of end.*) or both passive, then $\psi = \psi'_{<\eta}$ and $\psi^h = \psi'^h_{<i_h}$. Further, if they are both passive, $\eta = \eta'$ and $i_h = i'_h$, and if they are both active, for all b, $\eta(b) \leq \eta'(b)$ and $i_h \leq i'_h$.*

Proof. This is by an induction on the difference in length between p and p'. We then proceed by cases on the last action $\mathbf{a}$ in p'. By the constraints on p', if $\mathbf{C}'$ is active, $\mathbf{a}$ is either a τ transition or a *OA*-action. The case of a τ-transition is simple to verify. The most interesting one is when the $\mathbf{Conv}_F(p') = (\kappa'', \psi'', \eta'', \mu'', \psi''^h, i''_h, K[\text{end}(i'_h, \eta').\text{while}\, M_1\, \text{do}\, M_2], T)$. In this case, we can appeal to the I.H, but need to confirm that $i_h \leq i'_h$ and for all b, $\eta(b) \leq \eta(b')$. To do this consider the path p'' up to the configuration immediately after this iteration of the loop is expanded. Then we have that $\mathbf{Conv}_F(p'') = (\kappa''', \psi''', \eta', \mu''', \psi'''^h, i'_h, N''', T)$, and so by the I.H, $i_h \leq i'_h$ and for all b, $\eta(b) \leq \eta(b')$.

In the case of an *OA*-action $c(A)$, we partition $p'\ \mathbf{C}'$ into $p_1\ \mathbf{C}_1 \xrightarrow{\bar{f}(\overrightarrow{A},c)} \mathbf{C}_2\ p_2\ \mathbf{C}_3 \xrightarrow{c(A)} \mathbf{C}'$. Let $\mathbf{Conv}_F(p_1\ \mathbf{C}_1 \xrightarrow{\bar{f}(\overrightarrow{A},c)} \mathbf{C}_2) = (\kappa_2, \psi_2, \eta_2, \mu_2, \psi_2^h, i_{h2}, \emptyset, T_2)$, then by the I.H applied to $p_1\ \mathbf{C}_1 \xrightarrow{\bar{f}(\overrightarrow{A},c)} \mathbf{C}_2$ and $p_1\ \mathbf{C}_1 \xrightarrow{\bar{f}(\overrightarrow{A},c)} \mathbf{C}_2\ p_2\ \mathbf{C}_3$ (and Lemma 114) we have that $\mathbf{Conv}_F(p_1\ \mathbf{C}_1 \xrightarrow{\bar{f}(\overrightarrow{A},c)} \mathbf{C}_2\ p_2\ \mathbf{C}_3) = (\kappa_3, \psi_2, \eta_2, \mu_3, \psi_2^h, i_{h2}, \emptyset, T_2)$. From this is follows that if $\mathbf{Conv}_F(p_1\ \mathbf{C}_1) = (\kappa_1, \psi_1, \eta_1, \mu_1, \psi_1^h, i_{h1}, N_1, T_1)$, then $\psi' = \psi_1 \cdot \psi''$, $\psi'^h = \psi_1^h \cdot \psi''^h$ and for all b $\eta_1(b) \leq \eta'(b)$ and $i_{h1} \leq i'_h$. Thus the result holds by applying the I.H to p and $p_1\ \mathbf{C}_1$.

Similarly, if $\mathbf{C}'$ is passive, then $\mathbf{a}$ must be an *PA*-action $\bar{c}(A)$. We can partition p', $\mathbf{C}'$ into $p_1\ \mathbf{C}_1 \xrightarrow{f(\overrightarrow{A},c)} \mathbf{C}_2\ p_2\ \mathbf{C}_3 \xrightarrow{\bar{c}(A)} \mathbf{C}'$. Let $\mathbf{Conv}_F(p_1\ \mathbf{C}_1) = (\kappa_1, \psi_1, \eta_1, \mu_1, \psi_1^h, i_{h1}, \emptyset, T_1)$. Then by the I.H, $\eta_1 = \eta$, $i_{h1} = i_h$, and so $\psi_1 = \psi$, $\psi_1^h = \psi^h$. If $\mathbf{Conv}_F(p_1\ \mathbf{C}_1 \xrightarrow{f(\overrightarrow{A},c)} \mathbf{C}_2) = (\kappa_2, \psi_2, \eta_2, \mu_2, \psi_2^h, i_{h2}, N_1, T_1)$, then $\psi_{2<\eta} = \psi$ and $\psi^h_{2<i_h} = \psi^h$ and $\mu_2(c) = (i_h, \eta)$. If $\mathbf{Conv}_F(p_1\ \mathbf{C}_1 \xrightarrow{f(\overrightarrow{A},c)} \mathbf{C}_2\ p_2\ \mathbf{C}_3) = (\kappa_3, \psi_3, \eta_3, \mu_3, \psi_3^h, i_{h3}, N_3, T_1)$, then by Lemma 113, we have that $\mu_3(c) = \mu_2(c) = (i_h, \eta)$, $i'_h = i_h$ and $\eta' = \eta$. By the I.H applied to $p_1\ \mathbf{C}_1 \xrightarrow{f(\overrightarrow{A},c)} \mathbf{C}_2$ and $p_1\ \mathbf{C}_1 \xrightarrow{f(\overrightarrow{A},c)} \mathbf{C}_2\ p_2\ \mathbf{C}_3$, we obtain that $\psi' = \psi_{3<\eta} = \psi_{2<\eta} = \psi$ and $\psi'^h = \psi^h_{3<i_h} = \psi_{2<i_h} = \psi$. □

Using this, we can prove the following.

Lemma 116. *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, s.t. f is a level 2 name in $\text{Tr}(p)$ with originator g. If $\mathbf{Conv}_F(p\ \mathbf{C}) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$, then $\mu(g) = \mu(f)$.*

We can prove by a straight forward induction, the above Lemmata, Lemmata 43 and 44, the following property.

Lemma 117 (Lemma 53)**.** *Let (p, F) be a configuration of $\mathcal{L}^{\Delta}_{\text{Path}}$, and $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$. Then:*

- *if p ends in a active state $(\langle N, c, \gamma, \phi, h, H\rangle, S)$, then for $f \in \nu(N)$, we have $(\psi \circ \kappa)(f) = f$, $(\psi \circ \kappa)(c) = c$, and for location ℓ in N, $(\psi^h \circ \kappa)(\ell) = \ell$;*
- *if p ends in a passive state $(\langle \gamma, \phi, h, H, Fn\rangle, S)$, then for $f \in Fn$, we have $(\psi \circ \kappa)(f) = f$.*

Proof. This proof is done by an induction with a slightly stronger property. Define by mutual induction the sets

$$\begin{array}{rcl}\nu(N) & \subseteq & \mathrm{NamesP}_{\mathrm{H},\gamma}(N)\\ S & \subseteq & \mathrm{NamesO}_{\mathrm{H},\gamma}(\mathrm{S})\\ \mathrm{NamesP}_{\mathrm{H},\gamma}(N) & = & \nu(N) \cup \bigcup_{f\in\nu(N)} \mathrm{NamesO}_{\mathrm{H},\gamma}(\mathrm{H(f)})\\ \mathrm{NamesO}_{\mathrm{H},\gamma}(\mathrm{S}) & = & S \cup \bigcup_{f\in S} \mathrm{NamesP}_{\mathrm{H},\gamma}(\gamma(f))\end{array}$$

We can then also define

$$\begin{array}{rcl}\mathrm{Loc(M)} & = & \{\ell \mid \ell \text{ appears in } M\}\\ \mathrm{Loc}_\gamma(\mathrm{S}) & = & \bigcup_{f\in S, f\in\mathrm{dom}(\gamma)} \mathrm{Loc}(\gamma(\mathrm{f}))\end{array}$$

Our stronger inductive hypothesis is that:

1. if p ends in a active state $(\langle N, c, \gamma, \phi, h, H\rangle, S)$, then
   (a) for $f \in \mathrm{NamesP}_{\mathrm{H},\gamma}(N)$, $(\psi \circ \kappa)(f) = f$;
   (b) $(\psi \circ \kappa)(c) = c$;
   (c) for $f \in \mathrm{NamesO}_{\mathrm{H},\gamma}(\mathrm{H(c)})$, $(\psi \circ \kappa)(f) = f$;
   (d) for $\ell \in \mathrm{Loc(N)}$, $(\psi^h \circ \kappa)(\ell) = \ell$;
   (e) for $\ell \in \mathrm{Loc}_\gamma(\mathrm{NamesP}_\gamma(\mathrm{N}) \cup \mathrm{NamesO}_{\mathrm{H},\gamma}(\mathrm{H(c)}))$, $(\psi^h \circ \kappa)(\ell) = \ell$;
2. if p ends in a passive state $(\langle \gamma, \phi, h, H, Fn\rangle, S)$,
   (a) for $f \in \mathrm{NamesO}_{\mathrm{H},\gamma}(Fn)$, $(\psi \circ \kappa)(f) = f$;
   (b) for $\ell \in \mathrm{Loc}_\gamma(\mathrm{NamesO}_{\mathrm{H},\gamma}(Fn))$, $(\psi^h \circ \kappa)(\ell) = \ell$;
   (c) if $T = (c, (K, c')) : T'$, then
      - $(\psi \circ \kappa)(c') = c'$;
      - for $f \in \mathrm{NamesP}_{\mathrm{H},\gamma}(K)$, $(\psi \circ \kappa)(f) = f$;
      - for $\ell \in \mathrm{Loc(K)} \cup \mathrm{Loc}_\gamma(\mathrm{NamesP}_{\mathrm{H},\gamma}(\mathrm{K}))$, $(\psi^h \circ \kappa)(\ell) = \ell$;
   (d) if $T = (c, (K, c'), (i'_h, \eta', \psi'^h, \psi)) : T'$, then
      - $((\psi \cdot \psi') \circ \kappa)(c') = c'$;
      - for $f \in \mathrm{NamesP}_{\mathrm{H},\gamma}(K)$, $((\psi \cdot \psi') \circ \kappa)(f) = f$;
      - for $\ell \in \mathrm{Loc(K)} \cup \mathrm{Loc}_\gamma(\mathrm{NamesP}_{\mathrm{H},\gamma}(\mathrm{K}))$, $((\psi^h \cdot \psi'^h) \circ \kappa)(\ell) = \ell$;

The induction proceeds by considering the last action in p. We show some the two interesting cases.

- $p = p' \xrightarrow{\bar{f}(\overrightarrow{A},c)} (\langle \gamma, \phi, h, H, Fn\rangle, S) = \mathbf{C}$ with f a level 2 name. Let g be the originator of f.

  Let g be introduced in an action $f'(\overrightarrow{A'}, c')$, and so partition p' into $p_1\ \mathbf{C}_1 \xrightarrow{f'(\overrightarrow{A'},c')} p_2\ \mathbf{C}_2$, where $\mathbf{C}_1 = (\langle \gamma_1, \phi_1, h_1, H_1, Fn_1\rangle, S_1)$ and $\mathbf{C}_2 = (\langle K[\text{force } f\overrightarrow{A}], c_2, \gamma_2, \phi_2, h_2, H_2\rangle, S_2)$. Then by Lemma 44 and Lemma 16, $H_2(f) = Fn = Fn_1$, and $\gamma = \gamma_2, H = H_2$, (which are conservative extensions of γ_1,H_1) and $S = (c, (K, c_2)) : S_2$.

  Let $\mathbf{Conv}_F(p_1\ \mathbf{C}_1) = (\kappa_1, \psi_1, \eta_1, \mu_1, \psi_1^h, i_{h1}, N_1, T_1)$ and $\mathbf{Conv}_F(p') = (\kappa_2, \psi_2, \eta_2, \mu_2, \psi_2^h, i_{h2}, N_2, T_2)$. Then $N_2 = K'[\text{force } f\overrightarrow{A}]$ where $K = \tilde{K'}$. By Lemma 116, we obtain that $(i_h, \eta) = \mu_2(f) = (i_{h1}, \eta_1)$, so $\psi = \psi_{2<\eta}$, $\psi^h = \psi^h_{2<i_h}$ and $T = (c, (K', c_2), (i_{h2}, \eta_2, \psi^h_{2\geq i_h}, \psi_{2\geq\eta})) : T_2$.

  Observe that if $f'' \in \mathrm{NamesO}_{\mathrm{H},\gamma}(Fn)$, then $f'' \in \mathrm{NamesO}_{\mathrm{H}_2,\gamma_2}(Fn_2)$, and by Lemma 113 $\kappa f'' = \kappa_1 f'' = b^i$. By the I.H applied to $p_1\ \mathbf{C}_1$, it must be the case that $(\psi_1 \circ \kappa_1)(f'') = f''$, so by Lemma 111, we obtain that $i < \eta(b)$. So if $(\psi_2 \circ \kappa_2)(f'') = f''$, then $(\psi \circ \kappa)(f'') = (\psi_{2<\eta} \circ \kappa_2)(f'') = f''$. Similarly, for $\ell \in \mathrm{Loc}_\gamma(\mathrm{NamesO}_{\mathrm{H},\gamma}(Fn))$, $\kappa(\ell) < i_h$, and so if $(\psi_2^h \circ \kappa_2)(\ell) = \ell$, $(\psi^h \circ \kappa)(\ell) = \ell$.

  The result then hold by applying the inductive hypothesis to p', and for the stack, the observation that $\psi_{2<\eta} \cdot \psi_{2\geq\eta} = \psi_2$ and $\psi^h_{2<i_h} \cdot \psi^h_{2\geq i_h} = \psi^h_2$.

- $p = p' \xrightarrow{\bar{c}(A)} (\langle \gamma, \phi, h, H, Fn \rangle, S) = \mathbf{C}$ with $c \neq c'$. Due to the bracketing conditions, there exists an action $f(\overrightarrow{A}, c)$ that this action answers, so let $p' = p_1\ \mathbf{C}' \xrightarrow{f(\overrightarrow{A},c)} p_2$ where $\mathbf{C}' = (\langle \gamma', \phi', h', H', Fn' \rangle, S)$. It follows by Lemma 43 that $Fn' = Fn$, and by the construction of $\mathcal{L}_{\text{CBPV}}$ that γ, H are conservative extensions of γ', H'. By Lemma 114, $T = T'$.

  Let $\mathbf{Conv}_F(p_1\ \mathbf{C}') = (\kappa', \psi', \eta', \mu', \psi'^h, i'_h, N', T')$. Then by Lemma 113 κ is an extension of κ'. By Lemma 115, $\psi' = \psi_{<\eta'}$ and $\psi'^h = \psi^h_{<i'_h}$. The result then follows from applying the I.H. to $p_1\ \mathbf{C}'$ and using the observations that if $\psi'(\kappa'(d)) = d$ then $\psi(\kappa(d)) = \psi_{<\eta'}(\kappa(d)) = d$, and if $\psi'^h(\kappa'(\ell)) = \ell$ then $\psi^h(\kappa(\ell)) = \psi^h_{<i_h}(\kappa(\ell)) = \ell$.

□

We define the following function mapping extended stacks to stacks with the same structure as $\mathcal{L}^{\Delta}_{\text{PTR}}$.

$$\begin{array}{rcl}
\mathbf{Stack}_{\gamma,h,H,\mu}(\bot) & \triangleq & \bot \\
\mathbf{Stack}_{\gamma,h,H,\mu}((c,(K,c')) : T) & \triangleq & (c,(K,c')) : \mathbf{Stack}_{\gamma,h,H,\mu}(T) \\
\mathbf{Stack}_{\gamma,h,H,\mu}((c,(K,c'),(i_h,\eta,\psi^h,\psi)) : T) & \triangleq & (c,(K,c'),(i_h,\eta,\gamma\circ\psi,h\circ\psi^h,H\circ\psi,\mu\circ\psi)) : \mathbf{Stack}_{\gamma,h,H,\mu}(T)
\end{array}$$

Finally, we can define a functional bisimulation Θ from $\mathcal{L}_{\text{Path}}$ to $\mathcal{L}^{\Delta_{\Gamma,\sigma}}_{\text{PTR}}$:

$$\begin{array}{rcl}
\Theta((p\ \mathbf{C}, F)) & \triangleq & (\langle \kappa(N), \kappa(c), \kappa\circ\gamma\circ\psi, h\circ\psi^h, \kappa\circ H\circ\psi, i_h, \eta, \mu\circ\psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T))), \\
& & \text{if } \mathbf{C} = (\langle N', c, \gamma, \phi, h, H\rangle, S) \\
\Theta((p\ \mathbf{C}, F)) & \triangleq & (\langle \kappa\circ\gamma\circ\psi, h\circ\psi^h, \kappa\circ H\circ\psi, \kappa(Fn), i_h, \eta, \mu\circ\psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T))), \\
& & \text{if } \mathbf{C} = (\langle \gamma, \phi, h, H, Fn\rangle, S)
\end{array}$$

where $(\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T) = \mathbf{Conv}_F(p\ \mathbf{C})$ and if $F = \emptyset$ then $l = 0$, otherwise $l = 1$.

Lemma 118 (Lemma 54)**.** *Θ is a functional bisimulation between $\mathcal{L}^{\Delta}_{\text{Path}}$ and $\mathcal{L}^{\Delta}_{\text{PTR}}$*

Proof. It is easy to check that $\Theta(\mathsf{C}^{\rho,c_0}_M) = \mathsf{C}^{\text{PTR},\Delta_{\Gamma,\sigma}}_M$. We then need to check both directions of the bisimulation. Checking one direction (that $(p,F) \xrightarrow{\mathbf{a}} (p',F')$ implies $\Theta((p,F)) \xrightarrow{\mathbf{a}} \Theta(p',F))$ is straight froward given the construction. The other direction is more challenging, and given below.

Let $\mathbf{D} = \Theta((p,F))$ and $\mathbf{D}'$ be s.t $\mathbf{D} \xrightarrow{\mathbf{a}} \mathbf{D}'$, then we wish to show there exist (p',F') s.t. $(p,F) \xrightarrow{\mathbf{a}} (p',F')$ and $\mathbf{D}' = \Theta((p',F'))$. Let $\mathbf{Conv}_F(p) = (\kappa, \psi, \eta, \mu, \psi^h, i_h, N, T)$.

We proceed by cases. First we consider cases where $\mathbf{D}$ is active. Then $\mathbf{C} = (\langle N, c, \gamma, \phi, h, H\rangle, S)$, $p = p'\ \mathbf{C}$ (by Lemma 110) and $\mathbf{D} = (\langle \kappa(N), \kappa(c), \kappa\circ\gamma\circ\psi, h\circ\psi^h, \kappa\circ H\circ\psi, i_h, \eta, \mu\circ\psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$ where $l = 0$ if $F = \emptyset$ and $l = 1$ otherwise.

- $\mathbf{D} \xrightarrow{\tau} \mathbf{D}'$ where $\kappa(N) = K[\text{ref}\, V]$. Then there exists K' s.t $K = \kappa(K')$ and $N = K'[\text{ref}\, V]$. Thus

  $$\mathbf{D}' = (\langle K[i_h], \kappa(c), \kappa\circ\gamma\circ\psi, (h\circ\psi^h)\cdot[i_h \mapsto V], \kappa\circ H\circ\psi, i_h+1, \eta, \mu\circ\psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

  Now, for some ℓ, $\mathbf{C} \xrightarrow{\tau} (\langle \tilde{K}'[\ell], c, \gamma, \phi, h\cdot[\ell \mapsto V], H\rangle, S) = \mathbf{C}'$. Then $\mathbf{Conv}_F(p \xrightarrow{\tau} \mathbf{C}') = (\kappa\cdot[\ell \mapsto i_h+1], \psi, \eta, \mu, \psi^h\cdot[i_h+1 \mapsto \ell], i_h+1, K'[\ell], T)$. As $(h\circ\psi^h)\cdot[i_h \mapsto V] = (h\cdot[\ell \mapsto V])\circ(\psi^h\cdot[i_h+1 \mapsto \ell])$ we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\tau} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\tau} \mathbf{D}'$ where $\kappa(N) = K[\text{end}(i'_h, \eta').\text{while}\, N_1\, \text{do}\, N_2]$. Then these exists K', N'_1, N'_2 s.t. $K = \kappa(K')$, $N'_i = \kappa(N'_i)$ and $N = K'[\text{end}(i'_h, \eta').\text{while}\, N'_1\, \text{do}\, N'_2]$. Thus

  $$\begin{array}{l}
  \mathbf{D}' = (\langle K[N_1\ \text{to}\ x.\text{case}\ x\ \text{of return}\ (), (N_2\ \text{to}\ y.\text{end}(i'_h, \eta').\text{while}\, N_1\, \text{do}\, N_2)_{j>0}], \\
  \kappa(c), (\kappa\circ\gamma\circ\psi)_{<\eta'}, (h\circ\psi^h)_{<i_h}, (\kappa\circ H\circ\psi)_{<\psi}, i'_h, \eta', (\mu\circ\psi)_{<\eta'}, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))
  \end{array}$$

Now, $\mathbf{C} \xrightarrow{\tau} (\langle \tilde{K'}[N_1 \text{ to } x.\text{case } x \text{ of return } (), (N_2' \text{ to } y.\text{while } N_1' \text{ do } N_2')_{j>0}], c, \gamma, \phi, h, H\rangle, S) = \mathbf{C}'$, so

$$\mathbf{Conv}_F(p \xrightarrow{\tau} \mathbf{C}') = (\kappa, \psi_{<\eta'}, \eta', \mu, \psi^h_{<i'_h}, i'_h, K'[N_1' \text{ to } x.\text{case } x \text{ of return } (), \\ (N_2' \text{ to } y.\text{end}(i'_h, \eta').\text{while } N_1' \text{ do } N_2')_{j>0}], S)$$

As $(\chi \circ \psi)_{<\eta'} = \chi \circ (\psi_{<\eta'})$ and $(h \circ \psi^h)_{<i'_h} = h \circ (\psi^h_{<i_h})$, we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\tau} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\tau} \mathbf{D}'$ where $\kappa(N) = K[\text{while } N_1 \text{ do } N_2]$. Then these exists K', N_1', N_2' s.t. $K = \kappa(K')$, N_i and $N = K'[\text{while } N_1' \text{ do } N_2']$. Thus

$$\mathbf{D}' = \langle K[N_1 \text{ to } x.\text{case } x \text{ of return } (), (N_2 \text{ to } y.\text{end}(i_h, \eta).\text{while } N_1 \text{ do } N_2)_{j>0}], \\ \kappa(c), \kappa \circ \gamma \circ \psi, h \circ \psi^h, \kappa \circ H \circ \psi, i_h, \eta, \mu \circ \psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

Now, $\mathbf{C} \xrightarrow{\tau} (\langle \tilde{K'}[N_1 \text{ to } x.\text{case } x \text{ of return } (), (N_2' \text{ to } y.\text{while } N_1' \text{ do } N_2')_{j>0}], c, \gamma, \phi, h, H\rangle, S) = \mathbf{C}'$, so

$$\mathbf{Conv}_F(p \xrightarrow{\tau} \mathbf{C}') = (\kappa, \psi, \eta, \mu, \psi^h, i_h, K'[N_1' \text{ to } x.\text{case } x \text{ of return } (), \\ (N_2' \text{ to } y.\text{end}(i_h, \eta).\text{while } N_1' \text{ do } N_2')_{j>0}], S)$$

so we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\tau} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\tau} \mathbf{D}'$ where $\kappa(N) = K[!j]$. Then there exist K', ℓ s.t. $K = \kappa(K')$ $j = \kappa(\ell)$ and $N = K'[!\ell]$. Then

$$\mathbf{D}' = (\langle K[(h \circ \psi^h)(j)], \kappa(c), \kappa \circ \gamma \circ \psi, h \circ \psi^h, \kappa \circ H \circ \psi, i_h, \eta, \mu \circ \psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

Now $\mathbf{C} \xrightarrow{\tau} (\langle \tilde{K'}[h(\ell)], c, \gamma, \phi, h, H\rangle, S) = \mathbf{C}'$, so $\mathbf{Conv}_F(p \xrightarrow{\tau} \mathbf{C}') = (\kappa, \psi, \eta, \mu, \psi^h, i_h, K'[h(\ell)], T)$. Now, by Lemma 117, we have that $h(\ell) = (h \circ \psi^h)(\kappa(\ell))$, so we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\tau} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\tau} \mathbf{D}'$ where $\kappa(N) = K[j := V]$. Then there exist K', ℓ s.t. $K = \kappa(K')$ $j = \kappa(\ell)$ and $N = K'[\ell := V]$. Then

$$\mathbf{D}' = (\langle K[\text{return } ()], \kappa(c), \kappa \circ \gamma \circ \psi, (h \circ \psi^h)[j \mapsto V], \kappa \circ H \circ \psi, i_h, \eta, \mu \circ \psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

Now $\mathbf{C} \xrightarrow{\tau} (\langle \tilde{K'}[\text{return } ()], c, \gamma, \phi, h[\ell \mapsto V], H\rangle, S) = \mathbf{C}'$, so $\mathbf{Conv}_F(p \xrightarrow{\tau} \mathbf{C}') = (\kappa, \psi, \eta, \mu, \psi^h, i_h, K'[\text{return } ()], T)$. Now, by Lemma 117, we have that $h[\ell \mapsto V] = (h \circ \psi^h)[\kappa(\ell) \mapsto V]$, so we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\tau} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\tau} \mathbf{D}'$ otherwise. Then $\mathbf{D}' = (\langle N_1, \kappa(c), \kappa \circ \gamma \circ \psi, h \circ \psi^h, \kappa \circ H \circ \psi, i_h, \eta, \mu \circ \psi, l\rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$ where $(\kappa(N), h \circ \psi^h, i_h, \eta) \to_e (N_1, h \circ \psi^h, i_h, \eta)$. This reduction used neither names or locations, it follows that there exists N' s.t. $N_1 = \kappa(N')$ and $(N, h) \to_m (N', h)$, and so $(\tilde{N}, h) \to (\tilde{N'}, h)$. Then $\mathbf{C} \xrightarrow{\tau} (\langle \tilde{N'}, c, \gamma, \phi, h, H\rangle, S) = \mathbf{C}'$, so we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\tau} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\bar{f}(\vec{V}, b)} \mathbf{D}'$ where f is a level 2 name. Then $\kappa(N) = K[\text{force } f^i \vec{V}]$ and $b = \text{Suc}_{\mathrm{C}}(f)$. Then there exist K', f' s.t. $K = \kappa(K')$, $f^i = \kappa(f')$ and $N = K'[\text{force } f' \vec{V}]$. By Lemma 117, $(\mu \circ \psi)(\kappa(f')) = \mu(f') = (i'_h, \eta')$. Then

$$\mathbf{D}' = (\langle (\kappa \circ \gamma \circ \psi)_{<\eta'}, (h \circ \psi^h)_{<i'_h}, (\kappa \circ H \circ \psi)_{<\eta'}, (\kappa \circ H \circ \psi)(f'), i'_h, \eta', (\mu \circ \psi)_{<\eta'}, l\rangle, \\ (b^0, (K, \kappa(c)), P) : \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

where $P = (i_h, \eta, (\kappa \circ \gamma \circ \psi)_{\geq \eta'}, (h \circ \psi^h)_{\geq i'_h}, (\kappa \circ H \circ \psi)_{\geq \eta'}, (\mu \circ \psi)_{\geq \eta'})$.

It follows that for fresh continuation name c', $\mathbf{C} \xrightarrow{\bar{f}'(\overrightarrow{V},c')} (\langle \gamma, \phi \uplus \{c'\}, h, H, H(f')\rangle, (c', (\tilde{K}', c)) : S) = \mathbf{C}'$. By Lemma 109, $(p, F) \xrightarrow{\bar{f}(V,b)} (p \xrightarrow{\bar{f}'(\overrightarrow{V},c')} \mathbf{C}', F)$, and $\mathbf{Conv}_F(p \xrightarrow{\bar{f}'(\overrightarrow{V},c')} \mathbf{C}') = (\kappa \cdot [c' \mapsto b], \psi_{<\eta'}, \eta', \mu, \psi^h_{<i'_h}, i'_h, \emptyset, (c', (K', c), (i_h, \eta, \psi^h_{\geq i'_h}, \psi_{\geq \eta'})) : T)$.

By Lemma 117, $(\kappa \circ H \circ \psi)(f^i) = \kappa(H(f'))$, and $(\chi \circ \psi)_{<\eta'} = \chi \circ (\psi_{<\eta'})$ and $(h \circ \psi^h)_{<i'_h} = h \circ (\psi^h_{<i_h})$. Finally, as $(\chi \circ \psi)_{\geq \eta'} = \chi \circ (\psi_{\geq \eta'})$ and $(h \circ \psi^h)_{\geq i'_h} = h \circ (\psi^h_{\geq i_h})$, we have $(b^0, K, \kappa(c)), P) : \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)) = \kappa \cdot [c' \mapsto b^0](\mathbf{Stack}_{\gamma,h,H,\mu}((c', (K', c), (i_h, \eta, \psi^h_{\geq i'_h}, \psi_{\geq \eta'})) : T))$, so we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\bar{f}'(\overrightarrow{V},c')} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\bar{f}(\beta(\overrightarrow{A}),b)} \mathbf{D}'$ where f is not a level 2 name. Then $\kappa(N) = K[\text{force } f^i\ \overrightarrow{V}]$ where $(\overrightarrow{A}, \gamma', \eta') = \mathbf{IVal}^{\Delta}(f, \overrightarrow{V}, \eta)$ and $b = \mathrm{Suc}_{\mathrm{C}}(f)$. Then there exist $K', \overrightarrow{V'}, f'$ s.t. $K = \kappa(K'), \overrightarrow{V} = \kappa(\overrightarrow{V'}), f^i = \kappa(f')$ and $N = K'[\text{force } f'\overrightarrow{V'}]$. Then

$$\mathbf{D}' = (\langle (\kappa \circ \gamma \circ \psi) \cdot \gamma', h \circ \psi^h, \kappa \circ H \circ \psi, (\kappa \circ H \circ \psi)(f^i) \uplus \nu(\overrightarrow{A}), i_h, \eta', \mu \circ \psi, l\rangle, \\ (b^0, (K, \kappa(c))) : \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

There exist abstract value sequence $\overrightarrow{A'}$ and γ'' s.t. $(\overrightarrow{A'}, \gamma'') \in \mathbf{AVal}(\overrightarrow{V'})$, so $\mathbf{C} \xrightarrow{\bar{f}'(\overrightarrow{A'},c')} (\langle \gamma \cdot \gamma'', \phi \uplus \nu(\overrightarrow{A'}) \uplus \{c'\}, h, H, H(f') \uplus \nu(\overrightarrow{A'})\rangle, (c', (\tilde{K}', c)) : S) = \mathbf{C}'$. Following Lemmata 109 and 104, $(p, F) \xrightarrow{\bar{f}(\beta(\overrightarrow{A}),b)} (p \xrightarrow{\bar{f}'(\overrightarrow{A'},c')} \mathbf{C}', F)$ and $\gamma' = \kappa \circ \gamma'' \circ [\mathbf{Match}(A', A, g) \mapsto g]_{g \in \nu(\overrightarrow{A'})}$. As $(\overrightarrow{A}, \eta') \in \mathbf{IValSeq}^{\Delta}(f, \eta)$, we have $\mathbf{Conv}_F(p \xrightarrow{\bar{f}'(\overrightarrow{A'},c')} \mathbf{C}') = (\kappa \cdot \kappa', \psi \cdot \psi', \eta', \mu, \psi^h, i_h, \emptyset, (c', (K', c)) : T)$ where $\kappa' = [c' \mapsto b^0] \cdot [f \mapsto \mathbf{Match}(A', A, f)]_{f \in \nu(A')}$ and $\psi' = [\mathbf{Match}(A', A, f) \mapsto f]_{f \in \nu(A')}$. By Lemma 117, $(\kappa \circ H \circ \psi)(f^i) \uplus \nu(\overrightarrow{A}) = \kappa(H(f')) \uplus \kappa'(\nu(\overrightarrow{A'})) = (\kappa \cdot \kappa')(H(f') \uplus \nu(\overrightarrow{A'}))$. Further, we have that $(\kappa \circ \gamma \circ \psi) \cdot \gamma' = (\kappa \circ \gamma \circ \psi) \cdot (\kappa \circ \gamma'' \circ \psi') = \kappa \circ (\gamma \cdot \gamma'') \circ (\psi \cdot \psi') = (\kappa \cdot \kappa') \circ (\gamma \cdot \gamma'') \circ (\psi \cdot \psi')$, $\kappa \circ H \circ \psi = (\kappa \cdot \kappa') \circ H \circ (\psi \cdot \psi')$, and

$$\begin{array}{rcl} (\kappa \cdot \kappa')(\mathbf{Stack}_{\gamma \cdot \gamma'', h, H, \mu}((c', (K', c)) : T))) & = & (b^0, (K, \kappa(c)) : (\kappa \cdot \kappa')(\mathbf{Stack}_{\gamma \cdot \gamma'', h, H, \mu}(T))) \\ & = & (b^0, (K, \kappa(c)) : \kappa(\mathbf{Stack}_{\gamma, h, H, \mu}(T))) \end{array}$$

so we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\bar{f}'(\overrightarrow{A'},c')} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\bar{c_0}(\beta(A))} \mathbf{D}'$. Then $\kappa(N) = \text{return } V$ where $(A, \gamma', \eta') = \mathbf{IVal}^{\Delta}_{\sigma}(c_0, V, \eta)$, so it follows there exists V' s.t $V = \kappa(V')$ and $N = \text{return } V$. Then

$$\mathbf{D}' = (\langle (\kappa \circ \gamma \circ \psi) \cdot \gamma', h \circ \psi^h, \kappa \circ H \circ \psi, (\kappa \circ H \circ \psi)(c^0_0) \cup \nu(A), i_h, \eta', \mu \circ \psi, l\rangle, \\ \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

There exits abstract value A' and mapping γ'' s.t. $(A', \gamma'') \in \mathbf{AVal}_{\sigma}(V)$, so $\mathbf{C} \xrightarrow{\bar{c_0}(A')} (\langle \gamma \cdot \gamma'', \phi \uplus \nu(A'), h, H, H(c_0) \uplus \nu(A')\rangle, S) = \mathbf{C}'$. Following Lemma 109 and 103, $(p, F) \xrightarrow{\bar{c_0}(\beta(A))} (p \xrightarrow{\bar{c_0}(A')} \mathbf{C}', F)$ and $\gamma' = \kappa \circ \gamma'' \circ [\mathbf{Match}(A', A, f) \mapsto f]_{f \in \nu(A')}$. As $(A, \eta') \in \mathbf{IVals}^{\Delta}_{\sigma}(c_0, \eta)$, we have $\mathbf{Conv}_F(p \xrightarrow{\bar{c_0}(A')} \mathbf{C}') = (\kappa \cdot \kappa', \psi \cdot \psi', \eta', \mu, \psi^h, i_h, \emptyset, T)$ where $\kappa' = [f \mapsto \mathbf{Match}(A', A, f)]_{f \in \nu(A')}$ and $\psi' = [\mathbf{Match}(A', A, f) \mapsto f]_{f \in \nu(A')}$. As $(\kappa \circ H \circ \psi)(c^0_0) \uplus \nu(A) = (\kappa \cdot \kappa')(H(c_0) \uplus \nu(A'))$, $(\kappa \circ \gamma \circ \psi) \cdot \gamma' = (\kappa \circ \gamma \circ \psi) \cdot (\kappa \circ \gamma'' \circ \psi') = \kappa \circ (\gamma \cdot \gamma'') \circ (\psi \cdot \psi') = (\kappa \cdot \kappa') \circ (\gamma \cdot \gamma'') \circ (\psi \cdot \psi')$ and $\kappa \circ H \circ \psi = (\kappa \cdot \kappa') \circ H \circ (\psi \cdot \psi')$, we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\bar{c_0}(A')} \mathbf{C}', F))$.

- $\mathbf{D} \xrightarrow{\bar{b}(V)} \mathbf{D}'$ where $\kappa(c) = b^i$, c is not c_0. Then $N = \kappa(N) = \text{return } V$, and by Lemma 117, $(\mu \circ \psi)(\kappa(c)) = \mu(c) = (i'_h, \eta')$. So

$$\mathbf{D}' = (\langle (\kappa \circ \gamma \circ \psi)_{<\eta'}, (h \circ \psi^h)_{<i'_h}, (\kappa \circ H \circ \psi)_{<\eta'}, (\kappa \circ H \circ \psi)(b^i), i'_h, \eta', (\mu \circ \psi)_{<\eta'}, l \rangle, \\ \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

  Then $\mathbf{C} \xrightarrow{\bar{c}(V)} (\langle \gamma, \phi, h, H, H(c) \rangle, S) = \mathbf{C}'$, so by Lemma 109 $(p, F) \xrightarrow{\bar{b}(V)} (p \xrightarrow{\bar{c}(V)} \mathbf{C}', F)$, and $\mathbf{Conv}_F(p \xrightarrow{\bar{c}(V)} \mathbf{C}') = (\kappa, \psi_{<\eta'}, \eta', \mu, \psi^h_{<i'_h}, i'_h, \emptyset, T)$. As by Lemma 117, $(\kappa \circ H \circ \psi)(b^i) = \kappa(H(c))$, and $(\chi \circ \psi)_{<\eta'} = \chi \circ (\psi_{<\eta'})$ and $(h \circ \psi^h)_{<i'_h} = h \circ (\psi^h_{<i_h})$, we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{\bar{c}(V)} \mathbf{C}', F))$.

We now proceed to the cases where $\mathbf{D}$ is passive. Then $\mathbf{C} = (\langle \gamma, \phi, h, H, Fn \rangle, S)$, $p = p'$ $\mathbf{C}$ and $\mathbf{D} = (\langle \kappa \circ \gamma \circ \psi, h \circ \psi^h, \kappa \circ H \circ \psi, \kappa(Fn), i_h, \eta, \mu \circ \psi, l \rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$ where $l = 0$ if $F = \emptyset$ and $l = 1$ otherwise.

- $\mathbf{D} \xrightarrow{f(\beta(\overrightarrow{B}),c)} \mathbf{D}'$. Then $f^i \in \kappa(Fn)$, $(\overrightarrow{B'}, \eta') \in \mathbf{IValSeq}^{\Delta}(f, \eta)$,, and $c = \mathrm{Suc}_{\mathrm{C}}(f)$. If $l = 1$, then $\overrightarrow{B} = \overrightarrow{B'}$ and $l' = 1$, else $\overrightarrow{B} \in \mathbf{Select}(\overrightarrow{B'})$ and $l' = 1$ if a name is marked in $\overrightarrow{B}$, otherwise $l' = 0$. Let $j = \eta(c)$ and $V = (\gamma \circ \psi)(f^i)$, then

$$\mathbf{D}' = (\langle \text{force } \kappa(V) \overrightarrow{B}, b^j, \kappa \circ \gamma \circ \psi, h \circ \psi^h, (\kappa \circ H \circ \psi) \cdot [\nu(\overrightarrow{B}), c^j \mapsto \kappa(Fn)], i_h, \eta'', \\ (\mu \circ \psi) \cdot [\nu(\overrightarrow{B}), c^j \mapsto (i_h, \eta)], l' \rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T)))$$

  where $\eta'' = \eta'[c \mapsto j + 1]$.
  Now, there exist $f' \in Fn$ s.t. $f^i = \kappa(f')$, so by Lemma 109, $f = \mathbf{Base}^{\Delta}_{\mathrm{Tr}(p)}(f')$, and by Lemma 117 $\gamma(f') = (\gamma \circ \psi)(f^i) = V$. By Lemma 106, there exists $\overrightarrow{A}$ s.t $\mathbf{C} \xrightarrow{f'(\overrightarrow{A},c')} (\langle \text{force } V \overrightarrow{A}, c', \gamma, \phi, h, H \cdot [\nu(\overrightarrow{A}), c' \mapsto Fn] \rangle, S) = \mathbf{C}'$ and $\beta(\overrightarrow{B'}) = \mathbf{Base}^{\Delta}_t(\overrightarrow{A})$ where $t = \mathrm{Tr}(p \xrightarrow{f'(\overrightarrow{A},c')} \mathbf{C}')$. If no name in $\overrightarrow{B}$ is marked, then $F' = F$, otherwise $F' = \{g \in \nu(\overrightarrow{A}) \mid \mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, g) \text{ is marked}\}$. Then $(p, F) \xrightarrow{f(\beta(\overrightarrow{B}),c)} (p \xrightarrow{f'(\overrightarrow{A},c')} \mathbf{C}', F')$ and $\mathbf{Conv}_{F'}(p \xrightarrow{f'(\overrightarrow{A},c)} \mathbf{C}') = (\kappa \cdot \kappa', \psi \cdot \psi', \eta'', \mu \cdot \mu', \psi^h, i_h, \text{force } V \overrightarrow{A}, T)$ where $\kappa' = [c' \mapsto c^j] \cdot [g \mapsto \mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, g)]_{g \in \nu(\overrightarrow{A})}$, $\psi' = [c^j \mapsto c'] \cdot [\mathbf{Match}(\overrightarrow{A}, \overrightarrow{B}, g) \mapsto g]_{g \in \nu(\overrightarrow{A})}$ and $\mu' = [\nu(\overrightarrow{A}), c' \mapsto (i_h, \eta)]$.

  As $(\kappa \cdot \kappa')(\text{force } V \overrightarrow{A}) = \text{force } (\kappa \cdot \kappa')(V)(\kappa \cdot \kappa')(\overrightarrow{A}) = \text{force } \kappa(V) \overrightarrow{B}$, $\kappa \circ \gamma \circ \psi = (\kappa \cdot \kappa') \circ \gamma \circ (\psi \cdot \psi')$, $(\kappa \circ H \circ \psi) \cdot [\nu(\overrightarrow{B}), c^j \mapsto \kappa(Fn)] = \kappa \circ ((H \circ \psi) \cdot ([\nu(\overrightarrow{A}), c' \mapsto Fn] \circ \psi')) = (\kappa \cdot \kappa') \circ (H \cdot [\nu(\overrightarrow{A}), c' \mapsto Fn]) \circ (\psi \cdot \psi')$, $(\mu \circ \psi) \cdot [\nu(\overrightarrow{B}), c^j \mapsto (i_h, \eta)] = (\mu \circ \psi) \cdot (\mu' \cdot \psi') = (\mu \cdot \mu') \circ (\psi \cdot \psi')$, $l' = 0$ if F is empty and $l' = 1$ otherwise, we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{f'(\overrightarrow{A},c')} \mathbf{C}', F'))$.

- $\mathbf{D} \xrightarrow{c(\beta(B))} \mathbf{D}'$ where $\kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T))) = (c^0, (K, b^j)) : \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T')))$ and $T = (c', (K', c'')) : T'$. Then $c^0 : \sigma = \kappa(c')$, $K = \kappa(K')$, $b^j = \kappa(c'')$, and $(B', \eta') \in \mathbf{IVals}^{\Delta}_{\sigma}(c, \eta)$. If $l = 1$, then $B = B'$ and $l' = 1$, else $B \in \mathbf{Select}(B')$ and $l' = 1$ if a name is marked in B, otherwise $l' = 0$. Then

$$\mathbf{D}' = (\langle K[\text{return } B], b^j, \kappa \circ \gamma \circ \psi, h \circ \psi^h, (\kappa \circ H \circ \psi) \cdot [\nu(B) \mapsto \kappa(Fn)], i_h, \eta', \mu \circ \psi, l' \rangle, \\ \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T')))$$

  By Lemma 109, $c = \mathbf{Base}^{\Delta}_{\mathrm{Tr}(p)}(c')$, by Lemma 110, $S = (c', (\tilde{K}', c'')) : S'$, and by Lemma 105 there exists A s.t. $\mathbf{C} \xrightarrow{c'(A)} (\langle \tilde{K}'[\text{return } A], c'', \gamma, \phi, h, H \cdot [\nu(A) \mapsto Fn] \rangle, S') = \mathbf{C}'$ and

$\beta(B') = \mathbf{Base}^{\Delta}_{t}(A)$ where $t = \mathrm{Tr}(p \xrightarrow{c'(A)} \mathbf{C}')$. If no name in B is marked, then $F' = F$, otherwise $F' = \{g \in \nu(A) \mid \mathbf{Match}(A, B, g) \text{ is marked}\}$. Then $(p, F) \xrightarrow{c(\beta(B))} (p \xrightarrow{c'(A)} \mathbf{C}', F')$ and $\mathbf{Conv}_{F'}(p \xrightarrow{c'(A)} \mathbf{C}') = (\kappa \cdot \kappa', \psi \cdot \psi', \eta', \mu, \psi^h, i_h, \text{force } V \overrightarrow{A}, T)$ where $\kappa' = [g \mapsto \mathbf{Match}(A, B, g)]_{g \in \nu(A)}$, and $\psi' = [\mathbf{Match}(A, B, g) \mapsto g]_{g \in \nu(A)}$.

As $(\kappa \cdot \kappa')(K'[\text{return } A]) = K[\text{return } \kappa'(A)] = K[\text{return } B]$, $\kappa \circ \gamma \circ \psi = (\kappa \cdot \kappa') \circ \gamma \circ (\psi \cdot \psi')$, $(\kappa \circ H \circ \psi) \cdot [\nu(B) \mapsto \kappa(Fn)] = \kappa \circ ((H \circ \psi) \cdot ([\nu(A) \mapsto Fn] \circ \psi')) = (\kappa \cdot \kappa') \circ (H \cdot [\nu(A) \mapsto Fn]) \circ (\psi \cdot \psi')$, $\mu \circ \psi = \mu \circ (\psi \cdot \psi')$, $l' = 0$ if F is empty and $l' = 1$ otherwise, we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{c'(A)} \mathbf{C}', F'))$.

- $\mathbf{D} \xrightarrow{c(\beta(B))} \mathbf{D}'$ where $\kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T))) = (c^0, (K, b^j), (i'_h, \eta', h \circ \psi'^h, \kappa \circ \gamma \circ \psi', \kappa \circ H \circ \psi', \mu \circ \psi')) : \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T')))$ and $T = (c', (K', c''), (i'_h, \eta', \psi'^h, \psi')) : T'$. Then $c^0 : \sigma = \kappa(c'), K = \kappa(K'), b^j = \kappa(c'')$, and $(B', \eta'') \in \mathbf{IVals}^{\Delta}_{\sigma}(c, \eta')$. If $l = 1$, then $B = B'$ and $l' = 1$, else $B \in \mathbf{Select}(B')$ and $l' = 1$ if a name is marked in B, otherwise $l' = 0$. Then

$$\mathbf{D}' = (\langle K[\text{return } B], b^j, (\kappa \circ \gamma \circ \psi) \cdot (\kappa \circ \gamma \circ \psi'), (h \circ \psi^h) \cdot (h \circ \psi'^h), (\kappa \circ H \circ \psi) \cdot (\kappa \circ H \circ \psi') \cdot [\nu(B) \mapsto \kappa(Fn)], i_h, \eta', (\mu \circ \psi) \cdot (\mu \circ \psi') \cdot [\nu(B) \mapsto (i_h, \eta)], l' \rangle, \kappa(\mathbf{Stack}_{\gamma,h,H,\mu}(T')))$$

By Lemma 109, $c = \mathbf{Base}^{\Delta}_{\mathrm{Tr}(p)}(c')$, by Lemma 110, $S = (c', (\tilde{K}', c'')) : S'$, and by Lemma 105 there exists A s.t. $\mathbf{C} \xrightarrow{c'(A)} (\langle \tilde{K}'[\text{return } A], c'', \gamma, \phi, h, H \cdot [\nu(A) \mapsto Fn] \rangle, S') = \mathbf{C}'$ and $\beta(B') = \mathbf{Base}^{\Delta}_{t}(A)$ where $t = \mathrm{Tr}(p \xrightarrow{c'(A)} \mathbf{C}')$. If no name in B is marked, then $F' = F$, otherwise $F' = \{g \in \nu(A) \mid \mathbf{Match}(A, B, g) \text{ is marked}\}$. Then $(p, F) \xrightarrow{c(\beta(B))} (p \xrightarrow{c'(A)} \mathbf{C}', F')$ and $\mathbf{Conv}_{F'}(p \xrightarrow{c'(A)} \mathbf{C}') = (\kappa \cdot \kappa', \psi \cdot \psi' \cdot \psi'', \eta'', \mu \cdot \mu', \psi^h \cdot \psi'^h, i_h, \text{force } V \overrightarrow{A}, T)$ where $\kappa' = [g \mapsto \mathbf{Match}(A, B, g)]_{g \in \nu(A)}$, $\psi'' = [\mathbf{Match}(A, B, g) \mapsto g]_{g \in \nu(A)}$ and $\mu' = [\nu(A) \mapsto (i_h, \eta)]$.

As $(\kappa \cdot \kappa')(K'[\text{return } A]) = K[\text{return } \kappa'(A)] = K[\text{return } B]$, $(\kappa \circ \gamma \circ \psi) \cdot (\kappa \circ \gamma \circ \psi') = \kappa \circ \gamma \circ (\psi \cdot \psi') = (\kappa \cdot \kappa') \circ \gamma \circ (\psi \cdot \psi')$, $(\kappa \circ H \circ \psi) \cdot (\kappa \circ H \circ \psi') \cdot [\nu(B) \mapsto \kappa(Fn)] = \kappa \circ ((H \circ \psi) \cdot (H \circ \psi') \cdot ([\nu(A) \mapsto Fn] \circ \psi'')) = (\kappa \cdot \kappa') \circ (H \cdot [\nu(A) \mapsto Fn]) \circ (\psi \cdot \psi' \cdot \psi'')$, $(\mu \circ \psi) \cdot (\mu \circ \psi') \cdot [\nu(B) \mapsto (i_h, \eta)] = (\mu \circ \psi) \cdot (\mu \circ \psi') \cdot (\mu' \cdot \psi'') = (\mu \cdot \mu') \circ (\psi \cdot \psi' \cdot \psi'')$, $l' = 0$ if F is empty and $l' = 1$ otherwise, we obtain that $\mathbf{D}' = \Theta((p \xrightarrow{c'(A)} \mathbf{C}', F'))$.

□

Proof of Lemma 49. This follows from Lemma 108 and Lemma 118, as bisimulation implies trace equivalence for deterministic LTS. □

C.5 Canonical Forms

To simplify our work, we introduce a 'canonical form' for terms. The key property of this is that only thunks from the context will be able to bound to variables. In the CBN setting, β-normal,η-long form has been used for this purpose [10], whereas for CBV a slightly more subtle form is need [12]. We adopt a canonical form similar in style to the CBV case.

Definition 119 (Canonical Form)**.** A CBPV computation (value) is in canonical form if it can be generated by M (V) in the grammar in Figure 16.

A useful result is that all terms can be placed into a contextually equivalent canonical form.

Lemma 120 (Lemma 56)**.** *Given a CBPV computation $\Gamma \vdash^c M : \underline{\tau}$, there exists a computation $\Gamma \vdash^c \mathbf{Canon}(M) : \underline{\tau}$ in canonical form such that $\Gamma \vdash^c M \simeq^{\mathrm{CBPV}}_{ctx} \mathbf{Canon}(M) : \underline{\tau}$*

Our proof is modelled on the one in [28]. We first prove a helper lemma. In these proofs, the transformations used can all be proved to maintain contextual equivalence using the equational theory from Appendix A.3.

Ground Types	β	$\triangleq$	Unit | Int
Restricted Values	V_0	$\triangleq$	$x \mid () \mid \widehat{n} \mid \ell \mid \text{MkVar } V_0\ V_0$
Values	V	$\triangleq$	$V_0 \mid \text{thunk } M \mid \text{MkVar (thunk } M\text{) (thunk } M\text{)}$
Restricted Computations	M_0	$\triangleq$	$\text{force } V_0 \mid \text{return } V_0 \mid M_0 V \mid \text{ref } V \mid !V_0 \mid V_0 := V_0$
Computations	M	$\triangleq$	$M_0 \mid \text{return } V \mid \lambda x^\sigma.M \mid \text{let } x^\beta \text{ be } V.M$
			$\mid M \text{ to } x^\beta.M \mid M_0 \text{ to } x.M \mid \text{case } V \text{ of } (M_i)_{i\in I} \mid \text{while } M \text{ do } M$

Figure 16: The grammar for terms in canonical form

Lemma 121. *Given CBPV computation M in canonical form*

1. *for a value $V : \sigma$ also in canonical form, then if $M\{V/x\}$ is type-able, it can be placed into canonical form;*
2. *for a value $V : \sigma$ also in canonical form, then if $M\,V$ is type-able, it can be placed into canonical form;*
3. *for a computation $N : F\sigma$ also in canonical form, then if N to $x.M$ is type-able, it can be placed into canonical form;*

s.t Lemma 120 is satisfied.

Proof. We proceed by induction on σ, considering the points in turn.

1. If V has a base type β, then we are done. Similarly, if V is a restricted value, we are done, as restricted values can occur wherever variables can. Otherwise σ is of the form $U\underline{\tau}$ or Ref. We handle these two cases in turn.

   - If σ is of the form $U\underline{\tau}$ then $V = \text{thunk } M'$. Let us consider the substitution for the right-most occurrence of x. If this creates a violating, non-canonical sub-term, then that sub-term will have the form $(\text{force thunk } M')\overrightarrow{V}$ or $(\text{force thunk } M')\overrightarrow{V}$ to $y.M''$. Observe that we can reduce the second case to the first, as y must have a type smaller than σ, so we can apply the third point of the I.H. once $(\text{force thunk } M')\overrightarrow{V}$ is in canonical form. We can apply the transformation

     $$\text{force thunk } M' \mapsto M'$$

     to reduce this to placing $M'\overrightarrow{V}$ into canonical form. For this, we can repeatedly apply the second point of the I.H., as the types of $\overrightarrow{V}$ are all smaller than σ. This replaces the violating sub-term with a canonical one, leaving the whole term in canonical form.

     We have a way of obtaining a canonical form of M with V substituted for the rightmost occurrence of x. As it was the rightmost occurrence of x, none of the terms we transform involve x, so the total number of occurrences of x has decreased. Thus, we can repeatedly make right-most substitutions and apply this process to obtain the canonical form for $M\{V/x\}$.

   - If σ is Ref, then $V = \text{MkVar (thunk } M_{\text{read}}\text{) (thunk } M_{\text{write}}\text{)}$. As M is in canonical form, due to the typing rules, x either occurs in a position V can occur and remain in canonical form, or in a sub-term of the form $!x$, $x := V_0$, $!x$ to $y.N$, or $x := V_0$ to $y.N$. As above, the later two cases reduce to the first via the third point. We can then use the following transformations (which are exactly the operational reductions)

     $$\begin{array}{rcl} !\text{MkVar (thunk } M_{\text{read}}\text{) (thunk } M_{\text{write}}\text{)} & \mapsto & M_{\text{read}} \\ (\text{MkVar (thunk } M_{\text{read}}\text{) (thunk } M_{\text{write}}\text{)}) := V_0 & \mapsto & M_{\text{read}}\, V_0 \end{array}$$

In the first case, we are done. In the other, we can use the second point of the I.H.

2. We can assume that M is a restricted computation or has the from $\lambda z.N$ repeatedly apply the following transformations

$$\begin{array}{rcc} (\text{case } V_0 \text{ of } (M_i)_{i\in I})V & \mapsto & \text{case } V_0 \text{ of } (M_i V)_{i\in I} \\ (\text{let } y^\beta \text{ be } V'.M_1)V & \mapsto & \text{let } z^\beta \text{ be } V'.(M_1\{z/y\}V) \\ & & \textit{where } z \text{ is not free in } V \\ (M_1 \text{ to } y.M_2)V & \mapsto & M_1 \text{ to } z.(M_2\{z/y\}V) \\ & & \textit{where } z \text{ is not free in } V \end{array}$$

If M is a restricted computation, then we are done. If it has the from $\lambda z.N$, we use that

$$(\lambda z.N)V \mapsto N\{V/z\}$$

and appeal to the first point.

3. We can assume that N is a restricted computation, or of the form return V by repeated application of the following transformations:

$$\begin{array}{rcc} (\text{case } V \text{ of } (M_i)_{i\in I}) \text{ to } x.M & \mapsto & \text{case } V \text{ of } (M_i \text{ to } x.M)_{i\in I} \\ (\text{let } y^\beta \text{ be } V.M_1) \text{ to } x.M_2 & \mapsto & \text{let } z^\beta \text{ be } V.(M_1\{z/y\} \text{ to } x.M_2) \\ & & \textit{where } z \text{ is not free in } M_2, \text{ and } z \neq x \\ (M_1 \text{ to } y.M_2) \text{ to } x.M_3 & \mapsto & M_1 \text{ to } z.(M_2\{z/y\} \text{ to } x.M_3) \\ & & \textit{where } z \text{ is not free in } M_3, \text{ and } z \neq x \end{array}$$

If N is a restricted computation, then we are done. If it has the form return V, we can use the fact return V to $x.M \equiv M\{V/x\}$ and appeal to the first point.

□

Proof of Lemma 120. We prove this result by induction on the structure of terms. We strengthen the hypothesis to also apply to values. Terms of the form return V, thunk M, $\lambda x.M$, case V of $(M_i)_{i\in I}$, let V be $y^\beta.M$, and while M do N can be dealt with simply by applying the I.H. to the sub-terms. $M\,V$ and M to $x.N$ can be handled by apply the I.H. to the sub-terms, and applying Lemma 121. The other cases are:

- let x be $V.N$ Then we apply the I.H. to V and N, the transformation

$$\text{let } x \text{ be } V.N \mapsto N\{V/x\}$$

and appeal to the first point of Lemma 121.

- force V If V is a variable we are done, otherwise $V = \text{thunk } M$, in which case we use

$$\text{force thunk } M \mapsto M$$

and appeal to the I.H.

- $!V$ or $V := V_0$ If V is a variable we are done, otherwise we appeal to the I.H. to get an equivalent V' in canonical form. If $V' = \text{MkVar } x\ y$ we are done. Otherwise $V' = \text{MkVar thunk } M_{\text{read}} \text{ thunk } M_{\text{write}}$, so we use

$$\begin{array}{rcc} !\text{MkVar } (\text{thunk } M_{\text{read}})\ (\text{thunk } M_{\text{write}}) & \mapsto & M_{\text{read}} \\ (\text{MkVar } (\text{thunk } M_{\text{read}})\ (\text{thunk } M_{\text{write}})) := V_0 & \mapsto & M_{\text{read}}\, V_0 \end{array}$$

and Lemma 121.

- MkVar V_{read} V_{write} By the I.H we obtain $V'_{\text{write}}, V'_{\text{read}}$ in canonical form. If $V'_{\text{write}}, V'_{\text{read}}$ are both variables or both thunks, we are done. Otherwise, we can turn a variable x in to the thunk thunk force x

□

Given our canonical forms, we have a number of useful results, which can be proved by simple case analysis and inductions.

Lemma 122. *If M is a term in canonical form, then if $(M, h) \to^* (N, h')$, then N is in canonical form.*

Definition 123. A computation N is said to be in ***weak canonical form*** if it can be generated from the following grammar, potentially with thunk names substituted for variables:

$$N \triangleq M \mid (\text{force thunk } M)\,\overrightarrow{V_0} \mid M\,\overrightarrow{V_0} \mid \text{return } V_0 \text{ to } x.M$$

where M, V denote terms in canonical form, and V_0 a restricted value.

Lemma 124. *If $\Gamma \vdash^c M' : F\sigma$ is in canonical form, and ρ a Γ-assignment, then for any active configuration $(\langle N, c', \gamma, \phi, h, H\rangle, S)$ reachable from $\mathsf{C}^{\rho,c}_{M'}$, N is in weak canonical form.*

Lemma 125. *If M is a computation not of the form* while N_1 do N_2 *in (weak) canonical form, if $(M, h) \to (N, h')$, then N is smaller than M, where size is measured by the size of the derivation of the computation in the grammar, treating all restricted values as having the same size.*

Proof. The key to this proof is that substitutions never cause the duplication of a term of the form thunk M, so β-reductions do not cause terms to grow in size. □

C.6 Proof of exponential size for $\mathcal{L}^{\Delta}_{\mathrm{PTR}}$ (Lemma 57 and Lemma 63)

In giving the full detail of these proofs, we will provide the Lemmata used again, for clarity, and indicate the number under which they appear in the main body of the paper. To simplify the statement of results in this section, we will fix a PTR-computation $\Gamma \vdash^c M : F\sigma$, (Γ, σ)-name scheme $\Delta = (\mathrm{TBNames}, \mathrm{CBNames}, \rho, c_0, \mathrm{Suc_T}, \mathrm{Suc_C})$, and let $\mathbf{C}_0 = \mathsf{C}^{\mathrm{PTR},\Delta}_M$ and $N_O = \nu(\rho) \cup \{c_0\}$. We introduce the notation that for $\eta, \eta' : (\mathrm{TBNames} \cup \mathrm{CBNames}) \to \mathbb{N}$ we write $\eta \leq \eta'$ if for all $d \in \mathrm{TBNames} \cup \mathrm{CBNames}$, $\eta(d) \leq \eta'(d)$. We write $\eta < \eta'$ if $\eta \leq \eta'$ and for some $d \in \mathrm{TBNames} \cup \mathrm{CBNames}$, $\eta(d) < \eta'(d)$.

We first make the observation that, for any configuration (of form $(\langle N, c, \gamma, h, H, i_h, \eta, \mu, l\rangle, S)$ or $(\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle, S)$), reachable from $\mathbf{C}_0$ has that the domains of γ, H, μ only contain d^i s.t. $i < \eta(d)$. Similarly, the domain of h consists only of locations smaller than i_h.

We will also observe that, when analysing the asymptotic size of the reachable state space, we can effectively ignore the tag l, as it simply gives a factor of 2 increase.

We say two states differ only in the heap when all components other than the heap (h) and tag (l) are the same, and that they are distinct beyond the heap when some component other than the heap or tag is different. We extend these notions analogously to configurations.

Lemma 126. *Let η be an index component. Let p be a path in $\mathcal{L}_{\mathrm{PTR}}$ s.t. every configuration $\mathbf{C}$ in p with index component η' satisfies $\eta \leq \eta'$. Then the environment (γ), history (H), and reset (μ) components of all configurations in the path agree when restricted to η (i.e. $\gamma_{<\eta}$).*

Proof. The key observation is that no type of transition causes the mappings γ, H, μ in a configuration to be 'updated', that is to have an existing binding changed. The only way we can end up changing the binding for an indexed name n^i with $i < \eta(n)$ is for a *PA*-action to occur using a head name c^j s.t. $j < \eta(c)$. But this would involve a configuration in p with index component η' having $\eta \not\leq \eta'$, which is not permitted. Thus, all γ, H, μ components must agree when restricted by η. □

Lemma 127. *Let $\mathbf{C}$ be a configuration (of form $(\langle N, c, \gamma, h, H, i_h, \eta, \mu, l\rangle, S)$ or $(\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle, S)$) reachable from $\mathbf{C}_0$. Let $\mathbf{C}'$ (of form $(\langle N', c', \gamma', h', H', i'_h, \eta', \mu', l'\rangle, S')$ or $(\langle \gamma', h', H', Fn', i'_h, \eta', \mu', l'\rangle, S')$) be a configuration reachable from $\mathbf{C}$ via a path p s.t.*

- *no answer action in p answers a question not occurring in p;*
- *no τ-action is produced involving reducing an occurrence of* end.N *found in* $\mathbf{C}$*;*
- *p includes no unanswered PQ-actions on a level 2 name.*

Then $\gamma'_{<\eta} = \gamma$, $H'_{<\eta} = H$, and $\mu'_{<\eta} = \mu$.

Proof. We proceed by induction on the number of PQ-actions on the level 2 names. In the base case, we have no level 2 names. It is easy to confirm that the first two conditions on p ensure that all intermediate configurations have index component $\eta \leq \eta''$ (as the three actions which can reduce the index component are constrained so that they can only reduce it to a value it held after $\mathbf{C}$ which, inductively, is larger that held in $\mathbf{C}$). Thus, the result follows from Lemma 126.

In the inductive case, consider the last OA-action answering a level 2 question. Let this be $c_1(A)$ answering $\bar{f}(\overrightarrow{A}, c_1)$. Now, we can partition p into the following way: $p^1\ \mathbf{C}^1 \xrightarrow{\bar{f}(\overrightarrow{A},c_1)} \mathbf{C}^2\ p^2\ \mathbf{C}^3 \xrightarrow{c_1(A)} \mathbf{C}^4\ p^3$. Let $\mathbf{C}^1 = (\langle N^1, d^j, \gamma^1, h^1,$
$H^1, i_h^1, \eta^1, \mu^1, l^1\rangle, S^1)$. Then we can apply the inductive hypothesis (as $p^1\ \mathbf{C}^1$ includes no unanswered PQ-actions on a level 2 name and has strictly fewer answered question than p) to obtain $\gamma^1_{<\eta} = \gamma$, $H^1_{<\eta} = H$, and $\mu^1_{<\eta} = \mu$. Let $(i''_h, \eta'') = \mu(f)$, then $\mathbf{C}^2 = (\langle \gamma^1_{<\eta''}, h^1_{<i''_h}, H^1_{<\eta''}, Fn^2, i''_h, \eta'',$
$\mu^1_{<\eta''}, l^2\rangle, (c^0, (K, d^j), P) : S^1)$ where $P = (i^1_h, \eta^1, \gamma^1_{\geq\eta''}, h^1_{\geq i''_h}, H^1_{\geq\eta''}, \mu^1_{\geq\eta''})$.

We obtain that the index component of $\mathbf{C}^3$ is η'' from Lemma 115 and Lemma 118 (the bi-simulation relating $\mathcal{L}_{\text{Path}}$ to $\mathcal{L}^{\Delta}_{\text{PTR}}$). Applying the I.H. between $\mathbf{C}^2$ and $\mathbf{C}^3$, and the above result, means that $\mathbf{C}^3 = (\langle \gamma^1_{<\eta''}, h', H^1_{<\eta''}, Fn^3, i''_h, \eta'', \mu^1_{<\eta''}, l^3\rangle, (c^0, (K, d^j), P) : S^1)$. Therefore, as $\zeta_{<\eta''} \cdot \zeta_{\geq\eta''} = \zeta$, we obtain that $\mathbf{C}^4 = (\langle K[A], d^j, \gamma^1, h^4, H^1 \cdot [\nu(A) \mapsto Fn^3], i^1_h, \eta^1 \cdot \eta^A, \mu^1 \cdot [\nu(A) \mapsto$
$(i^1_h], l^1\rangle, S^1)$. Finally, we can apply Lemma 126 to $\mathbf{C}^4\ p^3$ (as all intermediate configuration have index component at least η), which gives the desired result. □

A simple application of Lemma 127 yields the following, which essentially tells us it is okay to consider on a single iteration of a while loop.

Lemma 128 (Lemma 59)**.** *Let $M_1 = N_1$ to x.case x of return $(), (N_2$ to y.end(i_h, η).while M do $N)_{j>0}$. Let $\mathbf{C} = (\langle K[M_1], c, \gamma, h, H, i_h, \eta, \mu, l\rangle, S)$ be reachable from $\mathbf{C}_0$. Let $\mathbf{C}' = (\langle K[M_1], c, \gamma', h', H', i'_h, \eta', \mu', l'\rangle, S)$ be a configuration reached from $\mathbf{C}$ by a path p which does not include a PA-action on c. Then $\mathbf{C}$ and $\mathbf{C}'$ differ only in the heap.*

Lemma 129 (Lemma 58)**.** *Let $\mathbf{C}$ be a passive configuration reachable from $\mathbf{C}_0$ in $\mathcal{L}^{\Delta}_{\text{PTR}}$. Let $\mathbf{C}'$ be a passive state reachable from $\mathbf{C}$ with the same stack component S as $\mathbf{C}$, and with the stack never being shorter than S in an intermediate configuration. Then $\mathbf{C}$ and $\mathbf{C}'$ differ only in the heap.*

Proof. Let $\mathbf{C} = (\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle, S)$ and $\mathbf{C}' = (\langle \gamma', h', H', Fn', i'_h, \eta', \mu', l'\rangle, S)$. It suffices to prove this hold when no passive configuration $\mathbf{C}''$ with the same stack S appears in the (shortest) path p between $\mathbf{C}$ and $\mathbf{C}'$. If $\mathbf{C} = \mathbf{C}'$, this is trivial. Otherwise, the first action in p is a OQ-action leading to a configuration $(\langle$force $V\overrightarrow{A}, c^j, \gamma, h, H \cdot [c^j \mapsto Fn], i_h, \eta', \mu', l\rangle, S)$ (where $\eta' = \eta[c \mapsto j+1]$ and $\mu' = \mu \cdot [c^j \mapsto (i_h, \eta)]$), and so the last action must be the answering PA-action from some $(\langle$return $A, c'', \gamma', h'', H'', i''_h, \eta'', \mu'', l''\rangle, S)$. The conditions in the statement ensure that Lemma 127 is applicable, so we have that $Fn' = H''(c^j) = H \cdot [c^j \mapsto Fn](c^j) = Fn$, $(i'_h, \eta') = \mu''(c^j) = \mu \cdot [c^j \mapsto (i_h, \eta)] = (i_h, \eta)$, and further that, as $\eta' = \eta$, $\gamma = \gamma'$, $H = H'$, $\eta = \eta'$, and $\mu = \mu'$, as required. □

Lemma 130. *Let $\mathbf{C} = (\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle, S)$ be reachable from $\mathbf{C}_0$, which occurs immediately before OQ-action $g(\overrightarrow{A}, c)$ introducing a level 2 name f'. Let f be a name which has f' as its originator. Let p be a path from $\mathbf{C}$ such that*

- *no answer action in p answers a question not occurring in p;*

- *no τ-action is produced involving reducing an occurrence of* end.N *found in* $\mathbf{C}$*;*
- *p includes no unanswered PQ-actions on a level 2 name;*
- *p ends in active* $\mathbf{C}' = (\langle K[\text{force } f^i\overrightarrow{V}], c', \gamma', h', H', i'_h, \eta', \mu', l'\rangle, S')$.

Then we have $\mu'(f^i) = (i_h, \eta)$, $H'(f^i) = Fn$.

Proof. We proceed inductively on the introduction chain showing that f' is the originator of f. In the base case, we have that $f = f'$. Then it follows that in the configuration $\mathbf{C}'' = (\langle N'', c'', \gamma'', h'', H'', i''_h, \eta'', \mu'', l''\rangle, S')$ immediately after $\mathbf{C}$, we have $\mu''(f^i) = (i_h, \eta)$ and $H''(f^i) = Fn$. Applying Lemma 127 to p between $\mathbf{C}''$ and $\mathbf{C}'$ gives the desired result.

For the inductive case, consider the action in which (the instance visible here of) f^i is introduced (i.e. an action $d(A)$ after which the term includes return B where $f^i \in B$, and after which no PQ-action answering a question from before $d(A)$ occurs). Let this action answer action $\bar{f''}(\overrightarrow{A''}, d)$. Then we can partition p as follows: $p^1\ \mathbf{C}^1\ \xrightarrow{\bar{f''}(\overrightarrow{A''},d)}\ \mathbf{C}^2\ p^2\ \mathbf{C}^3\ \xrightarrow{d(A)}\ \mathbf{C}^4\ p^3$. It must be the case that f' is the originator of f''. Then if $\mathbf{C}^1 = (\langle K[\text{force } f''^k\overrightarrow{V^1}], c'^j, \gamma^1, h^1, H^1, i^1_h, \eta^1, \mu^1, l^1\rangle, S^1)$, by the I.H. $\mu^1(f''^k) = (i_h, \eta)$, $H^1(f''^k) = Fn$. Thus, in $\mathbf{C}^2 = (\langle \gamma^1_{<\eta}, h^1_{<i_h}, H^1_{<\eta}, Fn, i_h, \eta, \mu^1_{<\eta}, l^1\rangle, (d^0, (K, c'^j), P) : S^1)$ where $P_1 = (i^1_h, \eta^1, \gamma_{\geq\eta}, h_{\geq i_h}, H_{\geq\eta}, \nu_{\geq\eta})$. Now, it must be the case that the stack in $\mathbf{C}^3$ is that same as the one in $\mathbf{C}^2$, due to bracketing. Therefore, by Lemma 129, we obtain that $\mathbf{C}^2$ and $\mathbf{C}^3$ are the same up to the heap. Thus, if $\mathbf{C}^4 = ((K[\text{return } A], c'^j, \gamma^4, h^4, H^4, i^4_h, \eta^4, \mu^4, l^4), S^1)$, we have $\mu^4(f^i) = (i_h, \eta)$ and $H^4(f^i) = Fn$. Finally, we can appeal to Lemma 127 to obtain that μ^4, μ', and H^4, H' must agree on f^i, from which the result follows.

□

From Lemma 130 and Lemma 127 we obtain the following

Lemma 131 (Lemma 60)**.** *Let* $\mathbf{C} = (\langle \gamma, h, H, Fn, i_h, \eta, \mu, l\rangle, S)$ *be reachable from* $\mathbf{C}_0$*, which occurs immediately before OQ-action* $g(\overrightarrow{A}, c)$ *introducing a level 2 name* f'*. Let* f *be a name which has* f' *as its originator. Let* p *be a path from* $\mathbf{C}$ *such that*

- *no answer action in p answers a question not occurring in p;*
- *no τ-action is produced involving reducing an occurrence of* end. *found in* $\mathbf{C}$*;*
- *p includes no unanswered PQ-actions on a level 2 name whose;*
- *p ends in active* $\mathbf{C}' = (\langle K[\text{force } f^i\overrightarrow{V}], c', \gamma', h', H', i'_h, \eta', \mu', l'\rangle, S')$.

Then we have if $\mathbf{C}' \xrightarrow{\bar{f}(\overrightarrow{A},d)} \mathbf{C}''$*, the states of* $\mathbf{C}$ *and* $\mathbf{C}''$ *differ only in the heap.*

Proof. By Lemma 127, we have that $\gamma'_{<\eta} = \gamma$, $H'_{<\eta} = H$, and $\mu'_{<\eta} = \mu$. By Lemma 130, we have $\mu'(f^i) = (i_h, \eta)$, $H'(f^i) = Fn$. Thus, the state of $\mathbf{C}''$ is $\langle \gamma, h'_{<i_h}, H, Fn, i_h, \eta, \mu, l'\rangle$, as required. □

Lemma 132 (Lemma 62)**.** *Let M be in canonical form. Let* $\mathbf{C}$ *be an active configuration reachable from* $\mathbf{C}_0$*, which occurs immediately after a OQ-action, or is* $\mathbf{C}_0$*. Let* $\mathbf{C}'$ *be an active configuration reachable from* $\mathbf{C}$ *by a path p in which no OQ or PA-actions occur, and each occurrence of the while construct is reduced at most once. Then the number of intermediate configurations which are distinct beyond the heap is polynomial in M.*

Proof. Let $\mathbf{C} = (\langle N, c, \gamma, h, H, i_h, \eta, \mu, l\rangle, S)$. We can easily show that the size of N is bounded by the size of M and the types in Γ and σ (the need for this is that a OQ-action creates applications to abstract value, but this is restricted by the types). Further, observe that any PQ-action must be followed immediately by an OA-action (as it cannot be an OQ-action). Thus, in the path, we always have reductions of the form $(\langle K[(\text{force } f)\overrightarrow{V}], c', \gamma', h', H', i'_h, \eta', \mu', l'\rangle, S) \xrightarrow{\bar{f}(\overrightarrow{A},i''_t)} \mathbf{C}'' \xrightarrow{i''_t(A)} (\langle K[\text{return } A], c', \gamma'', h', H'', i'_h, \eta'', \mu', l'\rangle, S)$

Let us consider a measure of the size of M which treats all values as having the same size. Recall from Lemma 124 that the term components are in weak canonical form. We can then see that every τ-action in p (save expanding whiles) decreases the size of the term component. Every PQ-action followed by an OA-action does not increase the size of the term component, and as every such pair is either the end of p, or is followed by a τ-action, we can conclude that the triple of actions decreases the size of the term component. In the absence of while, this would suffice to bound the number of distinct configurations. With while, we can appeal to the fact that each occurrence of the while construct is reduced at most once. This means we can treat expansions of while as if they took while M_1 do M_2 to M to x.case x of return $(), (N$ to $y.\text{end}(\tilde{i_h}, \tilde{\eta}).\Omega)_{j>0}$. Thus, expanding a while increases the size of the term component, but by a constant amount. As the number of while constructs is bound by the size of N, this gives us that the number of distinct configurations is linear in N. □

We now prove Lemma 61, the technques for which are independent of those used for the other results.

Lemma 133 (Lemma 61)**.** *Let $\Gamma \vdash^c M : F\sigma$ be a* PTR*-computation in canonical form, Γ-assignment ρ and continuation name c_0. Let $t \in \mathbf{Tr}(\mathsf{C}_M^{\rho,c_0})$ be a trace which does not have an unanswered PQ-action using a level 2 head name, then the number of unanswered PQ-actions in t is bounded by the size of M.*

Proof. Let p be the path which generates t.

Let $\mathbf{Forces}(N)$ be the function which counts the number of occurrences in weak canonical form term N, of the construct force x, $!x$, $x := V$, !MkVar x y, MkVar x $y := V$, and any of these where thunk names appear substituted for variables[1]. We will also allow the natural extension to reduction contexts K in weak canonical normal form, $\mathbf{Forces}(K)$. For a term (or context) N and mapping from names to thunks γ, define inductively the least set of reachable P-names,

$$\begin{array}{rcl} \mathbf{ReachT}_{\gamma,H}(N) & = & \bigcup_{\substack{g \in \nu(N) \\ g \text{ is not level 2}}} (\mathbf{ReachN}_{\gamma,H}(H(g))) \\ \mathbf{ReachN}_{\gamma,H}(S) & = & S \cup \bigcup_{f \in S} \mathbf{ReachT}_{\gamma,H}(\gamma(f)) \end{array}$$

which simply finds the P-names reachable from the top-most element of the stack (if there is one). In the natural way, we can lift γ to act on a set of P-names, and **Forces** to sets of terms by summing over the elements. Using this, we can then define the function $\mathbf{Forces}(\mathbf{C})$ on computations as

$$\begin{array}{rcl} \mathbf{Forces}((\langle \gamma, \phi, h, H, Fn\rangle, \bot)) & = & \mathbf{Forces}(\gamma(\mathbf{ReachN}_{\gamma,H}(Fn))) \\ \mathbf{Forces}((\langle \gamma, \phi, h, H, Fn\rangle, (c, (K, c')) : S)) & = & \mathbf{Forces}(\gamma(\mathbf{ReachN}_{\gamma,H}(Fn) \cup \mathbf{ReachT}_{\gamma,H}(K))) \\ & & +\mathbf{Forces}(K) \\ \mathbf{Forces}((\langle N, c, \gamma, \phi, h, H\rangle, S)) & = & \mathbf{Forces}(N) + \mathbf{Forces}(\gamma(\mathbf{ReachT}_{\gamma,H}(N))) \end{array}$$

We will now consider how $\mathbf{Forces}(\mathbf{C})$ changes as we move along the path p.

First, we observe that if $\mathbf{C} = (\langle \gamma, \phi, h, H, Fn\rangle, S)$ is the configuration immediately before an OQ-transition in p, and $\mathbf{C}' = (\langle \gamma', \phi', h', H', Fn'\rangle, S)$ is the configuration immediately after the answering PA-transition in p, then $\mathbf{Forces}(\mathbf{C}) = \mathbf{Forces}(\mathbf{C}')$. This is consequence of the fact that in the PTR-fragment, visibility entails that $Fn = Fn'$, and on all names already introduced at $\mathbf{C}$, γ, H and γ', H' agree (γ', H' extends γ, H). Thus, we can essentially 'skip' such sections of the path, and will not have to consider the case of a answering PA-transition.

Similarly, let $\mathbf{C} = (\langle K[(\text{force } f)\ \overrightarrow{V}], c, \gamma, \phi, h, H\rangle, S)$ be the state immediately before a PQ-action $\bar{f}(\overrightarrow{A}, c')$ where f is a level 2 name, and let $\mathbf{C}' = (\langle K[\text{return } A], c, \gamma', \phi', h', H'\rangle, S)$ be the state immediately following the answering OA-action $c'(A)$. Now, γ, H and γ', H' agree on all names already introduced by $\mathbf{C}$. Observe that any names in $\nu(A)$ must be level 2 names, so we have that $\mathbf{Forces}(\mathbf{C}') = \mathbf{Forces}(K[\text{return } A]) + \mathbf{Forces}(\gamma(\mathbf{ReachT}_{\gamma',H'}(K[\text{return } A]))) <$

[1] We need to count these uses of reference types, as they might reduce to a force f.

$\mathbf{Forces}(K[(\text{force } f')\ \overrightarrow{V}]) + \mathbf{Forces}(\gamma(\mathbf{ReachT}_{\gamma,H}(K[(\text{force } f')\ \overrightarrow{V}]))) = \mathbf{Forces}(\mathbf{C})$. Thus, we can also 'skip' these sections of a path, and as PQ-actions on level two names must be answered, we do not need to consider this case.

Finally, let $\mathbf{C}$ be a configuration in p where we are about to expand a while loop, which clearly will duplicate occurrences of force f etc. However, consider what occurs later in p. If we reach the point when we are expanding the same while loop again, we will have reached a configuration $\mathbf{C}'$ with $\mathbf{Forces}(\mathbf{C}) = \mathbf{Forces}(\mathbf{C}')$, which follows from the fact that when we reach the end of a loop, any name introduced in a loop does not escape the iteration it was introduced in. In the case that we never reduce the loop again (i.e. the condition is 0), then we observe that a path produced by instead expanding $\text{while } N_1 \text{ do } N_2$ into $K[N_1 \text{ to } x.\text{case } x \text{ of return } (), (N_2 \text{ to } y.\Omega)_{i>0}]$ produces the same trace, without increasing the value of $\mathbf{Forces}()$. Thus, we do not need to consider while loops.

What we will wish to show is that a PQ-transitions on non-level 2 names reduce the value of $\mathbf{Forces}(\mathbf{C})$, and all other types of transitions (other than answering PA-transitions, PQ-transitions on level 2 names and those expanding while loops) do not increase the value.

Let $\mathbf{C} = (\langle K[\text{force } f\overrightarrow{V}], c, \gamma, \phi, h, H\rangle, S) \xrightarrow{\bar{f}(\overrightarrow{A},c')} (\langle \gamma \cdot \gamma_A, \phi', h, H, H(f) \cup \nu(\overrightarrow{A})\rangle, (c', (K, c)) : S) = \mathbf{C}'$ occur in path p. Now,

$$\begin{array}{rcl}
\mathbf{Forces}(\mathbf{C}) & = & 1 + \mathbf{Forces}(K) + \mathbf{Forces}(\bigcup_{V\in\overrightarrow{V}}\{V\}) + \\
 & & \mathbf{Forces}(\gamma(\mathbf{ReachN}_{\gamma,H}(H(f)) \cup \bigcup_{V\in\overrightarrow{V}} \mathbf{ReachT}_{\gamma,H}(V) \cup \mathbf{ReachT}_{\gamma,H}(K))) \\
\mathbf{Forces}(\mathbf{C}') & = & \mathbf{Forces}(K) + \\
 & & \mathbf{Forces}(\gamma \cdot \gamma_A(\mathbf{ReachN}_{\gamma\cdot\gamma_A,H}(H(f) \cup \nu(\overrightarrow{A})) \cup \mathbf{ReachT}_{\gamma\cdot\gamma_A,H}(K)))
\end{array}$$

And as we have

$$\begin{array}{rcl}
\mathbf{ReachT}_{\gamma\cdot\gamma_A,H}(K) & = & \mathbf{ReachT}_{\gamma,H}(K) \\
\mathbf{ReachN}_{\gamma\cdot\gamma_A,H}(H(f) \cup \nu(\overrightarrow{A}))) & = & \mathbf{ReachN}_{\gamma\cdot\gamma_A,H}(H(f)) \cup \bigcup_{g\in\nu(\overrightarrow{A})} \mathbf{ReachT}_{\gamma\cdot\gamma_A,H}(\gamma \cdot \gamma_A(g)) \cup \nu(\overrightarrow{A}) \\
 & = & \mathbf{ReachN}_{\gamma,H}(H(f)) \cup \bigcup_{g\in\nu(\overrightarrow{A})} \mathbf{ReachT}_{\gamma,H}(\gamma_A(g)) \cup \nu(\overrightarrow{A}) \\
 & = & \mathbf{ReachN}_{\gamma,H}(H(f)) \cup \bigcup_{V\in\overrightarrow{V}} \mathbf{ReachT}_{\gamma,H}(V) \cup \nu(\overrightarrow{A})
\end{array}$$

$\mathbf{Forces}(\gamma \cdot \gamma_A(\mathbf{ReachT}_{\gamma\cdot\gamma_A,H}(K) \cup \mathbf{ReachN}_{\gamma\cdot\gamma_A,H}(H(f) \cup \nu(\overrightarrow{A}))))$

as names in $\nu(\overrightarrow{A})$ are mapped into $\overrightarrow{V}$ under γ_A

$= \mathbf{Forces}(\gamma(\mathbf{ReachT}_{\gamma,H}(K) \cup \mathbf{ReachN}_{\gamma,H}(H(f))) \cup \bigcup_{V\in\overrightarrow{V}} \mathbf{ReachT}_{\gamma,H}(V) \cup \bigcup_{V\in\overrightarrow{V}}\{V\})$

$= \mathbf{Forces}(\gamma(\mathbf{ReachT}_{\gamma,H}(K) \cup \mathbf{ReachN}_{\gamma,H}(H(f))) \cup \bigcup_{V\in\overrightarrow{V}} \mathbf{ReachT}_{\gamma,H}(V)) + \mathbf{Forces}(\bigcup_{V\in\overrightarrow{V}}\{V\})$

we can conclude that $\mathbf{Forces}(\mathbf{C}') < \mathbf{Forces}(\mathbf{C})$.

We can now check that the other cases do no cause an increase.

- τ-transition. It is easy to check that for N in weak canonical normal form, $(N, h) \to (N', h')$, then $\mathbf{Forces}(N) \geq \mathbf{Forces}(N')$. The key to this is that the weak canonical form ensures that reductions do not duplicate sub-terms of the from thunk M'. The case of expanding a while loop was handled above.

- OQ-transitions Suppose $\mathbf{C} = (\langle \gamma, \phi, h, H, Fn\rangle, S) \xrightarrow{f(\overrightarrow{A},c)} (\langle \text{force } V\overrightarrow{A}, c, \gamma, \phi \uplus \phi', h, H \cdot [\phi' \mapsto Fn]\rangle, S) = \mathbf{C}'$ in path p, with $V = \gamma(f)$ and $\phi' = \nu(\overrightarrow{A}) \uplus \{c'\}$. Now, all the names in $\nu(\overrightarrow{A})$ are level 2 names. Thus $\mathbf{Forces}(\mathbf{C}') = \mathbf{Forces}(V) + \mathbf{Forces}(\gamma(\mathbf{ReachT}_{\gamma,H}(V)))$. Now, as $f \in Fn$, and $\mathbf{Forces}(\mathbf{C}) \geq \mathbf{Forces}(\gamma(\mathbf{ReachN}_{\gamma,H}(Fn))) \geq \mathbf{Forces}(\gamma(\{f\} \cup \mathbf{ReachT}_{\gamma,H}(V))) = \mathbf{Forces}(\mathbf{C}')$ (as $f \in \nu(V)$)

- OA-transitions Suppose $\mathbf{C} = (\langle \gamma, \phi, h, H, Fn\rangle, (c, (K, c')) : S) \xrightarrow{c(A)} (\langle K[\text{return } A], c', \gamma, \phi \uplus \nu(A), h, H \cdot [\nu(A) \mapsto Fn]\rangle, S) = \mathbf{C}'$. Now, $\bigcup_{g\in\nu(A)} \mathbf{ReachN}_{\gamma,H\cdot[\nu(A)\mapsto Fn]}(H \cdot [\nu(A) \mapsto$

$Fn](A)) = \mathbf{ReachN}_{\gamma,H}(Fn)$, and $\mathbf{Forces}(\text{return } A) = 0$. So

$$\begin{array}{rcl}\mathbf{Forces}(\mathbf{C}') & = & \mathbf{Forces}(K) + \mathbf{Forces}(\text{return } A) + \\ & & \mathbf{Forces}(\gamma(\bigcup_{g\in\nu(A)} \mathbf{ReachN}_{\gamma,H\cdot[\nu(A)\mapsto Fn]}(H\cdot[\nu(A)\mapsto Fn](A)) \cup \mathbf{ReachT}_{\gamma,H\cdot[\nu(A)\mapsto Fn]}(K))) \\ & = & \mathbf{Forces}(K) + \mathbf{Forces}(\gamma(\mathbf{ReachN}_{\gamma,H}(Fn) \cup \mathbf{ReachT}_{\gamma,H}(K))) \\ & = & \mathbf{Forces}(\mathbf{C})\end{array}$$

- *PA*-transitions By the consideration above, we only need to check *PA*-transitions with head name c_0. Let $\mathbf{C} = (\langle \text{return } V, c_0, \gamma, \phi, h, H\rangle, \bot) \xrightarrow{\bar{c_0}(A)} (\langle \gamma\cdot\gamma', \phi \uplus \nu(A), h, H, H(c_0) \uplus \nu(A)\rangle, \bot) = \mathbf{C}'$ where $(A,\gamma') \in \mathbf{AVal}_\sigma(V)$. Now, $H(c_0) = \emptyset$, so

$$\begin{array}{rcl}\mathbf{Forces}(\mathbf{C}') & = & \mathbf{Forces}(\gamma\cdot\gamma'(\mathbf{ReachN}_{\gamma\cdot\gamma',H}(\nu(A)))) \\ & = & \mathbf{Forces}(\nu(A) \cup \gamma(\bigcup_{g\in\nu(A)} \mathbf{ReachT}_{\gamma\cdot\gamma',H}(\gamma\cdot\gamma'(g)))) \\ & = & \mathbf{Forces}(V \cup \gamma(\mathbf{ReachT}_{\gamma,H}(V))) \\ & = & \mathbf{Forces}(V) + \mathbf{Forces}(\gamma(\mathbf{ReachT}_{\gamma,H}(V))) \\ & = & \mathbf{Forces}(\mathbf{C})\end{array}$$

Now, we simply conclude by observing that $\mathbf{Forces}(\mathsf{C}_M^{\rho,c_0})$ is bounded in M, as it simply reduces to $\mathbf{Forces}(M)$. As for an *OQ*-actions to occur from $\mathbf{C}$, $\mathbf{Forces}(\mathbf{C}) > 0$, we obtain the desired bound on the number of unanswered *OQ*-actions in t. □

C.7 The IA and RML translations

We present the syntax of RML in Figure 17, and of IA in Figure 18. The languages have the usual operational semantics, which we shall write as $\to_{\text{RML}}$ and $\to_{\text{IA}}$ (or simply as $\to$ when clear from context). In particular, RML is operator first reduction. For termination (reducing to a value) we write $(M,h) \Downarrow^{\text{RML}}_{ter}$ and $(M,h) \Downarrow^{\text{IA}}_{ter}$. We can define contextual approximation in the standard way using the notions of context define below:

- For RML terms $\Gamma \vdash M_1, M_2 : \sigma, \Gamma \vdash M_1 \lesssim^{\text{RML}}_{ter} M_2$ holds when for all contexts $C \vdash \sigma \implies \sigma'$, $(C[M_1], h) \Downarrow^{\text{RML}}_{ter}$ implies $(C[M_2], h) \Downarrow^{\text{RML}}_{ter}$.
- For IA terms $\Gamma \vdash M_1, M_2 : \underline{\tau}$, $\Gamma \vdash M_1 \lesssim^{\text{IA}}_{ter} M_2$ holds when for all contexts $C \vdash \underline{\tau} \implies \underline{\text{com}}$, $(C[M_1], h) \Downarrow^{\text{IA}}_{ter}$ implies $(C[M_2], h) \Downarrow^{\text{IA}}_{ter}$.

RML Types $\sigma \triangleq \text{Unit} \mid \text{Int} \mid \text{Ref} \mid \sigma \to \sigma$

RML Value	V	$\triangleq$	$x \mid \widehat{n} \mid () \mid \ell \mid \text{MkVar } V\ V \mid \lambda x^\sigma.M$
RML Term	M	$\triangleq$	$V \mid M\,M \mid M := M \mid !M \mid \text{ref}\, M \mid \text{MkVar } M\ M \mid \text{while}\, M\, \text{do}\, M \mid \text{case } M \text{ of } (M_i)_{i\in I}$
RML Eval Ctx	K	$\triangleq$	$\bullet \mid K\,M \mid V\,K \mid M := K \mid K := V \mid !K \mid \text{ref}\, K \mid \text{MkVar } K\ M \mid \text{MkVar } V\ K$ $\mid \text{case } K \text{ of } (M_i)_{i\in I}$
RML Ctx	C	$\triangleq$	$\bullet \mid \text{MkVar } C\ M \mid \text{MkVar } M\ C \mid \lambda x^\sigma.C \mid C\,M \mid M\,C \mid C := M \mid M := C \mid !C \mid \text{ref}\, K$ $\text{case } C \text{ of } (M_i)_{i\in I} \mid \text{case } M \text{ of } (M_i)_{i<j}, C, (M_i)_{j<i} \mid \text{while}\, C\, \text{do}\, M \mid \text{while}\, M\, \text{do}\, C$

Notational conventions: $x, y \in \mathbf{Var}$, $\ell \in \mathbf{Loc}$, $n \in \mathbb{Z}$
Syntactic sugar: We write $\text{let}\, x = N \,\text{in}\, M$ for $(\lambda x.M)N$, and if x does not occur free in M, we write $N; M$ for $\text{let}\, x = N \,\text{in}\, M$, and Ω for $\text{while}\, \widehat{1}\, \text{do}\, ()$

Figure 17: RML syntax

We recall the translation of types is given in the following table.

$$\begin{array}{llll}
\text{IA types} & \underline{\tau} & \triangleq & \underline{\text{com}} \mid \underline{\text{expr}} \mid \underline{\text{var}} \mid \underline{\tau} \to \underline{\tau} \\
\\
\text{IA Value} & V & \triangleq & \text{skip} \mid x \mid \widehat{n} \mid () \mid \ell \mid \text{MkVar } M\ M \mid \lambda x^{\underline{\tau}}.M \\
\text{IA Term} & M & \triangleq & V \mid M\,M \mid M; M \mid M := M \mid !M \mid \text{new}\, x \,\text{in}\, M \mid \text{while}\, M \,\text{do}\, M \mid \text{case}\ M\ \text{of}\ (M_i)_{i\in I} \\
\text{IA Eval Ctx} & K & \triangleq & \bullet \mid K\,M \mid M := K \mid K := V \mid !K \mid \text{MkVar } K\ M \mid \text{MkVar } V\ K \mid \text{case}\ K\ \text{of}\ (M_i)_{i\in I} \\
\text{IA Ctx} & C & \triangleq & \bullet \mid \text{MkVar } C\ M \mid \text{MkVar } M\ C \mid \lambda x^{\sigma}.C \mid C\,M \mid M\,C \mid C; M \mid M; C \mid C := M \mid M := C \\
& & & \mid\ !C \mid \text{new}\, x \,\text{in}\, C \mid \text{case}\ C\ \text{of}\ (M_i)_{i\in I} \mid \text{case}\ M\ \text{of}\ (M_i)_{i<j}, C, (M_i)_{j<i} \\
& & & \mid\ \text{while}\, C \,\text{do}\, M \mid \text{while}\, M \,\text{do}\, C
\end{array}$$

Notational conventions: $x, y \in \mathbf{Var},\ \ell \in \mathbf{Loc},\ n \in \mathbb{Z}$
Syntactic sugar: We write Ω for $\text{while}\, \widehat{1}\, \text{do skip}$

Figure 18: IA syntax

RML type	CBPV value types
$\text{Int}, \text{Unit}, \text{Ref}$	$\text{Int}, \text{Unit}, \text{Ref}$
$\sigma_1 \to \sigma_2$	$U({\sigma_1}^{\text{RML}} \to F{\sigma_2}^{\text{RML}})$

IA type	CBPV computation types
$\underline{\text{expr}}, \underline{\text{com}}$	$F\text{Int}, F\text{Unit}$
$\underline{\text{var}}$	$\text{Int} \to \text{Int} \to F\text{Int}$
$\underline{\tau_1} \to \underline{\tau_2}$	$U{\underline{\tau_1}}^{\text{IA}} \to {\underline{\tau_2}}^{\text{IA}}$

We provide the translation $-^{\text{RML}}$ on terms in the table below.

RML term $M : \sigma$	CBPV computation $M^{\text{RML}} : F\sigma^{\text{RML}}$
x	return x
$()$	return $()$
$\widehat{n}$	return $\widehat{n}$
ℓ	return ℓ
$\lambda x.M$	return thunk $\lambda x.M^{\text{RML}}$
MkVar $M\ N$	M^{RML} to $f_r.N^{\text{RML}}$ to f_w.MkVar (thunk force $f_r()$) f_w
$M\,N$	M^{RML} to $f.N^{\text{RML}}$ to $x.(\text{force}\ f)\,x$
$M := N$	N^{RML} to $x.M^{\text{RML}}$ to $y.y := x$
$!M$	M^{RML} to $x.!x$
$\text{ref}\, M$	M^{RML} to $x.\text{ref}\, x$
$\text{while}\, M \,\text{do}\, N$	$\text{while}\, M^{\text{RML}} \,\text{do}\, N^{\text{RML}}$
$\text{case}\ M\ \text{of}\ (M_i)_{i\in I}$	M^{RML} to $x.\text{case}\ x\ \text{of}\ ({M_i}^{\text{RML}})_{i\in I}$

For a RML environment $\Gamma = \{x_1 : \sigma_1 \cdots x_k : \sigma_k\}$, we let $\Gamma^{\text{RML}} = \{x_1 : {\sigma_1}^{\text{RML}} \cdots x_k : {\sigma_k}^{\text{RML}}\}$. Thus, the sequent $\Gamma \vdash M : \sigma$ is translated into $\Gamma^{\text{RML}} \vdash^c M^{\text{RML}} : F\sigma^{\text{RML}}$.

For IA, we need to ensure terms are in η-long form: that is we η-expand so that all occurances of variables with function type are fully applied. η-expansion does not effect contexual equivalence in IA. This is so that any arguments that sub-terms of M take with type $\underline{\tau_1} \to \cdots \underline{\tau_k} \to \underline{\text{var}}$ are exposed. We provide the translation $-^{\text{IAP}}$ on terms in the table below, then define M^{IA} to be ${M'}^{\text{IAP}}$ where M has η-long form M'. Let $assert(x \sim n)$ be short for $\text{case}\ x\ \text{of}\ (\Omega)_{i<n}, (), (\Omega)_{n<i}$.

IA term $M : \underline{\tau}$	IA computation $M^{\mathrm{IAP}} : \underline{\tau}^{\mathrm{IA}}$
$x : \underline{\tau_1} \to \cdots \underline{\tau_k} \to \underline{\mathrm{var}}$	$\lambda y_1.\cdots\lambda y_k.\lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}$ $((assert(w \sim 0); (\mathrm{force}\ x)y_1 \cdots y_k\widehat{0}\widehat{0}), ((\mathrm{force}\ x)y_1 \cdots y_k\widehat{1}w\ \mathrm{to}\ z.assert(z \sim 0); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})$
x otherwise	$\mathrm{force}\ x$
skip	$\mathrm{return}\ ()$
$\widehat{n}$	$\mathrm{return}\ \widehat{n}$
ℓ	$\lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); !\ell), (\ell := w; \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})$
$\lambda x.M$	$\lambda x.M^{\mathrm{IAP}}$
MkVar $M\ N$	$\lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); M^{\mathrm{IAP}}), (N^{\mathrm{IAP}}(\mathrm{thunk}\ \mathrm{return}\ w); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})$
$M\,N$	$M^{\mathrm{IAP}}\,(\mathrm{thunk}\ N^{\mathrm{IAP}})$
$M; N$	$M^{\mathrm{IAP}}\ \mathrm{to}\ x.N^{\mathrm{IAP}}$
$M := \widehat{n}$	$M^{\mathrm{IAP}}\,\widehat{1}\,\widehat{n}\ \mathrm{to}\ y.assert(y \sim 0)$
$M := N$ otherwise	$N^{\mathrm{IAP}}\ \mathrm{to}\ x.M^{\mathrm{IAP}}\,\widehat{1}\,x\ \mathrm{to}\ y.assert(y \sim 0)$
$!M$	$M^{\mathrm{IAP}}\,\widehat{0}\,\widehat{0}$
$\mathrm{new}\,x\,\mathrm{in}\,M$	$\mathrm{ref}\,\widehat{0}\ \mathrm{to}\ x'.\mathrm{let}\ x\ \mathrm{be\ thunk}\ \lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); !x'), (x' := w; \mathrm{return}\ \widehat{0}), (\Omega)_{i>0}).$ M^{IAP}
$\mathrm{while}\,M\,\mathrm{do}\,N$	$\mathrm{while}\,M^{\mathrm{IAP}}\,\mathrm{do}\,N^{\mathrm{IAP}}$
$\mathrm{case}\ M\ \mathrm{of}\ (M_i)_{i\in I}$	$M^{\mathrm{IAP}}\ \mathrm{to}\ x.\mathrm{case}\ x\ \mathrm{of}\ ({M_i}^{\mathrm{IAP}})_{i\in I}$

For an IA environment $\Gamma = \{x_1 : \underline{\tau}_1 \cdots x_k : \underline{\tau}_k\}$, we let $\Gamma^{\mathrm{IA}} = \{x_1 : U{\underline{\tau}_1}^{\mathrm{IA}} \cdots x_k : U{\underline{\tau}_k}^{\mathrm{IA}}\}$. Thus, the sequent $\Gamma \vdash M : \underline{\tau}$ is translated into $\Gamma^{\mathrm{IA}} \vdash^c M^{\mathrm{IA}} : \underline{\tau}^{\mathrm{IA}}$

C.8 Full abstraction of the RML and IA translations

To prove Theorem 66, we will need to provide both soundness and completeness results.

We first present a useful result which deals with the somewhat unusual translation of variables with a type returning $\underline{\mathrm{var}}$.

Lemma 134. *For $\Sigma; \Gamma \vdash^c M^{\mathrm{IAP}} : U\tau_1 \to \cdots \to U\tau_k \to \mathrm{Int} \to \mathrm{Int} \to F\mathrm{Int}$ (where k can be 0), for any $\vdash^v V_i : U\tau_i$, $\Sigma \vdash \gamma : \Gamma$, and heap $h : \Sigma$, $(M^{\mathrm{IA}}\{\gamma\}\,V_1 \cdots V_k\,\widehat{j}\,\widehat{m}, h) \Downarrow_{ter}$ only if $j = 0$ and $m = 0$, or $j = 1$. If $j = 1$, then if this terminates, it reduces to $(\mathrm{return}\ 0, h')$.*

Proof. The proof of this is by induction on the structure of M. In the base cases, we have that M is either ℓ, MkVar M_1 M_2, or x. It is easy to verify the requirements hold in these cases. We show the last case.

$$\begin{array}{rcl}
(x^{\mathrm{IAP}}\{\gamma\}V_1 \cdots V_k\,\widehat{j}\,\widehat{m}, h) & = & ((\lambda y_1.\cdots\lambda y_k.\lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); (\mathrm{force}\ x\{\gamma\})y_1 \cdots y_k\widehat{0}\widehat{0}), \\
& & \quad ((\mathrm{force}\ x\{\gamma\})y_1 \cdots y_k\widehat{1}w\ \mathrm{to}\ z.assert(z \sim 0); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1}))\,V_1 \cdots V_k\,\widehat{j}\,\widehat{m}, h) \\
& \to^* & ((\lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); (\mathrm{force}\ x\{\gamma\})\,V_1 \cdots V_k\widehat{0}\widehat{0}), \\
& & \quad ((\mathrm{force}\ x\{\gamma\})\,V_1 \cdots V_k\widehat{1}w\ \mathrm{to}\ z.assert(z \sim 0); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1}))\,\widehat{j}\,\widehat{m}, h) \\
& \to^* & (\mathrm{case}\ \widehat{j}\ \mathrm{of}\ ((assert(\widehat{m} \sim 0); (\mathrm{force}\ x\{\gamma\})\,V_1 \cdots V_k\widehat{0}\widehat{0}), \\
& & \quad ((\mathrm{force}\ x\{\gamma\})\,V_1 \cdots V_k\widehat{1}\widehat{m}\ \mathrm{to}\ z.assert(z \sim 0); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})), h)
\end{array}$$

Observe that if $j \neq 0$ and $j \neq 1$, then this reduces to (Ω, h), which does not terminate. Further, if $j = 0$, then this reduces to $(assert(\widehat{m} \sim 0); (\mathrm{force}\ x\{\gamma\})\,V_1 \cdots V_k\widehat{0}\widehat{0}, h)$, and as $assert(\widehat{m} \sim 0)$ reduces to Ω if $m \neq 0$, for this to terminate, we get $m = 0$. Finally, if $j = 1$, then this reduces to $((\mathrm{force}\ x\{\gamma\})\,V_1 \cdots V_k\widehat{1}\widehat{m}\ \mathrm{to}\ z.assert(z \sim 0); \mathrm{return}\ \widehat{0}, h)$, which, if it terminates, reduces to $(\mathrm{return}\ \widehat{0}, h')$, as required.

The inductive cases for M having the form $N_1\,N_2$, $N_1; N_2$, case N of $(N_i)_{i\in I}$ are all straight forward. Observe that, due to the typing rule, M cannot have the form $\mathrm{new}\,x\,\mathrm{in}\,N$. We check the most interesting case, that M has the form $\lambda x.N$. In this case we have $(\lambda x.N)^{\mathrm{IAP}} = \lambda x.N^{\mathrm{IAP}}$. Thus, for any $\vdash^v V_i : U\tau_i$, $\Sigma \vdash \gamma : \Gamma$, and heap $h : \Sigma$, we have

$$((\lambda x.N)^{\mathrm{IAP}}\{\gamma\}\,V_1 \cdots V_k\,\widehat{j}\,\widehat{m}, h) = ((\lambda x.N^{\mathrm{IAP}}\{\gamma\})\,V_1 \cdots V_k\,\widehat{j}\,\widehat{m}, h) \to (N^{\mathrm{IAP}}\{\gamma \cdot [x \mapsto V_1]\}\,V_2 \cdots V_k\,\widehat{j}\,\widehat{m}, h)$$

Thus, by an appeal to the I.H on N^{IAP}, we obtain that $((\lambda x.N)^{\mathrm{IAP}}\{\gamma\}\, V_1 \cdots V_k\, \widehat{j}\, \widehat{m}, h)$ terminates only if $j = 0$ and $m = 0$, or $j = 1$, and if $j = 1$, then it terminates only if it reduces to $(\text{return } \widehat{0}, h')$. □

Lemma 135. *Let $\Gamma \vdash M : \underline{\tau} \to \underline{\tau'}$ be an* IA*-term. Then there exists an* IA*-term N s.t. $M^{\mathrm{IA}} =_\beta \lambda x.N^{\mathrm{IA}}$.*

Proof. This proceeds by induction on the structure of M.

In the base cases, are as follows. First, we have $M = \lambda x.N$, which is trivial, as $M^{\mathrm{IA}} = \lambda x.N^{\mathrm{IA}}$. The second case is $x\, N_1 \cdots N_k$ where $x : \underline{\tau_1} \to \cdots \to \underline{\tau_n} \to \underline{\tau}$. Then $M^{\mathrm{IA}} = \lambda x_{k+1}.\cdots \lambda x_n.x^{\mathrm{IAP}}(\text{thunk } {N_1}^{\mathrm{IA}}) \cdots (\text{thunk } {N_k}^{\mathrm{IA}})\,(\text{thunk } {x_{k+1}}^{\mathrm{IA}}) \cdots (\text{thunk } {x_n}^{\mathrm{IA}}) = \lambda x_{k+1}.(x\, N_1 \cdots N_k\, x_{k+1})^{\mathrm{IA}}$, as required.

The inductive cases are fairly straight forward, following mainly from the ς-rules. For example, in the case of $M = M_1; M_2$. By the I.H., we have N_2 s.t. ${M_2}^{\mathrm{IA}} =_{\beta\eta\varsigma} \lambda x.{N_2}^{\mathrm{IA}}$, so we have $M^{\mathrm{IA}} = {M_1}^{\mathrm{IA}}; {M_2}^{\mathrm{IA}} =_{\beta\eta\varsigma} {M_1}^{\mathrm{IA}}; \lambda x.{N_2}^{\mathrm{IA}} =_{\varsigma} \lambda x.({M_1}^{\mathrm{IA}}; {N_2}^{\mathrm{IA}}) = \lambda x.(M_1; N_2)^{\mathrm{IA}}$. The interesting case is $M = M_1\, M_2$. By the I.H., there exists N' s.t. ${M_1}^{\mathrm{IA}} =_{\beta\eta\varsigma} \lambda y.{N'}^{\mathrm{IA}}$. By the I.H., there exists N'' s.t. ${N'}^{\mathrm{IA}} =_{\beta\eta\varsigma} \lambda x.{N''}^{\mathrm{IA}}$. Thus, we have $M^{\mathrm{IA}} =_{\beta\eta\varsigma} (\lambda y.\lambda x.{N''}^{\mathrm{IA}})\text{thunk } {M_2}^{\mathrm{IA}} =_\beta \lambda x.{N''}^{\mathrm{IA}}\{\text{thunk } {M_2}^{\mathrm{IA}}/y\} =_\beta \lambda x.({N''}^{\mathrm{IA}}\text{thunk } {M_2}^{\mathrm{IA}}) = \lambda x.(N''\, M_2)^{\mathrm{IA}}$, as required. □

The following result, a kind of substitution lemma, is useful in what follows.

Lemma 136. *Let $\Gamma, x : \underline{\tau_N} \vdash M : \underline{\tau_M}$ and $\Gamma \vdash N : \underline{\tau_N}$ be* IA*-terms, then $\Gamma \vdash M^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\} \cong^{\mathrm{CBPV}}_{ter} (N\{M/x\})^{\mathrm{IA}}$.*

Sketch. This proof proceeds by induction on the type of x (its IA-type in M, $\underline{\tau_N}$) and secondarily on the structure of M.

- The base case is that $x : \underline{\mathrm{expr}}$, $x : \underline{\mathrm{com}}$, or $x : \underline{\mathrm{var}}$. For the induction on structure, the base case is that $M = x$. In the first two cases, $x^{\mathrm{IA}} = \text{force } x$, so we get $x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\} = \text{force thunk } N^{\mathrm{IA}} =_\eta N^{\mathrm{IA}}$. In the other case we have

$$\begin{array}{rcl} x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\} & = & \lambda mode.\lambda w.\text{case } mode \text{ of } ((assert(w \sim 0); (\text{force thunk } N^{\mathrm{IA}})\widehat{0}\widehat{0}), \\ & & \qquad ((\text{force thunk } N^{\mathrm{IA}})\widehat{1}w \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}), (\Omega)_{i>1}) \\ & =_\beta & \lambda mode.\lambda w.\text{case } mode \text{ of } ((assert(w \sim 0); (N^{\mathrm{IA}})\widehat{0}\widehat{0}), \\ & & \qquad ((N^{\mathrm{IA}})\widehat{1}w \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}), (\Omega)_{i>1}) \end{array}$$

  Now, by Lemma 3, it suffices to check CIU equivalence. So taken any $\Sigma \vdash K : \underline{\tau_M}^{\mathrm{IA}} \implies F\sigma,\ \Sigma \vdash \gamma : \Gamma^{\mathrm{IA}}$, and $h : \Sigma$. By the types, $K = K'[\bullet\, \widehat{j}\, \widehat{m}]$. We wish to show that $(K[N^{\mathrm{IA}}\{\gamma\}], h) \Downarrow_{ter}$ iff $(K[x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\}\{\gamma\}], h) \Downarrow_{ter}$.

  Going right-to-left, for the above it is immediate that $(K[x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\}\{\gamma\}], h) \Downarrow_{ter}$ only if $j = 0$ and $m = 0$ or $j = 1$. In the first case, we have $(K[x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\}\{\gamma\}], h) \to (K[N^{\mathrm{IA}}\{\gamma\}], h)$, and so are done. In the second case $(K[x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\}\{\gamma\}], h) \to (K'[N^{\mathrm{IA}}\{\gamma\}\widehat{1}\widehat{m} \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}], h)$. For this to terminate, it follows that $(N^{\mathrm{IA}}\{\gamma\}\widehat{1}\widehat{m}, h) \to (\text{return } \widehat{0}, h')$ and $(N^{\mathrm{IA}}\{\gamma\}\widehat{1}\widehat{m} \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}, h) \to (\text{return } \widehat{0}, h')$. Thus $(K[N^{\mathrm{IA}}\{\gamma\}], h) \Downarrow_{ter}$.

  Now going left-to-right, by Lemma 134, $(K[N^{\mathrm{IA}}\{\gamma\}], h) \Downarrow_{ter}$ implies that $j = 0$ and $m = 0$, or $j = 1$, and if $j = 1$, then $(N^{\mathrm{IA}}\{\gamma\}, h) \to (\text{return } \widehat{0}, h')$. If $j = 0$ and $m = 0$, then $(K[x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\}\{\gamma\}], h) \to (K[N^{\mathrm{IA}}\{\gamma\}], h)$, and so are done. If $j = 1$ then $(K[x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\}\{\gamma\}], h) \to (K'[N^{\mathrm{IA}}\{\gamma\}\widehat{1}\widehat{m} \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}], h)$, and as $(N^{\mathrm{IA}}\{\gamma\}\widehat{1}\widehat{m} \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}, h) \to (\text{return } \widehat{0}, h')$, we obtain $(K[x^{\mathrm{IA}}\{\text{thunk } N^{\mathrm{IA}}/x\}\{\gamma\}], h)$.

  The inductive cases on the structure of M are then straight-forward.

- The inductive case is that $x : \underline{\tau_1} \to \cdots \underline{\tau_n} \to \underline{\mathrm{expr}}$, $x : \underline{\tau_1} \to \cdots \underline{\tau_n} \to \underline{\mathrm{com}}$, or $x : \underline{\tau_1} \to \cdots \underline{\tau_n} \to \underline{\mathrm{var}}$.

For the inductive proof on the structure of M, we reduce the problem to showing that the equivalence holds when the right-most occurance of x is substituted. This means that the base case is $M = x\,N_1 \cdots N_k$, where x not free in N_i and $0 \leq k \leq n$. We have

$$M^{\mathrm{IA}} = \lambda x_{k+1} \cdots \lambda x_n.x^{\mathrm{IAP}}\,\mathrm{thunk}\ {N_1}^{\mathrm{IA}} \cdots \mathrm{thunk}\ {N_k}^{\mathrm{IA}} \mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}} \cdots \mathrm{thunk}\ {x_n}^{\mathrm{IA}}$$

for fresh x_i, and $(M\{N/x\})^{\mathrm{IA}} = N^{\mathrm{IA}}\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}})$.

For the first two cases, we have

$$\begin{array}{rcl} M^{\mathrm{IA}}\{\mathrm{thunk}\ N^{\mathrm{IA}}/x\} & = & (\lambda x_{k+1} \cdots \lambda x_n.\mathrm{force}\ x\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}}) \\ & & \qquad (\mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}}) \cdots ({x_n}^{\mathrm{IA}}))\{\mathrm{thunk}\ N^{\mathrm{IA}}/x\} \\ & = & \lambda x_{k+1} \cdots \lambda x_n.\mathrm{force}\ \mathrm{thunk}\ N^{\mathrm{IA}}\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}}) \\ & & \qquad (\mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {x_n}^{\mathrm{IA}}) \\ & =_\beta & \lambda x_{k+1} \cdots \lambda x_n.N^{\mathrm{IA}}\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}}) \\ & & \qquad (\mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {x_n}^{\mathrm{IA}}) \end{array}$$

By Lemma 135, we have that there exists some N' s.t. $\Gamma \vdash N^{\mathrm{IA}} \cong^{\mathrm{CBPV}}_{ter} \lambda x_1.\cdots\lambda x_n.{N'}^{\mathrm{IA}}$. Thus, we have that

$$\begin{array}{lll} \Gamma \vdash & M^{\mathrm{IA}}\{\mathrm{thunk}\ N^{\mathrm{IA}}/x\} & \\ =_\beta & \lambda x_{k+1} \cdots \lambda x_n.N^{\mathrm{IA}}\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}})(\mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {x_n}^{\mathrm{IA}}) & \\ \cong^{\mathrm{CBPV}}_{ter} & \lambda x_{k+1} \cdots \lambda x_n.(\lambda x_1.\cdots\lambda x_n.{N'}^{\mathrm{IA}})\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}}) & \\ & \qquad (\mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {x_n}^{\mathrm{IA}}) & \text{by above} \\ =_\beta & \lambda x_{k+1} \cdots \lambda x_n.({N'}^{\mathrm{IA}}\{\mathrm{thunk}\ {N_1}^{\mathrm{IA}}/x_1\} \cdots \{\mathrm{thunk}\ {N_k}^{\mathrm{IA}}/x_k\} & \\ & \qquad \{\mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}}/x_{k+1}\} \cdots \{\mathrm{thunk}\ {x_n}^{\mathrm{IA}}/x_n\}) & \\ \cong^{\mathrm{CBPV}}_{ter} & \lambda x_{k+1} \cdots \lambda x_n.((N'\{N_1/x_1\})^{\mathrm{IA}}\{\mathrm{thunk}\ {N_2}^{\mathrm{IA}}/x_2\} \cdots \{\mathrm{thunk}\ {N_k}^{\mathrm{IA}}/x_k\} & \\ & \qquad \{\mathrm{thunk}\ {x_{k+1}}^{\mathrm{IA}}/x_{k+1}\} \cdots \{\mathrm{thunk}\ {x_n}^{\mathrm{IA}}/x_n\}) & \text{by I.H} \\ \cong^{\mathrm{CBPV}}_{ter} & \lambda x_{k+1} \cdots \lambda x_n.((N'\{N_1/x_1\} \cdots \{N_k/x_k\}\{x_{k+1}/x_{k+1}\} \cdots \{x_n/x_n\})^{\mathrm{IA}}) & \text{by I.H} \\ = & \lambda x_{k+1} \cdots \lambda x_n.((N'\{N_1/x_1\} \cdots \{N_k/x_k\})^{\mathrm{IA}}) & \\ \cong^{\mathrm{CBPV}}_{ter} & \lambda x_{k+1} \cdots \lambda x_n.({N'}^{\mathrm{IA}}\{\mathrm{thunk}\ {N_1}^{\mathrm{IA}}/x_1\} \cdots \{\mathrm{thunk}\ {N_k}^{\mathrm{IA}}/x_k\}) & \text{by I.H} \\ = & (\lambda x_{k+1} \cdots \lambda x_n.{N'}^{\mathrm{IA}})\{\mathrm{thunk}\ {N_1}^{\mathrm{IA}}/x_1\} \cdots \{\mathrm{thunk}\ {N_k}^{\mathrm{IA}}/x_k\} & \\ =_\beta & (\lambda x_1.\cdots\lambda x_n.{N'}^{\mathrm{IA}})\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}}) & \\ \cong^{\mathrm{CBPV}}_{ter} & N^{\mathrm{IA}}\,(\mathrm{thunk}\ {N_1}^{\mathrm{IA}}) \cdots (\mathrm{thunk}\ {N_k}^{\mathrm{IA}}) & \text{by above} \\ = & (M\{N/x\})^{\mathrm{IA}} & \end{array}$$

For the case of $x : \underline{\tau_1} \to \cdots \underline{\tau_n} \to \underline{\mathrm{var}}$, the reasoning is the obvious combination of the preceeding and the reasoning for the case of $x : \underline{\mathrm{var}}$.

The inductive cases on the structure of M are again straight-forward.

□

We need to generalise the translations to be applicable to contexts. This is simple, we simply define $\bullet^{\mathrm{RML}} = \bullet$ and $\bullet^{\mathrm{IA}} = \bullet$.

Lemma 137.
- *For* RML *term* $\Gamma \vdash M : \sigma$ *and context* $\vdash C : \sigma \implies \sigma'$*, then* $(C[M])^{\mathrm{RML}} = C^{\mathrm{RML}}[M^{\mathrm{RML}}]$.
- *For* IA *term* $\Gamma \vdash M : \underline{\tau}$ *and context* $\vdash C : \underline{\tau} \implies \underline{\mathrm{com}}$*, then* $(C[M])^{\mathrm{IA}} \cong^{\mathrm{CBPV}}_{ter} C^{\mathrm{IA}}[M^{\mathrm{IA}}]$.

Proof. By induction on the structure of contexts. For IA, the resort to contextual equivalence arises due to the base case that M is not a fully applied variable, where we need to appeal to Lemma 136. □

Lemma 138.
- *for* RML *context* C*,* C *is an* RML *evaluation context iff* C^{RML} *is a* CBPV *evaluation context.*

- *for* IA *context* C, C *is an* IA *evaluation context iff* C^{IA} *is a* CBPV *evaluation context.*

Proof. By induction on the structure of contexts. □

We are now ready to establish that termination is preserved and reflected under the translations.

Lemma 139.
- *For closed* RML*-term* M, $(M, \emptyset) \Downarrow_{ter}^{\mathrm{RML}}$ *iff* $(M^{\mathrm{RML}}, \emptyset) \Downarrow_{ter}$.
- *For closed* IA*-term* M, $(M, \emptyset) \Downarrow_{ter}^{\mathrm{IA}}$ *iff* $(M^{\mathrm{IA}}, \emptyset) \Downarrow_{ter}$.

Sketch. This can be obtained by following the style of proof given by Levy in his thesis [29, Appendix A]. The RML-case is more straight-forward, and involves proving a simple substitution Lemma. In the IA case, we will to use Lemma 136 to handle substitutions.

To be concrete, we will sketch the proof for IA. We have to prove the more general result that for IA-term $\Sigma \vdash M : \underline{\tau}$, for any heap $h : \Sigma$, then $(M, h) \Downarrow_{ter}^{\mathrm{IA}}$ iff $(M^{\mathrm{IA}}, h) \Downarrow_{ter}$.

Now, observe that if N be the η-long form of M, $(M, h) \Downarrow_{ter}^{\mathrm{IA}}$ iff $(N, h) \Downarrow_{ter}^{\mathrm{IA}}$, and $M^{\mathrm{IA}} = N^{\mathrm{IA}}$. Thus, W.L.O.G assume that M is in η-long form, so $M^{\mathrm{IA}} = M^{\mathrm{IAP}}$.

We first prove left-to-right, so assume that $(M, h) \Downarrow_{ter}^{\mathrm{IA}}$. We proceed by an induction on the length of the reduction sequence from (M, h). In the base case, M is terminal, so has the form skip,$\widehat{n}$, ℓ, $\lambda x.N$, or MkVar N_1 N_2. It is trival to check that M^{IA} is terminal.

In the inductive cases we have that $(M, h) \to_{\mathrm{IA}} (M', h')$ where $(M', h') \Downarrow_{ter}^{\mathrm{IA}}$ in one fewer steps. The proof then proceeds on the rules for $\to_{\mathrm{IA}}$. We will show two of the interesting cases.

- $M = K[(\lambda x.M_1)\, M_2]$, $M' = K[M_1\{M_2/x\}]$ and $h = h'$. Now, $M^{\mathrm{IA}} = K^{\mathrm{IA}}[(\lambda x.{M_1}^{\mathrm{IA}})\, (\mathrm{thunk}\ {M_2}^{\mathrm{IA}})]$. By Lemma 138, K^{IA} is a reduction context, so $(M^{\mathrm{IA}}, h) \to (K^{\mathrm{IA}}[{M_1}^{\mathrm{IA}}\{\mathrm{thunk}\ {M_2}^{\mathrm{IA}}/x\}], h)$. Now, by Lemmata 136 and 137, $\vdash K^{\mathrm{IA}}[{M_1}^{\mathrm{IA}}\{\mathrm{thunk}\ {M_2}^{\mathrm{IA}}/x\}] \cong_{ter}^{\mathrm{CBPV}} K^{\mathrm{IA}}[(M_1\{M_2/x\})^{\mathrm{IA}}] \cong_{ter}^{\mathrm{CBPV}} {M'}^{\mathrm{IA}}$. Thus by I.H., $(K^{\mathrm{IA}}[{M_1}^{\mathrm{IA}}\{\mathrm{thunk}\ {M_2}^{\mathrm{IA}}/x\}], h) \Downarrow_{ter}$, so $(M^{\mathrm{IA}}, h) \Downarrow_{ter}$.

- $M = K[\ell := \widehat{n}]$, $M' = K[\mathrm{skip}]$, $h' = h[\ell \mapsto \widehat{n}]$.
  Now, $M^{\mathrm{IA}} = K^{\mathrm{IA}}[(\lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); !\ell), (\ell := w; \mathrm{return}\ \widehat{0}), (\Omega)_{i>1}))\, \widehat{1}\, \widehat{n}\ \mathrm{to}\ y.assert(y \sim 0)]$. By Lemma 138, K^{IA} is a reduction context, so

  $$(M^{\mathrm{IA}}, h) \to^* (K^{\mathrm{IA}}[\ell := \widehat{n}; \mathrm{return}\ \widehat{0}\ \mathrm{to}\ y.assert(y \sim 0)], h)$$
  $$\to (K^{\mathrm{IA}}[\mathrm{return}\ (); \mathrm{return}\ \widehat{0}\ \mathrm{to}\ y.assert(y \sim 0)], h[\ell \mapsto \widehat{n}]) \to^* (K^{\mathrm{IA}}[\mathrm{return}\ ()], h[\ell \mapsto \widehat{n}])$$

  As ${M'}^{\mathrm{IA}} = K^{\mathrm{IA}}[\mathrm{return}\ ()]$, by the I.H., we obtain that $(K^{\mathrm{IA}}[\mathrm{return}\ ()], h[\ell \mapsto \widehat{n}]) \Downarrow_{ter}$ and so $(M^{\mathrm{IA}}, h) \Downarrow_{ter}$.

We now turn to the right-to-left implication, so assume that $(M^{\mathrm{IA}}, h) \Downarrow_{ter}$. We proceed by an induction on the length of the reduction sequence from (M^{IA}, h).

In the base case, M^{IA} is terminal, so has the form return V or $\lambda x.M'$. By considering $-^{\mathrm{IAP}}$, it must have the form return (), return $\widehat{n}$, $\lambda x.N^{\mathrm{IA}}$, $\lambda mode.\lambda w.$
case $mode$ of $((assert(w \sim 0); !\ell), (\ell := w; \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})$, or $\lambda mode.\lambda w.\mathrm{case}\ mode$
of $((assert(w \sim 0); {N_1}^{\mathrm{IA}}), ({N_2}^{\mathrm{IA}}(\mathrm{thunk\ return}\ w); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})$ with M being skip, $\widehat{n}$, $\lambda x.N$, ℓ, or MkVar N_1 N_2, respectively, which are all terminal.

For the inductive case, we have that $(M^{\mathrm{IA}}, h) \to (M', h)$ where $(M', h') \Downarrow_{ter}^{\mathrm{IA}}$ in one fewer steps. The proof then proceeds on the rules for $\to$. We will show three cases.

- $M^{\mathrm{IA}} = K[(\lambda x.N)\, V_2]$, where $V_2 : U\underline{\tau}$, $M' = K[N\{V_2/x\}]$, and $h = h'$. By considering $-^{\mathrm{IAP}}$, we can observe that there must exist K_I s.t. $K = {K_I}^{\mathrm{IA}}$, M_1 s.t. $N = {M_1}^{\mathrm{IA}}$, and M_2 s.t. $V_2 = \mathrm{thunk}\ {M_2}^{\mathrm{IA}}$. By Lemma 138, K_I is a reduction context, so $(M, h) \to_{\mathrm{IA}} (K_I[M_1\{M_2/x\}], h)$. Now, by Lemmata 137 and 136, $\vdash (K_I[M_1\{M_2/x\}])^{\mathrm{IA}} \cong_{ter}^{\mathrm{CBPV}} {K_I}^{\mathrm{IA}}[(M_1\{M_2/x\})^{\mathrm{IA}}] \cong_{ter}^{\mathrm{CBPV}} M'$. By the I.H., $((K_I[M_1\{M_2/x\}])^{\mathrm{IA}}, h) \Downarrow_{ter}$, so $(M^{\mathrm{IA}}, h) \Downarrow_{ter}$.

- $M^{\mathrm{IA}} = K[(\lambda mode.N)\widehat{1}\widehat{n}]$, $M' = K[N\{\widehat{1}/mode\}]$, and $h = h'$. By consideration of $-^{\mathrm{IAP}}$, $N = \lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); !\ell), (\ell := w; \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})$ or

$$\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); M^{\mathrm{IA}}), (N^{\mathrm{IA}}(\mathrm{thunk\ return}\ w); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1})$$

and there exists K_I s.t. $K = {K_I}^{\mathrm{IA}}[\bullet\ \mathrm{to}\ y.assert(y \sim 0)]$. We will consider the first case. Thus, $M = K[\ell := \widehat{n}]$. By Lemma 138, K_I is a reduction context, so $(M, h) \to_{\mathrm{IA}} (K_I[\mathrm{skip}], h[\ell \mapsto \widehat{n}])$. Now, in this case we have

$$\begin{aligned}&(M^{\mathrm{IA}}, h) \to^* ({K_I}^{\mathrm{IA}}[\ell := \widehat{n}; \mathrm{return}\ \widehat{0}\ \mathrm{to}\ y.assert(y \sim 0)], h)\\ &\to ({K_I}^{\mathrm{IA}}[\mathrm{return}\ (); \mathrm{return}\ \widehat{0}\ \mathrm{to}\ y.assert(y \sim 0)], h[\ell \mapsto \widehat{n}]) \to^* ({K_I}^{\mathrm{IA}}[\mathrm{return}\ ()], h[\ell \mapsto \widehat{n}])\end{aligned}$$

As $(K_I[\mathrm{skip}])^{\mathrm{IA}} = {K_I}^{\mathrm{IA}}[\mathrm{return}\ ()]$, by the I.H., we obtain that $(K_I[\mathrm{skip}], h[\ell \mapsto \widehat{n}]) \Downarrow_{ter}$ and so $(M, h) \Downarrow_{ter}$.

- $M^{\mathrm{IA}} = K[\mathrm{return}\ \widehat{n}\ \mathrm{to}\ x.N]$, $M' = K[N\{\widehat{n}/x\}]$, and $h = h'$. By considering $-^{\mathrm{IAP}}$, K_I s.t. $K = {K_I}^{\mathrm{IA}}$, and that N must have the form case x of $({N_i}^{\mathrm{IA}})_{i\in I}$. In this case we must have $M = \mathrm{case}\ \widehat{n}\ \mathrm{of}\ (N_i)_{i\in I}$. Thus $(M, h) \to_{\mathrm{IA}} (K_I[N_n], h)$. We also have that $(M^{\mathrm{IA}}, h) \to (K[\mathrm{case}\ \widehat{n}\ \mathrm{of}\ ({N_i}^{\mathrm{IA}})_{i\in I}], h) \to (K[{N_i}^{\mathrm{IA}}], h)$, Thus, by Lemma 137 and the I.H., we are done.

□

Lemma 140 (Soundness)**.** - *Let* $\Gamma \vdash M_1, M_2 : \sigma$ *be* RML*-terms. If* $\Gamma^{\mathrm{RML}} \vdash {M_1}^{\mathrm{RML}} \lesssim_{ter}^{\mathrm{CBPV}} {M_2}^{\mathrm{RML}}$*, then* $\Gamma \vdash M_1 \lesssim_{ter}^{\mathrm{RML}} M_2$.

- *Let* $\Gamma \vdash M_1, M_2 : \underline{\tau}$ *be* IA*-terms. If* $\Gamma^{\mathrm{IA}} \vdash {M_1}^{\mathrm{IA}} \lesssim_{ter}^{\mathrm{CBPV}} {M_2}^{\mathrm{IA}}$*, then* $\Gamma \vdash M_1 \lesssim_{ter}^{\mathrm{RML}} M_2$.

Proof. Let T be RML or IA. Suppose that $(C[M_1], \emptyset) \Downarrow_{ter}^{T}$. Then by Lemma 139 (left-to-right), $((C[M_1])^T, \emptyset) \Downarrow_{ter}$. By Lemma 137 (left-to-right), $(C^T[M_1^T], \emptyset) \Downarrow_{ter}$, and so as $M_1^T \lesssim_{ter}^{\mathrm{CBPV}} M_2^T$, $(C^T[M_2^T], \emptyset) \Downarrow_{ter}$. By Lemma 137 (right-to-left), $((C[M_2])^T, \emptyset) \Downarrow_{ter}$, and so by Lemma 139 (right-to-left), $(C[M_1], \emptyset) \Downarrow_{ter}^{T}$ as required. □

We can now specialise our definability result when dealing with traces produced by image of RML and IA terms. For this we will need a few helper Lemmata.

We first deal with RML. We have the following result for sequencing in RML.

Lemma 141. *Let* $\mathrm{let}\, x = M \,\mathrm{in}\, N$ *be an* RML*-term. Then* $(\mathrm{let}\, x = M \,\mathrm{in}\, N)^{\mathrm{RML}} =_\beta M^{\mathrm{RML}}\ \mathrm{to}\ x.N^{\mathrm{RML}}$. *Specifically, if* $M; N$ *be an* RML*-term, then* $(M; N)^{\mathrm{RML}} =_\beta M^{\mathrm{RML}}; N^{\mathrm{RML}}$.

Proof. Recall that in RML, $\mathrm{let}\, x = M \,\mathrm{in}\, N$ syntactic sugar for $(\lambda x.N)M$. Then $(\mathrm{let}\, x = M \,\mathrm{in}\, N)^{\mathrm{RML}} = \mathrm{return}\ (\mathrm{thunk}\ \lambda x.N^{\mathrm{RML}})\ \mathrm{to}\ f.M^{\mathrm{RML}}\ \mathrm{to}\ x.\mathrm{force}\ f\ x =_\beta M^{\mathrm{RML}}\ \mathrm{to}\ x.(\lambda x.N^{\mathrm{RML}})\ x =_\beta M^{\mathrm{RML}}\ \mathrm{to}\ x.N^{\mathrm{RML}}$. □

Lemma 142. *For* RML*-terms* M, N *which are either a variable, an integer literal, a location, or* $()$*, we have the following:*

- $!M^{\mathrm{RML}} =_\beta !M$
- $M := N^{\mathrm{RML}} =_\beta M := N$
- $(\mathrm{case}\ M\ \mathrm{of}\ (M_i)_{i\in I})^{\mathrm{RML}} =_\beta \mathrm{case}\ M\ \mathrm{of}\ ({M_i}^{\mathrm{RML}})_{i_I}$

Proof. The proof of all of these is similar, so we show the first case. $(!M)^{\mathrm{RML}} = M^{\mathrm{RML}}\ \mathrm{to}\ x.!x = \mathrm{return}\ M\ \mathrm{to}\ x.!x =_\beta !M$. □

Lemma 143 (Definability for RML)**.** *Suppose $\phi \subseteq$ TNames, $c : \sigma$ is s.t. the types are in the image of $-^{\mathrm{RML}}$. Suppose t is an even-length P-visible, P-bracketed, O-visible, P-visible $(\{\circ\}, \phi \uplus \{c\})$-trace starting with an O-action, such that $t = t' \; \bar{\circ}(A)$ and t' is complete. There exists a passive configuration $\mathbf{C}$ such that $\mathbf{Tr}^{\mathrm{even}}(\mathbf{C})$ is the even-length prefixes of t (along with their renamings via permutations on* Names *that fix $\phi \uplus \{\circ\}$).*

Moreover, $\mathbf{C} = \langle \gamma^{\mathrm{RML}}, \phi \uplus \{c\}, h, [\circ \mapsto \emptyset], \phi \rangle, (c, (K^{\mathrm{RML}}, \circ)) : \bot)$ for some h, K, γ, where return $\gamma^{\mathrm{RML}}(x) = \gamma(x)^{\mathrm{RML}}$.

Proof. This proof stems form the fact the constructions in Lemma 100 when dealing with the image of RML types are themselves the image of RML terms (up to β-equality). That is, we will show that there exists K, γ so that $K^{\mathrm{RML}}, \gamma^{\mathrm{RML}}$ are equal up to β-equality to the ones found in Lemma 22 (and so produce the same trace). First, observe that for the translation of RML types, that abstract value sequences $\overrightarrow{A}$ always consist of exactly one entry. We check the constructions that are used below.

- The non-terminating term at any type $(\Omega; M)$ is, by Lemma 141, β-equal to $(\Omega; M')^{\mathrm{RML}}$, where M' is s.t. ${M'}^{\mathrm{RML}} = M$.
- The constructions for operations like + can be seen to be the translation of the RML construction.
- The construction inc *time* =!*time* to $t.(t+1)$ to t'.*time* := t' is β-equal to the translation of the term let t =!*time* in let $t' = t + \widehat{1}$ in *time* := t', by appeal to Lemmata 141 and 142.
- The construction *assert*$(x \sim A)$ for Int is β-equal to the translation of case x of $(\Omega)_{i<A}, (), (\Omega)_{A<i}$ by Lemma 142.
- The substitution $M\{\overrightarrow{x}/\overrightarrow{A}\}$ is such that if M is β-equal to the image under $-^{\mathrm{RML}}$ of an RML-term (with names substituted for free variables), then the substitution also is β-equal to the image under $-^{\mathrm{RML}}$ of an RML-term (with names substituted for free variables).
- Similarly, the substitution $A\{\gamma\}$ where γ is s.t. return $\gamma(x)$ is β-equal to the image under $-^{\mathrm{RML}}$ of an RML-term (with names substituted for free variables) then return $A\{\gamma\}$ is β-equal to the image under $-^{\mathrm{RML}}$ of an RML-term (with names substituted for free variables).
- Consider (force g) $A\{\gamma\}$ where γ is s.t. return $\gamma(x)$ is β-equal to the image under $-^{\mathrm{RML}}$ of an RML-term (with names substituted for free variables). Let M be a RML-term s.t. return $A\{\gamma\}$ is β-equal M^{RML} (with names substituted for free variables). Then (force g) $A\{\gamma\}$ is β-equal to the image under $-^{\mathrm{RML}}$ of $y\,M$, with g substituted for y.

Given the above observation, and Lemmata 141 and 142, we can see that

- the inductive construction for γ_i satisfies the property that for all $x \in \mathrm{dom}(\gamma_i)$, return $\gamma_i(x)$ is β-equal to the image under $-^{\mathrm{RML}}$ of an RML-term (with names substituted for free variables), and
- the inductive construction for S_i satisfies the property that for all contexts $K \in S_i$, K is β-equal to the image under $-^{\mathrm{RML}}$ of an RML-context (with names substituted for free variables).

Formally this can be done by a mutual induction. The desired result then follows form the case of $i = 0$. □

Lemma 144 (Completeness for RML)**.** *Let $\Gamma \vdash M_1, M_2 : \sigma$ be* RML*-terms. If $\Gamma \vdash M_1 \lesssim_{ter}^{\mathrm{RML}} M_2$, then $\Gamma^{\mathrm{RML}} \vdash {M_1}^{\mathrm{RML}} \lesssim_{ter}^{\mathrm{CBPV}} {M_2}^{\mathrm{RML}}$.*

Proof. Suppose $\Gamma \vdash M_1 \lesssim_{ter}^{\mathrm{IA}(ciu)} M_2$. Let ρ be a Γ^{RML}-configuration, $A_i = \rho(x_i)$, $c : \sigma^{\mathrm{RML}}$ and $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\rho_{\vec{A_i}},c}_{{M_1}^{\mathrm{RML}}})$ s.t. t is complete. Then t is a $(\nu(\rho) \uplus \{c\}, \emptyset)$-trace s.t. the types are in the image of $-^{\mathrm{RML}}$. W.L.O.G, due to the closure of $\mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\rho_{\vec{A_i}},c}_{{M_1}^{\mathrm{RML}}})$ under renaming, we can ensure that $\circ$ does not appear in t. Let $\circ : \mathrm{Unit}$ and $t_1 = t^{\perp}\, \bar{\circ}(())$, a $(\{\circ\}, \nu(\rho) \uplus \{c\})$-trace. Then, as t is O-visible, O-bracketed, P-visible, P-bracketed and complete, so is $t^{\perp}$ and so t_1 is. Thus, we can appeal to Lemma 143 to get a passive configuration $\mathbf{C}_O = \langle {\gamma_O}^{\mathrm{RML}}, \phi \uplus \{c\}, h, [\circ \mapsto \emptyset], \phi \rangle, (c, (K^{\mathrm{RML}}, \circ)) : \perp)$ for some $h, K^{\mathrm{RML}}, \gamma_O^{\mathrm{RML}}$, such that $\mathbf{Tr}^{\mathrm{even}}_{\mathrm{CBPV}}(\mathbf{C}_O)$ consists of all even-length prefixes of t_1, up to renamings which preserve $\nu(\rho) \uplus \{c, \circ\}$.

Observe that $\mathbf{C}_O = \mathsf{C}^{\vec{\gamma_i},c}_{h,K^{\mathrm{RML}},\gamma^{\mathrm{RML}}}$ where $\gamma(x_i) = A_i\{\gamma_O^{\mathrm{RML}}\}$ and $\gamma_i^{\mathrm{RML}} = \gamma_O^{\mathrm{RML}} \restriction \nu(A_i)$. Then we have the $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\rho_{\vec{A_i}},c}_{{M_1}^{\mathrm{RML}}})$ and $t_1 \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\vec{\gamma_i},c}_{h,K^{\mathrm{RML}},\gamma^{\mathrm{RML}}})$. We can then apply Lemma 20 (right-to-left) to obtain $(K^{\mathrm{RML}}[{M_1}^{\mathrm{RML}}]\{\gamma^{\mathrm{RML}}\}, h) \Downarrow_{ter}$. We can then define the (closed) RML context $C = \mathrm{let}\, t = \mathrm{ref}\, 0 \,\mathrm{in}\, (\mathrm{let}\, x_1 = \gamma(x_1) \,\mathrm{in} \cdots \mathrm{let}\, x_n = \gamma(x_n) \,\mathrm{in}\, K)\{t/time\}$, which is such that C^{RML} is equivalent to $\mathrm{ref}\, 0$ to $t.(\mathrm{return}\ \gamma(x_1)^{\mathrm{RML}}\ \mathrm{to}\ x. \cdots \mathrm{return}\ \gamma(x_n)^{\mathrm{RML}}\ \mathrm{to}\ x_n.K^{\mathrm{RML}})\{t/time\}$, by Lemma 141. Thus $(C^{\mathrm{RML}}[{M_1}^{\mathrm{RML}}], h) \Downarrow_{ter}$.

So by Lemma 137 (right-to-left) and Lemma 139 (right-to-left), $(C[M_1], h) \Downarrow^{\mathrm{RML}}_{ter}$. Thus by the assumption, $(C[M_2], h) \Downarrow^{\mathrm{RML}}_{ter}$, and so by Lemma 139 (left-to-right) and Lemma 137 (left-to-right) $(C^{\mathrm{RML}}[{M_2}^{\mathrm{RML}}], h) \Downarrow_{ter}$ and so $(K^{\mathrm{RML}}[{M_2}^{\mathrm{RML}}]\{\gamma^{\mathrm{RML}}\}, h) \Downarrow_{ter}$. By Lemma 20 (left-to-right), we have a complete trace $t' \in\in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\rho_{\vec{A_i}},c}_{{M_2}^{\mathrm{RML}}})$ such that ${t'}^{\perp}\, \bar{\circ}(()) \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\vec{\gamma_i},c}_{h,K^{\mathrm{RML}},\gamma^{\mathrm{RML}}})$. By the definition of $\mathbf{C}_O$, all complete traces in $\mathbf{Tr}_{\mathrm{CBPV}}(\mathbf{C}_O)$ are equal to t_1 up to renaming of names preserving $\nu(\rho) \uplus \{c, \circ\}$, so t' is equal to t (up to a renaming of names preserving $\nu(\rho) \uplus \{c\}$). Therefore, by the closure property, we have $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\rho_{\vec{A_i}},c}_{{M_2}^{\mathrm{RML}}})$.

Thus, we obtain that $\mathbf{Tr}_{\mathrm{CBPV}}(\Gamma^{\mathrm{RML}} \vdash^c {M_1}^{\mathrm{RML}}) \subseteq \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma^{\mathrm{RML}} \vdash^c {M_2}^{\mathrm{RML}})$, and so by Theorem 21 (Soundness), we have $\Gamma^{\mathrm{RML}} \vdash^c {M_1}^{\mathrm{RML}} \lesssim_{ter}^{\mathrm{CBPV}(ciu)} {M_2}^{\mathrm{RML}}$, as required. □

We can now turn to IA, where we will prove similarly prove some helpful lemmata.

Lemma 145. *For* IA*-terms M, N which are either an integer literal or a location, then we have the following:*

- $(!M)^{\mathrm{IA}} =_\beta !M$
- $(M := N)^{\mathrm{IA}} =_\beta M := N$
- $(\mathrm{case}\ M\ \mathrm{of}\ (M_i)_{i \in I})^{\mathrm{IA}} =_\beta \mathrm{case}\ M\ \mathrm{of}\ ({M_i}^{\mathrm{IA}})_{i \in I}$

Proof. We handle the first case. Recall that $!M^{\mathrm{IA}} = M^{\mathrm{IA}}\, \widehat{0}\, \widehat{0}$. If $M = \ell$, then $\ell^{\mathrm{IA}} = \lambda mode.\lambda w.\mathrm{case}\ mode$ of $((assert(w \sim 0); !\ell), (\ell := w; \mathrm{return}\ \widehat{0}), (\Omega)_i)$, so $(!M)^{\mathrm{IA}} =_\beta \mathrm{case}\ \widehat{0}$ of $((assert(\widehat{0} \sim 0); !\ell), (\ell := \widehat{0}; \mathrm{return}\ \widehat{0}), (\Omega)_i) =_\beta assert(\widehat{0} \sim 0); !\ell =_\beta !\ell$. □

Lemma 146. *Let* $\mathrm{new}\, x \,\mathrm{in}\, M$ *be an* IA*-term (in which x is not rebound in M) and all occurrences of x have the form $!x$ or $x := N$. Recall that*

$$(\mathrm{new}\, x \,\mathrm{in}\, M)^{\mathrm{IA}} = \mathrm{ref}\, \widehat{0}\ \mathrm{to}\ x'.\mathrm{let}\ x\ \mathrm{be\ thunk}\ \lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}$$
$$((assert(w \sim 0); !x'), (x' := w; \mathrm{return}\ \widehat{0}), (\Omega)_i).M^{\mathrm{IA}}$$

This is $\beta\eta\varsigma$-equivalent to $\mathrm{ref}\, \widehat{0}\ \mathrm{to}\ x'.M'$*, where M' is M^{IA} but with occurrences of $!x$ translated as $!x'$ and $x := N$ as N^{IA} to $y.x' := y$*

Proof. First, observe that $(\mathrm{new}\, x \,\mathrm{in}\, M)^{\mathrm{IA}}$ is β-equivalent to $\mathrm{ref}\, \widehat{0}\ \mathrm{to}\ x'.M^{\mathrm{IA}}\{\mathrm{thunk}\ N/x\}$ where $N = \lambda mode.\lambda w.\mathrm{case}\ mode$ of $((assert(w \sim 0); !x'), (x' := w; \mathrm{return}\ \widehat{0}), (\Omega)_(i > 1))$. Now, an occurrence of $!x$ in M becomes

$$(\lambda mode.\lambda w.\mathrm{case}\ mode\ \mathrm{of}\ ((assert(w \sim 0); (\mathrm{force}\ x)\widehat{0}\widehat{0}),$$
$$((\mathrm{force}\ x)\widehat{1}w\ \mathrm{to}\ z.assert(z \sim 0); \mathrm{return}\ \widehat{0}), (\Omega)_{i>1}))\, \widehat{0}\, \widehat{0}$$

in M^{IA} so in $M^{\mathrm{IA}}\{\text{thunk } M'/x\}$ becomes

$$(\lambda mode.\lambda w.\text{case } mode \text{ of } ((assert(w \sim 0); (\text{force thunk } M')\widehat{0}\widehat{0}),$$
$$((\text{force thunk } M')\widehat{1}w \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}), (\Omega)_{i>1}))\,\widehat{0}\,\widehat{0}$$
$$=_\beta \text{case } \widehat{0} \text{ of } ((assert(\widehat{0} \sim 0); (\text{force thunk } M')\widehat{0}\widehat{0}),$$
$$((\text{force thunk } M')\widehat{1}\widehat{0} \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}), (\Omega)_{i>1})$$
$$=_\beta assert(\widehat{0} \sim 0); (\text{force thunk } M')\widehat{0}\widehat{0} =_\beta (\text{force thunk } M')\widehat{0}\widehat{0} =_\beta M'\widehat{0}\widehat{0}$$
$$=_\beta \text{case } \widehat{0} \text{ of } ((assert(\widehat{0} \sim 0); !x'), (x' := \widehat{0}; \text{return } \widehat{0}), (\Omega)_{i>1}) =_\beta assert(\widehat{0} \sim 0); !x' =_\beta !x'$$

Similarly, an occurrence of $x := N$ in M(where N is not $\widehat{n}$) becomes

$$N^{\mathrm{IA}} \text{ to } y.(\lambda mode.\lambda w.\text{case } mode \text{ of } ((assert(w \sim 0); (\text{force } x)\widehat{0}\widehat{0}),$$
$$((\text{force } x)\widehat{1}w \text{ to } z.assert(z \sim 0)); \text{return } \widehat{0}, (\Omega)_{i>1}))\,\widehat{1}\,y \text{ to } z.assert(z \sim \widehat{0})$$

in M^{IA} so in $M^{\mathrm{IA}}\{\text{thunk } M'/x\}$ becomes

$$N^{\mathrm{IA}} \text{ to } y.(\lambda mode.\lambda w.\text{case } mode \text{ of } ((assert(w \sim 0); (\text{force thunk } M')\widehat{0}\widehat{0}),$$
$$((\text{force thunk } M')\widehat{1}w \text{ to } z.assert(z \sim 0)); \text{return } \widehat{0}, (\Omega)_{i>1}))\,\widehat{1}\,y \text{ to } z.assert(z \sim \widehat{0})$$
$$=_\beta N^{\mathrm{IA}} \text{ to } y.\text{case } \widehat{1} \text{ of } ((assert(y \sim 0); (\text{force thunk } M')\widehat{0}\widehat{0}),$$
$$((\text{force thunk } M')\widehat{1}y \text{ to } z.assert(z \sim 0)); \text{return } \widehat{0}, (\Omega)_{i>1}) \text{ to } z.assert(z \sim \widehat{0})$$
$$=_\beta N^{\mathrm{IA}} \text{ to } y.((\text{force thunk } M')\widehat{1}y \text{ to } z.assert(z \sim 0)); \text{return } \widehat{0} \text{ to } z.assert(z \sim \widehat{0})$$
$$=_\beta N^{\mathrm{IA}} \text{ to } y.((M')\widehat{1}y \text{ to } z.assert(z \sim 0)); \text{return } \widehat{0} \text{ to } z.assert(z \sim \widehat{0})$$
$$=_\beta N^{\mathrm{IA}} \text{ to } y.((\text{case } \widehat{1} \text{ of } ((assert(w \sim 0); !x'), (x' := y; \text{return } \widehat{0}), (\Omega)_{i>1}))$$
$$\text{to } z.assert(z \sim 0)); \text{return } \widehat{0} \text{ to } z.assert(z \sim \widehat{0})$$
$$=_\beta N^{\mathrm{IA}} \text{ to } y.((x' := y; \text{return } \widehat{0}) \text{ to } z.assert(z \sim 0)); \text{return } \widehat{0} \text{ to } z.assert(z \sim \widehat{0})$$
$$=_\varsigma N^{\mathrm{IA}} \text{ to } y.x' := y; \text{return } \widehat{0} \text{ to } z.assert(z \sim 0); \text{return } \widehat{0} \text{ to } z.assert(z \sim \widehat{0})$$
$$=_\beta N^{\mathrm{IA}} \text{ to } y.x' := y; assert(\widehat{0} \sim 0); assert(\widehat{0} \sim 0) =_\eta N^{\mathrm{IA}} \text{ to } y.x' := y$$

The case where $N = \widehat{n}$ is analogous. □

Definition 147. A $(\phi \uplus \{c\}, \emptyset)$-trace t is an IA trace if $\phi \subseteq \mathrm{TNames}$, $c : \sigma$ is s.t. the types are in the image of $-^{\mathrm{IA}}$, and for $f : U(U_{\underline{\tau}_1} \to \cdots \to U_{\underline{\tau}_k} \to \mathrm{Int} \to \mathrm{Int} \to F\mathrm{Int})$ (where k can be 0):

- Any OQ-action in t with head name f has the form $f(A_1 \cdots A_k\,\widehat{0}\,\widehat{0}, c)$ or $f(A_1 \cdots A_k\,\widehat{1}\widehat{n}, c)$;
- Any PQ-action in t with head name f has the form $\bar{f}(A_1 \cdots A_k\,\widehat{0}\widehat{0}, c)$ or $\bar{f}(A_1 \cdots A_k\,\widehat{1}\widehat{n}, c)$;
- Any OA-action answering a PQ-action of the form $\bar{f}(A_1 \cdots A_k\,\widehat{1}\,\widehat{n}, c)$ has the form $c(\widehat{0})$;
- Any PA-action answering a OQ-action of the form $f(A_1 \cdots A_k\,\widehat{1}\,\widehat{n}, c)$ has the form $\bar{c}(\widehat{0})$.

Lemma 148. *Let $\Gamma \vdash^c N : F\sigma$ be s.t. $\Gamma, N, F\sigma$ are the image of an* IA *term under $-^{\mathrm{IA}}$. Let ρ be a Γ-assignment, and c a continuation name. Then complete trace $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_M^{\rho,c})$ is an* IA*-trace.*

Proof. We will first consider the case of an OQ-action. Suppose that $f(\overrightarrow{A}\,\widehat{j}\,\widehat{n}, c)$ is in t. Now, consider $\mathbf{C} = (\langle \gamma, \phi, h, H, Fn\rangle, S)$ as being the configuration immediately before $f(\overrightarrow{A}\,\widehat{j}\,\widehat{n}, c)$ in the path p generating t. We have that $\gamma(f) = N$, where N is equivalent to some N'^{IA} with names substituted for free variables (which we can prove by induction on paths, using Lemma 136

to handle OQ-actions). The term component of the configuration after $\mathbf{C}'$ is $N\,\overrightarrow{A}\,\widehat{j}\,\widehat{n}$, so by consideration similar to Lemma 134, we can determine that p can only produce the answering PA-action if $j = 0$ and $n = 0$, or $j = 1$ (as we would otherwise reach a point where the term component is non-terminating). As t is complete, it must be the case $j = 0$ and $n = 0$, or $j = 1$. Further more, if $j = 1$, it must be the case that the configuration before the answering PA-action in p is return $\widehat{0}$, and so the answering PA-action is $\bar{c}(\widehat{0})$.

Now, consider the case of a PQ-action. Suppose that $f(\overrightarrow{A}\,\widehat{j}\,\widehat{n}, c)$ is in t. Then let $\mathbf{C} = (\langle N, c', \gamma, \phi, h, H\rangle, S)$ as being the configuration immediately before $f(\overrightarrow{A}\,\widehat{j}\,\widehat{n}, c)$ in the path p generating t. It must be the case that $N = K[(\text{force } f)\overrightarrow{V}\,\widehat{j}\,\widehat{n}]$. By considering the types, we observe that this must result from the translation of IA term with the form $!N$, $N_1 := N_2$, or a varible x. As $(!N)^{\mathrm{IA}} = N^{\mathrm{IA}}\,\widehat{0}\,\widehat{0}$, $(N_1 := N_2)^{\mathrm{IA}} = {N_2}^{\mathrm{IA}}$ to $x.{N_1}^{\mathrm{IA}}\,\widehat{1}\,x$ to $z.assert(z \sim 0)$, and for a suitable x, $\lambda y_1.\cdots\lambda y_k.\lambda mode.\lambda w.\text{case } mode \text{ of } ((assert(w \sim 0); (\text{force } x)y_1\cdots y_k\widehat{0}\widehat{0}), ((\text{force } x)y_1\cdots y_k\widehat{1}w \text{ to } z.assert(z \sim 0); \text{return } \widehat{0}), (\Omega)_{i>1})$ we can conclude that $j = 0$ and $n = 0$, or $j = 1$.

Now suppose that $j = 1$. Then it must be that $K = K'[\bullet \text{ to } z.assert(z \sim 0)]$. Let $c(\widehat{m})$ be the answering action in t, and $\mathbf{C}'$ the configuration after this action in the path p generating t. It follows from bracketing that the term component of $\mathbf{C}'$ is $K'[\text{return } \widehat{m} \text{ to } z.assert(z \sim 0)]$. Therefore, before the next PA or PQ action, p must reach a configuration with term component $K'[assert(m \sim 0)]$. For t to be complete, $assert(m \sim 0)$ must terminate, and so $m = 0$. □

Lemma 149 (Definability for IA)**.** *Suppose t is an even-length P-visible, P-bracketed, O-visible, P-visible $(\{\circ\}, \phi \uplus \{c\})$-trace starting with an O-action, such that $t = t'^{\perp}\ \bar{\circ}(A)$ and t' is an* IA*-trace. There exists a passive configuration $\mathbf{C}$ such that $\mathbf{Tr}^{\mathrm{even}}(\mathbf{C})$ is the even-length prefixes of t (along with their renamings via permutations on* Names *that fix $\phi \uplus \{\circ\}$).*

Moreover, $\mathbf{C} = \langle \gamma^{\mathrm{IA}}, \phi \uplus \{c\}, h, [\circ \mapsto \emptyset], \phi\rangle, (c, (K^{\mathrm{IA}}, \circ)) : \perp)$ for some h, K, γ, where $\gamma^{\mathrm{IA}}(x) =$ thunk $\gamma(x)^{\mathrm{IA}}$*.*

Proof. This proof stems form the fact the constructions in Lemma 100 when considering an IA-trace are (contextually) equivalent to the image of IA terms (we names substituted for free variables). That is, we will show that there is K, γ so that $K^{\mathrm{IA}}, \gamma^{\mathrm{IA}}$ is equivalent to the one found in Lemma 22 (and so produces the same trace).

We check that the constructions we use are the image of IA terms.

- The non-terminating term at any type $(\Omega; M)$ is $(\Omega; M')^{\mathrm{IA}}$, where M' is s.t. ${M'}^{\mathrm{IA}} = M$.
- The constructions for operations like $+$ are equivalent to the translation from IA, in particular we have that $M + \widehat{n}^{\mathrm{IA}}$ is equivalent to M to $y.y + \widehat{n}$, as the implementation is done in terms of case.
- The construction inc $time$ =!$time$ to $t.(t+1)$ to $t'.time := t'$ is equivalent to the translation of the term $time$:=!$time + 1$, by appeal to Lemma 145 and the above point.
- The translation is s.t. we never have a type of the form $FU\underline{\tau}$ or $F\mathrm{Ref}$. Thus occurrences of return A in our constructions will only have A of the form $(), \widehat{n}$, in which cases return A is simply the translation of A under $-^{\mathrm{IA}}$.
- Further, in the construction of K for S_i, x will only be used in the assertion, as the restriction observed above ensures that x has type Unit or Int, so this equivalent to the translation of $assert(\bullet \sim A)$.
- The substitution $M\{\overrightarrow{x}/\overrightarrow{A}\}$ is such that if M is equivalent to the image under $-^{\mathrm{IA}}$ of an IA-term (with names substituted for free variables), then the substitution also is equivalent to the image under $-^{\mathrm{IA}}$ of an IA-term (with names substituted for free variables).
- Consider (force g) $\overrightarrow{A}\{\gamma\}$ where γ is s.t. $\gamma(x)$ is equivalent to the thunk N^{IA} for IA-term N (with names substituted for free variables). Now, it must be the case that all of the values in $\overrightarrow{A}$ have types of the form $U\underline{\tau}$, or (force g) $\overrightarrow{A}\{\gamma\} \equiv$ (force g) $\overrightarrow{A'}\{\gamma\}\,\widehat{j}\,\widehat{n}$.

In the first case, all values in $\overrightarrow{A}$ are names, and so assume it equals $f_1 \cdots f_k$. Assume N_k is an IA term s.t. $\gamma(f_i)$ is equivalent to thunk ${N_i}^{\mathrm{IA}}$ (with names substituted for free variables). Then $(y\, N_1 \cdots N_k)^{\mathrm{IA}} = (\lambda x_1. \cdots \lambda x_k.(\text{force } y)(\text{thunk } {x_1}^{\mathrm{IA}}) \cdots (\text{thunk } {x_k}^{\mathrm{IA}}))\text{thunk } {N_1}^{\mathrm{IA}} \cdots \text{thunk } {N_k}^{\mathrm{IA}}$, so by repeated appeals to Lemma 136, we obtain that this is equivalent to (force y)(thunk ${N_1}^{\mathrm{IA}}$) $\cdots$ (thunk ${N_k}^{\mathrm{IA}}$). Thus force $g\, \overrightarrow{A}\{\gamma\}$ is equivalent to $(y\, N_1 \cdots N_k)^{\mathrm{IA}}$ (with g substituted for y, and names substituted for free variables).

Consider now the second case. It follows from a similar argument to the first case that $(y\, N_1 \cdots N_k)^{\mathrm{IA}}$ is equivalent to $y^{\mathrm{IAP}}(\text{thunk } {N_1}^{\mathrm{IA}}) \cdots (\text{thunk } {N_k}^{\mathrm{IA}})$. By the fact t' is an IA-trace, we have that $j = 1$, or $j = 0$ and $n = 0$. Then we can easily check that (force y) $\overrightarrow{A'}\{\gamma\}\,\widehat{0}\,\widehat{0}$ is equivalent to $(y^{\mathrm{IAP}})\, \overrightarrow{A'}\{\gamma\}\,\widehat{0}\,\widehat{0}$, so we obtain that (force g) $\overrightarrow{A'}\{\gamma\}\,\widehat{0}\,\widehat{0}$ is equivalent to $!(y\, N_1 \cdots N_k)^{\mathrm{IA}}$ (with g substituted for y, and names substituted for some variables).

Similarly, we have that (force g) $\overrightarrow{A'}\{\gamma\}\,\widehat{1}\,\widehat{n}$ to $z.assert(z \sim 0)$ is equivalent to $(y\, N_1 \cdots N_k) := \widehat{n}^{\mathrm{IA}}$ (with g substituted for y, and names substituted for free variables). Now, recall that contexts K on the stack have the form $\bullet$ to $x.assert(x \sim 0); N$, equivalent to the image of $assert(\bullet \sim 0); N'$, where N' is equivalent to N^{IA}. Then it is the case that $K[\text{force } g\, \overrightarrow{A'}\{\gamma\}\,\widehat{1}\,\widehat{n}]$ is equivalent to $((y\, N_1 \cdots N_k) := \widehat{n}; N')^{\mathrm{IA}}$ (with g substituted for y, and names substituted for free variables).

- Observe that the construction $\gamma_i(f)$ where $f : U(U_{\underline{\tau}_1} \to \cdots \to U_{\underline{\tau}_k} \to \mathrm{Int} \to \mathrm{Int} \to F\mathrm{Int})$ is equivalent to

$$\text{thunk } (\lambda x_1. \cdots \lambda x_k.(\text{inc } time; !time \text{ to } t.\text{case } t \text{ of } (M'_j)_{0 \le j \le n})$$

where M'_j is either Ω, or of the form $\lambda mode.\lambda w.assert(mode \sim m_j); assert(w \sim n_j); N_j$, for some m_j, n_j, where N_j is the case dependent inductive construction. By the fact that t is an IA-trace, we have that $m_j = 0$ and $n_j = 0$, or $m_j = 1$.

In the case $m_j = 0$ and $n_j = 0$, then M'_j is equivalent to

$$\begin{aligned}
&\lambda mode.\lambda w.(\text{case } mode \text{ of } (\text{return } ()), (\Omega)_{i>0}); assert(w \sim 0); N_j \\
&\quad =_\varsigma \lambda mode.\lambda w.\text{case } mode \text{ of } (\text{return } (); assert(w \sim 0); N_j), (\Omega)_{i>0} \\
&\quad =_\beta \lambda mode.\lambda w.\text{case } mode \text{ of } (assert(w \sim 0); N_j), (\Omega)_{i>0}
\end{aligned}$$

Then if N_j is equivalent to $(N'_j)^{\mathrm{IA}}$ for IA-term N'_j (with names substituted for free variables), then M_j is equivalent to $(\text{MkVar } N'_j\ \Omega)^{\mathrm{IA}}$ (with name substituted for free variables).

In the second case, then M'_j is equivalent to

$$\begin{aligned}
&\lambda mode.\lambda w.(\text{case } mode \text{ of } (\Omega), (\text{return } ()), (\Omega)_{i>1}); assert(w \sim n_j); N_j \\
&\quad =_\varsigma \lambda mode.\lambda w.\text{case } mode \text{ of } (\Omega), (\text{return } (); assert(w \sim n_j); N_j), (\Omega)_{i>1} \\
&\quad =_\beta \lambda mode.\lambda w.\text{case } mode \text{ of } (\Omega), (assert(w \sim n_j); N_j), (\Omega)_{i>1} \\
&\quad =_\beta \lambda mode.\lambda w.\text{case } mode \text{ of } (\Omega), \\
&\qquad ((\lambda x.\text{force } x \text{ to } x'.assert(w' \sim n_j))(\text{thunk return } w); N_j), (\Omega)_{i>1}
\end{aligned}$$

Now, by the fact t is an IA-trace, we can conclude that $N_j; \text{return } \widehat{0}$ is equivalent to N_j, as it must be that case that N_j terminates, it returns 0. Then if N_j is equivalent to $(N'_j)^{\mathrm{IA}}$ for IA-term N'_j (with names substituted for free variables), then M_j is equivalent to $(\text{MkVar } \Omega\ (\lambda x.assert(x \sim n_j); N'_j))^{\mathrm{IA}}$ (with name substituted for free variables).

Given the above observation, and Lemma 145, we can see that

- the inductive construction for γ_i satisfies the property that for all $x \in \mathrm{dom}(\gamma_i)$, $\gamma_i(x)$ is equivalent to thunk M^{IA} for an IA-term M (with names substituted for free variables), and
- the inductive construction for S_i satisfies the property that for all contexts $K \in S_i$, K is equivalent to the image under $-^{\mathrm{IA}}$ of an IA-context (with names substituted for free variables).

Formally this can be done by a mutual induction. The desired result then follows form the case of $i = 0$, so taking $\gamma = \gamma_0$ and K s.t $S_0 = (c, (K, \circ)) : \bot$. □

For IA, we must handle the fact that the image of the translation of types includes those of the from $\sigma_1 \to \cdots \to \sigma_k \to F\sigma$, for which we cannot directly use our trace model. The following Lemma allows us to resolve this.

Lemma 150. *Let* $\Gamma \vdash M\underline{\tau} \to \underline{\tau'}$. *Then* $\Gamma, (x, \sigma) \vdash (M\,x)^{\mathrm{IA}} \cong^{\mathrm{CBPV}}_{ter} M^{\mathrm{IA}}\,x$.

Proof. By Lemma 135, $M^{\mathrm{IA}} =_{\beta\eta\varsigma} \lambda x.N^{\mathrm{IA}}$. Thus, $(M\,x)^{\mathrm{IA}} =_{\beta\eta\varsigma} N^{\mathrm{IA}}\{\mathrm{thunk}\,x^{\mathrm{IA}}/x\}$ and $M^{\mathrm{IA}}\,x =_{\beta\eta\varsigma} N^{\mathrm{IA}}\{x/x\} = N^{\mathrm{IA}} = (N\{x/x\})^{\mathrm{IA}}$. We can the appeal to Lemma 136 to get the desired result. □

Lemma 151 (Completeness for IA)**.** *Let* $\Gamma \vdash M_1, M_2 : \underline{\tau}$ *be* IA*-terms. If* $\Gamma \vdash M_1 \lesssim^{\mathrm{RML}}_{ter} M_2$, *then* $\Gamma^{\mathrm{IA}} \vdash {M_1}^{\mathrm{IA}} \lesssim^{\mathrm{CBPV}}_{ter} {M_2}^{\mathrm{IA}}$.

Proof. Suppose $\Gamma \vdash M_1 \lesssim^{\mathrm{IA}(ciu)}_{ter} M_2$. Recall Lemma 4. Therefore, if ${M_i}^{\mathrm{IA}}$ has type $\sigma_1 \to \cdots \to \sigma_k \to F\sigma$, we consider $N_i = {M_i}^{\mathrm{IA}}\,y_1 \cdots y_k$ and $\Gamma' = \Gamma^{\mathrm{IA}}, (y_1, \sigma_1), \cdots, (y_k, \sigma_k)$, where $\Gamma^{\mathrm{IA}} = (x_1, \sigma'_1), \cdots, (x_m, \sigma'_m)$. Observe, because of the Lemma 150, for any IA context C, that $(C^{\mathrm{IA}}[(M_i\,y_1 \cdots y_n)^{\mathrm{IA}}], h) \Downarrow_{ter}$ iff $(C^{\mathrm{IA}}[{M_i}^{\mathrm{IA}}\,y_1 \cdots y_n], h) \Downarrow_{ter}$.

Let ρ be a Γ'-configuration, $A_i = \rho(x_i)$, $c : \sigma$ and $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{N_1}^{\rho_{\vec{A_i}}, c})$ s.t. t is complete. Then by Lemma 148, t is a $(\nu(\rho) \uplus \{c\}, \emptyset)$ IA-trace. W.L.O.G, due to the closure of $\mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{N_1}^{\rho_{\vec{A_i}}, c})$ under renaming, we can ensure that $\circ$ does not appear in t'. Let $\circ : \mathrm{Unit}$ and $t_1 = t^{\bot}\,\bar{\circ}(())$, a $(\{\circ\}, \nu(\rho) \uplus \{c\})$-trace. Then, as t is O-visible, O-bracketed, P-visible, P-bracketed and complete, so is $t^{\bot}$ and so t_1 is. Thus, we can appeal to Lemma 149 to get a passive configuration $\mathbf{C}_O = \langle {\gamma_O}^{\mathrm{IA}}, \phi \uplus \{c\}, h, [\circ \mapsto \emptyset], \phi \rangle, (c, (K^{\mathrm{IA}}, \circ)) : \bot)$ for some $h, K^{\mathrm{IA}}, \gamma_O^{\mathrm{IA}}$, such that $\mathbf{Tr}^{\mathrm{even}}_{\mathrm{CBPV}}(\mathbf{C}_O)$ consists of all even-length prefixes of t (up to renamings which preserve $\nu(\rho) \uplus \{c, \circ\}$).

Observe that $\mathbf{C}_O = \mathsf{C}^{\vec{\gamma_i}, c}_{h, K^{\mathrm{IA}}, \gamma^{\mathrm{IA}}}$ where $\gamma(x_i)^{\mathrm{IA}} = A_i\{\gamma_O^{\mathrm{IA}}\}$ and $\gamma_i = \gamma_O^{\mathrm{IA}} \restriction \nu(A_i)$. Then we have that $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{N_1}^{\rho_{\vec{A_i}}, c})$ and $t_1 \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}^{\vec{\gamma_i}, c}_{h, K^{\mathrm{IA}}, \gamma^{\mathrm{IA}}})$. We can then apply Lemma 20 (right-to-left) to obtain $(K^{\mathrm{IA}}[N_1]\{\gamma^{\mathrm{IA}}\}, h) \Downarrow_{ter}$.

Consider the case that ${M_i}^{\mathrm{IA}}$ has type $U\underline{\tau}_1 \to \cdots \to U\underline{\tau}_n \to F\sigma$ (so $k = n$). We can then define the (closed) IA context

$$C = \mathrm{new}\,t\,\mathrm{in}\,((\lambda x_1. \cdots \lambda x_m. \lambda y_1. \cdots \lambda y_n. K)\,\gamma(x_1) \cdots \gamma(x_m) \gamma(y_1) \cdots \gamma(y_n))\{t/\mathit{time}\}$$

which is such that C^{IA} is equivalent to $\mathrm{ref}\,\widehat{0}$ to $t.((\lambda x_1. \cdots \lambda x_m. K^{\mathrm{IA}})\,\gamma(x_1)^{\mathrm{IA}} \cdots \gamma(x_m)^{\mathrm{IA}})\{t/\mathit{time}\}$, by Lemma 146. Thus $(C^{\mathrm{IA}}[N_1], h) \Downarrow_{ter}$.

In the case that ${M_i}^{\mathrm{IA}}$ has type $U\underline{\tau}_1 \to \cdots \to U\underline{\tau}_n \to \mathrm{Int} \to \mathrm{Int} \to F\mathrm{Int}$ (so $k = n + 2$). Then by Lemma 134, for $(K^{\mathrm{IA}}[N_1]\{\gamma^{\mathrm{IA}}\}, h) \Downarrow_{ter}$ to terminate, we must have $\gamma^{\mathrm{IA}}(y_{n+1}) = \widehat{0}$ and $\gamma^{\mathrm{IA}}(y_{n+2}) = \widehat{0}$, or $\gamma^{\mathrm{IA}}(y_{n+1}) = \widehat{1}$. Thus we can define (closed) IA context $C = \mathrm{new}\,t\,\mathrm{in}\,(!(\lambda x_1. \cdots \lambda x_m. \lambda y_1. \cdots \lambda y_n. K)\,\gamma(x_1) \cdots \gamma(x_m)\,\gamma(y_1) \cdots \gamma(y_n))\{t/\mathit{time}\}$ or $C = \mathrm{new}\,t\,\mathrm{in}\,(((\lambda x_1. \cdots \lambda y_m. \lambda y_1. \cdots \lambda y_n. K)\,\gamma(x_1) \cdots \gamma(x_m)\,\gamma(y_1) \cdots \gamma(y_n)) := \gamma(y_{n+2}))\{t/\mathit{time}\}$ (respectively), which are such that, by Lemma 146, C^{IA} is equivalent to

$$\mathrm{ref}\,\widehat{0}\ \mathrm{to}\ t.((\lambda x_1. \cdots \lambda x_m. \lambda y_1. \cdots \lambda y_n. K^{\mathrm{IA}}[\bullet\,\gamma(y_{n+1})^{\mathrm{IA}}\,\gamma(y_{n+2})^{\mathrm{IA}}])\,\gamma(x_1)^{\mathrm{IA}} \cdots \gamma(x_m)^{\mathrm{IA}}\,\gamma(y_1)^{\mathrm{IA}} \cdots \gamma(y_n)^{\mathrm{IA}})\{t/\mathit{time}\}$$

By the observation above, $(C^{\mathrm{IA}}[(M_1\, y_1 \cdots y_n)^{\mathrm{IA}}], h) \Downarrow_{ter}$, so by Lemma 137 (right-to-left) and Lemma 139 (right-to-left), $(C[M_1\, y_1 \cdots y_n], h) \Downarrow_{ter}^{\mathrm{IA}}$. Thus by the assumption that $\Gamma \vdash M_1 \lesssim_{ter}^{\mathrm{IA}(ciu)} M_2$, we have $(C[M_2\, y_1 \cdots y_n], h) \Downarrow_{ter}^{\mathrm{IA}}$, and so by Lemma 139 (left-to-right) and Lemma 137 (left-to-right) $(C^{\mathrm{IA}}[(M_2\, y_1 \cdots y_n)^{\mathrm{IA}}], h) \Downarrow_{ter}$, and by the observation above, $(C^{\mathrm{IA}}[(M_2)^{\mathrm{IA}}\, y_1 \cdots y_n], h) \Downarrow_{ter}$. Thus we have $(K^{\mathrm{IA}}[N_2]\{\gamma^{\mathrm{IA}}\}, h) \Downarrow_{ter}$. By Lemma 20 (left-to-right), we have a complete trace $t' \in\in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{N_2}^{\rho_{\vec{A}_i}, c})$ such that $t'^{\perp}\, \overline{\circ}(()) \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{h, K^{\mathrm{IA}}, \gamma^{\mathrm{IA}}}^{\vec{\gamma}_i, c})$. By the definition of $\mathbf{C}_O$, we obtain that t' is equal to t (up to a renaming of names preserving $\nu(\rho) \uplus \{c\}$). Therefore, by the closure property, we have $t \in \mathbf{Tr}_{\mathrm{CBPV}}(\mathsf{C}_{N_2{}^{\mathrm{IA}}}^{\rho_{\vec{A}_i}, c})$.

Thus, we obtain that $\mathbf{Tr}_{\mathrm{CBPV}}(\Gamma' \vdash^c N_1) \subseteq \mathbf{Tr}_{\mathrm{CBPV}}(\Gamma' \vdash^c N_2)$, and so by Theorem 21 (Soundness), we have $\Gamma' \vdash^c N_1 \lesssim_{ter}^{\mathrm{CBPV}(ciu)} N_2$. By Lemma 4, we obtain $\Gamma^{\mathrm{IA}} \vdash^c M_1{}^{\mathrm{IA}} \lesssim_{ter}^{\mathrm{CBPV}(ciu)} M_2{}^{\mathrm{IA}}$ as required. □

Proof of Theorem 66. Follows from Lemmata 144, 151 and 140. □