# Towards a theory of bisimulation for local names

Alan Jeffrey
CTI, DePaul University
243 South Wabash Ave
Chicago IL 60604, USA
ajeffrey@cs.depaul.edu

Julian Rathke
COGS, University of Sussex
Brighton BN1 9QH, UK
julianr@cogs.susx.ac.uk

Report 2/00, January 2000

**Abstract**

Pitts and Stark have proposed the ν-calculus as a language for investigating the interaction of unique name generation and higher-order functions. They developed a sound model based on logical relations, but left completeness as an open problem. In this paper, we develop a complete model based on bisimulation for a labelled transition system semantics. We show that bisimulation is complete, but not sound, for the ν-calculus. We also show that by adding assignment to the ν-calculus, bisimulation becomes sound and complete. The analysis used to obtain this result illuminates the difficulties involved in finding fully abstract models for ν-calculus proper.

**Keywords:** semantics, bisimulation, nominal calculi.

## 1 Introduction

The use of localised information in computing is endemic. In a workplace where the prevalence of networks and mobile agents is increasing, the concepts of private and public data are of great importance, and the question of how secrecy can be maintained is a significant one. Issues of privacy are important for notions of locally defined names such as communication channels [11], references [15], encryption keys [1], or locations [4]. Gordon has described such languages as *nominal calculi* [7].

Using ideas from an unpublished manuscript by Stoughton, Pitts and Stark developed a core higher-order nominal language, the ν-calculus [13, 14, 21], by extending the simply typed λ-calculus with an abstract type of names, together with a name generator and an equality test. Even this small language allows one to model and reason about visibility and scoping quite thoroughly. In fact, the interaction between locally declared names and higher-order functions is extremely complex, and at present there is no known fully abstract model of the ν-calculus beyond first-order types.

In this paper we revisit the ν-calculus and provide a novel treatment of the semantics of local names using a labelled transition systems (lts). In order for the standard notion of weak bisimulation to be fully abstract for contextual equivalence we have to demonstrate:

- *Completeness*. We must show that contextual equivalence implies weak bisimilarity. To do this we show that each transition $\xRightarrow{\gamma}$ corresponds to a small piece of context $C_\gamma[t]$ such that $t \xRightarrow{\gamma} v$ iff $C_\gamma[t] \Rightarrow (v, \mathtt{true})$. (We call such lts's *contextual*: the notion that transition labels should correspond to small contexts appears to be folklore, and has only recently been investigated formally by Sewell [20].) This formal relationship between labelled observations and reduction in contexts yields completeness because non-bisimilar terms have a distinguishing trace of labelled actions, yielding a distinguishing context.
- For the converse, *soundness*, we must show that bisimilarity implies contextual equivalence, for which it is sufficient to demonstrate that bisimilarity is a congruence.

We note that our approach to characterising contextual equivalence is already in sharp contrast to Pitts and Stark. They propose logical relations as an operational proof technique for establishing contextual equivalence of ν-calculus terms. The logical relation can easily be construed as a form of bisimulation on an lts, but the labels which would have to be used are not contextual—this compromises completeness in order to obtain a direct proof of soundness for their technique.

In the case of the λ-calculus, we revisit Gordon [6] and Bernstein and Stark's [2] presentation of an lts semantics of the λ-calculus. Completeness is routine, and soundness follows by using Howe's [10] technique to show bisimulation to be a congruence.

For the ν-calculus, there is a simple intuitive extension of the lts for the λ-calculus to give a reasonable semantics for local names up to first-order types, but it transpires that bisimulation on this lts fails to be a congruence for types of second-order and above. We make explicit the reason for this failure and argue that the problem arises due to the paucity of observational contexts of the ν-calculus and that, by extending the base calculus with more realistic features, the problem dissolves. By 'more realistic features' we mean any side-effecting operators which have the capability to model leaking secrets.

The particular language extension we select for study in this paper is the νref-calculus, given by adding global references which store names. Leaking a secret name is implemented by assigning it to a shared reference. We consider this to be a minimal extension of the language for which our proof techniques are successful. We demonstrate that bisimulation for the extended language is actually a congruence and thus achieve a full abstraction result for contextual equivalence.

We would like to thank Karen Bernstein, Matthew Hennessy, Guy McCusker, Ian Stark and Allen Stoughton for discussions about this paper.

## 2 λ-calculus

In this section we consider a simply typed call-by-value lambda-calculus with booleans and pairing. We present the usual small-step reduction semantics, and a slight modification of Gordon's [6] and Bernstein and Stark's [2] labelled transition system semantics. We then show that bisimulation (defined on the lts semantics) is both sound and complete for contextual equivalence (defined on the reduction semantics). This is the result already shown in both [6] and [2] save for a few technical differences. We reiterate the result here for presentational purposes.

We show completeness (contextual equivalence implies bisimulation) by observing that each label of the lts semantics corresponds to a small piece of context, and so each lts transition can be matched by contextual equivalence. This is proof is inspired by Hennessy's [8] proofs for completeness of may and must testing.

Soundness (bisimulation implies contextual equivalence) follows immediately once we can show that bisimulation is a congruence. We briefly discuss Gordon's [6] presentation of Howe's technique [10] for this proof, concentrating on the aspect of the proof which is problematical for the $\nu$-calculus.

## 2.1 Syntax and type rules

The grammar of types is given by:

$$\sigma ::= \mathtt{unit} \mid \mathtt{bool} \mid \sigma \times \sigma \mid \sigma \to \sigma$$

The grammar of values is given by:

$$v ::= x \mid () \mid \mathtt{true} \mid \mathtt{false} \mid (v, v) \mid \lambda x : \sigma \,.\, t$$

where $x$ is drawn from some infinite set of variables, and the grammar of terms is given by:

$$t ::= v \mid \mathtt{if}\ v\ \mathtt{then}\ t\ \mathtt{else}\ t \mid \mathtt{fst}\ v \mid \mathtt{snd}\ v \mid vv \mid \mathtt{let}\ x\ =\ t\ \mathtt{in}\ t$$

We have taken the liberty of limiting much of the syntax to values: we can use obvious syntax sugar (in the spirit of Moggi's [12] computational monads) to extend these to terms. For example the term $(t, t')$ is defined:

$$(t, t') \overset{\text{def}}{=} \mathtt{let}\ x\ =\ t\ \mathtt{in}\ \mathtt{let}\ x'\ =\ t'\ \mathtt{in}\ (x, x')$$

The type system for this $\lambda$-calculus is standard, and is given by judgements of the form $\Gamma \vdash t : \sigma$ where $\Gamma$ is a type environment of the form $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ for disjoint $x_1, \ldots, x_n$.

## 2.2 Reduction semantics

We now consider the operational semantics of the $\lambda$-calculus, given as *small-step* reductions $t \xrightarrow{\tau} t'$. First, we give the atomic rules for reduction:

$$\begin{array}{rcl}
\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ t\ \mathtt{else}\ t' & \xrightarrow{\tau} & t \\
\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ t\ \mathtt{else}\ t' & \xrightarrow{\tau} & t' \\
\mathtt{fst}\ (v, v') & \xrightarrow{\tau} & v \\
\mathtt{snd}\ (v, v') & \xrightarrow{\tau} & v' \\
(\lambda x : \sigma \,.\, t)v & \xrightarrow{\tau} & t[v/x] \\
\mathtt{let}\ x\ =\ v\ \mathtt{in}\ t & \xrightarrow{\tau} & t[v/x]
\end{array}$$

then, following Wright and Felleisen [22] we allow reduction in any *evaluation context* given by the grammar:

$$\mathcal{E} ::= \cdot \mid \mathtt{let}\ x\ =\ \mathcal{E}\ \mathtt{in}\ t$$

with the rule:

$$\frac{t \xrightarrow{\tau} t'}{\mathcal{E}[t] \xrightarrow{\tau} \mathcal{E}[t']}$$

Let $\Rightarrow$ denote the reflexive, transitive closure of $\xrightarrow{\tau}$.

We can define the usual notion of contextual equivalence: two terms are considered equivalent whenever there is no boolean context which can distinguish between them: for terms $\Gamma \vdash t : \sigma$ and $\Gamma \vdash t' : \sigma$, we define $\Gamma \vdash t \approx_{ctx} t' : \sigma$ if for all closing contexts $\mathcal{C}$ of type $\mathtt{bool}$, we have $\mathcal{C}[t] \Rightarrow \mathtt{true}$ iff $\mathcal{C}[t'] \Rightarrow \mathtt{true}$.

## 2.3 Labelled transition system semantics

We now introduce a lts semantics for well-typed λ-calculus terms.

For closed terms, transitions are of the form $(\vdash v : \sigma) \xrightarrow{\gamma} (\vdash t : \sigma')$ where the grammar of labels is given by:

$$\gamma ::= \texttt{true} \mid \texttt{false} \mid @v \mid \texttt{copy} \mid \texttt{discard} \mid \texttt{l}.\gamma \mid \texttt{r}.\gamma$$

Atomic rules for reductions:

$$\begin{array}{lcll} (\vdash v : \texttt{bool}) & \xrightarrow{v} & (\vdash () : \texttt{unit}) & \\ (\vdash v : \sigma \to \sigma') & \xrightarrow{@v'} & (\vdash v(v') : \sigma') & (\text{where } \vdash v' : \sigma) \end{array}$$

diagonal and terminal transitions:

$$\begin{array}{lcl} (\vdash v : \sigma) & \xrightarrow{\texttt{copy}} & (\vdash (v, v) : \sigma \times \sigma) \\ (\vdash v : \sigma) & \xrightarrow{\texttt{discard}} & (\vdash () : \texttt{unit}) \end{array}$$

and inference rules for pair contexts:

$$\frac{(\vdash v_1 : \sigma_1) \xrightarrow{\gamma} (\vdash t_1 : \sigma_1')}{(\vdash (v_1, v_2) : \sigma_1 \times \sigma_2) \xrightarrow{\texttt{l}.\gamma} (\vdash (t_1, v_2) : \sigma_1' \times \sigma_2)}$$
$$\frac{(\vdash v_2 : \sigma_2) \xrightarrow{\gamma} (\vdash t_2 : \sigma_2')}{(\vdash (v_1, v_2) : \sigma_1 \times \sigma_2) \xrightarrow{\texttt{r}.\gamma} (\vdash (v_1, t_2) : \sigma_1 \times \sigma_2')}$$

recalling that $(t, v)$ and $(v, t)$ can be defined as syntax sugar using let.

We can then include the reduction semantics to give reductions of the form $t \xrightarrow{\alpha} t'$ where:

$$\alpha ::= \tau \mid \gamma$$

Define $t \overset{\alpha}{\Longrightarrow} t'$ as $t \Rightarrow v \xrightarrow{\alpha} t'$. Define $t \overset{\hat{\alpha}}{\Longrightarrow} t'$ as $t \Rightarrow t'$ when $\alpha$ is $\tau$ and $t \overset{\alpha}{\Longrightarrow} t'$ otherwise.

A *type-indexed relation* on terms $R$ is a family of relations $R_{\Gamma,\sigma}$ between type judgements $(\Gamma \vdash t : \sigma)$. We shall usually write $\Gamma \vDash t \mathrel{R} t' : \sigma$ for $(\Gamma \vdash t : \sigma) \mathrel{R_{\Gamma,\sigma}} (\Gamma \vdash t' : \sigma)$, and often elide the type information where it is obvious from context.

A *simulation* is a type-indexed relation $R$ such that the following diagram can be completed:

$$\begin{tikzcd}
(\vdash t_1 : \sigma) \arrow[r, leftrightarrow, "R"] \arrow[dd, "\alpha"'] & (\vdash t_2 : \sigma) & & (\vdash t_1 : \sigma) \arrow[r, leftrightarrow, "R"] \arrow[dd, "\alpha"'] & (\vdash t_2 : \sigma) \arrow[dd, Rightarrow, "\hat{\alpha}"'] \\
& & \text{as} & & \\
(\vdash t_1' : \sigma') & & & (\vdash t_1' : \sigma') \arrow[r, leftrightarrow, "R"] & (\vdash t_2' : \sigma')
\end{tikzcd}$$

A *strong simulation* is a type-indexed relation $R$ such that the following diagram can be completed:

$$\begin{tikzcd}
(\vdash t_1 : \sigma) \arrow[r, leftrightarrow, "R"] \arrow[dd, "\alpha"'] & (\vdash t_2 : \sigma) & & (\vdash t_1 : \sigma) \arrow[r, leftrightarrow, "R"] \arrow[dd, "\alpha"'] & (\vdash t_2 : \sigma) \arrow[dd, "\alpha"'] \\
& & \text{as} & & \\
(\vdash t_1' : \sigma') & & & (\vdash t_1' : \sigma') \arrow[r, leftrightarrow, "R"] & (\vdash t_2' : \sigma')
\end{tikzcd}$$

A (strong) bisimulation is a (strong) simulation whose inverse is a simulation.

Let $\approx$ be the largest bisimulation, and let $\sim$ be the largest strong bisimulation.

We can extend these relations from closed terms to open terms by closing with any appropriately typed values. A type-indexed relation $R$ on closed terms can be extended to a relation $R^\circ$ on open terms:

$$\begin{array}{l} \Gamma \vDash t \, R^\circ \, t' : \sigma \\ \quad \text{iff for all } \vdash [\overline{v}/\overline{x}] : \Gamma \text{ we have:} \\ \qquad \vDash (t[\overline{v}/\overline{x}]) \, R \, (t'[\overline{v}/\overline{x}]) : \sigma \end{array}$$

where we write $\vdash [\overline{v}/\overline{x}] : (\overline{x} : \overline{\sigma})$ whenever $\vdash \overline{v} : \overline{\sigma}$.

## 2.4 Example

Let *not* be defined:

$$\mathit{not} \overset{\text{def}}{=} \lambda x : \texttt{bool} \,.\, \texttt{if}\; x \;\texttt{then}\; \texttt{false}\; \texttt{else}\; \texttt{true}$$

then one sample reduction of *not* is:

$$\begin{array}{lcl} \mathit{not} & \xRightarrow{\texttt{copy}} & (\mathit{not}, \mathit{not}) \\ & \xRightarrow{\texttt{l.@true}} & (\mathit{not}(\texttt{true}), \mathit{not}) \\ & \xRightarrow{\tau} & (\texttt{false}, \mathit{not}) \\ & \xRightarrow{\texttt{r.@false}} & (\texttt{false}, \mathit{not}(\texttt{false})) \\ & \xRightarrow{\tau} & (\texttt{false}, \texttt{true}) \\ & \xRightarrow{\texttt{l.false}} & ((), \texttt{true}) \\ & \xRightarrow{\texttt{r.true}} & ((), ()) \end{array}$$

showing how *not* evaluates when applied to `true` or `false`.

## 2.5 Completeness

In this section, we shall show that bisimulation is *complete*, that is:

$$\text{if } t \approx_{ctx} t' \text{ then } t \approx^\circ t'$$

First we observe that the $\lambda$-calculus is deterministic and normalizing, and so bisimulation and trace equivalence coincide.

We then show that contextual equivalence implies trace equivalence by constructing a context $C_{\overline{\gamma}}$ for each sequence of labels $\overline{\gamma}$ so that the context induces reductions for each label:

**Lemma 2.1** *For every sequence $\overline{\gamma}$ of transition labels there is a context $C_{\overline{\gamma}}$ such that:*

$$(\vdash t : \sigma) \xRightarrow{\overline{\gamma}} \Rightarrow (\vdash v : \sigma') \quad \textit{iff} \quad (\vdash C_{\overline{\gamma}}[t] : (\sigma' \times \texttt{bool})) \Rightarrow (\vdash (v, \texttt{true}) : (\sigma' \times \texttt{bool}))$$

**Proof:** We describe the context $C_{\overline{\gamma}}$ by first giving contexts for each individual label, using some obvious syntax sugar on terms:

$$\begin{aligned}
\mathcal{C}_{\mathtt{true}}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,x \;=\; t\;\mathtt{in}\,(x,x)\\
\mathcal{C}_{\mathtt{false}}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,x \;=\; t\;\mathtt{in}\,(x,\mathit{not}(x))\\
\mathcal{C}_{@v}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,x \;=\; t(v)\;\mathtt{in}\,(x,\mathtt{true})\\
\mathcal{C}_{\mathtt{copy}}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,x \;=\; t\;\mathtt{in}\,((x,x),\mathtt{true})\\
\mathcal{C}_{\mathtt{discard}}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,x \;=\; t\;\mathtt{in}\,((),\mathtt{true})\\
\mathcal{C}_{\mathtt{l}.\gamma}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,(x_1,x_2) \;=\; t\\
&\phantom{\stackrel{\text{def}}{=}}\ \mathtt{in}\,\mathtt{let}\,(x_1',x_2') \;=\; \mathcal{C}_\gamma[x_1]\;\mathtt{in}\,((x_1',x_2),x_2')\\
\mathcal{C}_{\mathtt{r}.\gamma}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,(x_1,x_2) \;=\; t\\
&\phantom{\stackrel{\text{def}}{=}}\ \mathtt{in}\,\mathtt{let}\,(x_1',x_2') \;=\; \mathcal{C}_\gamma[x_2]\;\mathtt{in}\,((x_1,x_1'),x_2')
\end{aligned}$$

We then prove that $\mathcal{C}_\gamma$ has the required property, by induction on $\gamma$. This is straightforward, as an example we demonstrate the case where the label is $\mathtt{l}.\gamma$.

Suppose $t \xRightarrow{\mathtt{l}.\gamma} \Rightarrow v$. We know that $t$ must converge to a value, and by construction, we know that this value must be a pair, $(v_1,v_2)$ say, such that $v_1 \xRightarrow{\gamma} \Rightarrow v'$, where $v = (v',v_2)$. Now,

$$\mathcal{C}_{\mathtt{l}.\gamma}[t] \Rightarrow \mathtt{let}\,(x_1',x_2') \;=\; \mathcal{C}_\gamma[v_1]\;\mathtt{in}\,((x_1',v_2),x_2').$$

By induction we know that $\mathcal{C}_\gamma[v_1] \Rightarrow (v',\mathtt{true})$, thus we have

$$\mathtt{let}\,(x_1',x_2') \;=\; \mathcal{C}_\gamma[v_1]\;\mathtt{in}\,((x_1',v_2),x_2') \Rightarrow ((v',v_2),\mathtt{true})$$

which is to say $\mathcal{C}_{\mathtt{l}.\gamma}[t] \Rightarrow (v,\mathtt{true})$.

Conversely, suppose that $\mathcal{C}_{\mathtt{l}.\gamma}[t] \Rightarrow (v,\mathtt{true})$. By inspection of the context we note that $t \Rightarrow (v_1,v_2)$ for some values and $\mathcal{C}_\gamma[v_1] \Rightarrow (v',\mathtt{true})$ such that $v$ is $(v',v_2)$. From this we know by induction that $v_1 \xRightarrow{\gamma} \Rightarrow v'$, whence $t \Rightarrow (v_1,v_2) \xRightarrow{\mathtt{l}.\gamma} ((v',v_2),\mathtt{true})$ as required.

For a sequence of labels, we define:

$$\begin{aligned}
\mathcal{C}_\varepsilon[t] &\stackrel{\text{def}}{=} \mathtt{let}\,x \;=\; t\;\mathtt{in}\,(x,\mathtt{true})\\
\mathcal{C}_{\overline{\gamma},\overline{\gamma}'}[t] &\stackrel{\text{def}}{=} \mathtt{let}\,(x_1,x_2) \;=\; \mathcal{C}_{\overline{\gamma}}[t]\\
&\phantom{\stackrel{\text{def}}{=}}\ \mathtt{in}\,\mathtt{let}\,(x_1',x_2') \;=\; \mathcal{C}_{\overline{\gamma}'}[x_1]\;\mathtt{in}\,(x_1',x_2 \wedge x_2')
\end{aligned}$$

The result follows by induction on the length of $\overline{\gamma}$. □

**Theorem 2.2 (completeness for λ-calculus)** *If* $\Gamma \vDash t \approx_{ctx} t' : \sigma$ *then* $\Gamma \vDash t \approx^{\circ} t' : \sigma$.

**Proof:** It suffices to show the result for closed terms. Let $\overline{\gamma}$ be a trace of $t$:

$$(\vdash t : \sigma) \xRightarrow{\overline{\gamma}}$$

$$\begin{array}{lll}
\text{so} & (\vdash t:\sigma) \xRightarrow{\overline{\gamma}} \Rightarrow (\vdash v:\sigma') & (\lambda\text{-calculus is terminating})\\
\text{so} & (\vdash \mathcal{C}_{\overline{\gamma}}[t] : (\sigma' \times \mathtt{bool})) \Rightarrow (\vdash (v,\mathtt{true}) : (\sigma' \times \mathtt{bool})) & (\text{Lemma 2.1})\\
\text{so} & (\vdash \mathtt{snd}\,(\mathcal{C}_{\overline{\gamma}}[t]) : \mathtt{bool}) \Rightarrow (\vdash \mathtt{true} : \mathtt{bool}) & (\text{Defn of snd})\\
\text{so} & (\vdash \mathtt{snd}\,(\mathcal{C}_{\overline{\gamma}}[t']) : \mathtt{bool}) \Rightarrow (\vdash \mathtt{true} : \mathtt{bool}) & (t \approx_{ctx} t')\\
\text{so} & (\vdash \mathcal{C}_{\overline{\gamma}}[t'] : (\sigma' \times \mathtt{bool})) \Rightarrow (\vdash (v',\mathtt{true}) : (\sigma' \times \mathtt{bool})) & (\text{Defn of snd})\\
\text{so} & (\vdash t':\sigma) \xRightarrow{\overline{\gamma}} \Rightarrow (\vdash v':\sigma') & (\text{Lemma 2.1})
\end{array}$$

Similarly, any trace of $t'$ is a trace of $t$, so the terms are trace equivalent. Since the λ-calculus is deterministic, trace equivalence and bisimulation coincide, so $t \approx t'$. □

## 2.6 Soundness

In this section, we shall show that bisimulation is *sound*, that is:

$$\text{if } t \approx^\circ t' \text{ then } t \approx_{ctx} t'$$

This result is immediate from the result that bisimulation is a congruence, for which we adopt Howe's technique [10], following Gordon [6].

For any type-indexed relation $R$, let $\widehat{R}$ be defined such that for each type rule in the language:

$$\frac{\overline{\Gamma} \vdash \overline{t} : \overline{\sigma}}{\Gamma \vdash op(\overline{t}) : \sigma}$$

we have:

$$\frac{\overline{\Gamma} \vDash \overline{t}\,R\,\overline{t'} : \overline{\sigma}}{\Gamma \vDash op(\overline{t})\,\widehat{R}\,op(\overline{t'}) : \sigma}$$

For any type-indexed relation $R$, let $R^\bullet$ be defined:

$$\frac{t_1\,\widehat{R^\bullet}\,t_2\,R^\circ\,t_3}{t_1\,R^\bullet\,t_3}$$

Howe's proof depends first on showing that $\approx^\bullet$ is *substitutive on values*:

if $t_1 \approx^\bullet t_2$
and $v_1 \approx^\bullet v_2$
then $t_1[v_1/x] \approx^\bullet t_2[v_2/x]$

and then showing that $\approx^\bullet$ is a bisimulation. The only tricky case is let-β, where we complete:

$$\begin{tikzcd}
\texttt{let}\,x = v_1\,\texttt{in}\,t_1 \arrow[r, leftrightarrow, "\widehat{\approx^\bullet}"] \arrow[d, "\tau"'] & \texttt{let}\,x = v_2\,\texttt{in}\,t_2 \arrow[r, leftrightarrow, "\approx"] & t_3 \\
t_1[v_1/x] & &
\end{tikzcd}$$

as:

$$\begin{tikzcd}
\texttt{let}\,x = v_1\,\texttt{in}\,t_1 \arrow[r, leftrightarrow, "\widehat{\approx^\bullet}"] \arrow[d, "\tau"'] & \texttt{let}\,x = v_2\,\texttt{in}\,t_2 \arrow[r, leftrightarrow, "\approx"] \arrow[d, "\tau"'] & t_3 \arrow[d, Rightarrow] \\
t_1[v_1/x] \arrow[r, leftrightarrow, "\approx^\bullet"] & t_2[v_2/x] \arrow[r, leftrightarrow, "\approx"] & t_3'
\end{tikzcd}$$

which commutes because $\approx^\bullet$ is substitutive on values. From this, it is routine to show that bisimulation is a congruence, and so is sound.

**Theorem 2.3 (soundness for λ-calculus)** *If* $\Gamma \vDash t \approx^\circ t' : \sigma$ *then* $\Gamma \vDash t \approx_{ctx} t' : \sigma$.

## 2.7 Comments

The astute reader will notice that the `copy` and `discard` transitions are redundant in this setting. In fact, it is a well known property of pure functional languages that 'operational extensionality' holds, that is, contextual equivalence can be verified by using applicative contexts alone. This does certainly not hold true of the extensions to the $\lambda$-calculus which we will consider later in this paper where operational extensionality fails.

In a similar vein, we notice that the use of `l.` and `r.` tags rather than Gordon's `fst` and `snd` transitions is also unnecessary here because pairing forms a product on values. In later sections, because of the presence of side-effects, the pairing operator is no longer a product, but is symmetric monoidal.

It is an important feature of the transition systems being used here, and also those of [6, 2] that they are *applicative* in nature. That is, any arbitrary pieces of code being carried in the label is always of lower order type than the term under scrutiny.

# 3 $\nu$-calculus

We now extend the $\lambda$-calculus with unique name generation and equality testing, in order to investigate Pitts and Stark's [13] $\nu$-calculus.

Pitts and Stark have demonstrated that finding a sound and complete semantics for the $\nu$-calculus is a difficult open problem. They provide a sound (but incomplete) semantics using logical relations. In this section, we provide an 'upper bound' to complement their 'lower bound' by presenting a bisimulation which is complete (but only sound up to first-order). We observe that our complete bisimulation provides a more investigative proof method for establishing contextual inequivalence which allows one to construct distinguishing contexts in a piecemeal fashion. This useful feature of the semantics avoids the need to build these, sometimes elaborate, contexts completely by making much of the construction automatic.

## 3.1 Syntax and type rules

Extend the grammar of types with:

$$\sigma ::= \cdots \mid \mathtt{name}$$

Extend the grammar of values with:

$$v ::= \cdots \mid n$$

Extend the grammar of terms with:

$$t ::= \cdots \mid \nu n \,.\, t \mid v = v$$

Extend the type judgements $\Gamma \vdash t : \sigma$ to include a *name context* $\Delta$ of the form $n_1, \ldots, n_n$ for distinct $n_i$, so judgements are now of the form $\Gamma; \Delta \vdash t : \sigma$. The type rules for the new terms are:

$$\frac{}{\Gamma; \Delta, n, \Delta' \vdash n : \mathtt{name}} \qquad \frac{\Gamma; \Delta, n \vdash t : \sigma}{\Gamma; \Delta \vdash \nu n \,.\, t : \sigma}$$

$$\frac{\Gamma; \Delta \vdash v : \mathtt{name} \quad \Gamma; \Delta \vdash v' : \mathtt{name}}{\Gamma; \Delta \vdash v = v' : \mathtt{bool}}$$

The other rules do not change the name context.

## 3.2 Reduction semantics

Terms no longer reduce to values, instead they now reduce to *prevalues* of the form:

$$p ::= \nu\overline{n} \,.\, v$$

Extend the reduction relation with (when $n \neq n'$):

$$\begin{array}{rcl} n = n & \xrightarrow{\tau} & \texttt{true} \\ n = n' & \xrightarrow{\tau} & \texttt{false} \end{array}$$

Extend the grammar of evaluation contexts by:

$$\mathcal{E} ::= \cdots \mid \nu n \,.\, \mathcal{E}$$

Replace the let-β reduction rule by:

$$\texttt{let}\, x \;=\; \nu\overline{n} \,.\, v\, \texttt{in}\, t \quad \xrightarrow{\tau} \quad \nu\overline{n} \,.\, t[v/x]$$

where we α-convert $\nu\overline{n} \,.\, v$ if necessary to ensure that none of the free names in $t$ are captured. It is in this rule that *scope extrusion* of the static name binder occurs. There is an obvious translation from Pitts and Stark's ν-calculus into ours (theirs does not include pairing), and it is routine to show that this translation is adequate.

The definition of contextual equivalence remains the same, except that the results of a test can include some private names: $t \approx_{ctx} t'$ whenever for all closing contexts $\mathcal{C}$ of type `bool`, we have $\mathcal{C}[t] \Rightarrow \nu\overline{n} \,.\, \texttt{true}$ iff $\mathcal{C}[t'] \Rightarrow \nu\overline{n}' \,.\, \texttt{true}$.

## 3.3 Labelled transition system semantics

We can no longer define the lts semantics as judgements $(\,\vdash v : \sigma) \xrightarrow{\gamma} (\,\vdash t : \sigma')$, for two reasons:

- Terms may reduce down to prevalues now, rather than values, so transitions should be of the form $(\,\vdash p : \sigma) \xrightarrow{\gamma} (\,\vdash t : \sigma')$.
- One of the allowed transitions allows a private name to become public, and we α-convert the name to ensure it does not clash with any existing public names. To do this, we carry an environment of existing public names, so transitions should be of the form

  $$(\Delta \vdash p : \sigma) \xrightarrow{\gamma} (\Delta, \Delta' \vdash t : \sigma')$$

  Note that transitions can add new public names, but not remove any.

We extend the grammar of labels by:

$$\gamma ::= \cdots \mid n \mid \nu n$$

These labels can be read as 'the term announces a public name' and 'the term announces a private name, and makes it public'. Labels now contain bound names:

$$bn(\nu n) = \{n\} \qquad bn(\overline{\gamma}, \overline{\gamma}') = bn(\overline{\gamma}) \cup bn(\overline{\gamma}')$$

and free names:

$$fn(n) = \{n\} \qquad fn(@v) = fn(v) \qquad fn(\overline{\gamma},\overline{\gamma}') = fn(\overline{\gamma}) \cup fn(\overline{\gamma}') \setminus bn(\overline{\gamma})$$

A public name can be announced:

$$(\Delta \vdash n : \mathtt{name}) \quad \xrightarrow{n} \quad (\Delta \vdash () : \mathtt{unit})$$

The context $\nu n \,.\, \cdot$ is an observation context:

$$\frac{(\Delta, n \vdash p : \sigma) \xrightarrow{\gamma} (\Delta, n, \Delta' \vdash t : \sigma')}{(\Delta \vdash \nu n \,.\, p : \sigma) \xrightarrow{\gamma} (\Delta, \Delta' \vdash \nu n \,.\, t : \sigma')}\ [n \text{ not in } \gamma]$$

Private names can announce themselves and become public:

$$\frac{(\Delta, n \vdash p : \sigma) \xrightarrow{\iota.n} (\Delta, n \vdash t : \sigma')}{(\Delta \vdash \nu n \,.\, p : \sigma) \xrightarrow{\iota.\nu n} (\Delta, n \vdash t : \sigma')}$$

where $\iota.$ is a sequence of $\mathtt{l.}$ and $\mathtt{r.}$ tags. The side-condition on application is weakened to allow values to have free public names:

$$(\Delta \vdash v : \sigma \to \sigma') \quad \xrightarrow{@v'} \quad (\Delta \vdash v(v') : \sigma') \qquad (\text{where } \Delta \vdash v' : \sigma)$$

Note that for any well-typed $(\Delta \vdash t)$, if $(\Delta \vdash t) \overset{\overline{\gamma}}{\Longrightarrow} (\Delta, \Delta' \vdash t')$ then the free names of $\overline{\gamma}$ are contained in $\Delta$ and the bound names of $\overline{\gamma}$ are $\Delta'$.

We define bisimulation as before between configurations $(\Delta \vdash t : \sigma)$.

Since we are allowing free names in terms, we have to change the definition of open extension $R^\circ$ to allow the environment to introduce new names:

$$\begin{array}{l} \Gamma; \Delta \vDash t \; R^\circ \; t' : \sigma \\ \quad \text{iff for all } \Delta, \Delta' \vdash [\overline{v}/\overline{x}] : \Gamma \text{ we have:} \\ \qquad \Delta, \Delta' \vDash (t[\overline{v}/\overline{x}]) \; R \; (t'[\overline{v}/\overline{x}]) : \sigma \end{array}$$

This is necessary, for example, in order to distinguish:

$$x : \mathtt{name}; \Delta \vDash x \in \Delta \not\approx^\circ \mathtt{true} : \mathtt{bool}$$

we need to introduce a new name $n$ to substitute for $x$:

$$\Delta, n \vDash n \in \Delta \not\approx \mathtt{true} : \mathtt{bool}$$

If $R^\circ$ were not allowed to introduce new names, then these terms could not be distinguished, and so $\approx^\circ$ would not be a congruence, even at first order.

## 3.4 Examples

Consider the terms:

$$\nu n \,.\, \lambda x : \mathtt{unit} \,.\, n \not\approx \lambda x : \mathtt{unit} \,.\, \nu n \,.\, n$$

These are not bisimilar because the first term has the reduction:

$$\begin{array}{rcl}
\nu n \,.\, \lambda x : \mathtt{unit} \,.\, n & \overset{\mathtt{copy}}{\Longrightarrow} & \nu n \,.\, (\lambda x : \mathtt{unit} \,.\, n, \lambda x : \mathtt{unit} \,.\, n) \\
& \overset{\mathtt{l}.@()}{\Longrightarrow} \Rightarrow & \nu n \,.\, (n, \lambda x : \mathtt{unit} \,.\, n) \\
& \overset{\mathtt{r}.@()}{\Longrightarrow} \Rightarrow & \nu n \,.\, (n, n) \\
& \overset{\mathtt{l}.\nu n}{\Longrightarrow} & ((), n) \\
& \overset{\mathtt{r}.n}{\Longrightarrow} & ((), ())
\end{array}$$

which the second term can only match:

$$\begin{array}{rcl}
\lambda x : \mathtt{unit} \,.\, \nu n \,.\, n & \overset{\mathtt{copy}}{\Longrightarrow} & (\lambda x : \mathtt{unit} \,.\, \nu n \,.\, n, \lambda x : \mathtt{unit} \,.\, \nu n \,.\, n) \\
& \overset{\mathtt{l}.@()}{\Longrightarrow} \Rightarrow & (\nu n \,.\, n, \lambda x : \mathtt{unit} \,.\, \nu n \,.\, n) \\
& \overset{\mathtt{r}.@()}{\Longrightarrow} \Rightarrow & \nu n \,.\, (n, \nu n' \,.\, n') \\
& \overset{\mathtt{l}.\nu n}{\Longrightarrow} & \nu n' \,.\, ((), n')
\end{array}$$

At this point the term cannot match the last $\overset{\mathtt{r}.n}{\Longrightarrow}$ transition performed by the first term because its only move is:

$$\nu n' \,.\, ((), n') \overset{\mathtt{r}.\nu n'}{\Longrightarrow} ((), ())$$

Note that this example relies crucially on the use of $\mathtt{copy}$, $\mathtt{l}.\gamma$ and $\mathtt{r}.\gamma$ transitions.

## 3.5 Completeness

Completeness for the $\nu$-calculus follows in much the same way as it does for the $\lambda$-calculus however the inducing contexts must now carry information about the new names which are introduced during reduction.

**Lemma 3.1** *For every sequence $\overline{\gamma}$ of transition labels with free names in $\Delta$ and bound names $\Delta'$, there is a context $C^{\Delta}_{\overline{\gamma}}$ such that:*

$$(\Delta \vdash t : \sigma) \overset{\overline{\gamma}}{\Longrightarrow} \Rightarrow (\Delta, \Delta' \vdash p : \sigma')$$

*iff:*

$$(\Delta \vdash C^{\Delta}_{\overline{\gamma}}[t] : (\sigma' \times \mathtt{bool})) \Rightarrow (\Delta \vdash \nu\Delta' \,.\, (\Delta', p, \mathtt{true}) : (\mathtt{name} \times \cdots \times \mathtt{name} \times \sigma' \times \mathtt{bool}))$$

**Proof:** We modify all the contexts $C_{\gamma}$ from Lemma 2.1 in the obvious way and add:

$$\begin{array}{rcl}
C^{\Delta}_{n}[t] & \overset{\text{def}}{=} & \mathtt{let}\, x \;=\; t \,\mathtt{in}\, ((), (), x = n) \\
C^{\Delta}_{\nu n}[t] & \overset{\text{def}}{=} & \mathtt{let}\, x \;=\; t \,\mathtt{in}\, (x, (), x \notin \Delta)
\end{array}$$

The result follows for contexts $C_{\gamma}$ by induction on $\gamma$ as for the $\lambda$-calculus. For sequences of labels, we define $C^{\Delta}_{\overline{\gamma}}$ (where $\Delta'$ is the bound variables of $\overline{\gamma}$ and $\Delta''$ is the bound variables of $\overline{\gamma}'$) as:

$$\begin{array}{rcl}
C^{\Delta}_{\varepsilon} & \overset{\text{def}}{=} & \mathtt{let}\, x \;=\; t \,\mathtt{in}\, ((), x, \mathtt{true}) \\
C^{\Delta}_{\gamma,\overline{\gamma}'} & \overset{\text{def}}{=} & \mathtt{let}\, (\Delta', x_1, x_2) \;=\; C^{\Delta}_{\overline{\gamma}}[t] \,\mathtt{in}\, \mathtt{let}\, (\Delta'', x_1', x_2') \;=\; C^{\Delta,\Delta'}_{\overline{\gamma}'}[x_1] \,\mathtt{in}\, (\Delta', \Delta'', x_1', x_2 \wedge x_2')
\end{array}$$

using some syntax sugar such as:

$$\mathtt{let}\, n = t \,\mathtt{in}\, t' \overset{\mathsf{def}}{=} \mathtt{let}\, x = t \,\mathtt{in}\, (t'[x/n])$$

The result then follows by induction on $\overline{\gamma}$. □

**Theorem 3.2 (completeness for ν-calculus)** *If* $\Gamma;\Delta \vDash t \approx_{ctx} t' : \sigma$ *then* $\Gamma;\Delta \vDash t \approx^{\circ} t' : \sigma$.

## 3.6 Partial soundness

It is a fairly simple matter to show that bisimulation is sound for the ν-calculus at first order, by showing that bisimulation coincides with Pitts and Stark's logical relation semantics.

**Theorem 3.3 (soundness for ν-calculus at first order)** *For first order* $\sigma$*, if* $\Gamma;\Delta \vDash t \approx^{\circ} t' : \sigma$ *then* $\Gamma;\Delta \vDash t \approx_{ctx} t' : \sigma$.

Unfortunately, bisimulation is *not* sound for the ν-calculus. The counterexample is from Pitts and Stark [21]. Consider three functions of type $(\mathtt{name} \to \mathtt{bool}) \to \mathtt{bool}$:

$$\begin{aligned} t_1 &\overset{\mathsf{def}}{=} \lambda f : \mathtt{name} \to \mathtt{bool} \,.\, \mathtt{true} \\ t_2 &\overset{\mathsf{def}}{=} \nu n \,.\, \nu n' \,.\, \lambda f : \mathtt{name} \to \mathtt{bool} \,.\, f(n) = f(n') \\ t_3 &\overset{\mathsf{def}}{=} \nu n \,.\, \lambda f : \mathtt{name} \to \mathtt{bool} \,.\, \nu n' \,.\, f(n) = f(n') \end{aligned}$$

These three terms are all bisimilar. To see this we note that Pitts and Stark showed that $t_1$ and $t_2$ are contextually equivalent so by completeness it follows that the terms are bisimilar. We leave it to the reader to convince oneself that terms $t_2$ and $t_3$ cannot be distinguished by any sequence of transitions. However, we do note that $t_2 \not\approx_{ctx} t_3$, since the following context distinguishes them:

$$C[\cdot] \overset{\mathsf{def}}{=} \mathtt{let}\, F = \cdot \,\mathtt{in}\, F(\lambda x : \mathtt{name} \,.\, F(\lambda x' : \mathtt{name} \,.\, x = x'))$$

This is the same counterexample that Pitts and Stark use to show that logical relations are incomplete, since logical relations identify none of these terms.

Bisimulation is unsound because it is not a congruence. To see why Howe's proof fails, we have to observe that the crucial step in Howe is that $\approx^{\bullet}$ matches let-$\beta$ reductions. In the ν-calculus we would have to complete the diagram:

$$\begin{tikzcd}
\mathtt{let}\, x = \nu\overline{n}_1 \,.\, v_1 \,\mathtt{in}\, t_1 \arrow[r, leftrightarrow, "\widehat{\approx^{\bullet}}"] \arrow[d, "\tau"] & \mathtt{let}\, x = \nu\overline{n}_2 \,.\, v_2 \,\mathtt{in}\, t_2 \arrow[r, leftrightarrow, "\approx"] & t_3 \\
\nu\overline{n}_1 \,.\, t_1[v_1/x] & &
\end{tikzcd}$$

as:

$$\begin{tikzcd}
\mathtt{let}\, x = \nu\overline{n}_1 \,.\, v_1 \,\mathtt{in}\, t_1 \arrow[r, leftrightarrow, "\widehat{\approx^{\bullet}}"] \arrow[d, "\tau"] & \mathtt{let}\, x = \nu\overline{n}_2 \,.\, v_2 \,\mathtt{in}\, t_2 \arrow[r, leftrightarrow, "\approx"] \arrow[d, "\tau"] & t_3 \arrow[d, Rightarrow] \\
\nu\overline{n}_1 \,.\, t_1[v_1/x] \arrow[r, leftrightarrow, "???"] & \nu\overline{n}_2 \,.\, t_2[v_2/x] \arrow[r, leftrightarrow, "\approx"] & t_3'
\end{tikzcd}$$

but in order to complete the diagram we need to know that $\approx^\bullet$ is *substitutive on prevalues*:

$$\begin{array}{l}\text{if } t_1 \approx^\bullet t_2 \\ \text{and } \nu\overline{n}_1 \,.\, v_1 \approx^\bullet \nu\overline{n}_2 \,.\, v_2 \\ \text{then } \nu\overline{n}_1 \,.\, t_1[v_1/x] \approx^\bullet \nu\overline{n}_2 \,.\, t_2[v_2/x]\end{array}$$

which is not true of the ν-calculus, as witnessed by the counterexample described above.

## 3.7 Comments

We now consider how our bisimulation compares with logical relations. The example given above serves to demonstrate that logical relations are strictly finer than bisimulation, and that they only coincide at first order. The main difference between the two approaches is the view of privacy they adopt. In the bisimulation approach names are considered private until the secret is leaked by a $\nu n$ transition, whereas the logical relations use a more overt proof method whereby the environment has access to secrets but can only test using them restrictedly. For terms of first-order type the test values are of ground type and the restrictions imposed by the logical relations are strong enough to disallow any testing with secrets altogether, thus realigning the method with the more covert approach of bisimulation.

There is also an evident analogue of Sangiorgi's context bisimulation [18], which could be formulated for the ν-calculus. In essence, this says that whenever we need to test a λ-abstraction we must consider its behaviour in every context. This could easily be formulated in an lts by allowing suitably typed transitions of the form $v \xrightarrow{f@} f(v)$. This of course, defies our insistence that labels be applicative because the type of $f$ here must be of higher-order than $v$. In fact, bisimulation on such an lts could easily be shown to be fully abstract but this would, in effect, be no more than a restatement of the **ciu**-theorem of [13] following [9]. Sangiorgi did find an applicative characterisation of context bisimulation for higher-order π-calculus which he refers to as normal bisimulation [18]. This characterisation exploits the very simple nature of evaluation context in π-calculus and is tailored to that setting. It would therefore be inappropriate in the current work.

Although one obvious reason for bisimulation failing to be a congruence is that the labels in our lts do not provide sufficient distinguishing power, it is difficult to see how the label set could be effectively enlarged without using non-applicative values such as are used for context bisimulation.

As an alternative, we extend the ν-calculus to include imperative side-effects, to get the νref-calculus, and show that the extra $@v$ transitions give us full abstraction. This is the subject of the next section.

# 4 νref-calculus

We have shown that bisimulation is complete (but unsound) for the ν-calculus. In this section we show that adding imperative side-effects to the language allows us to recover a sound and complete semantics.

The reason why assignments allow us to recover completeness is that the counterexamples rely on the fact that $n$ is a 'secret' in terms such as:

$$\nu n \,.\, \lambda f : \texttt{name} \to \texttt{bool} \,.\, f(n)$$

despite the fact that some ‘foreign’ code $f$ is being applied to $n$. By adding assignment, $f$ can leak the secret name $n$ to the environment.

We believe that any form of side-effect which allows secrets to leak like this will help to make bisimulation sound and complete, for example call-cc, communication channels or imperative objects. Although the extent to which any additional features are required is as yet unclear. We have chosen to investigate global assignment as it is the simplest addition which is still deterministic and terminating.

## 4.1 Syntax and type rules

Extend the grammar of terms by:

$$t ::= \cdots \mid r := v \,.\, t \mid ?r$$

where $r$ ranges over an infinite set of *references*. These operations allow a name to be written to, or read from, a reference. We do not introduce references themselves as values and thus have no need for introducing a type of references.

We introduce a *use-def* type system to ensure that all references are written to before they are read from. The type judgements are now of the form $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma$ where R is the set of references which are read by the term, and W is the set of references which are written by the term. Most of the type rules remain unchanged from the ν-calculus, the notable changes are in let and function bodies:

$$\frac{\Gamma, x : \sigma;\Delta;\mathrm{R}; \vdash t : \sigma'}{\Gamma;\Delta;\mathrm{R}; \vdash (\lambda x : \sigma \,.\, t) : (\sigma \to \sigma')} \qquad \frac{\Gamma;\Delta;\mathrm{R};\mathrm{W} \cup \overline{r} \vdash t : \sigma \quad \Gamma, x : \sigma;\Delta;\mathrm{R} \cup \overline{r};\mathrm{W}' \vdash t' : \sigma'}{\Gamma;\Delta;\mathrm{R};\mathrm{W} \cup \mathrm{W}' \vdash \mathtt{let}\, x \,=\, t \,\mathtt{in}\, t' : \sigma'}$$

The type judgements for the new operators are:

$$\frac{\Gamma;\Delta;\mathrm{R}; \vdash v : \mathtt{name} \quad \Gamma;\Delta;\mathrm{R} \cup r;\mathrm{W} \setminus r \vdash t : \sigma}{\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash r := v \,.\, t : \sigma} \qquad \frac{}{\Gamma;\Delta;\mathrm{R} \cup r; \vdash ?r : \mathtt{name}}$$

Note that we can derive the following weakening rule for references:

$$\frac{\Gamma;\Delta;\mathrm{R};\mathrm{W} \cup \mathrm{W}' \vdash t : \sigma}{\Gamma;\Delta;\mathrm{R} \cup \mathrm{R}';\mathrm{W} \vdash t : \sigma}$$

We shall refer to any term which can be typed $\Delta;;\mathrm{W} \vdash t : \sigma$ as being *ref-closed*, and other terms as being *ref-open*. For example $?r$ is ref-open, but $r := n \,.\, ?r$ is ref-closed.

## 4.2 Reduction semantics

Prevalues are now terms of the form:

$$p ::= d \,.\, v \qquad d ::= \nu\overline{n} \,.\, \overline{r} := \overline{n}' \,.$$

The reduction semantics for the νref-calculus is defined up to a structural equivalence, following Berry and Boudol’s Chemical Abstract Machine [3]. Let the *heating* relation, $\Rrightarrow$ be the least preorder given by:

$$\begin{array}{rcll}
r := n \,.\, \nu n' \,.\, t & \Rrightarrow & \nu n' \,.\, r := n \,.\, t & (n \neq n') \\
r_1 := n_1 \,.\, r_2 := n_2 \,.\, t & \Rrightarrow & r_2 := n_2 \,.\, r_1 := n_1 \,.\, t & (r_1 \neq r_2) \\
r := n_1 \,.\, r := n_2 \,.\, t & \Rrightarrow & r := n_2 \,.\, t & \\
\nu n \,.\, \nu n' \,.\, t & \Rrightarrow & \nu n' \,.\, \nu n \,.\, t & \\
d.\mathtt{let}\, v \,=\, t \,\mathtt{in}\, t' & \Rrightarrow & \mathtt{let}\, v \,=\, d.t \,\mathtt{in}\, t' &
\end{array}$$

(where the bound names in $d$ do not clash with free names in $t'$) and:

$$\frac{t_1 \Rrightarrow t_2}{\mathcal{E}[t_1] \Rrightarrow \mathcal{E}[t_2]}$$

Let $\equiv$ be the least equivalence generated by $\Rrightarrow$.

Extend the evaluation contexts to include assignment:

$$\mathcal{E} ::= \cdots \mid r := v \,.\, \mathcal{E}$$

Extend the reduction semantics with a rule for dereferencing:

$$r := n \,.\, ?r \quad \xrightarrow{\tau} \quad r := n \,.\, n$$

Since we have modified the prevalues, we need to modify the let-β rule:

$$\texttt{let}\, x \;=\; d \,.\, v\, \texttt{in}\, t \quad \xrightarrow{\tau} \quad d \,.\, t[v/x]$$

Add a structural equivalence rule:

$$\frac{t_1 \equiv t_2 \quad t_2 \xrightarrow{\tau} t_3 \quad t_3 \equiv t_4}{t_1 \xrightarrow{\tau} t_4}$$

The definition of contextual equivalence remains the same, except that the results of a test can include some assignments: $t_1 \approx_{ctx} t_2$ whenever for all ref-closing contexts $\mathcal{C}$ of type `bool`, we have $\mathcal{C}[t_1] \Rightarrow d_1 \,.\, \texttt{true}$ iff $\mathcal{C}[t_2] \Rightarrow d_2 \,.\, \texttt{true}$.

**Lemma 4.1** *Any derivation* $t \xrightarrow{\tau} t'$ *can be deduced* $t \Rrightarrow t'' \xrightarrow{\tau} t''' \equiv t'$ *where* $t'' \xrightarrow{\tau} t'''$ *can be deduced without using structural equivalence.*

**Proof:** A simple analysis of the rules which generate $\Rrightarrow$ suffices to show that any reduction which may occur on the left-hand side of a rule may also occur on the right, so naught is to be gained by *cooling*. □

The reader may like to note that the νref-calculus contains closed terms which may not necessarily converge to a prevalue, such as $\texttt{let}\, x \;=\; ?r\, \texttt{in}\, t$. However, all such terms are ref-open, and our reduction semantics is only used for ref-closed terms.

## 4.3 Labelled transition system semantics

We need to provide a semantics for terms with references, so judgements are now of the form $(\Delta; \mathrm{R}; \mathrm{W} \vdash p : \sigma) \xrightarrow{\gamma} (\Delta, \Delta'; \mathrm{R}; \mathrm{W} \vdash t : \sigma')$. Note that since terms cannot generate new references, that the reference environments are not changed by transitions.

Extend the grammar of labels with:

$$\gamma ::= \cdots \mid r{:=}n \mid ?r$$

The new transitions allow a name to be assigned:

$$(\Delta; \mathrm{R}; \vdash () : \texttt{unit}) \xrightarrow{r:=n} (\Delta; \mathrm{R}; \vdash r := n \,.\, () : \texttt{unit}) \qquad (\text{where } n \in \Delta)$$

and to be read:

$$(\Delta;\mathrm{R};\vdash ():\texttt{unit}) \xrightarrow{?r} (\Delta;\mathrm{R};\vdash ?r:\texttt{name}) \qquad (\text{where } r\in\mathrm{R})$$

We weaken the side-condition on application to allow the argument to include free references:

$$(\Delta;\mathrm{R};\vdash v:\sigma\to\sigma') \xrightarrow{@v'} (\Delta;\mathrm{R};\vdash v(v'):\sigma') \qquad (\text{where } \Delta;\mathrm{R};\vdash v':\sigma)$$

Transitions are allowed in assignment contexts:

$$\frac{(\Delta;\mathrm{R}\cup r;\mathrm{W}\setminus r\vdash p:\sigma)\xrightarrow{\gamma}(\Delta,\Delta';\mathrm{R}\cup r;\mathrm{W}\setminus r\vdash t:\sigma')}{(\Delta;\mathrm{R};\mathrm{W}\vdash r:=n\,.\,p:\sigma)\xrightarrow{\gamma}(\Delta,\Delta';\mathrm{R};\mathrm{W}\vdash r:=n\,.\,t:\sigma')}$$

We define bisimulation as before between configurations $(\Delta;;\mathrm{W}\vdash t:\sigma)$: note we are only defining bisimulation on ref-closed terms. For ref-open terms, we extend the definition of $R^\circ$ to include ref-closing assignments:

$\Gamma;\Delta;\mathrm{R};\mathrm{W}\vDash t\ R^\circ\ t':\sigma$
iff for all $\Delta,\Delta';\mathrm{R}\cup\mathrm{R}';\vdash[\overline{v}/\overline{x}]:\Gamma$ and $\Delta,\Delta'\vdash(\overline{r}:=\overline{n}):(\mathrm{R}\cup\mathrm{R}')$ we have:
$\Delta,\Delta';;\mathrm{R}\cup\mathrm{R}'\cup\mathrm{W}\vDash(\overline{r}:=\overline{n}\,.\,t[\overline{v}/\overline{x}])\ R\ (\overline{r}:=\overline{n}\,.\,t'[\overline{v}/\overline{x}]):\sigma$

where we write $\Delta\vdash(\overline{r}:=\overline{n}):(\overline{r})$ whenever $\overline{n}\subseteq\Delta$.

## 4.4 Example

We can distinguish the problem cases from the ν-calculus:

$$\begin{array}{rcl}
t_1 & \overset{\text{def}}{=} & \lambda f:\texttt{name}\to\texttt{bool}\,.\,\texttt{true}\\
t_2 & \overset{\text{def}}{=} & \nu n\,.\,\nu n'\,.\,\lambda f:\texttt{name}\to\texttt{bool}\,.\,f(n)=f(n')\\
t_3 & \overset{\text{def}}{=} & \nu n\,.\,\lambda f:\texttt{name}\to\texttt{bool}\,.\,\nu n'\,.\,f(n)=f(n')
\end{array}$$

It is easy to distinguish $t_1$ from the others, since we just let $f$ be a function with a side-effect. To distinguish $t_2$ from $t_3$ we define:

$$v_2\overset{\text{def}}{=}\lambda f:\texttt{name}\to\texttt{bool}\,.\,f(n)=f(n')$$

and so for fresh $r_0$ and $n_0$:

$$\begin{array}{lrl}
r_0:=n_0\,.\,t_2 & \equiv & \nu n\,.\,\nu n'\,.\,r_0:=n_0\,.\,v_2\\
 & \xRightarrow{\texttt{copy}} & \nu n\,.\,\nu n'\,.\,r_0:=n_0\,.\,(v_2,v_2)\\
 & \xRightarrow{\texttt{l}.@\lambda x:\texttt{name}.r_0:=x.\texttt{true}} & \nu n\,.\,\nu n'\,.\,r_0:=n_0\,.\,(v_2(\lambda x:\texttt{name}\,.\,r_0:=x\,.\,\texttt{true}),v_2)\\
 & \xRightarrow{\tau} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(\texttt{true},v_2)\\
 & \xRightarrow{\texttt{l.discard}} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,((),v_2)\\
 & \xRightarrow{\texttt{l}.?r_0} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(?r_0,v_2)\\
 & \xRightarrow{\tau} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(n',v_2)\\
 & \xRightarrow{\texttt{r}.@\lambda x:\texttt{name}.r_0:=x.\texttt{true}} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(n',v_2(\lambda x:\texttt{name}\,.\,r_0:=x\,.\,\texttt{true}))\\
 & \xRightarrow{\tau} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(n',\texttt{true})\\
 & \xRightarrow{\texttt{r.discard}} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(n',())\\
 & \xRightarrow{\texttt{r}.?r_0} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(n',?r_0)\\
 & \xRightarrow{\tau} & \nu n\,.\,\nu n'\,.\,r_0:=n'\,.\,(n',n')
\end{array}$$

whereas when we try to emulate these transitions in $t_3$ we end with:

$$\nu n \,.\, \nu n' \,.\, \nu n'' \,.\, r_0 := n'' \,.\, (n', n'')$$

which are easily distinguished.

## 4.5 Completeness

Completeness for the $\nu$ref-calculus follows as it does for the $\lambda$-calculus and $\nu$-calculus. The contexts corresponding to the extra transitions for reference manipulation are immediate.

**Theorem 4.2 (completeness for $\nu$ref-calculus)**

$$\textit{If} \quad \Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_{ctx} t' : \sigma \quad \textit{then} \quad \Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx^{\circ} t' : \sigma.$$

## 4.6 Soundness

The subject of the next section of this paper is to establish that bisimulation is a congruence for the $\nu$ref-calculus, from which soundness immediately follows.

**Theorem 4.3 (soundness for $\nu$ref-calculus)**

$$\textit{If} \quad \Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx^{\circ} t' : \sigma \quad \textit{then} \quad \Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_{ctx} t' : \sigma.$$

The remainder of the paper is given over to proving this theorem.

## 4.7 Comments

In the definition of Standard ML [17] a model of references is presented using *configurations* of the form $(S,t)$ where $S$ is a state (a mapping of references to values). In this paper, we have included states in the syntax of terms, so the configuration $(\{\overline{r} \mapsto \overline{v}\}, t)$ is modelled by the term $\overline{r} := \overline{v} \,.\, t$. A similar use of syntax to model configuration is used in Ferreira, Hennessy and Jeffrey's [5] lts model of Reppy's [16] configuration-based model of Concurrent ML.

A recent paper of Pitts and Stark [15] also concerns itself with an operational study of locality using references. The difference here is that they use a language of integer references so that all equality tests between local names are definable from primitive operations, such as assignment and equality test on integers. The logical relations presented in [15] are much stronger than those of the $\nu$-calculus, largely because of their non-applicative nature. They are closer in spirit to context bisimulation than to the bisimulations presented here.

The use of global references in this paper is not intended as a serious language extension, rather a means of demonstrating the nature of private names in the $\nu$-calculus. We are certainly looking to extend this work to a setting in which the references themselves may be made local, or indeed other side-effects are used. For instance, we rely on a termination property in the proof of soundness which would fail in the presence of recursively defined functions. In order to relax the termination requirement we would be obliged to use side-effects with a certain amount of control, such as exceptions or call-cc. Unfortunately these side-effects alone do not enjoy other properties of references which we employ in our proof. For instance, we make explicit use of references as boolean flags in order to codify test values which behave differently on each instantiation:

$$(\Delta;;\mathrm{W} \vdash \lambda x : \sigma \to \sigma'.(xv_1, xv_2)$$

could actually be tested with a term which simulates the test

$$(\Delta;;\mathrm{W} \vdash (fv_1, gv_2))$$

for different $f$ and $g$, by applying a function which behaved like $f$ provided a certain flag held true and like $g$ otherwise simply by initialising the flag to true and lowering it after the first call. It is unclear at present whether such a property is crucial for obtaining a bisimulation congruence or just an artefact of the proof method we employ. This certainly bears further investigation.

# 5 Congruence of bisimulation for the νref-calculus

## 5.1 Active and passive names

To show that bisimulation is a congruence, we need to perform some analysis on the names generated by the prevalues.

For ref-closed terms, we define $\Delta$ to be *active* in $(\Delta, \Delta';;\mathrm{W} \vdash t : \sigma)$ if there is some sequence of transitions $\overline{\gamma}$ with free names in $\Delta'$ and some $n$ in $\Delta$ such that:

$$(\Delta, \Delta;;\mathrm{W} \vdash t : \sigma) \xRightarrow{\overline{\gamma}} \xRightarrow{\iota.n}$$

For ref-open terms $(\Gamma; \Delta, \Delta'; \mathrm{R}; \mathrm{W} \vdash t : \sigma)$, we define $\Delta$ to be active iff there exists some ref-closing substitution $\Delta', \Delta''; \mathrm{R} \cup \mathrm{R}'; \vdash [\overline{v}/\overline{x}] : \Gamma$ and ref-closing assignment $\Delta', \Delta'' \vdash \overline{r} := \overline{n} : \mathrm{R} \cup \mathrm{R}'$ such that $\Delta$ is active in $(\Delta, \Delta', \Delta'';;\mathrm{R} \cup \mathrm{R}' \cup \mathrm{W} \vdash \overline{r} := \overline{n} \,.\, t[\overline{v}/\overline{x}] : \sigma)$.

A name environment is *passive* if it is not active. We shall often use $\Pi$ to range over passive name environments.

Intuitively, if a name is passive in a term then it is a secret which the term never reveals. Unfortunately, this intuition does not hold for the ν-calculus, where we can construct terms:

$$\begin{aligned} F &\stackrel{\text{def}}{=} \lambda f : \mathtt{name} \to \mathtt{bool} \,.\, \nu n' \,.\, \mathtt{if}\ f(n') = f(n)\ \mathtt{then}\ n'\ \mathtt{else}\ n \\ f &\stackrel{\text{def}}{=} \lambda x : \mathtt{name} \,.\, x = n \end{aligned}$$

then we have $n$ is passive in $F$ and $f$, but is active in $F(f)$. This is not only very counterintuitive, but strikes at the heart of the problem for finding a fully abstract model for ν-calculus. The logical relations approach fails to be complete because when new names are generated it must be guessed whether the names are active (global) or passive (secret). The environment is then allowed to test with secret names provided they occur passively in the test value. During passive testing though the names may change their status from passive to active so no consistent guess can be made.

By comparison, in the νref-calculus, although $n$ is passive in $f$, it is *not* passive in $F$ since we have the transitions:

$$\begin{aligned} F &\xRightarrow{@\lambda x{:}\mathtt{name}.r := x.\mathtt{true}} \Rightarrow \nu n' \,.\, r := n \,.\, n' \\ &\xRightarrow{\mathtt{discard}} \nu n' \,.\, r := n \,.\, () \\ &\xRightarrow{?r} \Rightarrow \nu n' \,.\, r := n \,.\, n \\ &\xRightarrow{n} \nu n' \,.\, r := n \,.\, () \end{aligned}$$

We can now state the following key proposition, which is *not* true of the ν-calculus:

**Proposition 5.1** *If* $\Pi$ *is passive in* $(\Gamma;\Delta;\mathrm{R};\vdash v:\sigma)$ *and in* $(\Gamma,x:\sigma;\Delta;\mathrm{R};\mathrm{W}\vdash t:\sigma')$ *then* $\Pi$ *is passive in* $(\Gamma;\Delta;\mathrm{R};\mathrm{W}\vdash t[v/x]:\sigma')$.

**Proof:** See the appendix. □

## 5.2 Overt bisimulation

Motivated by the distinction between active and passive names, we present an alternative bisimulation for the $\nu$ref-calculus, which is more complex, but turns out to be more suitable for Howe-style proof.

This relation is inspired by Pitts and Stark's logical relations semantics although there are some subtle differences, which we will discuss later.

A type-indexed family of relations $R^{\Pi}$ (where if $\Delta;;\mathrm{W}\vDash t\ R^{\Pi}\ t':\sigma$ then $\Pi\subseteq\Delta$) is an *overt simulation* if:

1. We can complete the following diagram:

$$\begin{tikzcd}
(\Delta;;\mathrm{W}\vdash t_1:\sigma) \arrow[r, leftrightarrow, "R^{\Pi}"] \arrow[d, "\alpha"'] & (\Delta;;\mathrm{W}\vdash t_2:\sigma) \\
(\Delta';;\mathrm{W}\vdash t_1':\sigma') &
\end{tikzcd}$$

   where the free names of $\alpha$ are disjoint from $\Pi$ as:

$$\begin{tikzcd}
(\Delta;;\mathrm{W}\vdash t_1:\sigma) \arrow[r, leftrightarrow, "R^{\Pi}"] \arrow[d, "\alpha"'] & (\Delta;;\mathrm{W}\vdash t_2:\sigma) \arrow[d, Rightarrow, "\hat{\alpha}"'] \\
(\Delta';;\mathrm{W}\vdash t_1':\sigma') \arrow[r, leftrightarrow, "R^{\Pi}"] & (\Delta';;\mathrm{W}\vdash t_2':\sigma')
\end{tikzcd}$$

2. If $\Delta;;\mathrm{W}\vDash \nu n\,.\,p_1\ R^{\Pi}\ p_2:\sigma$ then either:

   (a) $p_2\equiv\nu n\,.\,p_3$ and $\Delta,n;;\mathrm{W}\vDash p_1\ R^{\Pi}\ p_3:\sigma$, or

   (b) $\Delta,n;;\mathrm{W}\vDash p_1\ R^{\Pi,n}\ p_2:\sigma$.

3. If $\Delta;;\mathrm{W}\vDash p_1\ R^{\Pi}\ p_2:\sigma$ and $(\Delta;;\mathrm{W}\vdash p_1:\sigma)\xrightarrow{\iota.n}$ then $n\notin\Pi$.

Let $\approx_o^{\Pi}$ be the largest overt bisimulation. We shall write $\approx_o$ when $\Pi$ is empty.

Intuitively, the definition of an overt bisimulation says:

1. $\approx_o^{\Pi}$ is a bisimulation,

2. If $\Delta;;\mathrm{W}\vDash\nu n\,.\,p_1\approx_o^{\Pi}p_2:\sigma$ then either:

   (a) $n$ is active in $p_1$, so $p_2$ has to match it by having an appropriate name binder (which is added to the active name environment $\Delta$), or

(b) $n$ is passive in $p_1$, so $p_2$ can match it by ignoring the name (which is added to the passive name environment $\Pi$).

3. $\Pi$ only contains passive names.

Overt bisimulation is a partial equivalence relation, and we can show a generalization of transitivity, as evidenced in the following lemma.

**Lemma 5.2** *If* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi_0,\Pi_1}{}^{\circ} \approx_o^{\Pi_0,\Pi_2} u : \sigma$ *then* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi_0,\Pi_1,\Pi_2}{}^{\circ} u : \sigma$.

**Proof:** It suffices to show the result for ref-closed terms, since we can then close up under all closing substitutions and ref-closing assignments. Define:

$$R^{\Pi'} = \left\{(t,u) \mid t \approx_o^{\Pi_0,\Pi_1} \approx_o^{\Pi_0,\Pi_2} u,\ \text{and}\ \Pi' = \Pi_0,\Pi_1,\Pi_2\right\}.$$

It is not difficult to check that $R$ forms an overt bisimulation. □

Since an overt simulation is a simulation, it is easy to see that $\approx_o$ is a finer relation than $\approx$. In fact, we can show that overt bisimulation coincides with bisimulation.

**Proposition 5.3** $\approx$ *is the same as* $\approx_o$

**Proof:** Define $\Delta,\Pi;;\mathrm{W} \vDash t_1 \approx^{\Pi} t_2 : \sigma$ whenever $\Delta;;\mathrm{W} \vDash \nu\Pi\,.\,t_1 \approx \nu\Pi\,.\,t_2 : \sigma$ and $\Pi$ is passive in $t_1$ and $t_2$. It is routine to verify that this is an overt bisimulation, and that it coincides with bisimulation when $\Pi$ is empty. □

## 5.3 Congruence of overt bisimulation

The proof that overt bisimulation is a congruence uses Howe's technique, but the definition of $\approx^{\bullet}$ is rather more complex, since we have to allow names to move between the passive and active name environments.

Define $\approx_o^{\Pi\bullet}$ by two rules:

$$\frac{\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t_1 \widehat{\approx_o^{\Pi\bullet}} t_2 \qquad \Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t_2 \approx_o^{\Pi,\Pi'\circ} t_3}{\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t_1 \approx_o^{\Pi,\Pi'\bullet} t_3}$$

and:

$$\frac{\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vDash t_1 \approx_o^{\Pi,n\bullet} t_2 \qquad \Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash \nu n\,.\,t_2 \approx_o^{\Pi\circ} t_3}{\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash \nu n\,.\,t_1 \approx_o^{\Pi\bullet} t_3}$$

This relation satisfies the usual [6] properties required of this relation, that is it contains the ˆ closure of itself and it contains overt bisimulation.

**Lemma 5.4** *If* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi_0,\Pi_1}{}^{\bullet} \approx_o^{\Pi_0,\Pi_2}{}^{\circ} u : \sigma$ *then* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi_0,\Pi_1,\Pi_2}{}^{\bullet} u : \sigma$.

**Proof:** Suppose $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi_0,\Pi_1}{}^{\bullet} t_0 \approx_o^{\Pi_0,\Pi_2} u : \sigma$ and proceed by induction on the structure of $t$. There are two main cases to consider based on how the Howe relation decomposes.

Firstly, suppose $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \widehat{\approx_o^{\Pi'\bullet}} t'_0 \approx_o^{\Pi_0,\Pi_1}{}^{\circ} t_0 : \sigma$. In this case we use Lemma 5.2 to establish $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t'_0 \approx_o^{\Pi_0,\Pi_1,\Pi_2}{}^{\circ} u : \sigma$ and then use the Howe relation to fold these terms back up to $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi_0,\Pi_1,\Pi_2}{}^{\bullet} u : \sigma$.

Secondly, consider the case in which $t$ is $\nu n \,.\, t'$ and the latter Howe rule is used. This means that there is some $t'_0$ such that

$$\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vDash t' \approx_o^{\Pi_0,\Pi_1,n^\bullet} t'_0 : \sigma \quad \text{and} \quad \Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash \nu n \,.\, t'_0 \approx_o^{\Pi_0,\Pi_1{}^\circ} t_0 : \sigma.$$

We can apply the induction hypothesis to

$$\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vDash t' \approx_o^{\Pi_0,\Pi_1,n^\bullet} t'_0 \approx_o^{\Pi_0,\Pi_1,\Pi_2,n} t'_0 : \sigma$$

to yield $\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vDash t' \approx_o^{\Pi_0,\Pi_1,\Pi_2,n^\bullet} t'_0 : \sigma$, and use Lemma 5.2 again to obtain

$$\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash \nu n \,.\, t'_0 \approx_o^{\Pi_0,\Pi_1,\Pi_2{}^\circ} u : \sigma.$$

From here we apply the second Howe rule to finish. □

First, we show some technical lemmas, which extend obvious properties of bisimulation on ref-closed terms to ref-open terms. All have routine proofs.

**Lemma 5.5** *If* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi^\circ} t' : \sigma$ *then* $\Pi$ *are passive in* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma$.

**Lemma 5.6** *If* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash \nu n \,.\, p \approx_o^{\Pi^\circ} p' : \sigma$ *then either:*

1. $p' \equiv \nu n \,.\, p''$ *and* $\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vDash p \approx_o^{\Pi^\circ} p'' : \sigma$, *or*
2. $\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vDash p \approx_o^{\Pi,n^\circ} p' : \sigma$.

**Lemma 5.7** *If* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash r := n \,.\, p \approx_o^{\Pi^\circ} p' : \sigma$ *then* $p' \equiv r := n \,.\, p''$ *and*

$$\Gamma;\Delta;\mathrm{R} \cup r;\mathrm{W} \setminus r \vDash p \approx_o^{\Pi^\circ} p'' : \sigma.$$

**Lemma 5.8** *If* $\Pi,\overline{n}$ *are passive in* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma$ *then* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \approx_o^{\Pi^\circ} \nu\overline{n} \,.\, t : \sigma$.

We can now verify that $\approx_o^{\Pi^\bullet}$ is substitutive on prevalues:

**Proposition 5.9** $\approx_o^{\Pi^\bullet}$ *is substitutive on prevalues, that is:*

$$\begin{array}{l}
\textit{if}\ \Gamma,x:\sigma;\Delta;\mathrm{R} \cup \overline{r};\mathrm{W}' \vDash t_1 \approx_o^{\Pi^\bullet} t_2 : \sigma' \\
\textit{and}\ \Gamma;\Delta;\mathrm{R};\mathrm{W} \cup \overline{r} \vDash d_1 \,.\, v_1 \approx_o^{\Pi^\bullet} d_2 \,.\, v_2 : \sigma \\
\textit{then}\ \Gamma;\Delta;\mathrm{R};\mathrm{W} \cup \mathrm{W}' \vDash d_1 \,.\, t_1[v_1/x] \approx_o^{\Pi^\bullet} d_2 \,.\, t_2[v_2/x] : \sigma'
\end{array}$$

**Proof:** We use induction on the size of the declaration $d_1$.

**Case:** $d_1$ is empty.

In this case we know that $\Gamma;\Delta;\mathrm{R};\ \vDash v_1 \approx_o^{\Pi^\bullet} d_2.v_2 : \sigma$. It is fairly easy to see that $d_2$ must be of the form $\nu\overline{n}$, that is it contains no assignments, otherwise this would break the hypothesis that $d_2.v_2$ is bisimilar to a value, which certainly has no immediate assignments. Suppose then that $d_2 = \nu\overline{n}$ so that:

$$\Gamma;\Delta;\mathrm{R};\ \vDash v_1 \widehat{\approx_o^{\Pi'^\bullet}} v_3 \approx_o^{\Pi^\circ} \nu\overline{n} \,.\, v_2 : \sigma$$

for some value $v_3$. By Lemma 5.6 we have:

$$\Gamma;\Delta,\overline{n};\mathrm{R};\vDash v_3 \approx_o^{\Pi,\overline{n}^\circ} v_2 : \sigma$$

so by weakening and definition of $\approx_o^{\Pi^\bullet}$ we have:

$$\Gamma;\Delta,\overline{n};\mathrm{R};\vDash v_1 \approx_o^{\Pi,\overline{n}^\bullet} v_2 : \sigma$$

so by substitutivity on values we obtain:

$$\Gamma;\Delta,\overline{n};\mathrm{R};\mathrm{W}' \vDash t_1[v_1/x] \approx_o^{\Pi,\overline{n}^\bullet} t_2[v_2/x] : \sigma$$

By by Lemma 5.5 and weakening, $\Pi,\overline{n}$ are passive in $\Gamma,x:\sigma;\Delta,\overline{n};\mathrm{R};\mathrm{W}' \vdash t_2 : \sigma'$ and $\Gamma;\Delta,\overline{n};\mathrm{R};\vdash v_2 : \sigma$, so by weakening and Proposition 5.1, we know that $\Pi,\overline{n}$ are passive in $\Gamma;\Delta,\overline{n};\mathrm{R};\mathrm{W}' \vdash t_2[v_2/x] : \sigma$. This means that by Lemma 5.8:

$$\Gamma;\Delta,\overline{n};\mathrm{R};\mathrm{W}' \vDash t_2[v_2/x] \approx_o^{\Pi,\overline{n}} \nu\overline{n}\,.\,t_2[v_2/x] : \sigma'$$

Therefore, by weakening and definition of $\approx_o^{\Pi^\bullet}$, we can conclude:

$$\Gamma;\Delta;\mathrm{R};\mathrm{W}' \vDash t_1[v_1/x] \approx_o^{\Pi^\bullet} \nu\overline{n}\,.\,t_2[v_2/x] : \sigma'$$

**Case:** $d_1$ is $r := n_0\,.\,d_3$.

In must be the case that:

$$\Gamma;\Delta;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash r := n_0\,.\,d_3.v_1 \widehat{\approx_o^{\Pi'^\bullet}} r := n_0\,.\,d_4.v_3 \approx_o^{\Pi^\circ} d_2.v_2 : \sigma$$

By Lemma 5.7 we have:

$$d_2.v_2 \equiv r := n_0\,.\,d_5.v_2 \qquad \Gamma;\Delta;\mathrm{R}\cup r;\mathrm{W}\cup\overline{r}\setminus r \vDash d_4.v_3 \approx_o^{\Pi^\circ} d_5.v_2 : \sigma$$

We can therefore apply the induction hypothesis to obtain:

$$\Gamma;\Delta;\mathrm{R}\cup r;\mathrm{W}\setminus r\cup\mathrm{W}' \vDash d_3.t_1[v_1/x] \approx_o^{\Pi^\bullet} d_5.t_2[v_2/x] : \sigma'$$

From here we use weakening and the definition of $\approx_o^{\Pi^\bullet}$ to get:

$$\Gamma;\Delta;\mathrm{R};\mathrm{W}\cup\mathrm{W}' \vDash r := n_0\,.\,d_3.t_1[v_1/x] \approx_o^{\Pi^\bullet} r := n_0\,.\,d_5.t_2[v_2/x] : \sigma'$$

which is exactly what we require.

**Case:** $d_1$ is $\nu n_0\,.\,d_3$.

We can decompose the Howe relation in two ways so we take each in turn. Firstly, suppose:

$$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_3.v_1 \approx_o^{\Pi'^\bullet} t_0 : \sigma \qquad \Gamma;\Delta;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash \nu n_0\,.\,t_0 \approx_o^{\Pi^\circ} d_2.v_2 : \sigma$$

and suppose, without loss of generality that $t_0$ is actually a prevalue, $d_4.v_3$. By Lemma 5.6, there are two subcases to consider:

1. $d_2.v_2 \equiv \nu n_0 \,.\, d_5.v_4$ and $\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_4.v_3 \approx_o^{\Pi^\circ} d_5.v_4 : \sigma$.
   In this case, Lemma 5.4 gives us:

   $$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_3.v_1 \approx_o^{\Pi^\bullet} d_5.v_4 : \sigma$$

   so by induction:

   $$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\mathrm{W}' \vDash d_3.t_1[v_1/x] \approx_o^{\Pi^\bullet} d_5.t_2[v_4/x] : \sigma'$$

   which gives us:

   $$\Gamma;\Delta;\mathrm{R};\mathrm{W}\cup\mathrm{W}' \vDash \nu n_0 \,.\, d_3.t_1[v_1/x] \,\widehat{\approx_o^{\Pi^\bullet}}\, \nu n_0 \,.\, d_5.t_2[v_4/x] \equiv d_2.t_2[v_2/x] : \sigma'$$

   and so we can use the definition of $\approx_o^{\Pi^\bullet}$ to conclude.
2. $\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_4.v_3 \approx_o^{\Pi,n_0{}^\circ} d_2.v_2 : \sigma$.
   In this case, Lemma 5.4 gives us:

   $$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_3.v_1 \approx_o^{\Pi,n_0{}^\bullet} d_2.v_2 : \sigma$$

   so by induction:

   $$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\mathrm{W}' \vDash d_3.t_1[v_1/x] \approx_o^{\Pi,n_0{}^\bullet} d_2.t_2[v_2/x] : \sigma'$$

   and so:

   $$\Gamma;\Delta;\mathrm{R};\mathrm{W}\cup\mathrm{W}' \vDash \nu n_0 \,.\, d_3.t_1[v_1/x] \,\widehat{\approx_o^{\Pi^\bullet}}\, \nu n_0 \,.\, d_2.t_2[v_2/x] \approx_o^{\Pi^\circ} d_2.t_2[v_2/x] : \sigma'$$

   (where the latter equivalence holds since $n_0$ does not occur free in $d_2.t_2[v_2/x]$) and so we can use the definition of $\approx_o^{\Pi^\bullet}$ to conclude.

Alternatively, suppose that we have the decomposition

$$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_3.v_1 \approx_o^{\Pi,n_0{}^\bullet} t_0 : \sigma \qquad \Gamma;\Delta;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash \nu n_0 \,.\, t_0 \approx_o^{\Pi^\circ} d_2.v_2 : \sigma$$

and suppose, without loss of generality that $t_0$ is actually a prevalue, $d_4.v_3$. Again, by Lemma 5.6, there are two subcases to consider:

1. $d_2.v_2 \equiv \nu n_0 \,.\, d_5.v_4$ and $\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_4.v_3 \approx_o^{\Pi^\circ} d_5.v_4 : \sigma$.
   In this case, Lemma 5.4 gives us:

   $$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r} \vDash d_3.v_1 \approx_o^{\Pi,n_0{}^\bullet} d_5.v_4 : \sigma$$

   so by induction:

   $$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\mathrm{W}' \vDash d_3.t_1[v_1/x] \approx_o^{\Pi,n_0{}^\bullet} d_5.t_2[v_4/x] : \sigma'$$

   and since:

   $$\nu n_0 \,.\, d_5.t_2[v_4/x] \equiv d_2.t_2[v_2/x]$$

   and we can use the definition of $\approx_o^{\Pi^\bullet}$ to conclude.

2. $\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r}\vDash d_4.v_3 \approx_o^{\Pi,n_0{}^\circ} d_2.v_2 : \sigma$.
   In this case, Lemma 5.4 gives us:

$$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\overline{r}\vDash d_3.v_1 \approx_o^{\Pi,n_0{}^\bullet} d_2.v_2 : \sigma$$

   so by induction:

$$\Gamma;\Delta,n_0;\mathrm{R};\mathrm{W}\cup\mathrm{W}'\vDash d_3.t_1[v_1/x] \approx_o^{\Pi,n_0{}^\bullet} d_2.t_2[v_2/x] : \sigma'$$

   and so:

$$\Gamma;\Delta;\mathrm{R};\mathrm{W}\cup\mathrm{W}'\vDash \nu n_0\,.\,d_3.t_1[v_1/x] \mathrel{\widehat{\approx_o^{\Pi\bullet}}} \nu n_0\,.\,d_2.t_2[v_2/x] \approx_o^{\Pi^\circ} d_2.t_2[v_2/x] : \sigma'$$

   (where the latter equivalence holds since $n_0$ does not occur free in $d_2.t_2[v_2/x]$) and so we can use the definition of $\approx_o^{\Pi\bullet}$ to conclude. □

We can then show that $\approx^\bullet$ is a bisimulation up to $(\equiv,=)$ [19], from which it is routine to show that overt bisimulation, and hence bisimulation, is a congruence.

**Proposition 5.10** *On ref-closed terms, $\approx_o{}^\bullet$ is an overt bisimulation up to $(\equiv,=)$.*

**Proof:** Take $\Delta;;\mathrm{W}\vDash t\approx_o^{\Pi\bullet} u:\sigma$. It is fairly easy to see that the latter two conditions for being an overt bisimulation are satisfied, and we concentrate on showing that any transition of $t$ can be matched by a transition of $u$.

We will show a slightly more general result, which is that if:

$$\Delta;\overline{r};\mathrm{W}\setminus\overline{r}\vDash t\approx_o^{\Pi\bullet} u:\sigma \qquad (\Delta;;\mathrm{W}\vdash\overline{r}:=\overline{n}\,.\,t:\sigma)\xrightarrow{\alpha}(\Delta,\Delta';;\mathrm{W}\vdash t':\sigma)$$

then we can find $u'$ such that:

$$\Delta,\Delta';;\mathrm{W}\vDash t'\equiv\approx_o^{\Pi\bullet} u':\sigma' \qquad (\Delta;;\mathrm{W}\vdash\overline{r}:=\overline{n}\,.\,u:\sigma)\overset{\hat{\alpha}}{\Longrightarrow}(\Delta,\Delta';;\mathrm{W}\vdash u':\sigma')$$

In particular, note that we can take $\overline{r}$ to be empty and get the desired result.

We proceed by induction on the proof of $\approx_o^{\Pi\bullet}$. For most of the cases this is a completely standard rule induction so we only detail the situations which vary from the usual approach.

In fact, we shall prove this property for a variant transition system, where we add extra transitions $(\Delta;\mathrm{R}\vdash v:\sigma)\xrightarrow{\mathtt{id}}(\Delta;\mathrm{R}\vdash v:\sigma)$. It is easy to see that adding these transitions makes no difference to the semantic theory, and it makes this proof slightly easier.

**Case:** Suppose the transition in question is:

$$(\Delta;;\mathrm{W}\vdash\overline{r}:=\overline{n}\,.\,\nu n_0\,.\,t:\sigma)\xrightarrow{\iota.\nu n_0}(\Delta,n_0;;\mathrm{W}\vdash\overline{r}:=\overline{n}\,.\,t':\sigma')$$

derived from:

$$(\Delta,n_0;\overline{r};\mathrm{W}\setminus\overline{r}\vdash t:\sigma)\xrightarrow{\iota.n_0}(\Delta,n_0;\overline{r};\mathrm{W}\setminus\overline{r}\vdash t':\sigma')$$

We know that $\Delta;\overline{r};\mathrm{W}\setminus\overline{r}\vDash \nu n_0\,.\,t\approx_o^{\Pi\bullet} u:\sigma$ so we consider how this could be. There are two evident ways of establishing these terms to be in the $\approx_o^{\Pi\bullet}$ relation, the latter of which applies when the private name $n_0$ is passive. This of course is not the case here so it must be that:

$$\Delta,n_0;\overline{r};\mathrm{W}\setminus\overline{r}\vDash t\approx_o^{\Pi'\bullet} t_0:\sigma \qquad \Delta;\overline{r};\mathrm{W}\setminus\overline{r}\vDash \nu n_0\,.\,t_0\approx_o^{\Pi^\circ} u:\sigma$$

Since we have:

$$(\Delta, n_0;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, t : \sigma) \xrightarrow{\iota.n_0} (\Delta, n_0;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, t' : \sigma')$$

we can apply induction to get that:

$$\Delta, n_0;;\mathrm{W} \vDash \overline{r} := \overline{n} \,.\, t' \equiv\approx_o^{\Pi^\bullet} t_0' : \sigma'$$

and

$$(\Delta, n_0;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, t_0 : \sigma) \overset{\iota.n_0}{\Longrightarrow} (\Delta, n_0;;\mathrm{W} \vdash t_0' : \sigma')$$

so:

$$(\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, \nu n_0 \,.\, t_0 : \sigma) \equiv (\Delta;;\mathrm{W} \vdash \nu n_0 \,.\, \overline{r} := \overline{n} \,.\, t_0 : \sigma) \overset{\iota.\nu n_0}{\Longrightarrow} (\Delta, n_0;\mathrm{W} \vdash t_0' : \sigma')$$

and since $\Delta;\overline{r};\mathrm{W}\setminus\overline{r} \vDash \nu n_0 \,.\, t_0 \approx_o^{\Pi^\circ} u : \sigma$ we can find $u'$ such that:

$$\Delta, n_0;;\mathrm{W} \vDash t_0' \approx_o^{\Pi} u' : \sigma' \qquad (\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, u : \sigma) \overset{\iota.\nu n_0}{\Longrightarrow} (\Delta, n_0;;\mathrm{W} \vdash u' : \sigma')$$

We use Lemma 5.4 to finish.

**Case:** Suppose we have the transition:

$$(\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, \nu n_0 \,.\, t : \sigma) \xrightarrow{\gamma} (\Delta, \Delta';;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, \nu n_0 \,.\, t' : \sigma')$$

derived from:

$$(\Delta, n_0;\overline{r};\mathrm{W}\setminus\overline{r} \vdash t : \sigma) \xrightarrow{\gamma} (\Delta, n_0, \Delta';\overline{r};\mathrm{W}\setminus\overline{r} \vdash t' : \sigma')$$

and suppose that $\Delta;\overline{r};\mathrm{W}\setminus\overline{r} \vDash \nu n_0 \,.\, t \approx_o^{\Pi^\bullet} u : \sigma$ arises from:

$$\Delta, n_0;\overline{r};\mathrm{W}\setminus\overline{r} \vDash t \approx_o^{\Pi,n_0\bullet} t_0 : \sigma \qquad \Delta;\overline{r};\mathrm{W}\setminus\overline{r} \vDash \nu n_0 \,.\, t_0 \approx_o^{\Pi^\circ} u : \sigma$$

Since we have:

$$(\Delta, n_0;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, t : \sigma) \xrightarrow{\gamma} (\Delta, n_0, \Delta';;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, t' : \sigma')$$

we can apply induction to get that:

$$\Delta, n_0, \Delta';;\mathrm{W} \vDash \overline{r} := \overline{n} \,.\, t' \equiv\approx_o^{\Pi,n_0\bullet} t_0' : \sigma'$$

and

$$(\Delta, n_0;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, t_0 : \sigma) \overset{\gamma}{\Longrightarrow} (\Delta, n_0, \Delta';;\mathrm{W} \vdash t_0' : \sigma')$$

so:

$$(\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, \nu n_0 \,.\, t_0 : \sigma) \equiv (\Delta;;\mathrm{W} \vdash \nu n_0 \,.\, \overline{r} := \overline{n} \,.\, t_0 : \sigma) \overset{\gamma}{\Longrightarrow} (\Delta, \Delta';;\mathrm{W} \vdash \nu n_0 \,.\, t_0' : \sigma')$$

and since $\Delta;\overline{r};\mathrm{W}\setminus\overline{r} \vDash \nu n_0 \,.\, t_0 \approx_o^{\Pi^\circ} u : \sigma$ we can find $u'$ such that:

$$\Delta, \Delta';;\mathrm{W} \vDash \nu n_0 \,.\, t_0' \approx_o^{\Pi} u' : \sigma' \qquad (\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n} \,.\, u : \sigma) \overset{\gamma}{\Longrightarrow} (\Delta, \Delta';;\mathrm{W} \vdash u' : \sigma')$$

Using the second rule for the Howe relation gives:

$$\Delta, \Delta';;\mathrm{W} \vDash \overline{r} := \overline{n} \,.\, \nu n_0 \,.\, t' \equiv \nu n_0 \,.\, \overline{r} := \overline{n} \,.\, t' \equiv\approx_o^{\Pi^\bullet} u' : \sigma'$$

and so we have the desired result.

**Case:** Suppose $(\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n}\,.\,t : \sigma) \xrightarrow{\tau} (\Delta;;\mathrm{W} \vdash t' : \sigma)$. The most interesting case occurs when this is an instance of the let block $\beta$-reduction, that is:

$$(\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n}\,.\,\texttt{let}\,x\;=\;d_1.v_1\;\texttt{in}\,t_1 : \sigma) \xrightarrow{\tau} (\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n}\,.\,d_1.t_1[v_1/x] : \sigma)$$

We know, by definition of the Howe relation, that there exists some $t_0, t_2$ such that:

$$\Delta;\overline{r};\mathrm{W}' \cup \overline{r}' \vDash d_1.v_1 \approx_o^{\Pi'^{\bullet}} t_0 : \sigma' \qquad x : \sigma';\Delta;\overline{r} \cup \overline{r}';\mathrm{W}'' \vDash t_1 \approx_o^{\Pi'^{\bullet}} t_2 : \sigma$$
$$\Delta;\overline{r};\mathrm{W} \setminus \overline{r} \vDash \texttt{let}\,x\;=\;t_0\;\texttt{in}\,t_1 \approx_o^{\Pi} u : \sigma$$

for some $\Pi' \subseteq \Pi$, and $\mathrm{W}' \cup \mathrm{W}'' = \mathrm{W} \setminus \overline{r}$. Since:

$$(\Delta;;\mathrm{W}' \cup \overline{r}' \vdash \overline{r} := \overline{n}\,.\,d_1.v_1 : \sigma') \xrightarrow{\texttt{id}} (\Delta;;\mathrm{W}' \cup \overline{r}' \vdash \overline{r} := \overline{n}\,.\,d_1.v_1 : \sigma')$$

by induction we can find $d_2$ and $v_2$ such that:

$$(\Delta;;\mathrm{W}' \cup \overline{r}' \vdash \overline{r} := \overline{n}\,.\,t_0 : \sigma') \xRightarrow{\texttt{id}} (\Delta;;\mathrm{W}' \cup \overline{r}' \vdash d_2.v_2 : \sigma')$$

and

$$\Delta;;\mathrm{W}' \cup \overline{r}' \vDash \overline{r} := \overline{n}\,.\,d_1.v_1 \equiv\approx_o^{\Pi'^{\bullet}} d_2.v_2 : \sigma'$$

and so:

$$\begin{array}{rcl}
(\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n}\,.\,\texttt{let}\,x\;=\;t_0\;\texttt{in}\,t_1 : \sigma) & \equiv & (\Delta;;\mathrm{W} \vdash \texttt{let}\,x\;=\;\overline{r} := \overline{n}\,.\,t_0\;\texttt{in}\,t_1 : \sigma) \\
 & \Rightarrow & (\Delta;;\mathrm{W} \vdash \texttt{let}\,x\;=\;d_2.v_2\;\texttt{in}\,t_1 : \sigma) \\
 & \xrightarrow{\tau} & (\Delta;;\mathrm{W} \vdash d_2.t_1[v_2/x] : \sigma)
\end{array}$$

and so we can find a $u'$ such that:

$$\Delta;;\mathrm{W} \vDash d_2.t_1[v_2/x] \approx_o^{\Pi} u' : \sigma' \qquad (\Delta;;\mathrm{W} \vdash \overline{r} := \overline{n}\,.\,u : \sigma) \Rightarrow (\Delta;;\mathrm{W} \vdash u' : \sigma)$$

We can now apply Proposition 5.9 to observe that:

$$\Delta;;\mathrm{W} \vDash \overline{r} := \overline{n}\,.\,d_1.t_1[v_1/x] \equiv\approx_o^{\Pi'^{\bullet}} d_2.t_2[v_2/x] \approx_o^{\Pi} u' : \sigma$$

and we use Lemma 5.4 to finish.

**Case:** We demonstrate how the Howe relation is preserved by structural congruence. In fact, we know by Lemma 4.1 that we need only consider the heating rules and show that if $t \Rrightarrow t'$ and $t \approx_o{}^{\bullet} u$ then $t' \approx_o{}^{\bullet} u$ also. We use the following case as a typical example. Suppose:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash r := n\,.\,\texttt{let}\,x\;=\;t\;\texttt{in}\,t' : \sigma) \Rrightarrow (\Delta;\mathrm{R};\mathrm{W} \vdash \texttt{let}\,x\;=\;r := n\,.\,t\;\texttt{in}\,t' : \sigma)$$

We know that there is some $t_0$ such that:

$$\Delta;\mathrm{R} \cup r;\mathrm{W} \setminus r \vDash \texttt{let}\,x\;=\;t\;\texttt{in}\,t' \approx_o^{\Pi'^{\bullet}} t_0 : \sigma \qquad \Delta;\mathrm{R};\mathrm{W} \vDash r := n\,.\,t_0 \approx_o^{\Pi^{\circ}} u : \sigma$$

where $\Pi' \subseteq \Pi$. We decompose the former further to obtain terms $t_0'$ and $t_0''$ and $\mathrm{W}, \mathrm{W}''$ such that $\mathrm{W}' \cup \mathrm{W}'' = \mathrm{W}$ and:

$$\Delta;\mathrm{R} \cup r;\mathrm{W}' \setminus r \vDash t \approx_o^{\Pi''^{\bullet}} t_0' : \sigma' \qquad x : \sigma;\Delta;\mathrm{R} \cup r;\mathrm{W}'' \vDash t' \approx_o^{\Pi''^{\bullet}} t_0'' : \sigma$$
$$\Delta;\mathrm{R} \cup r;\mathrm{W} \setminus r \vDash \texttt{let}\,x\;=\;t_0'\;\texttt{in}\,t_0'' \approx_o^{\Pi'^{\circ}} t_0 : \sigma$$

We observe that $\approx_o$ is easily seen to be congruent with respect to assignment so we can obtain:

$$\begin{array}{rcl}
\Delta;\mathrm{R};\mathrm{W} \vDash \mathtt{let}\, x \;=\; r := n\,.\,t\;\mathtt{in}\; t' & \approx_o^{\Pi''^{\bullet}} & \mathtt{let}\, x \;=\; r := n\,.\,t'_0\;\mathtt{in}\; t''_0 \\
 & \equiv & r := n\,.\,\mathtt{let}\, x \;=\; t'_0\;\mathtt{in}\; t''_0 \\
 & \approx_o^{\Pi'^{\circ}} & r := n\,.\,t_0 \\
 & \approx_o^{\Pi^{\circ}} & u : \sigma
\end{array}$$

and so we are finished. □

**Corollary 5.11** $\approx^{\circ}$ *is a congruence for the* $\nu$*ref-calculus.*

## 5.4 Comments

Earlier in the paper we described the logical relations of [14] as an *overt* proof technique for $\nu$-calculus. We can see now that there are similarities between our overt bisimulation and the logical relations. In particular, both techniques make use of a predicate to track the private names of terms under test. In the logical relations the predicate takes the form of a partial injection between the free names of terms—secret names are those not in the domain (or range) of this injection.

The key point is that when a new name is to be generated, in both overt bisimulation and logical relations, it must be guessed whether the name will be active or passive. Where the two approaches differ greatly however is in the tests allowed in the $@v$ transitions. Logical relations allow the environment to use secrets passively, even though, morally, they have no knowledge of them. Overt bisimulations forbid this and insist that a secret is a secret and, until the environment knows it, no testing can be made with it at all. It would be interesting to show that these two approaches coincide for the $\nu$ref-calculus. In light of Proposition 5.1 there is strong evidence to suggest that they do, but we leave this as an open problem.

# Appendix : Passivity is preserved by substitution

Let us first generalise the notion of passivity to allow a restriction on the names used for testing. This will enable us to express our proof invariant precisely. We say that $\overline{n}$ are passive from $\Theta$ in $\Delta;;\mathrm{W} \vdash t : \sigma$ if for all sequences of transitions $(\Delta;;\mathrm{W} \vdash t : \sigma) \overset{\overline{\gamma}}{\Longrightarrow} \Delta,\Delta';;\mathrm{W} \vdash t' : \sigma')$ such that $fn(\overline{\gamma}) \cap \Delta \subseteq \Theta$, there is no transition $(\Delta,\Delta';;\mathrm{W} \vdash t' : \sigma') \overset{\iota.n}{\Longrightarrow}$ . This says that no matter how a term is provoked using only names from $\Theta$ (or fresh names), the term will not release the identity of the name $n$.

We then lift this definition of passivity to open terms by defining $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma)$ if for all closing value instantiations $\Psi,\Delta';\mathrm{R}\cup\mathrm{R}';\vdash [\overline{v}/\overline{x}] : \Gamma$ and all $\Psi,\Delta' \vdash \overline{r} := \overline{n}_0 : \mathrm{R}$ and $\Theta,\Delta' \vdash \overline{r}' := \overline{n}_0{}' : \mathrm{R}'$ we have $\overline{n}$ passive from $\Theta$ in

$$(\Delta,\Delta';;\mathrm{R}\cup\mathrm{R}'\cup\mathrm{W} \vdash \overline{r} := \overline{n}_0\,.\,\overline{r}' := \overline{n}'_0\,.\,t[\overline{v}/\overline{x}] : \sigma).$$

We note immediately that the notion of passivity introduced earlier is simply an instance of this general definition in that $\overline{n}$ are passive in $(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma)$ if and only if $\overline{n}$ are passive from $\Delta \subseteq \Delta$ in $(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma)$.

Extend the definition of passivity from terms to capture-free evaluation contexts $\mathcal{E}$ with typing:

$$\frac{\Gamma;\Delta;R';W' \vdash \cdot : \sigma'}{\Gamma;\Delta;R;W \vdash \mathcal{E}[\cdot] : \sigma}$$

Define $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $\mathcal{E}$ iff $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $\Gamma;\Delta,\Delta';R;W \vdash \mathcal{E}[t] : \sigma$ for any $\Gamma;\Psi,\Delta';R';W' \vdash t : \sigma'$.

A partial equivalence relation (PER) on names $R$ is a transitive, symmetric relation. We shall write $R : \overline{n} \leftrightarrow \overline{n}$ whenever $\overline{n}$ is the domain of $R$.

Given a PER $R$: $\overline{n} \leftrightarrow \overline{n}$, define the *passive bisimulation* $\sim_R$ to be the type-indexed relation given by:

- If $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $(\Gamma;\Delta;R;W \vdash t : \sigma)$ then $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R;W \vDash t \sim_R t : \sigma$.
- If $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R;\vDash v \sim_R v' : \sigma$ and $\Gamma,x:\sigma;\Psi \subseteq \Theta \subseteq \Delta;R;W \vDash t \sim_R t' : \sigma'$ then $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R;W \vDash t[v/x] \sim_R t'[v'/x] : \sigma'$.
- If $\Gamma;\Psi \subseteq \Theta,n \subseteq \Delta,n;R;W \vDash t \sim_R t' : \sigma$
  or $\Gamma;\Psi \subseteq \Theta \subseteq \Delta,n;R;W \vDash t \sim_R t' : \sigma$
  then $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R;W \vDash \nu n\,.\,t \sim_R \nu n\,.\,t' : \sigma$.
- If $\mathcal{E}$ is an evalutation context with $\overline{n}$ passive from $\Psi \subseteq \Theta$ with typing:

  $$\frac{\Gamma;\Delta;R';W' \vdash \cdot : \sigma'}{\Gamma;\Delta;R;W \vdash \mathcal{E}[\cdot] : \sigma}$$

  and $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R';W' \vDash t \sim_R t' : \sigma'$
  then $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R;W \vDash \mathcal{E}[t] \sim_R \mathcal{E}[t'] : \sigma$.
- If $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R;W \vDash t \sim_R t' : \sigma$ and $[\overline{n}'/\overline{n}] \subseteq R$ is a bijective substitution then $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;R;W \vDash t \sim_R t'[\overline{n}/\overline{n}'] : \sigma$.

We can now give a proof of the main proposition, using properties of $\sim_R$ we will show later.

**Proposition A.1** *If* $\Pi$ *is passive in* $(\Gamma;\Delta;R;\vdash v:\sigma)$ *and in* $(\Gamma,x:\sigma;\Delta;R;W \vdash t:\sigma')$ *then* $\Pi$ *is passive in* $(\Gamma;\Delta;R;W \vdash t[v/x] : \sigma')$.

**Proof:** Assume without loss of generality that $\Gamma$ and R are empty. Let $\Pi'$ be fresh names, and let $R : (\Pi,\Pi') \leftrightarrow (\Pi,\Pi')$ be the PER generated by equating $\Pi$ and $\Pi'$. We can use Proposition A.3 to complete the diagram: (note that the $\Psi$ and $\Theta$ component of the type index are initially both $\Delta$).

$$\begin{tikzcd}
(\Delta,\Delta';;W \vdash t[v/x] : \sigma') \arrow[r, leftrightarrow, "\sim_R"] \arrow[d, Rightarrow, "\overline{\gamma}"'] & (\Delta,\Delta';;W \vdash t[v[\Pi'/\Pi]/x] : \sigma') \\
\cdot &
\end{tikzcd}$$

(where $\Pi, \Pi'$ are not free in $\overline{\gamma}$) as:

$$\begin{tikzcd}
(\Delta, \Delta';; \mathrm{W} \vdash t[v/x] : \sigma') \arrow[r, leftrightarrow, "\sim_R"] \arrow[d, Rightarrow, "\overline{\gamma}"'] & (\Delta, \Delta';; \mathrm{W} \vdash t[v[\Pi'/\Pi]/x] : \sigma') \arrow[d, Rightarrow, "\overline{\gamma}"'] \\
\cdot \arrow[r, leftrightarrow, "\sim_R"] & \cdot
\end{tikzcd}$$

so by Proposition A.2 we have that $\Pi$ is passive in $t[v/x]$. □

This proof relies on the fact that if $t \sim_R u$ then $t$ cannot perform a $n$ transition for any $n$ in the domain of $R$:

**Proposition A.2** *For any $R : \overline{n} \leftrightarrow \overline{n}$, if $\Delta; \mathrm{R}; \mathrm{W} \vDash t \sim_R u : \sigma$ and $(\Delta; \mathrm{R}; \mathrm{W} \vdash t : \sigma) \xrightarrow{\iota.n}$ then $n \notin \overline{n}$.*

**Proof:** A straightforward induction on the proof of $\sim_R$. □

It now suffices to show that passive bisimulation is a strong bisimulation. We do this by introducing an auxiliary *one-step passive bisimulation* relation $\sim_R^1$ defined on closed terms such that if $\Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vDash t \sim_R^1 u : \sigma$ then:

1. $\Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vDash t \sim_R t : \sigma$, and
2. we can complete the following diagram:

$$\begin{tikzcd}
(\Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vdash t : \sigma) \arrow[r, leftrightarrow, "\sim_R^1"] \arrow[d, "\alpha"'] & (\Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vdash u : \sigma) \\
(\Psi \subseteq \Theta, \Delta' \subseteq \Delta, \Delta'; \mathrm{R}; \mathrm{W} \vdash t' : \sigma) &
\end{tikzcd}$$

as

$$\begin{tikzcd}
(\Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vdash t : \sigma) \arrow[r, leftrightarrow, "\sim_R^1"] \arrow[d, "\alpha"'] & (\Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vdash u : \sigma) \arrow[d, "\alpha"'] \\
(\Psi \subseteq \Theta, \Delta' \subseteq \Delta, \Delta'; \mathrm{R}; \mathrm{W} \vdash t' : \sigma) \arrow[r, leftrightarrow, "\sim_R"] & (\Psi \subseteq \Theta, \Delta' \subseteq \Delta, \Delta'; \mathrm{R}; \mathrm{W} \vdash u' : \sigma)
\end{tikzcd}$$

when $\overline{n}$ are not free in $\alpha$.

and on open terms as:

$$\begin{aligned}
&\Gamma; \Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vDash t \sim_R^1 u : \sigma \\
&\quad \text{iff for all } \Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \vDash [\overline{v}/\overline{x}] \sim_R^1 [\overline{w}/\overline{x}] : \Gamma \\
&\qquad \text{we have } \Psi \subseteq \Theta \subseteq \Delta; \mathrm{R}; \mathrm{W} \vDash t[\overline{v}/\overline{x}] \sim_R^1 u[\overline{w}/\overline{x}] : \sigma
\end{aligned}$$

We shall show that one-step passive bisimulation coincides with passive bisimulation, and hence that passive bisimulation is a strong bisimulation:

**Proposition A.3** *If* $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R};\mathrm{W} \vDash t \sim_R u : \sigma$ *then* $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R};\mathrm{W} \vDash t \sim^1_R u : \sigma$.

**Proof:** We proceed by induction on the proof of $\sim_R$ and notice that the type index $\Psi \subseteq \Theta$ plays only a small, but crucial, role in this proof so we will leave it implicit for the sake of readability and only draw attention to it in the appropriate places.

**Case:** $\overline{n}$ are passive in $(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma)$, and $t = u$, so we use Proposition A.4.

**Case:** $t = t'[v/x]$ and $u = u'[w/x]$ where $\Gamma;\Delta;\mathrm{R}; \vDash v \sim_R w : \sigma'$ and $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t' \sim_R w : \sigma'$. By induction, we get that $\Gamma;\Delta;\mathrm{R}; \vDash v \sim^1_R w : \sigma'$ and $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t' \sim^1_R w : \sigma'$, so by the definition of $\sim^1_R$ on open terms, we have that $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t'[v/x] \sim^1_R u'[w/x] : \sigma$, as required.

**Case:** $t = \nu n\,.\,t'$ and $u = \nu n\,.\,u'$ where $\Gamma;\Psi \subseteq \Theta' \subseteq \Delta, n;\mathrm{R};\mathrm{W} \vDash t' \sim_R u' : \sigma$ with $\Theta'$ possibly containing $n$. The induction hypothesis ensures that $\Gamma;\Psi \subseteq \Theta' \subseteq \Delta, n;\mathrm{R};\mathrm{W} \vDash t' \sim^1_R u' : \sigma$. We notice that any transition from $\nu n\,.\,t'$ must originate from $t'$ thus $u'$ and hence $\nu n\,.\,u'$ will match such a transition. In the case in which the transition is actually a $\nu n$ transition we can assume that $n$ is contained in the testable name set $\Theta$.

**Case:** $t = \mathcal{E}[t']$ and $u = \mathcal{E}[u']$ for some $\mathcal{E}$ satisfying the required properties for $\sim_R$. In particular, we have that $\Gamma;\Delta;\mathrm{R}';\mathrm{W}' \vDash t' \sim_{R,R'} u' : \sigma$ and so by induction $\Gamma;\Delta;\mathrm{R}';\mathrm{W}' \vDash t' \sim^1_R u' : \sigma$. We then apply Proposition A.5 to finish.

**Case:** $u = u'[\overline{n}'/\overline{n}]$ and $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \sim_R u' : \sigma$ for some bijective substitution $[\overline{n}'/\overline{n}]$. By induction, $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \sim^1_R u' : \sigma$, and the result follows from Proposition A.6.

The result follows. □

We now have a number of technical propositions left to prove.

**Proposition A.4** *If* $\overline{n}$ *are passive from* $\Psi \subseteq \Theta$ *in* $(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma)$, *and* $R : \overline{n} \leftrightarrow \overline{n}$ *then*

$$\Gamma;\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R};\mathrm{W} \vdash t \sim^1_R t : \sigma.$$

**Proof:** We have to show that for any $\Delta;\mathrm{R}; \vDash [\overline{v}/\overline{x}] \sim^1_R [\overline{w}/\overline{x}] : \Gamma$ we can complete the diagram:

$$\begin{tikzcd}
(\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R};\mathrm{W} \vdash t[\overline{v}/\overline{x}] : \sigma) \arrow[r, leftrightarrow, "\sim_R"] \arrow[d, "\alpha"'] & (\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R};\mathrm{W} \vdash t[\overline{w}/\overline{x}] : \sigma) \\
(\Psi \subseteq \Theta, \Delta' \subseteq \Delta, \Delta';\mathrm{R};\mathrm{W} \vdash t' : \sigma) &
\end{tikzcd}$$

as

$$\begin{tikzcd}
(\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R};\mathrm{W} \vdash t[\overline{v}/\overline{x}] : \sigma) \arrow[r, leftrightarrow, "\sim_R"] \arrow[d, "\alpha"'] & (\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R};\mathrm{W} \vdash t[\overline{w}/\overline{x}] : \sigma) \arrow[d, "\alpha"'] \\
(\Psi \subseteq \Theta, \Delta' \subseteq \Delta, \Delta';\mathrm{R};\mathrm{W} \vdash t' : \sigma) \arrow[r, leftrightarrow, "\sim_R"] & (\Psi \subseteq \Theta, \Delta' \subseteq \Delta, \Delta';\mathrm{R};\mathrm{W} \vdash u' : \sigma)
\end{tikzcd}$$

when $fn(\alpha) \cap \Delta \subseteq \Theta$. The interesting cases are:

**Case:** $t = x$, so the result follows by the definition of $\sim^1_R$.

**Case:** $(\Delta;\mathrm{R};\mathrm{W} \vdash t[\overline{v}/\overline{x}] : \sigma) \xrightarrow{\alpha} (\Delta;\mathrm{R};\mathrm{W} \vdash t' : \sigma')$ is given from some open transition:

$$(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma) \xrightarrow{\alpha}{}^{\circ} (\Gamma;\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash t'' : \sigma')$$

where $t' = t''[\overline{v}/\overline{x}]$. By Proposition A.11 $\overline{n}$ are passive in $(\Gamma;\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash t'' : \sigma')$, so we have:

$$\begin{tikzcd}
(\Delta;\mathrm{R};\mathrm{W} \vdash t[\overline{v}/\overline{x}] : \sigma) \arrow[r, leftrightarrow, "\sim_R"] \arrow[d, "\alpha"'] & (\Delta;\mathrm{R};\mathrm{W} \vdash t[\overline{w}/\overline{x}] : \sigma) \arrow[d, "\alpha"'] \\
(\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash t''[\overline{v}/\overline{x}] : \sigma) \arrow[r, leftrightarrow, "\sim_R"] & (\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash t''[\overline{w}/\overline{x}] : \sigma)
\end{tikzcd}$$

**Case:** The transition is a $\beta$-reduction of the form:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[x(v_0)][\overline{v}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[t_0[v_0/x_0]][\overline{v}/\overline{x}] : \sigma)$$

where $x[\overline{v}/\overline{x}] = \lambda x_0 : \sigma_0 \,.\, t_0$ and $\mathcal{E}$ has typing:

$$\frac{\Gamma;\Delta,\Delta';\mathrm{R}';\mathrm{W}' \vdash \cdot : \sigma'}{\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[\cdot] : \sigma}$$

Let $u_0$ be such that $x[\overline{w}/\overline{x}] = \lambda x_0 : \sigma_0 \,.\, u_0$. Since $\overline{v} \sim^1_R \overline{w}$, by Proposition A.12 we have:

$$x_0 : \sigma_0;\Delta;\mathrm{R}; \vDash t_0 \sim_R u_0 : \sigma'$$

and by $\beta$-reduction we have:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[x(v_0)][\overline{w}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[u_0[v_0/x_0]][\overline{w}/\overline{x}] : \sigma)$$

By Proposition A.7 $\overline{n}$ are passive in $\mathcal{E}$ and that $\overline{n}$ are passive in $v_0$. So, we have

$$\Gamma;\Delta;\mathrm{R}';\mathrm{W}' \vDash v_0 \sim_R v_0 : \sigma_0$$

which means that we can use the definition of $\sim_R$ to get:

$$\Delta;\mathrm{R};\mathrm{W} \vDash \mathcal{E}[t_0[v_0/x_0]][\overline{v}/\overline{x}] \sim_R \mathcal{E}[u_0[v_0/x_0]][\overline{w}/\overline{x}] : \sigma$$

as required.

**Case:** The transition is a reduction of the form:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[x = n][\overline{v}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[b][\overline{v}/\overline{x}] : \sigma)$$

where $x[\overline{v}/\overline{x}] = n'$. By Proposition A.2 $n' \notin \overline{n}$, and since $\overline{v} \sim^1_R \overline{w}$, we have $x[\overline{w}/\overline{x}] = n'$, so

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[x = n][\overline{v}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[b][\overline{v}/\overline{x}] : \sigma)$$

By Proposition A.11 $\overline{n}$ are passive in $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[v][n'/x] : \sigma$, so

$$\Delta;\mathrm{R};\mathrm{W} \vDash \mathcal{E}[b][\overline{v}/\overline{x}] \sim_R \mathcal{E}[b][\overline{w}/\overline{x}] : \sigma$$

as required.

**Case:** The transition is a reduction of the form:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[x = x'][\overline{v}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[b][\overline{v}/\overline{x}] : \sigma)$$

which is handled similarly to the previous case.

**Case:** The transition is a reduction of the form:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[\mathtt{if}\, x\, \mathtt{then}\, t_1\ \mathtt{else}\, t_2][\overline{v}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[t_1][\overline{v}/\overline{x}] : \sigma)$$

where $x[\overline{v}/\overline{x}] = \mathtt{true}$. Again we notice that, $x[\overline{w}/\overline{x}] = \mathtt{true}$ and by Proposition A.11 we know that $\overline{n}$ are passive in $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[t_1][\mathtt{true}/x]$. The result follows easily from this and a similar argument applies when $x[\overline{v}/\overline{x}] = \mathtt{false}$.

**Case:** The transition is a reduction of the form:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[\mathtt{fst}\, x][\overline{v}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[v_1][\overline{v}/\overline{x}] : \sigma)$$

where $x[\overline{v}/\overline{x}] = (v_1, v_2)$. This follows by noticing that $\overline{n}$ are passive in

$$\Gamma, y : \sigma_1, z : \sigma_2; \Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[(y,z)/x][y].$$

We know that $\overline{v} \sim^1_R \overline{w}$ so we have $x[\overline{w}/\overline{x}] = (w_1, w_2)$ with $v_i \sim_R w_i$ for $i = 1,2$. Therefore we know that $\Gamma; z : \sigma_2; \Delta;\mathrm{R};\mathrm{W} \vDash \mathcal{E}[(v_1,z)/x][v_1] \sim_R \mathcal{E}[(w_1,z)/x][w_1] : \sigma$. This guarantees that

$$(\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[\mathtt{fst}\, x][\overline{w}/\overline{x}] : \sigma) \xrightarrow{\tau} (\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[w_1][\overline{w}/\overline{x}] : \sigma)$$

provides the matching transition. A similar argument follows for `snd` also.

**Case:** We write $V\langle t\rangle$ to be a term of the grammar

$$V\langle t\rangle ::= (v, V\langle t\rangle) \mid (V\langle t\rangle, v) \mid t$$

So suppose the transition is of the form:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash d.V\langle x\rangle[\overline{v}/\overline{x}] : \sigma_1 \times \sigma \times \sigma_2) \xrightarrow{\iota.\gamma} (\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash d.V\langle t''\rangle[\overline{v}/\overline{x}] : \sigma_1 \times \sigma' \times \sigma_2)$$

derived from a transition:

$$(\Delta;\mathrm{R};\ \vdash x[\overline{v}/\overline{x}] : \sigma) \xrightarrow{\gamma} (\Delta,\Delta';\mathrm{R};\ \vdash t'' : \sigma')$$

so since $\overline{v} \sim^1_R \overline{w}$ we have:

$$(\Delta;\mathrm{R};\ \vdash x[\overline{w}/\overline{x}] : \sigma) \xrightarrow{\gamma} (\Delta,\Delta';\mathrm{R};\ \vdash u'' : \sigma') \qquad \Delta,\Delta';\mathrm{R};\ \vDash t'' \sim^1_R u'' : \sigma'$$

which means:

$$(\Delta;\mathrm{R};\mathrm{W} \vdash d.V\langle x\rangle[\overline{w}/\overline{x}] : \sigma_1 \times \sigma \times \sigma_2) \xrightarrow{\iota.\gamma} (\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash d.V\langle u''\rangle[\overline{w}/\overline{x}] : \sigma_1 \times \sigma' \times \sigma_2)$$

By Propositions A.10, A.9 and A.8 we have $\overline{n}$ passive in:

$$(\Gamma, x' : \sigma';\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash d.V\langle x'\rangle : \sigma_1 \times \sigma' \times \sigma_2)$$

so

$$\Delta,\Delta';\mathrm{R};\mathrm{W} \vDash d.V\langle t''\rangle[\overline{v}/\overline{x}] \sim_R d.V\langle u''\rangle[\overline{w}/\overline{x}] : \sigma_1 \times \sigma' \times \sigma_2)$$

as required. □

**Proposition A.5** *For any $\mathcal{E}$ with $\overline{n}$ passive typed:*

$$\frac{\Gamma;\Delta;\mathrm{R}';\mathrm{W}' \vdash \cdot : \sigma'}{\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[\cdot] : \sigma}$$

*if* $\Gamma;\Delta;\mathrm{R}';\mathrm{W}' \vDash t \sim^1_R t' : \sigma'$ *then* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash \mathcal{E}[t] \sim^1_R \mathcal{E}[t'] : \sigma$.

**Proof:** Similar to the proof of Proposition A.4 with the addition of two cases which arise as an interaction between $\mathcal{E}$ and $t$.

**Case:** $\mathcal{E}$ is $\mathcal{E}_1[r := v.\,[\cdot]]$ and $t$ is $\mathcal{E}_2[?r]$ so that

$$\Gamma;\Delta;\mathrm{R}',r;\mathrm{W}' \vDash \mathcal{E}_2[?r] \sim_R \mathcal{E}_2'[?r]$$

with $r$ not assigned to in $\mathcal{E}_2, \mathcal{E}_2'$. We observe that $v$ cannot be a name in $\overline{n}$ and we easily get $\Gamma;\Delta;;\vDash v \sim_R v$. By definition of $\sim_R$ it follows that

$$\Gamma, y : \mathtt{name};\Delta;\mathrm{R}',r;\mathrm{W}' \vDash \mathcal{E}_2[y] \sim_R \mathcal{E}_2'[y]$$

because any $n$ which can be instantiated for $y$ can be supplied to $?r$ using a closing assignment. Given this it is a simple matter to use the definition of $\sim_R$ to yield

$$\Gamma, y : \mathtt{name};\Delta;\mathrm{R};\mathrm{W}' \vDash \mathcal{E}[\mathcal{E}_2[y] \sim_R \mathcal{E}[\mathcal{E}_2'[y]$$

and the result follows.

**Case:** $\mathcal{E}$ is $\mathcal{E}'[\mathtt{let}\ x\ =\ [\cdot]\ \mathtt{in}\ u]$ and $t$ is $d_1.v_1$ so that $\Gamma;\Delta;\mathrm{R}';\mathrm{W}' \vDash d_1.v_1 \sim_R d_2.v_2$. We use Proposition A.10 to observe that $\Gamma;\Delta,\Delta';\mathrm{R}',\mathrm{R}'';\vDash v_1 \sim_R v_2$ for appropriate $\Delta', \mathrm{R}''$. It is easy to see that the hypothesis tells us that $\overline{n}$ are passive in $\Gamma, x : \sigma';\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}'[u]$ therefore

$$\Gamma;\Delta,\Delta';\mathrm{R},\mathrm{R}'',\mathrm{W} \vDash \mathcal{E}'[u[v_1/x]] \sim_R \mathcal{E}'[u[v_2/x].$$

We use the definition of $\sim_R$ to obtain

$$\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash d_1.\mathcal{E}'[u[v_1/x] \sim_R d_2.\mathcal{E}'[u[v_2/x]$$

and structural congruence to finish. □

**Proposition A.6** *If* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \sim^1_R t' : \sigma$ *and* $[\overline{n}'/\overline{n}] \subseteq R : \overline{n} \leftrightarrow \overline{n}$ *is a bijective substitution then* $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \sim^1_R t'[\overline{n}'/\overline{n}] : \sigma$.

**Proof:** For closed terms, this goes through immediately, since transitions are invariant under bijective substitutions.

For open terms, consider any substitutions $\Delta;\mathrm{R};\vDash [\overline{v}/\overline{x}] \sim^1_R [\overline{w}/\overline{x}] : \Gamma$. Since $\overline{v}$ and $\overline{w}$ are closed, we have that:

$$\Delta;\mathrm{R};\vDash [\overline{v}/\overline{x}] \sim^1_R [\overline{w}[\overline{n}/\overline{n}']/\overline{x}] : \Gamma$$

and so since $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vDash t \sim^1_R t' : \sigma$ we have:

$$\Delta;\mathrm{R};\mathrm{W} \vDash t[\overline{v}/\overline{x}] \sim^1_R t'[\overline{w}[\overline{n}/\overline{n}']/\overline{x}] : \sigma$$

and again we have closed terms, so:

$$\Delta;\mathrm{R};\mathrm{W} \vDash t[\overline{v}/\overline{x}] \sim^1_R t'[\overline{w}[\overline{n}/\overline{n}']/\overline{x}][\overline{n}'/\overline{n}] = t'[\overline{n}'/\overline{n}][\overline{w}/\overline{x}] : \sigma$$

as required. □

**Proposition A.7** *If $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash \mathcal{E}[x(v)] : \sigma$ then $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $\mathcal{E}$ and $\Gamma;\Delta;\mathrm{R}';\vdash v : \sigma'$.*

**Proof:** Firstly, we can suppose that $x$ does not appear freely in $\mathcal{E}$ or $v$. This is no real restriction because if this were the case then we could instantiate two occurrences with a function built using a fresh reference, for example:

$$\lambda x.\texttt{if } ?r = n_0 \texttt{ then } r := n_1 \,.\, fx \texttt{ else } gx$$

This function will act, in the context $r := n_0$, like $f$ for the first time it is applied and $g$ thereafter.

We shall prove the result for $\mathcal{E}$ and $v$ closed, since the result for open $\mathcal{E}$ and $v$ follows directly.

To show the former statement we simply observe that any instantiation $t$ for $\mathcal{E}$ could be attained by instantiating $\mathcal{E}[xv]$ with $\lambda z.t$ with $z \notin t$. The latter is harder to demonstrate. We aim for a contradiction by supposing that for some $n \in \overline{n}$ we have transitions $(\Delta;;\vdash v : \sigma') \overset{\overline{\gamma}}{\Longrightarrow} \overset{\iota.n}{\Longrightarrow}$ with $fn(\overline{\gamma}) \cap \Delta \subseteq \Theta$. It is not too difficult to see that, using Theorem 4.2, one can build a value $\Theta;;,r \vdash w_{\overline{\gamma}} : \sigma' \to \sigma''$ such that

$$(\Delta;;\vdash w_{\overline{\gamma}}v : \sigma'') \Rightarrow (\Delta;;\vdash d.r := n \,.\, v_0 : \sigma'')$$

for some canonical value $v_0$ of the appropriate type and fresh $r$. We then observe that, since reduction in this language is terminating,

$$(\Delta;;\mathrm{W} \vdash \mathcal{E}[w_{\overline{\gamma}}v]) \Rightarrow (\Delta;;\mathrm{W} \vdash d.r := n \,.\, p)$$

for some prevalue $p$ which does not assign to $r$. We can now discard and read from $r$ to realise a transition $\xrightarrow{n}$. Notice that the free names of $w_{\overline{\gamma}}$ may contain names from $\Theta$ which are not necessarily in $\Psi$. This would prevent us getting our contradiction to the passivity of $\overline{n}$ in $\mathcal{E}[xv]$, were it not for the fact that $x : \sigma' \to \sigma''$ has function type and the restriction to only instantiating names from $\Psi$ may be circumvented at higher types by means of references. To see this we simply use a fresh reference for each name $n \in \Theta \setminus \Psi$ and replace any occurrence of these names in $w_{\overline{\gamma}}$ with a read on the corresponding reference. The closing initialisation of these new references actually allows us to use names from $\Theta$, thus each of these references can be initialised with the name which they represent. □

**Proposition A.8** *If $\overline{n}$ are passive in $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma$ then $\overline{n}$ are passive in $\Gamma,\Gamma';\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash t : \sigma$.*

**Proposition A.9** *If $\overline{n}$ are passive in $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash (v_1, v_2) : \sigma_1 \times \sigma_2$ then $\overline{n}$ are also passive in both $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash v_i : \sigma_i$ for $(i = 1,2)$.*

**Proposition A.10** *If $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash d.p$ then $\overline{n}$ are also passive from $\Psi \subseteq \Theta$ in $\Gamma;\Delta,\Delta';\mathrm{R}';\mathrm{W}' \vDash p$ for suitable $\Delta',\mathrm{R}',\mathrm{W}'$.*

**Proof:** Straightforward. □

**Proposition A.11** *If $\overline{n}$ are passive from $\Psi \subseteq \Theta$ in $(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma)$ and*

$$(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma) \xrightarrow{\alpha}{}^{\circ} (\Gamma;\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash t' : \sigma')$$

*then $\overline{n}$ are passive from $\Psi \subseteq \Theta,\Delta'$ in $(\Gamma;\Delta,\Delta';\mathrm{R};\mathrm{W} \vdash t' : \sigma')$.*

**Proof:** This is simple for all cases save

$$(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t : \sigma) \xrightarrow{\iota.\nu n}{}^{\circ} (\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vdash t' : \sigma').$$

Suppose for contradiction that we could find a closing instantiation $\Psi,\Delta'';\mathrm{R}';\mathrm{W} \vdash [\overline{v}/\overline{x}] : \Gamma$, assignments $\Psi,\Delta'' \vdash \overline{r} := \overline{n}_0 : \mathrm{R}$, and $\Theta,n,\Delta'' \vdash \overline{r}' := \overline{n}'_0 : \mathrm{R}'$ (notice the essential use of $\Psi$ here) such that, for some $n' \in \overline{n}$ we get

$$(\Delta,\Delta'';;\mathrm{W} \vdash \overline{r} := \overline{n}_0 \,.\, \overline{r}' := \overline{n}'_0 \,.\, t'[\overline{v}/x]) \xRightarrow{\overline{\gamma}} \xRightarrow{\iota.n'}$$

such that $fn(\overline{\gamma}) \cap \Delta, n \subseteq \Theta, n$. We cannot use these transitions to contradict passivity of $\overline{n}$ in $t$ immediately because $\overline{\gamma}$ may contain $n$ and the initialisation assignments to $\mathrm{R}'$ may also contain $n$. However, by assumption we can ensure that $n$ is leaked to the environment first, thus permitting the contradictory tests. Note that $t$ must be a prevalue of the form $\nu n \,.\, d.w$, so

$$(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash t) \xrightarrow{\mathtt{copy}}{}^{\circ} (\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash \nu n \,.\, d.(w,w))$$

and

$$(\Gamma;\Delta;\mathrm{R};\mathrm{W} \vdash \nu n \,.\, d.(w,w)) \xrightarrow{\mathtt{l}.\iota.\nu n}{}^{\circ} (\Gamma;\Delta,n;\mathrm{R};\mathrm{W} \vdash d.(w',w))$$

for some $w'$. At this point we use assignment transitions to ensure that any of the initialising assignments to $\mathrm{R}'$ which contain $n$ are made and we follow this with the sequence of $\overline{\gamma}$ transitions tagged with the `r.` identifier. Thus we obtain a $\xrightarrow{\mathtt{r}.\iota.n'}$ transition to give a contradiction. □

**Proposition A.12** *If* $\Gamma;\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R} \vDash \lambda x : \sigma \,.\, t \sim^{1}_{R} \lambda x : \sigma \,.\, u : \sigma \to \sigma'$ *then*

$$\Gamma,x:\sigma;\Psi \subseteq \Theta \subseteq \Delta;\mathrm{R} \vDash t \sim_R u : \sigma'.$$

**Proof:** Straightforward enough, but notice that the converse does not hold in general as $\Psi \subseteq \Theta$. Any value supplied to the latter term can be supplied to the former by using an apply transition but not vice-versa. □

# References

[1] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. Research Report 149, Digital Equipment Corporation Systems Research Center, 1998. To appear in *Information and Computation.*

[2] K.L. Bernstein and E.W. Stark. Operational semantics of a focussing debugger. In *Proc. MFPS 95*. Springer-Verlag, 1995. Vol 1. Electronic Notes in Comp. Sci.

[3] G. Berry and G. Boudol. The chemical abstract machine. In *Proc. 17th Ann. Symp. Principles of Programming Languages*, 1990.

[4] L. Cardelli and A. Gordon. Mobile ambients. In *Proc. FoSSaCS '98*, LNCS. Springer-Verlag, 1998.

[5] W. Ferreira, M. Hennessy, and A.S.A Jeffrey. A theory of weak bisimulation for core CML. In *Proc. ACM SIGPLAN Int. Conf. Functional Programming*. ACM Press, 1996. To appear in J. Functional Programming.

[6] A. Gordon. Bisimilarity as a theory of functional programming. In *Proc. MFPS 95*, number 1 in Electronic Notes in Comp. Sci. Springer-Verlag, 1995.

[7] A. Gordon. Nominal calculi for security and mobility. In *Proc. DARPA Workshop on Foundations for Secure Mobile Code*, pages 10–14, 1997.

[8] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, Cambridge, Massachusetts, 1988.

[9] F. Honsell, I.A. Mason, S. Smith, and C. Talcott. A variable typed logic of effects. *Information and Computation*, 119(1):55–90, 1995.

[10] Douglas Howe. Equality in lazy computation systems. In *Proc. LICS '89*, pages 198–203. IEEE Computer Society Press, 1989.

[11] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Part I + II. *Information and Computation*, 100(1):1–77, 1992.

[12] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[13] A. M. Pitts and I. D. B. Stark. Observable properties of higher order functions that dynamically create local names, or: What's new? In *Proc. MFCS 93*, pages 122–141. Springer-Verlag, 1993. LNCS 711.

[14] A. M. Pitts and I. D. B. Stark. On the observable properties of higher order functions that dynamically create local names (preliminary report). In *Workshop on State in Programming Languages, Copenhagen, 1993*, pages 31–45. ACM SIGPLAN, 1993. Yale Univ. Dept. Computer Science Technical Report YALEU/DCS/RR-968.

[15] A.M. Pitts and I.D.B. Stark. Operational reasoning for functions with local state. In A.D. Gordon and A.M. Pitts, editors, *Higher Order Operational Techniques in Semantics*, pages 227–273. Cambridge University Press, 1998. Publications of the Newton Institute.

[16] J. Reppy. *Higher-Order Concurrency*. PhD thesis, Cornell University, June 1992. Technical Report TR 92-1285.

[17] M.Tofte, R.Milner and R.Harper. *The Definition of Standard ML*. MIT Press, 1990.

[18] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1993.

[19] D. Sangiorgi and R. Milner. On the problem of 'weak bisimulation up to'. In W.R. Cleaveland, editor, *Proceedings CONCUR 92,* Stony Brook, NY, USA, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 1992.

[20] P. Sewell. From rewrite rules to bisimulation congruences. In D. Sangiorgi and R. de Simone, editors, *Proceedings CONCUR 98,* Nice, volume 1466 of *Lecture Notes in Computer Science*, pages 269–284. Springer-Verlag, 1998.

[21] I.D.B. Stark. *Names and Higher-Order Functions*. PhD thesis, University of Cambridge, 1994. Also published as Computer Laboratory Tech. Report 363.

[22] Andrew K. Wright and Matthias Felleisen. A syntactic approach to type soundness. Technical Report TR91-160, Rice University, 1991.