Observable Properties of Higher Order Functions that Dynamically Create Local Names, or: What's *new*?

Andrew M. Pitts* and Ian D. B. Stark**

University of Cambridge Computer Laboratory,
Pembroke Street, Cambridge CB2 3QG, England

Abstract. The research reported in this paper is concerned with the problem of reasoning about properties of higher order functions involving state. It is motivated by the desire to identify what, if any, are the difficulties created purely by *locality* of state, independent of other properties such as side-effects, exceptional termination and non-termination due to recursion. We consider a simple language (equivalent to a fragment of Standard ML) of typed, higher order functions that can dynamically create fresh *names*; names are created with local scope, can be tested for equality and can be passed around via function application, but that is all. Despite the extreme simplicity of the language and its operational semantics, the observable properties of such functions are shown to be very subtle. A notion of 'logical relation' is introduced which incorporates a version of *representation independence* for local names. We show how to use it to establish observational equivalences. The method is shown to be complete (and decidable) for expressions of first order types, but incomplete at higher types.

1 Introduction

Programming languages combining higher order features with the manipulation of local state present severe problems for the traditional techniques of programming language semantics and logics of programs. For denotational semantics, the problems manifest themselves as a lack of abstraction in existing semantic models: some expressions that are observationally equivalent (i.e. that can be interchanged in any program without affecting its behaviour when executed) are assigned different denotations in the model. For operational semantics, the problems manifest themselves partly in the fact that simple techniques for analyzing observational equivalence in the case of purely functional languages (such as Milner's 'Context Lemma' [8], or more generally, notions of applicative bisimulation [1]) break down in the presence of state-based features. Furthermore, operationally based approaches to properties of programs are often inconveniently intensional, e.g. the familiar congruence properties of equational logic fail to hold. (See [6, Sect. 5(A)], for example.) These problems have been intensively studied for the case of local variables in block-structured, Algol-like languages and to a lesser extent for the case of languages

* Supported by UK SERC grant GR/G53279 and CEC ESPRIT project CLICS-II
** Supported by UK SERC studentship 91307943 and CEC SCIENCE project PL910296

involving the dynamic creation of mutable locations (such as ML-style *references*). See [17, 2, 7, 3, 18, 12, 13, 6, 4]. Our interest in this subject stems primarily from a desire to improve and deepen the techniques which are available for reasoning about program behaviour in the 'impure' functional language Standard ML [9].

Our motivation here is to try to identify what, if any, are the difficulties created purely by *locality* of state, independent of other properties such as side-effects, exceptional termination and non-termination due to recursion. Accordingly we consider higher order functions which can dynamically create fresh names of things, but which ignore completely what kind of thing (references, exceptions, etc.) is being named. Names are created with local scope, can be tested for equality, and are passed around via function application, but that is all. Because of this limited framework, there is some hope of obtaining *definitive* results—fully abstract models and complete proof techniques. As the vehicle for this study we formulate an extension of the call-by-value, simply typed lambda calculus, called the *nu-calculus* and introduced in Sect. 2. In ML terms, it contains higher order functions over ground types `bool` and `unit ref`—the latter being the type of dynamically created references to the unique element of type `unit`. This acts as a type of 'names' because only one thing can be (and is) stored in such a reference, so that its only characteristic is its name. We have purposely excluded recursion from the nu-calculus and as a result any closed expression evaluates to an essentially unique canonical form. Indeed, the nu-calculus appears at first sight to be an extremely simple system. On closer inspection, we find that nu-calculus expressions can exhibit very subtle behaviour with respect to an appropriate notion of observational equivalence. Thus our first contribution is somewhat in the spirit of Meyer and Seiber [7]: we observe that even for this extremely simple case of local state there are observationally equivalent expressions which traditional denotational techniques will fail to identify (Example 4).

In Sect. 3 we introduce a notion of 'logical relation' for the nu-calculus incorporating a version of *representation independence* for local names. Our technique is a syntactic version of the relationally parametric semantics of O'Hearn and Tennent [13]. There are also interesting similarities with Plotkin and Abadi's parametricity schema for existential types [16, Theorem 7]. We use our version of logical relations to establish the termination properties of the nu-calculus (Theorem 12) and to provide a useful notion of 'applicative' equivalence between nu-calculus expressions which implies observational equivalence (Theorem 14). Although the two notions of equivalence differ at higher order types (Example 6), they coincide for expressions of first order types (Theorem 22) and are decidable there (Corollary 23). The proof of this occupies Sect. 4 and is surprisingly hard work: although applicative equivalence provides a compositional explanation of (observational equivalence classes of) first order functions, even these can have complicated behaviour (see Example 1).

Note. This paper is an expanded version of the operationally-based results announced in [14]. That reference also contains an outline of our approach to the denotational semantics of the nu-calculus.

2 The nu-calculus

Syntactically, the nu-calculus is a kind of simply typed lambda calculus. The types, σ, are built up from a ground type o of *booleans* and a ground type ν of *names*, by forming *function types*, $\sigma \to \sigma'$. Expressions take the form

$M ::= x$		variable
$\mid n$		name
$\mid \mathit{true}$	$\mid \mathit{false}$	truth values
$\mid \mathit{if}\ M\ \mathit{then}\ M\ \mathit{else}\ M$		conditional
$\mid M = M$		equality of names
$\mid \nu n\ .\ M$		local name declaration
$\mid \lambda x : \sigma\ .\ M$		function abstraction
$\mid MM$		function application

where $x \in \mathit{Var}$, an infinite set whose elements are called *variables*, and $n \in \mathit{Nme}$, an infinite set (disjoint from Var) whose elements are called *names*. Function abstraction is a variable-binding construct (occurrences of x in M are bound in $\lambda x : \sigma\ .\ M$), whereas local name declaration is a name-binding construct (occurrences of n in M are bound in $\nu n\ .\ M$). We write $\mathit{Var}(M)$ and $\mathit{Nme}(M)$ for the finite subsets of Var and Nme consisting of the free variables and the free names in an expression M. We denote by $M[M'/x]$ (respectively $M[M'/n]$) the result of substituting an expression M' for all free occurrences of x (respectively n) in M, renaming bound variables and bound names if necessary, to avoid variable and name capture.

Note. Henceforward, we implicitly identify expressions that differ up to α-conversion of bound variables and bound names. Thus when we refer to an expression M we really mean an α-equivalence class of expressions, referred to via one of its representatives M.

Expressions will be assigned types via typing assertions of the form

$$s, \Gamma \vdash M : \sigma$$

where s is a finite subset of Nme, Γ is a finite function from variables to types, σ is a type, and M is an expression (more precisely, an α-equivalence class of expressions) satisfying $\mathit{Nme}(M) \subseteq s$ and $\mathit{Var}(M) \subseteq \mathit{dom}(\Gamma)$ (the domain of definition of Γ). The rules generating the valid typing assertions are given in Table 1. In these rules $s \oplus \{n\}$ indicates the finite set of names obtained from s by adjoining $n \notin s$; and $\Gamma \oplus [x : \sigma]$ denotes the finite function obtained by extending Γ by mapping $x \notin \mathit{dom}(\Gamma)$ to σ. Clearly, if $s, \Gamma \vdash M : \sigma$ holds, then σ is uniquely determined by s, Γ and M. We write

$$\mathrm{Exp}_\sigma(s) \stackrel{def}{=} \{M \mid s, \emptyset \vdash M : \sigma\}$$

for the set of *closed nu-calculus expression of type σ with free names in the set s*. The subset

$$\mathrm{Can}_\sigma(s) \subseteq \mathrm{Exp}_\sigma(s)$$

Table 1. Rules for assigning types in the nu-calculus

$$\frac{}{s, \Gamma \vdash x : \Gamma(x)}\,(x \in dom(\Gamma)) \qquad \frac{}{s, \Gamma \vdash n : \nu}\,(n \in s) \qquad \frac{}{s, \Gamma \vdash b : o}\,(b = true, false)$$

$$\frac{s, \Gamma \vdash B : o \quad s, \Gamma \vdash M : \sigma \quad s, \Gamma \vdash M' : \sigma}{s, \Gamma \vdash \mathit{if}\ B\ \mathit{then}\ M\ \mathit{else}\ M' : \sigma} \qquad \frac{s, \Gamma \vdash N : \nu \quad s, \Gamma \vdash N' : \nu}{s, \Gamma \vdash (N = N') : o}$$

$$\frac{s \oplus \{n\}, \Gamma \vdash M : \sigma}{s, \Gamma \vdash \nu n\ .\ M : \sigma} \qquad \frac{s, \Gamma \oplus [x : \sigma] \vdash M : \sigma'}{s, \Gamma \vdash \lambda x : \sigma\ .\ M : \sigma \to \sigma'} \qquad \frac{s, \Gamma \vdash F : \sigma \to \sigma' \quad s, \Gamma \vdash M : \sigma}{s, \Gamma \vdash FM : \sigma'}$$

of *canonical* nu-calculus expressions of type σ with free names in the set s consists of those closed expressions which are either names (in s), or the booleans constants *true* and *false*, or function abstractions.

We give the operational semantics of the nu-calculus in terms of an inductively defined evaluation relation which matches the computational behaviour of equivalent ML expressions. The ML equivalent of the expression $\nu n\ .\ M$ is

$$\texttt{let n=ref() in}\ M\ \texttt{end}$$

(using the ML type `unit ref` for the type of names). In other words the effect of evaluating $\nu n\ .\ M$ should be to create a fresh name n and then use it in evaluating M. In the definition of ML [9] environments are used to bind identifiers (variables) to addresses (names), whereas here we have chosen to simplify the form of the evaluation relation by using 'extended' expressions containing names explicitly. It would be possible to simplify the syntax of the nu-calculus even further by identifying the syntactic category of names with that of variables of type ν. We choose not to do so because names and variables have different semantic properties. For example, the operational semantics we give commutes with arbitrary substitutions on variables, but only with restricted forms of substitutions on names (viz. essentially just permutations of names): see Remark 2.

An appropriate notion of state for this simple language is just a finite subset of *Nme*, indicating the names which have been created so far. So we will use an evaluation relation of the form

$$s \vdash M \Downarrow_\sigma (s')C \tag{1}$$

where s and s' are *disjoint* finite sets of names, $M \in \mathrm{Exp}_\sigma(s)$ and $C \in \mathrm{Can}_\sigma(s \oplus s')$.

Note. Throughout this paper, we write $s \oplus s'$ to indicate the union of two sets s and s' that are *disjoint*.

The intended meaning of (1) is: 'in state s, expression M evaluates to canonical form C creating fresh, local names s' in the process'. The rules for generating the relation are given in Table 2. In rule (EQ) we use the notation $\delta_{nn'}$, where

$$\delta_{nn'} \stackrel{def}{=} \begin{cases} true \text{ if } n = n' \\ false \text{ if } n \neq n' \end{cases} .$$

It is important to note that the rules in Table 2 refer to the collection of judgements as in (1) that are well-formed, i.e. satisfy the conditions mentioned above. For example, in rule (LOCAL) the well-formedness of the hypothesis and the conclusion entail that n is not an element of either s or s_1.

Table 2. Rules for evaluating nu-calculus expressions

(CAN)
$$\frac{}{s \vdash C \Downarrow_\sigma C}$$

(COND1)
$$\frac{s \vdash B \Downarrow_o (s_1)true \quad s \oplus s_1 \vdash M \Downarrow_\sigma (s_2)C}{s \vdash \mathit{if}\ B\ \mathit{then}\ M\ \mathit{else}\ M' \Downarrow_\sigma (s_1 \oplus s_2)C}$$

(COND2)
$$\frac{s \vdash B \Downarrow_o (s_1)false \quad s \oplus s_1 \vdash M' \Downarrow_\sigma (s_2)C'}{s \vdash \mathit{if}\ B\ \mathit{then}\ M\ \mathit{else}\ M' \Downarrow_\sigma (s_1 \oplus s_2)C'}$$

(EQ)
$$\frac{s \vdash N \Downarrow_\nu (s_1)n \quad s \oplus s_1 \vdash N' \Downarrow_\nu (s_2)n'}{s \vdash (N = N') \Downarrow_o (s_1 \oplus s_2)\delta_{nn'}}$$

(LOCAL)
$$\frac{s \oplus \{n\} \vdash M \Downarrow_\sigma (s_1)C}{s \vdash \nu n\ .\ M \Downarrow_\sigma (\{n\} \oplus s_1)C}$$

(APP)
$$\frac{\begin{array}{c} s \vdash F \Downarrow_{\sigma \to \sigma'} (s_1)\lambda x : \sigma\ .\ M' \qquad s \oplus s_1 \vdash M \Downarrow_\sigma (s_2)C \\ s \oplus s_1 \oplus s_2 \vdash M'[C/x] \Downarrow_{\sigma'} (s_3)C' \end{array}}{s \vdash FM \Downarrow_{\sigma'} (s_1 \oplus s_2 \oplus s_3)C'}$$

It is easy to see that evaluation is deterministic up to renaming created names, in the following sense:

Lemma 1. *If $s \vdash M \Downarrow_\sigma (s_1)C$ and $s \vdash M \Downarrow_\sigma (s_2)C'$, then there is a bijection $R : s_1 \leftrightarrow s_2$ so that C' is α-convertible with the expression $C[n'/n \mid (n, n') \in R]$.*

Remark 2 (States are affine linear). The initial state s in the evaluation (1) has the structural properties of an *affine linear logic context*, in the sense that derived rules of weakening and exchange are valid, but a rule of contraction is not. (Compare

the use made of affine linear logic by O'Hearn in [11].) Thus

$$\text{(WEAK)} \quad \frac{s \vdash M \Downarrow_\sigma (s_1)C}{s \oplus \{n\} \vdash M \Downarrow_\sigma (s_1)C}$$

$$\text{(EXCH)} \quad \frac{s \oplus \{n\} \oplus \{n'\} \vdash M \Downarrow_\sigma (s_1)C}{s \oplus \{n'\} \oplus \{n\} \vdash M \Downarrow_\sigma (s_1)C}$$

are correct derived rules (the second trivially so, because we are using states that are sets rather than lists), but

$$\text{(CONTR)} \quad \frac{s \oplus \{n\} \oplus \{n'\} \vdash M \Downarrow_\sigma (s_1)C}{s \oplus \{n''\} \vdash M[n''/n, n''/n'] \Downarrow_\sigma (s_1)C[n''/n, n''/n']}$$

is not a correct derived rule — as can be seen, for example, by taking s and s_1 to be $\emptyset$, σ to be o, M to be $n = n'$ and C to be *false*.

More generally, given a function $f : s \to s'$ and letting $M[f]$ denote the substituted expression $M[f(n)/n \mid n \in s]$, we have that the rule (SUBST) below is a correct derived rule *provided that f is an injective function.*

$$\text{(SUBST)} \quad \frac{s \vdash M \Downarrow_\sigma (s_1)C}{s' \vdash M[f] \Downarrow_\sigma (s_1)C[f]}$$

Remark 3 (Sequentiality condition). The evaluation rules in Table 2 follow the state convention of Standard ML [9, p. 50], i.e. order of evaluation is from left to right, with state accumulating sequentially. We have formulated the operational semantics of the nu-calculus in this way to emphasize that it is (equivalent to) a fragment of ML. However, because we are dealing with state that can be created but cannot be mutated, some of this sequentiality is spurious. Table 3 gives 'de-sequentialized' versions of rules (COND1), (COND2), (EQ), and (APP). We claim that using these rules instead of the corresponding rules in Table 2 does not affect the collection of instances of evaluation that are derivable. This claim follows from the fact that a converse of the weakening rule (WEAK) is derivable:

$$\text{(STREN)} \quad \frac{s \oplus s' \vdash M \Downarrow_\sigma (s_1)C}{s \vdash M \Downarrow_\sigma (s_1)C} \, (Nme(M) \subseteq s \text{ and } Nme(C) \subseteq s \oplus s_1) \quad .$$

The evaluation relation (1) can be used to define a Morris-style contextual equivalence between nu-calculus expressions: two expressions are equivalent if they can be interchanged in any program without affecting the observable result of evaluating it. Here we will take a 'program' to be a closed expression of type o, and the possible observable results of evaluating a program to be the booleans *true* and *false*, disregarding any local names that are created in the process of evaluation. (It would not change the notion of observational equivalence given below if we also allowed programs to be of type ν and observable results to include pre-existing names.) In the following definition, as usual the 'context' $B[-]$ is an expression in which some subexpressions have been replaced by a place-holder, $-$; and then $B[M]$ denotes the result of filling the place-holder with an expression M.

Table 3. 'De-sequentialized' evaluation rules

$$\text{(COND1}') \qquad \frac{s \vdash B \Downarrow_o (s_1)\mathit{true} \quad s \vdash M \Downarrow_\sigma (s_2)C}{s \vdash \mathit{if}\ B\ \mathit{then}\ M\ \mathit{else}\ M' \Downarrow_\sigma (s_1 \oplus s_2)C}$$

$$\text{(COND2}') \qquad \frac{s \vdash B \Downarrow_o (s_1)\mathit{false} \quad s \vdash M' \Downarrow_\sigma (s_2)C'}{s \vdash \mathit{if}\ B\ \mathit{then}\ M\ \mathit{else}\ M' \Downarrow_\sigma (s_1 \oplus s_2)C'}$$

$$\text{(EQ}') \qquad \frac{s \vdash N \Downarrow_\nu (s_1)n \quad s \vdash N' \Downarrow_\nu (s_2)n'}{s \vdash (N = N') \Downarrow_o (s_1 \oplus s_2)\delta_{nn'}}$$

$$\text{(APP}') \qquad \frac{\begin{array}{c} s \vdash F \Downarrow_{\sigma \to \sigma'} (s_1)\lambda x : \sigma\ .\ M' \qquad s \vdash M \Downarrow_\sigma (s_2)C \\ s \oplus s_1 \oplus s_2 \vdash M'[C/x] \Downarrow_{\sigma'} (s_3)C' \end{array}}{s \vdash FM \Downarrow_{\sigma'} (s_1 \oplus s_2 \oplus s_3)C'}$$

Definition 4 (Observational equivalence). Given $M_1, M_2 \in \mathrm{Exp}_\sigma(s)$, we write

$$s \vdash M_1 \approx_\sigma M_2$$

to mean that for all $B[-]$ and all $b \in \{\mathit{true}, \mathit{false}\}$,

$$\exists s_1(s \vdash B[M_1] \Downarrow_o (s_1)b) \Leftrightarrow \exists s_2(s \vdash B[M_2] \Downarrow_o (s_2)b) \quad .$$

In this case we say that M_1 and M_2 are *observationally equivalent*.

The following result shows that one need only consider contexts that immediately evaluate their arguments in order to establish observational equivalence. It is the analogue of Theorem (**ciu**) in [4].

Lemma 5. *$s \vdash M_1 \approx_\sigma M_2$ if and only if for all $b \in \{\mathit{true}, \mathit{false}\}$ and all $\lambda x : \sigma\ .\ B \in \mathrm{Can}_{\sigma \to o}(s)$*

$$\exists s_1(s \vdash (\lambda x : \sigma\ .\ B)M_1 \Downarrow_o (s_1)b) \Leftrightarrow \exists s_2(s \vdash (\lambda x : \sigma\ .\ B)M_2 \Downarrow_o (s_2)b) \quad .$$

The following instances of observational equivalence are easily established using the lemma.

Corollary 6. *1. If $M \in \mathrm{Exp}_\sigma(s)$ and $n \notin s$, then $s \vdash \nu n\ .\ M \approx_\sigma M$.*
2. If $M \in \mathrm{Exp}_\sigma(s \oplus \{n\} \oplus \{n'\})$, then $s \vdash \nu n\ .\ \nu n'\ .\ M \approx_\sigma \nu n'\ .\ \nu n\ .\ M$.
3. If $s \vdash M \Downarrow_\sigma (s')C$, then $s \vdash M \approx_\sigma \nu s'\ .\ C$. Here $\nu s'\ .\ C$ stands for $\nu n_1 \ldots \nu n_k\ .\ C$ if $s' = \{n_1, \ldots, n_k\}$ for some $k > 0$, and stands for C if $s' = \emptyset$. (By part 2, up to observational equivalence, it does not matter which order we enumerate the elements of s' in $\nu s'\ .\ C$.)
4. If $s, [x : \sigma] \vdash M : \sigma'$ and $C \in \mathrm{Can}_\sigma(s)$, then $s \vdash (\lambda x : \sigma\ .\ M)C \approx_{\sigma'} M[C/x]$.

In the next section we will show that evaluation of nu-calculus expressions always terminates (Theorem 12). It follows from this and the above corollary that, up to observational equivalence, the only closed expressions of type o are *true* and *false* and the only closed expression of type ν not involving any free names is

$$new \stackrel{def}{=} \nu n \,.\, n \quad .$$

However, at higher types things become more complicated. The following example gives infinitely many expressions of type $\nu \to \nu$ which are mutually observationally inequivalent.

Example 1. For each $p \geq 1$, consider the nu-calculus expression of type $\nu \to \nu$ which first creates $p+1$ local names $n_0, \ldots, n_p$ and then acts as the function cyclically permuting these names and mapping any other name to n_0:

$$\begin{aligned} F_p \stackrel{def}{=} \nu n_0 \ldots \nu n_p \,.\, \lambda x : \nu \,.\, & \mathit{if}\ x = n_0\ \mathit{then}\ n_1\ \mathit{else} \\ & \mathit{if}\ x = n_1\ \mathit{then}\ n_2\ \mathit{else} \\ & \quad \ldots \\ & \quad \mathit{if}\ x = n_p\ \mathit{then}\ n_0\ \mathit{else}\ n_0 \quad . \end{aligned}$$

Then $\emptyset \vdash F_p \not\approx_{\nu \to \nu} F_{p'}$ whenever $p \neq p'$, because

$$B_q \stackrel{def}{=} \lambda f : \nu \to \nu \,.\, \nu n \,.\, (f^{(q+2)}(n) = f(n))$$

has the property that for all $q \in \{1, \ldots, p\}$, $\emptyset \vdash B_q F_p \Downarrow_o (\{n_0, \ldots, n_p, n\})\mathit{true}$ if and only if $q = p$. (In B_q, $f^{(q+2)}$ indicates f iterated $q+2$ times.)

Example 2. Here is a simple example to illustrate the fact that local name declaration and function abstraction in general do not commute up to observational equivalence. The expressions

$$M \stackrel{def}{=} \nu n \,.\, \lambda x : \nu \,.\, n \quad \text{and} \quad N \stackrel{def}{=} \lambda x : \nu \,.\, \nu n \,.\, n$$

are not observationally equivalent, because $B \stackrel{def}{=} \lambda f : \nu \to \nu \,.\, (f\,new = f\,new)$ has the property that $\emptyset \vdash BM \Downarrow_o (\{n, n_1, n_2\})\mathit{true}$ whereas $\emptyset \vdash BN \Downarrow_o (\{n, n_1, n_2\})\mathit{false}$.

Example 3. The rule (APP) in Table 2 embodies a form of strict, or 'call-by-value', application. Part 4 of Corollary 6 shows that the appropriate restricted form of beta-conversion (Plotkin's β_v [15]) holds up to observational equivalence. Although there is no non-termination in our simple language, the general form of beta-conversion fails for the nu-calculus, because of the dynamics of name creation. For example, the beta redex $(\lambda x : \nu \,.\, x = x)new$ is not observationally equivalent to the corresponding reduct $new = new$ since

$$\begin{aligned} &\emptyset \vdash (\lambda x : \nu \,.\, x = x)new \Downarrow_o (\{n_1\})\mathit{true} \\ &\emptyset \vdash (new = new) \Downarrow_o (\{n_1, n_2\})\mathit{false} \quad . \end{aligned}$$

For the simple functional language PCF, Milner's context lemma [8] shows that observational equivalence may be established by testing just with *applicative contexts*, i.e. those of the form $[-]C_1C_2 \ldots C_k$. Not surprisingly, this fails in the nu-calculus. For example, the expressions F_p in Example 1 are in fact indistinguishable by such applicative contexts, even though they can be distinguished by more complicated contexts (like $B_q([-])$) which carry out 'anonymous' manipulation of the private names $n_0, \ldots, n_p$. It would seem that the properties of higher order functions which create and pass around private names can be quite subtle. Two contrasting examples of observational equivalence, more subtle than those in Corollary 6, are given below. The first one illustrates the fact that local names are always distinct from externally supplied names; the second illustrates the fact that any two local names are indiscernible by externally supplied boolean tests. (This second equivalence is quite delicate—it certainly would not hold in languages where evaluation of functions can have side-effects on mutable state.) The methods developed in the next section suffice to prove (2), but not (3). At the moment, the only method known to us for establishing this second equivalence is denotational, i.e. via a specific model of the nu-calculus: see [14, Sect. 4].

Example 4.

$$\emptyset \vdash \nu n \,.\, \lambda x : \nu \,.\, (x = n) \;\approx_{\nu \to o}\; \lambda x : \nu \,.\, \mathit{false} \tag{2}$$

$$\emptyset \vdash \nu n \,.\, \nu n' \,.\, \lambda f : \nu \to o \,.\, (fn = fn') \;\approx_{(\nu \to o) \to o}\; \lambda f : \nu \to o \,.\, \mathit{true} \quad . \tag{3}$$

In (3), the boolean equality test $fn = fn'$ is an abbreviation for

$$\mathit{if}\ fn\ \mathit{then}\ (\mathit{if}\ fn'\ \mathit{then}\ \mathit{true}\ \mathit{else}\ \mathit{false})\ \mathit{else}\ (\mathit{if}\ fn'\ \mathit{then}\ \mathit{false}\ \mathit{else}\ \mathit{true}) \quad .$$

3 Representation independence for local names

This section develops a notion of (binary) logical relation for the nu-calculus and shows how to use it to establish instances of observational equivalence between nu-calculus expressions.

Given finite subsets $s_1, s_2 \subseteq \mathit{Nme}$ of names, we write $R : s_1 \rightleftharpoons s_2$ to indicate that R is (the graph of) a *partial bijection* from s_1 to s_2. In other words, $R \subseteq s_1 \times s_2$ satisfies

$$m_1 \; R \; m_2 \wedge n_1 \; R \; n_2 \Rightarrow (m_1 = n_1 \Leftrightarrow m_2 = n_2) \quad . \tag{4}$$

(We use infix notation for binary relations.) Note that $R \oplus R'$ is a partial bijection $s_1 \oplus s_1' \rightleftharpoons s_2 \oplus s_2'$ when $R : s_1 \rightleftharpoons s_2$ and $R' : s_1' \rightleftharpoons s_2'$ are disjoint partial bijections. The *identity* partial bijection, $I_s : s \rightleftharpoons s$, is given by:

$$n_1 \; I_s \; n_2 \Leftrightarrow n_1 = n_2 \quad . \tag{5}$$

The domain and codomain of definition of a partial bijection $R : s_1 \rightleftharpoons s_2$ will be denoted

$$\mathit{dom}(R) \stackrel{\mathit{def}}{=} \{n_1 \in s_1 \mid \exists n_2 \in s_2 \,.\, n_1 \; R \; n_2\} \tag{6}$$

$$\mathit{cod}(R) \stackrel{\mathit{def}}{=} \{n_2 \in s_2 \mid \exists n_1 \in s_1 \,.\, n_1 \; R \; n_2\} \quad . \tag{7}$$

Thus R is a bijection just in case $dom(R) = s_1$ and $cod(R) = s_2$, in which case we write $R : s_1 \leftrightarrow s_2$.

Definition 7 (Logical relations). For each type σ we define a family of binary relations between canonical expressions

$$(R_\sigma \subseteq \mathrm{Can}_\sigma(s_1) \times \mathrm{Can}_\sigma(s_2) \mid R : s_1 \rightleftharpoons s_2)$$

by induction on the structure of σ as in (9), (10) and (11) below; clause (11) makes use of associated relations between expressions, $\overline{R}_\sigma \subseteq \mathrm{Exp}_\sigma(s_1) \times \mathrm{Exp}_\sigma(s_2)$ defined by (8).

$$\begin{aligned} M_1 \; \overline{R}_\sigma \; M_2 \Leftrightarrow \exists R' : s_1' \rightleftharpoons s_2', C_1 \in \mathrm{Can}_\sigma(s_1 \oplus s_1'), C_2 \in \mathrm{Can}_\sigma(s_2 \oplus s_2') \; . \\ s_1 \vdash M_1 \Downarrow_\sigma (s_1')C_1 \wedge s_2 \vdash M_2 \Downarrow_\sigma (s_2')C_2 \wedge C_1 \; (R \oplus R')_\sigma \; C_2 \end{aligned} \tag{8}$$

$$b_1 \; R_o \; b_2 \Leftrightarrow b_1 = b_2 \tag{9}$$

$$n_1 \; R_\nu \; n_2 \Leftrightarrow n_1 \; R \; n_2 \tag{10}$$

$$\begin{aligned} \lambda x : \sigma \; . \; M_1 \; R_{\sigma \to \sigma'} \; \lambda x : \sigma \; . \; M_2 \Leftrightarrow \\ \forall R' : s_1' \rightleftharpoons s_2', C_1 \in \mathrm{Can}_\sigma(s_1 \oplus s_1'), C_2 \in \mathrm{Can}_\sigma(s_2 \oplus s_2') \; . \\ C_1 \; (R \oplus R')_\sigma \; C_2 \Rightarrow M_1[C_1/x] \; (\overline{R \oplus R'})_{\sigma'} \; M_2[C_2/x] \quad . \end{aligned} \tag{11}$$

(It is implicit in (8) and (11) that each s_i' is required to be disjoint from s_i.)

The family $(\overline{R}_\sigma \mid \sigma)$ is a form of binary 'logical relation' for nu-calculus expressions. Since we choose in (9) to take the logical relation to be the identity at the ground type o, the whole family is determined by what we take at the other ground type ν. We wish related expressions to be mapped to related expressions by any nu-calculus function, and we have to impose the restriction (4) on the relation R to ensure this property holds for the function testing equality of names. The following proposition expresses this fundamental property of our notion of logical relation.

Proposition 8 (Fundamental property of logical relations). *Suppose*

$$[x_1 : \sigma_1, \ldots, x_k : \sigma_k] \vdash M : \sigma \quad .$$

Then for all $R : s_1 \rightleftharpoons s_2$, $C_i \in \mathrm{Can}_{\sigma_i}(s_1)$ *and* $D_i \in \mathrm{Can}_{\sigma_i}(s_2)$ *(*$i = 1, \ldots, k$*), one has*

$$\left(\bigwedge_{i=1}^{k} C_i \; R_{\sigma_i} \; D_i \right) \Rightarrow M[C_1/x_1, \ldots, C_k/x_k] \; \overline{R}_\sigma \; M[D_1/x_1, \ldots, D_k/x_k] \quad .$$

Proof. The proof proceeds by induction on the derivation of the typing assertion $[x_1 : \sigma_1, \ldots, x_k : \sigma_k] \vdash M : \sigma$, and makes use of (the only if part of) the following lemma, which is itself proved by induction on the structure of the type σ. We omit the details. □

Lemma 9. *Given $R : s_1 \rightleftharpoons s_2$ and $R' : s_1' \rightleftharpoons s_2'$ with s_i and s_i' disjoint (for $i = 1, 2$), then for all types σ and all canonical expressions $C_i \in \mathrm{Can}_\sigma(s_i)$ ($i = 1, 2$), $C_1\ R_\sigma\ C_2$ if and only if $C_1\ (R \oplus R')_\sigma\ C_2$.*

Similarly, for all $M_i \in \mathrm{Exp}_\sigma(s_i)$, $M_1\ \overline{R}_\sigma\ M_2$ if and only if $M_1\ (\overline{R \oplus R'})_\sigma\ M_2$.

Remark. The main interest in Definition 7 lies in clause (8) where the relation $\overline{R}_\sigma$ on expressions is defined in terms of the relation R_σ on canonical expressions. This clause embodies a form of 'representation independence' for the dynamically created local names. (Cf. Plotkin and Abadi's parametricity schema for existential types [16, Theorem 7].) One might have expected to see not (8), but rather

$$\begin{aligned} M_1\ \overline{R}_\sigma\ M_2 \Leftrightarrow\ & (\forall s_1', C_1 \in \mathrm{Can}_\sigma(s_1 \oplus s_1')\ .\ s_1 \vdash M_1 \Downarrow_\sigma (s_1')C_1 \Rightarrow \\ & \quad \exists s_2', R' : s_1' \rightleftharpoons s_2', C_2 \in \mathrm{Can}_\sigma(s_2 \oplus s_2')\ . \\ & \qquad s_2 \vdash M_2 \Downarrow_\sigma (s_2')C_2 \wedge C_1\ (R \oplus R')_\sigma\ C_2) \\ \wedge\ & \\ & (\forall s_2', C_2 \in \mathrm{Can}_\sigma(s_2 \oplus s_2')\ .\ s_2 \vdash M_2 \Downarrow_\sigma (s_2')C_2 \Rightarrow \\ & \quad \exists s_1', R' : s_1' \rightleftharpoons s_2', C_1 \in \mathrm{Can}_\sigma(s_1 \oplus s_1')\ . \\ & \qquad s_1 \vdash M_1 \Downarrow_\sigma (s_1')C_1 \wedge C_1\ (R \oplus R')_\sigma\ C_2) \end{aligned} \tag{12}$$

This deals appropriately with the possibility of non-termination. However, the simple language we are considering here has the property (Theorem 12) that all expressions converge to canonical forms which are essentially unique (by Lemma 1), in which case (12) is equivalent to the simpler form (8).

Clause (11) of Definition 7 is a syntactic version of O'Hearn and Tennent's approach to relational parametricity in [13]. It also exhibits the characteristic feature of 'logical relations', in that two functions are defined to be related if they send related arguments to related results. To be more in keeping with the definition of applicative bisimulation in [1], one might consider an alternative definition in which two functions are related when they give related results for all arguments. For pure functional languages, such as the lazy lambda calculus, one expects the two approaches to be equivalent, and to equal observational equivalence: see [1, 5]. Here, the notion of 'applicative equivalence' we define below using Definition 7 is contained in, but not equal to observational equivalence; and we believe that replacing clause (11) by a 'related if related on all arguments' version (which we will not formulate precisely here) results in an even weaker notion of equivalence.

We will need to use Proposition 8 in the more general form given in the corollary below. Its statement makes use of the following notation for *renaming* expressions along the bijection $R : \mathit{dom}(R) \leftrightarrow \mathit{cod}(R)$ obtained from a partial bijection $R : s_1 \rightleftharpoons s_2$ by restricting it to its domain of definition (cf. definitions (6) and (7)).

Definition 10. Given a partial bijection $R : s_1 \rightleftharpoons s_2$, for any nu-calculus expression M, let $M[R]$ denote the result of simultaneously substituting for each name in $\mathit{dom}(R)$ the corresponding name in $\mathit{cod}(R)$:

$$M[R] \stackrel{def}{=} M[n'/n \mid n\ R\ n'] \quad .$$

Corollary 11. *Suppose* $s_1, [x_1 : \sigma_1, \ldots, x_k : \sigma_k] \vdash M : \sigma$*, that* $R : s_1 \leftrightarrow s_2$ *is a bijection and that* $R' : s_1' \rightleftharpoons s_2'$ *is a partial bijection disjoint from* R*. Then for all* $C_i \in \mathrm{Can}_{\sigma_i}(s_1 \oplus s_1')$ *and* $D_i \in \mathrm{Can}_{\sigma_i}(s_2 \oplus s_2')$ *(*$i = 1, \ldots, k$*) one has*

$$\left(\bigwedge_{i=1}^{k} C_i \ (R \oplus R')_{\sigma_i} \ D_i\right) \Rightarrow$$
$$M[C_1/x_1, \ldots, C_k/x_k] \ (\overline{R \oplus R'})_\sigma \ M[R][D_1/x_1, \ldots, D_k/x_k] \quad .$$

Proof. Apply Proposition 8 to

$$[y_1 : \nu, \ldots, y_\ell : \nu, x_1 : \sigma_1, \ldots, x_k : \sigma_k] \vdash M[y_j/n_j \mid 1 \leq j \leq \ell] : \sigma$$

where $s = \{n_1, \ldots, n_\ell\}$. □

Theorem 12 (Termination). *For all closed expressions* M*, of type* σ *and with free names in the set* s *say, there is some set of names* s' *(disjoint from* s*) and some canonical expression* $C \in \mathrm{Can}_\sigma(s \oplus s')$ *such that* $s \vdash M \Downarrow_\sigma (s')C$*.*

Proof. The $k = 0$ case of Corollary 11 implies that $M \ (\overline{I_s})_\sigma \ M$ for all $M \in \mathrm{Exp}_\sigma(s)$. Termination follows from this, given the definition of $\overline{R}_\sigma$ in (8). □

We now show how the fundamental property of our notion of logical relation embodied in Proposition 8 can be used to establish observational equivalences.

Definition 13 (Applicative equivalence). We say that two expressions $M_1, M_2 \in \mathrm{Exp}_\sigma(s)$ are *applicatively equivalent* if $M_1 \ (\overline{I_s})_\sigma \ M_2$, where $I_s : s \rightleftharpoons s$ is the identity partial bijection on s defined in (5).

Theorem 14. *Applicative equivalence implies observational equivalence.*

Proof. Suppose $M_1 \ (\overline{I_s})_\sigma \ M_2$. We employ Lemma 5 to see that $s \vdash M_1 \approx_\sigma M_2$. By (8) there is some $R' : s_1' \rightleftharpoons s_2'$, and C_1, C_2 with $s \vdash M_i \Downarrow_\sigma (s_i')C_i$ $(i = 1, 2)$ and $C_1 \ (I_s \oplus R')_\sigma \ C_2$. Then for any $\lambda x : \sigma \,.\, B \in \mathrm{Can}_{\sigma \to o}(s)$, applying Corollary 11 with $R = I_s$ we get $B[C_1/x] \ (\overline{I_s \oplus R'})_o \ B[C_2/x]$. Hence by (8) again, there is some $R'' : s_1'' \rightleftharpoons s_2''$ and b_1, b_2 with $s \oplus s_i' \vdash B[C_i/x] \Downarrow_o (s_i'')b_i$ $(i = 1, 2)$ and $b_1 \ (I_s \oplus R' \oplus R'')_o \ b_2$, i.e. with $b_1 = b_2$ (by (9)). Applying the rules in Table 2, we deduce that $s \vdash (\lambda x : \sigma \,.\, B)M_i \Downarrow_o (s_i' \oplus s_i'')b_i$ with $b_1 = b_2$. Thus Lemma 5 and the deterministic nature of the evaluation relation (Lemma 1) imply that $M_1 \approx_\sigma M_2$. □

Example 5. Theorem 14 provides quite a powerful method for establishing some observational equivalences, since the relation $(\overline{I_s})_\sigma$ is much easier to deal with than $\approx_\sigma$. For example, the observational equivalence (2) can be established by this method. For with

$$C_1 \stackrel{def}{=} \lambda x : \nu \,.\, (x = n) \qquad \text{and} \qquad C_2 \stackrel{def}{=} \lambda x : \nu \,.\, \mathit{false}$$

it is not hard to see that $C_1 \ (I_\emptyset \oplus R)_{\nu \to o} \ C_2$ where $R : \{n\} \rightleftharpoons \emptyset$ is necessarily the empty partial bijection; hence $\nu n \,.\, C_1 \ (\overline{I_\emptyset})_{\nu \to o} \ C_2$, as required.

However, not every observational equivalence can be established via Theorem 14, as the following example shows. Thus *applicative equivalence is in general a strictly weaker relation than observational equivalence*. Nevertheless, as we shall see below (Theorem 22), the converse of Theorem 14 does hold when σ is a *first order type*, i.e. of the form $\sigma_k \to \sigma_{k-1} \to \cdots \to \sigma_0$ with each σ_i either ν or o.

Example 6. The pair of second order expressions in (3) are observationally equivalent (this can be established via the denotational methods sketched in [14, Sect. 4]), but they are not related by $(\overline{I_\emptyset})_{(\nu\to o)\to o}$. For the only possible partial bijection $R : \{n, n'\} \rightleftharpoons \emptyset$ is $R = \emptyset$; but $\lambda f : \nu \to o \,.\, (fn = fn')$ and $\lambda f : \nu \to o \,.\, \mathit{true}$ are not related by $(I_\emptyset \oplus R)_{(\nu\to o)\to o}$, because for the canonical expressions C_1 and C_2 defined in Example 5, $C_1\ (I_\emptyset \oplus R)_{\nu\to o}\ C_2$, whereas it is not the case that $(fn = fn')[C_1/f]\ (\overline{I_\emptyset \oplus R})_o\ \mathit{true}[C_2/f]$.

4 Observational relations

To investigate further the relationship between observational and applicative equivalence, we introduce the following generalization of the notion of observational equivalence which we will see satisfies all the defining properties of applicative equivalence in Definition 7 except (11).

Definition 15. Given a partial bijection $R : s_1 \rightleftharpoons s_2$ and expressions $M_i \in \mathrm{Exp}_\sigma(s_i)$ $(i = 1, 2)$, we write

$$M_1 R^{\mathrm{obs}}_\sigma M_2$$

to mean that for all $\tau \in \{o, \nu\}$ and all $\lambda x : \sigma \,.\, P \in \mathrm{Can}_{\sigma\to\tau}(\mathit{dom}(R))$

$$(\lambda x : \sigma \,.\, P)M_1\ \overline{R}_\tau\ (\lambda x : \sigma \,.\, P[R])M_2 \quad .$$

In this case we say that M_1 and M_2 are *observationally R-related*. Note that because τ is a ground type, the relation $\overline{R}_\tau$, defined using (8), (9) and (10), takes a particularly simple form:

- For all $B_i \in \mathrm{Exp}_o(s_i)$, $B_1\ \overline{R}_o\ B_2$ if and only if there is some $b \in \{\mathit{true}, \mathit{false}\}$ so that for each $i = 1, 2$ $s_i \vdash B_i \Downarrow_o (s'_i)b$ for some s'_i.
- For all $N_i \in \mathrm{Exp}_\nu(s_i)$, $N_1\ \overline{R}_\nu\ N_2$ if and only if for each $i = 1, 2$ $s_i \vdash N_i \Downarrow_\nu (s'_i)n_i$ for some s'_i and some $n_i \in s_i \oplus s'_i$ satisfying

$$n_1\ R\ n_2 \text{ or } (n_1 \in s'_1 \text{ and } n_2 \in s'_2) \quad .$$

The following proposition substantiates the claim that observational relations generalize the notion of observational equivalence.

Proposition 16. *Observational equivalence coincides with being observationally I_s-related. In other words, for any $M_1, M_2 \in \mathrm{Exp}_\sigma(s)$*

$$s \vdash M_1 \approx_\sigma M_2 \Leftrightarrow M_1 (I_s)^{\mathrm{obs}}_\sigma M_2 \quad .$$

Proof. Comparing Definition 15 with the characterization of observational equivalence in Lemma 5, it suffices to show that when $s \vdash M_1 \approx_\sigma M_2$ then $(\lambda x : \sigma \,.\, P)M_1\ (\overline{I_s})_\nu\ (\lambda x : \sigma \,.\, P)M_2$, for any $\lambda x : \sigma \,.\, P \in \mathrm{Can}_{\sigma\to\nu}(s)$. Certainly $s \vdash M_1 \approx_\sigma M_2$ implies $s \vdash (\lambda x : \sigma \,.\, P)M_1 \approx_\nu (\lambda x : \sigma \,.\, P)M_2$. So in fact it suffices to show for any $N_1, N_2 \in \mathrm{Exp}_\nu(s)$ that

$$s \vdash N_1 \approx_\nu N_2 \Rightarrow N_1\ (\overline{I_s})_\nu\ N_2 \quad . \tag{13}$$

To proof (13), first use Theorem 12 to find s_i and n_i such that $s \vdash N_i \Downarrow_\nu (s_i)n_i$. For any $n \in s$ one thus has $s \vdash (\lambda x : \nu \,.\, x = n)N_i \Downarrow_\nu (s_i)b_i$, where $b_i = \mathit{true}$ if and only if $n = n_i$. Since $s \vdash N_1 \approx_\nu N_2$, $b_1 = b_2$; hence either $n_1 = n_2 \in s$, or $n_1 \in s_1$ and $n_2 \in s_2$. Thus $N_1\ (\overline{I_s})_\nu\ N_2$, as required. □

Lemma 17. *For any partial bijection $R : s_1 \rightleftharpoons s_2$ and any $M_i \in \mathrm{Exp}_\sigma(s_i)$ $(i = 1, 2)$*

$$M_1\ \overline{R}_\sigma\ M_2 \Rightarrow M_1 R^{\mathrm{obs}}_\sigma M_2 \quad . \tag{14}$$

Moreover, when $\sigma \in \{o, \nu\}$ the reverse implication holds.

Proof. The implication (14) follows immediately from Corollary 11. To see that the second part of the lemma holds, note that in case $\sigma \in \{o, \nu\}$, if $M_1 R^{\mathrm{obs}}_\sigma M_2$ then in Definition 15 we can take P to be x to conclude that $(\lambda x : \sigma \,.\, x)M_1\ \overline{R}_\sigma\ (\lambda x : \sigma \,.\, x)M_2$ and hence that $M_1\ \overline{R}_\sigma\ M_2$ (since M_i and $(\lambda x : \sigma \,.\, x)M_i$ have the same behaviour under evaluation). □

Lemma 18. *For any $R : s_1 \rightleftharpoons s_2$, $M_i \in \mathrm{Exp}_\sigma(s_i)$ $(i = 1, 2)$ and $\lambda x : \sigma \,.\, N \in \mathrm{Can}_{\sigma\to\nu}(\mathit{dom}(R))$, suppose*

$$\begin{aligned} s_i \vdash M_i \Downarrow_\sigma (s_i')C_i \qquad (i = 1, 2)\\ s_1 \oplus s_1' \vdash N[C_1/x] \Downarrow_\nu (s_1'')n_1\\ s_2 \oplus s_2' \vdash N[R][C_2/x] \Downarrow_\nu (s_2'')n_2 \quad . \end{aligned}$$

If $M_1 R^{\mathrm{obs}}_\sigma M_2$, then $n_1 \in s_1''$ if and only if $n_2 \in s_2''$.

Proof. Consider the boolean expression

$$B \stackrel{def}{=} (\lambda x : \sigma \,.\, N)x = (\lambda x : \sigma \,.\, N)x \quad .$$

For each $i = 1, 2$ let s_i''' be a fresh set of names in bijection with s_i'', via $R_i : s_i'' \leftrightarrow s_i'''$ say. Then

$$\begin{aligned} s_1 \vdash (\lambda x : \sigma \,.\, B)M_1 \Downarrow_o (s_1' \oplus s_1'' \oplus s_1''')b_1\\ s_2 \vdash (\lambda x : \sigma \,.\, B[R])M_2 \Downarrow_o (s_2' \oplus s_2'' \oplus s_2''')b_2 \end{aligned}$$

where $b_i = \mathit{false}$ if and only if $n_i \neq n_i[R_i]$, i.e. if and only if $n_i \in s_i''$. If $M_1 R^{\mathrm{obs}}_\sigma M_2$ then we must have $b_1 = b_2$, from which the result follows. □

The following proposition expresses a key property of observational relations which is a precise analogue of the characteristic clause (8) in the definition of logical relation that we have been using. It shows why partial bijections between states (sets of names) play a prominent role in studying observational properties of the nu-calculus, since they can be used to explain observational equivalence (i.e. being observationally I_s-related, by Proposition 16) between general expressions in terms of observational relations between canonical expressions. The proof of the proposition is quite intricate and we give it in some detail.

Proposition 19. *For any partial bijection $R : s_1 \rightleftharpoons s_2$ and any $M_i \in \mathrm{Exp}_\sigma(s_i)$ $(i = 1, 2)$*

$$M_1 R_\sigma^{\mathrm{obs}} M_2 \Leftrightarrow \exists R' : s_1' \rightleftharpoons s_2', C_1 \in \mathrm{Can}_\sigma(s_1 \oplus s_1'), C_2 \in \mathrm{Can}_\sigma(s_2 \oplus s_2') \,. \quad (15)$$
$$s_1 \vdash M_1 \Downarrow_\sigma (s_1')C_1 \wedge s_2 \vdash M_2 \Downarrow_\sigma (s_2')C_2 \wedge C_1 (R \oplus R')_\sigma^{\mathrm{obs}} C_2 \quad .$$

Proof. Suppose that $M_1 R_\sigma^{\mathrm{obs}} M_2$. By Theorem 12, $s_i \vdash M_i \Downarrow_\sigma (s_i')C_i$ for some $C_i \in \mathrm{Can}_\sigma(s_i \oplus s_i')$ $(i = 1, 2)$. We begin by constructing a suitable partial bijection $R' : s_1' \rightleftharpoons s_2'$.

Let R' consist of those pairs of names $(n, n') \in s_1' \times s_2'$ for which there is some $\lambda x : \sigma \,.\, N \in \mathrm{Can}_{\sigma \to \nu}(dom(R))$ with

$$s_1 \oplus s_1' \vdash (\lambda x : \sigma \,.\, N)C_1 \Downarrow_\nu (s_1'')n \quad (16)$$
$$s_2 \oplus s_2' \vdash (\lambda x : \sigma \,.\, N[R])C_2 \Downarrow_\nu (s_2'')n' \quad . \quad (17)$$

To see that R' is a partial bijection, suppose $n \; R' \; n'$, witnessed by a canonical expression $\lambda x : \sigma \,.\, N$ satisfying (16) and (17), and suppose also $m \; R' \; m'$, witnessed by some $\lambda x : \sigma \,.\, M$. Applying the test $\lambda x : \sigma \,.\, (N = M) \in \mathrm{Can}_{\sigma \to o}(dom(R))$ to $M_1 R_\sigma^{\mathrm{obs}} M_2$, we have $(\lambda x : \sigma \,.\, (N = M))M_1 \; \overline{R}_o \; (\lambda x : \sigma \,.\, (N = M)[R])M_2$; from this it follows that $n = m$ if and only if $n' = m'$. Thus R' is indeed a partial bijection.

Next we show that $C_1 (R \oplus R')_\sigma^{\mathrm{obs}} C_2$. Given any $\lambda x : \sigma \,.\, P \in \mathrm{Can}_{\sigma \to \tau}(dom(R \oplus R'))$ with $\tau \in \{o, \nu\}$, we have to show that $(\lambda x : \sigma \,.\, P)C_1 \; (\overline{R \oplus R'})_\tau \; (\lambda x : \sigma \,.\, P[R \oplus R'])C_2$. Enumerate R' as $\{(n_i, n_i') \mid 1 \leq i \leq k\}$ for some $k \geq 0$, and for each i let $\lambda x : \sigma \,.\, N_i \in \mathrm{Can}_{\sigma \to \nu}(dom(R))$ witness that $n_i \; R' \; n_i'$ (as in (16) and (17)). Consider

$$P' \stackrel{def}{=} (\lambda y_1 : \nu \,.\, \cdots \lambda y_k : \nu \,.\, P[y_i / n_i \mid 1 \leq i \leq k])N_1 \cdots N_k$$

Suppose that

$$s_1 \oplus s_1' \vdash (\lambda x : \sigma \,.\, P)C_1 \Downarrow_\tau (s_1'')D_1 \quad (18)$$
$$s_2 \oplus s_2' \vdash (\lambda x : \sigma \,.\, P[R \oplus R'])C_2 \Downarrow_\tau (s_2'')D_2 \quad . \quad (19)$$

Then by construction of P', we also have

$$s_1 \vdash (\lambda x : \sigma \,.\, P')M_1 \Downarrow_\tau (s_1' \oplus s \oplus s_1'')D_1 \quad (20)$$
$$s_2 \vdash (\lambda x : \sigma \,.\, P'[R])M_2 \Downarrow_\tau (s_2' \oplus s' \oplus s_2'')D_2 \quad (21)$$

for some s and s'. Since $\lambda x : \sigma \,.\, P' \in \mathrm{Can}_{\sigma \to \tau}(dom(R))$ and $M_1 R_\sigma^{\mathrm{obs}} M_2$, we have $(\lambda x : \sigma \,.\, P')M_1 \; \overline{R}_\tau \; (\lambda x : \sigma \,.\, P'[R])M_2$. Hence by (20) and (21),

$$D_1 \; (R \oplus S)_\tau \; D_2 \quad (22)$$

for some $S : s_1' \oplus s \oplus s_1'' \rightleftharpoons s_2' \oplus s' \oplus s_2''$. We consider the cases $\tau = o$ and $\tau = \nu$ separately.

When $\tau = o$, (22) immediately gives $D_1 = D_2$, and hence by (18) and (19), $(\lambda x : \sigma \,.\, P)C_1\ (\overline{R \oplus R'})_o\ (\lambda x : \sigma \,.\, P[R \oplus R'])C_2$, as required.

When $\tau = \nu$, (22) implies either $D_1\ R\ D_2$, or $D_1 \in s_1' \oplus s \oplus s_1''$ and $D_2 \in s_2' \oplus s \oplus s_2''$. But in this second case, by Lemma 18

$$(D_1 \in s_1' \text{ and } D_2 \in s_2') \text{ or } (D_1 \in s \oplus s_1'' \text{ and } D_2 \in s' \oplus s_2'') \quad .$$

By definition of R', if $D_i \in s_i'$ $(i = 1, 2)$, then $D_1\ R'\ D_2$. So when $\tau = \nu$ we have

$$D_1\ R \oplus R'\ D_2 \text{ or } (D_1 \in s \oplus s_1'' \text{ and } D_2 \in s' \oplus s_2'')$$

and hence by (18) and (19), $(\lambda x : \sigma \,.\, P)C_1\ (\overline{R \oplus R'})_\nu\ (\lambda x : \sigma \,.\, P[R \oplus R'])C_2$, as required.

This completes the proof of the implication $\Rightarrow$ in (15). The proof of the reverse implication is quite straightforward and we omit it. □

Combining Proposition 19 with Lemma 17, we have that R_σ^{obs} satisfies the defining clauses (8)–(10) of R_σ and $\overline{R}_\sigma$ in Definition 7. It cannot also satisfy clause (11) for function types, since then R_σ^{obs} and $\overline{R}_\sigma$ would coincide for all σ, and hence (by Proposition 16) observational equivalence would coincide with applicative equivalence; but by Example 6 we know that in general this is not the case. However, for function types $\sigma \to \sigma'$ with $\sigma \in \{o, \nu\}$ we can simplify clause (11) as in Proposition 21 below. To establish this proposition we need the following property of the relations $\overline{R}_\sigma$ under relabelling along a bijection; it is easily established by induction on the structure of σ, using the derived rule (SUBST) from Remark 2.

Lemma 20. *Suppose given a partial bijection $R : s_1 \rightleftharpoons s_2$, and bijections $R_1 : s_1 \leftrightarrow s_1'$ and $R_2 : s_2 \leftrightarrow s_2'$. Then for all $M_i \in \text{Exp}_\sigma(s_i)$ $(i = 1, 2)$*

$$M_1\ \overline{R}_\sigma\ M_2 \Leftrightarrow M_1[R_1]\ (\overline{R_2 \circ R \circ R_1^{-1}})_\sigma\ M_2[R_2]$$

where $R_2 \circ R \circ R_1^{-1}$ is the composed relation $\{(n_1', n_2') \mid \exists (n_1, n_2) \in R \,.\, (n_i, n_i') \in R_i (i = 1, 2)\}$.

Proposition 21. *Suppose given $R : s_1 \rightleftharpoons s_2$ and $C_i \in \text{Can}_{\sigma \to \sigma'}(s_i)$ $(i = 1, 2)$.*

1. *When $\sigma = o$, $C_1\ R_{o \to \sigma'}\ C_2$ if and only if for all $b \in \{\mathit{true}, \mathit{false}\}$, $C_1 b\ \overline{R}_{\sigma'}\ C_2 b$.*
2. *When $\sigma = \nu$, $C_1\ R_{\nu \to \sigma'}\ C_2$ if and only if*
   (a) for all $(n_1, n_2) \in R$, $C_1 n_1\ \overline{R}_{\sigma'}\ C_2 n_2$, and
   (b) $C_1 n\ (\overline{R \oplus I_{\{n\}}})_{\sigma'}\ C_2 n$
   where n is some name not in $s_1 \cup s_2$.

Proof. The 'only if' direction of each statement follows almost immediately from definition (11). For the 'if' direction, suppose given $R' : s_1 \rightleftharpoons s_2'$ and $D_i \in \text{Can}_\sigma(s_i \oplus s_i')$ $(i = 1, 2)$ with

$$D_1\ (R \oplus R')_\sigma\ D_2 \quad . \tag{23}$$

It suffices to show that

$$C_1 D_1\ (\overline{R \oplus R'})_{\sigma'}\ C_2 D_2 \quad . \tag{24}$$

In case $\sigma = o$, (23) implies $D_1 = D_2 \in \{true, false\}$, hence $C_1 D_1 \ \overline{R}_{\sigma'} \ C_2 D_2$ holds by hypothesis, and therefore so does (24), by Lemma 9.

In case $\sigma = \nu$, (23) implies either $(D_1, D_2) \in R$ or $(D_1, D_2) \in R'$. The first possibility yields (24) much as in the case $\sigma = o$. In the second case, we can express R' as $R_1 \oplus R_2$ where $R_1 = \{(D_1, D_2)\}$ and $R_2 = R' \setminus \{(D_1, D_2)\}$. Then Lemma 20 and the assumption that $C_1 n \ (\overline{R \oplus I_{\{n\}}})_{\sigma'} \ C_2 n$ implies $C_1 D_1 \ (\overline{R \oplus R_1})_{\sigma'} \ C_2 D_2$; hence by Lemma 9, (24) holds since $R' = R_1 \oplus R_2$. □

Theorem 22. *Observational equivalence coincides with applicative equivalence for expressions of first order types. In other words, if σ is of the form $\sigma_k \to \sigma_{k-1} \to \cdots \to \sigma_0$ with each σ_i either ν or o, then for all $M_1, M_2 \in \mathrm{Exp}_\sigma(s)$*

$$s \vdash M_1 \approx_\sigma M_2 \Leftrightarrow M_1 \ (\overline{I_s})_\sigma \ M_2 \quad .$$

Proof. By Theorem 14 and Proposition 16, it suffices to prove for first order σ, and any $R : s_1 \rightleftharpoons s_2$ and $M_i \in \mathrm{Exp}_\sigma(s_i)$, that

$$M_1 R^{\mathrm{obs}}_\sigma M_2 \Rightarrow M_1 \ \overline{R}_\sigma \ M_2 \quad .$$

We do this by induction on the structure of σ. The base cases $\sigma = o, \nu$ are covered by the last part of Lemma 17. For the induction step we have to show that the property holds of $\tau \to \sigma$ ($\tau \in \{o, \nu\}$) when it does of σ. For this, by Propositions 19 and 21 it suffices to check that $R^{\mathrm{obs}}_{\tau \to \sigma}$ satisfies the analogue of the 'only if' part of the latter proposition. In other words it suffices to check that if $C_1 R^{\mathrm{obs}}_{\tau \to \sigma} C_2$, then

- when $\tau = o$, $C_1 b \ \overline{R}_\sigma \ C_2 b$ for all $b \in \{true, false\}$; and
- when $\tau = \nu$
  - for all $(n_1, n_2) \in R$, $C_1 n_1 \ \overline{R}_\sigma \ C_2 n_2$, and
  - $C_1 n \ (\overline{R \oplus I_{\{n\}}})_\sigma \ C_2 n$

  where n is any name not in $s_1 \cup s_2$.

We indicate the proof of the last of these properties (the others being straightforward to establish). So suppose $C_1 R^{\mathrm{obs}}_{\nu \to \sigma} C_2$ and $n \notin s_1 \cup s_2$. Given any $\tau \in \{o, \nu\}$ and any $\lambda x : \sigma \,.\, P \in \mathrm{Can}_{\sigma \to \tau}(dom(R \oplus I_{\{n\}}))$, we have to show

$$(\lambda x : \sigma \,.\, P)(C_1 n) \ (\overline{R \oplus I_{\{n\}}})_\tau \ (\lambda x : \sigma \,.\, P)(C_2 n) \quad . \tag{25}$$

Consider

$$P' \stackrel{def}{=} \nu n \,.\, (\lambda x : \sigma \,.\, P)(f n)$$

Since $\lambda f : \nu \to \sigma \,.\, P' \in \mathrm{Can}_{(\nu \to \sigma) \to \tau}(dom(R))$, we have

$$(\lambda f : \nu \to \sigma \,.\, P') C_1 \ \overline{R}_\tau \ (\lambda f : \nu \to \sigma \,.\, P') C_2 \quad . \tag{26}$$

So if

$$s_1 \oplus \{n\} \vdash (\lambda x : \sigma \,.\, P)(C_1 n) \Downarrow_\tau (s_1') D_1$$
$$s_2 \oplus \{n\} \vdash (\lambda x : \sigma \,.\, P[R \oplus I_{\{n\}}])(C_2 n) \Downarrow_\tau (s_2') D_2$$

then by definition of P', (26) implies $D_1\ (R \oplus R')_\tau\ D_2$ for some $R' : \{n\} \oplus s'_1 \rightleftharpoons \{n\} \oplus s'_2$. In case $\tau = o$ this immediately gives $D_1 = D_2$ and hence that (25) holds, as required. In case $\tau = \nu$, it suffices to show that

$$D_1 = n \Leftrightarrow D_2 = n \quad . \tag{27}$$

For then $D_1\ (R \oplus I_{\{n\}} \oplus R'')_\tau\ D_2$ for some R'' (namely $R'' = R' \setminus \{(n, n)\}$) and hence (25) holds, as required. To see that (27) holds, consider applying the test

$$\lambda f : \nu \to \sigma\ .\ \nu n\ .\ ((\lambda x : \sigma\ .\ P)(fn) = n) \in \mathrm{Can}_{(\nu \to \sigma) \to o}(dom(R))$$

to $C_1 R^{\mathrm{obs}}_{\tau \to \sigma} C_2$. □

Corollary 23. *The relation of observational equivalence between nu-calculus expressions of first order type is decidable.*

Proof. By the above theorem, it suffices to check that the relations $\overline{R}_\sigma$ are decidable for first order σ. For this, it is sufficient to establish the decidability of the relations R_σ (for first order σ) since Theorem 12 ensures that we can calculate s'_1 and s'_2 in clause (8), and then there are only finitely R' for which a decidable property has to be checked. The decidability of R_σ can be established by induction on the structure of the first order type σ, the base cases being trivial, and the induction step following from Proposition 21. □

5 Conclusion

The nu-calculus combines higher order functions with an extremely simple kind of dynamically created local state. Our original motivation for introducing and studying such a computationally simple language was as a vehicle for understanding what, if any, are the difficulties introduced by pure locality of state when reasoning about properties of higher order functions. Our expectation that the difficulties would not be very great has proved to be incorrect, as the results and examples in this paper show.

On a more positive note, we have developed a useful notion of logical relation which builds in a version of 'representation independence' for local names. We showed that it can be used to establish observational equivalence between expressions (Theorem 14). We expect that extensions of this logical relations approach will prove useful for studying observational equivalence in computationally more interesting languages (such as a larger fragment of ML with dynamically created references and exception names).

For the nu-calculus, this method of establishing observational equivalence is incomplete in general (Example 6), but is complete for expressions of first order type (Theorem 22). Of course, the fundamental problem is that (canonical) expressions $\lambda x : \sigma \to \sigma'\ .\ M$ of function type are not in general determined up to observational equivalence by their *extensional* behaviour, i.e. by the function on closed expressions $C \mapsto M[C/x]$ that they determine via application. Nevertheless, it may be that observational equivalence at function types, $\approx_{\sigma \to \sigma'}$, can be explained *compositionally*

by applying some construction to $\approx_\sigma$ and $\approx_{\sigma'}$. Clearly this compositionality property is enjoyed by the notion of applicative equivalence (Definition 13). We leave as an open question whether observational equivalence also has this property.

This paper has taken an operationally-based approach. Section 4 of [14] outlines an approach to the denotational semantics of the nu-calculus which builds on work of Moggi [10] using categorical *monads*. The monadic approach enforces a distinction between denotations of values (expressions in canonical form) and denotations of computations (arbitrary expressions). This is helpful, since it allows us to identify explicitly and simply what structure is needed in a model to give a static meaning for the key dynamic aspect of the nu-calculus, viz. *the action of computing a new name*. Further details will appear elsewhere.

Acknowledgements We are grateful to Eugenio Moggi, Peter O'Hearn, Allen Stoughton and Robert Tennent for making their unpublished work available to us. We have benefited from many conversations with them on the topic of this paper.

References

1. S. Abramsky. The Lazy Lambda Calculus. In D. Turner (ed.), *Research Topics in Functional Programming* (Addison-Wesley, 1990), pp 65–116.
2. H.-J. Boehm. Side-effects and aliasing can have simple axiomatic descriptions, *ACM Trans. Prog. Lang. Syst.* 7(1985) 637–655.
3. M. Felleisen and D. P. Friedman. A Syntactic Theory of Sequential State, *Theoretical Computer Science* 69(1989) 243–287.
4. F. Honsell, I. A. Mason, S. Smith and C. Talcott. A Variable Typed Logic of Effects. In *Proc. Computer Science Logic 1992*, Lecture Notes in Computer Science (Springer-Verlag, Berlin, 1993), *to appear*.
5. D. J. Howe. Equality in Lazy Computation Systems. In *Proc. 4th Annual Symp. on Logic in Computer Science*, Asilomar, 1989 (IEEE Computer Society Press, Washington, 1989) pp 198–203.
6. I. A. Mason and C. Talcott. References, local variables and operational reasoning. In *Proc. 7th Annual Symp. on Logic in Computer Science*, Santa Cruz, 1992 (IEEE Computer Society Press, Washington, 1992) pp 186–197.
7. A. Meyer and K. Sieber. Towards fully abstract semantics for local variables: preliminary report. In *Conf. Record 15th Symp. on Principles of Programming Languages*, San Diego, 1988 (ACM, New York, 1988) pp 191-203.
8. R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science* 4(1977) 1–22.
9. R. Milner, M. Tofte and R. Harper. *The Definition of Standard ML* (MIT Press, 1990).
10. E. Moggi. Notions of Computation and Monads, *Information and Computation* 93(1991) 55–92.
11. P. W. O'Hearn. A Model for Syntactic Control of Interference, *Mathematical Structures in Computer Science*, to appear.
12. P. W. O'Hearn and R. D. Tennent. Semantics of Local Variables. In M. P. Fourman, P. T. Johnstone and A. M. Pitts (eds), *Applications of Categories in Computer Science*, L.M.S. Lecture Note Series 177 (Cambridge University Press, 1992), pp 217–238.
13. P. W. O'Hearn and R. D. Tennent. Relational Parametricity and Local Variables. In *Conf. Record 20th Symp. on Principles of Programming Languages*, Charleston, 1993 (ACM, New York, 1993) pp 171–184.
14. A. M. Pitts and I. D. B. Stark. On the Observational Properties of Higher Order Functions that Dynamically Create Local Names (preliminary report). In *Proceedings of the ACM SIGPLAN Workshop on State in Programming Languages*, Copenhagen, 1993, Yale Univ. Dept. Computer Science Tech. Report.
15. G. D. Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theoretical computer Science* 1(1975) 125–159.
16. G. D. Plotkin and M. Abadi. A Logic for Parametric Polymorphism. In *Proceedings of the Conference on Typed Lambda Calculus and its Applications*, Utrecht, 1993, Lecture Notes in Computer Science Vol. 664 (Springer-Verlag, Berlin, 1993) pp 361-375.
17. J. C. Reynolds. Syntactic Control of Interference. In *Conf. Record 5th Symp. on Principles of Programming Languages*, Tucson, 1978 (ACM, New York, 1978) pp 39–46.
18. R. D. Tennent. Semantic Analysis of Specification Logic, *Information and Computation* 85(1990) 135–162.

This article was processed using the LaTeX macro package with LLNCS style