

# Extending Howe's Method to Early Bisimulations for Typed Mobile Embedded Resources with Local Names<sup>\*</sup>

Jens Chr. Godskesen and Thomas Hildebrandt

Department of Theoretical Computer Science  
IT University of Copenhagen  
{jcg, hilde}@itu.dk

**Abstract.** We extend Howe's method to prove that *input-early* strong and -delay contextual bisimulations are congruences for the Higher-order mobile embedded resources (Homer) calculus, a typed higher order process calculus with active mobile processes, nested locations and local names which conservatively extends the syntax and semantics of higher-order calculi such as Plain CHOCS and HO $\pi$ . We prove that the input-early strong and -delay contextual bisimulation congruences are sound co-inductive characterisations of barbed bisimulation congruence and in fact *complete* in the strong case. The extension of Howe's method provides considerably simpler congruence proofs than established previously for similar calculi for mobile processes in nested locations.

## 1 Introduction

The ability to reason compositionally about the behaviour of programs and processes and compare the behaviour of processes in any context are key issues in semantics. A way to achieve this is to give a co-inductive characterisation of a behavioural congruence, for process calculi typically a notion of barbed bisimulation congruence [1], in terms of a labelled transition bisimulation.

In the present paper we study this problem for higher-order process calculi allowing to represent active, copyable (non-linear) mobile processes with local names and nested locations as found in the Seal calculus [2], the M-calculus [3] and its recent successor the Kell calculus[4]. This has proven to be a difficult problem.

Our main contribution is to extend *Howe's method* [5–8], a classical technique for proving that applicative bisimulation is a congruence, to *early* bisimulations for a core higher-order calculus with local names, extended with non-linear active process mobility and explicit, nested locations. We call the calculus Homer as short for *Higher-Order Mobile Embedded Resources*<sup>1</sup>. Thereby we also propose a calculus which conservatively extends the standard syntax and semantics of higher-order process calculi such as Plain CHOCS [9] and HO $\pi$  [10], which has been one of the main design criteria behind

---

<sup>\*</sup> Preliminary version submitted for publication, june 2005. Funded in part by the Danish Research Agency (grant no.: 2059-03-0031) - the LaCoMoCo project. Available at [www.itu.dk/research/theory/CoMo/Papers/homerbisim.pdf](http://www.itu.dk/research/theory/CoMo/Papers/homerbisim.pdf)

<sup>1</sup> and reference to the dangerous mobile embedded resources in the legend of Troy.

Homer. The result is a calculus with considerable simpler syntax and semantics than present calculi with comparable expressive power.

*Active process mobility* is introduced in Homer by the prefix  $n\langle r \rangle$  denoting a resource  $r$  residing at the location (or address)  $n$ , which may be *moved* by the complementary prefix,  $\bar{n}(x)$ , expressed by a reaction rule

$$n\langle r \rangle p \parallel \bar{n}(x)q \searrow p \parallel q[r/x] ,$$

where  $r$  is a process and  $x$  is a process variable binding any number of occurrences of  $x$  in  $q$ . The rule complements the usual reaction rule for higher-order process calculi

$$\bar{n}\langle r \rangle p \parallel n(x)q \searrow p \parallel q[r/x] ,$$

where the process  $r$  is assumed to be *passive*, meaning that it can neither compute internally nor interact with other processes before it has been sent. In the usual way the receiver may activate or forward any number of copies of the process, but once a copy has started computing, it cannot be moved again. This kind of mobility is known as *code mobility* or *passive process mobility*. To allow *active process mobility* the process  $r$  in the prefix  $n\langle r \rangle$  can perform internal computations, that is  $r \searrow r'$  implies  $n\langle r \rangle p \searrow n\langle r' \rangle p$  as in [11]. We allow interaction with (arbitrarily) nested active mobile resources by the use of *nested names* as introduced in [12]. For instance, a resource  $r$  may be sent to the subaddress *server* : 80 by the reaction

$$\overline{\text{server}} : 80\langle r \rangle p \parallel \text{server}\langle 80(x)q' \parallel q'' \rangle \searrow p \parallel \text{server}\langle q'[r/x] \parallel q'' \rangle .$$

Dually, a resource  $r$  may be taken from the subaddress *server* : 80 by the reaction

$$\text{server}\langle 80\langle r \rangle q' \parallel q'' \rangle \parallel \overline{\text{server}} : 80(x)p \searrow \text{server}\langle q' \parallel q'' \rangle \parallel p[r/x] .$$

As usual we let  $(n)p$  denote a process  $p$  in which the name  $n$  is local. This provides the means to control access to resources. A standard example is the *perfect firewall equation* [13]:  $(n)(n\langle p \rangle) \approx \mathbf{0}$ , expressing that a resource computing at a location with a *local name* has no observable behaviour.

When a resource is moved from a location it may be necessary to extend the scope of a name through a location boundary, which we will refer to as *vertical scope extension*. For instance, if the resource  $r$  below contains the name  $n$ , we will expect the reaction

$$m\langle (n)(m'\langle r \rangle \parallel p) \rangle \parallel \overline{m} : m'(x)q \searrow (n)(m\langle p \rangle \parallel q[r/x]) \quad (1)$$

in which the scope of  $n$  is extended. If mobile processes can *not* be copied during computation, such as in the Mobile Ambients calculus [13], vertical scope extension can simply be dealt with in the structural congruence by introducing the equation

$$m\langle (n)p \rangle \equiv (n)m\langle p \rangle , \text{ if } n \neq m .$$

However, as also identified in [2, 14, 4] this equation is unsound if mobile processes can be copied. If placed in a context with a copying process  $(-) \parallel \bar{m}(x)(x \parallel x)$  then the left hand process in (1) above will reduce to  $(n)p \parallel (n)p$  while the right hand process will reduce to  $(n)(p \parallel p)$ , which in general will not be equivalent.

$(inactive) \frac{}{0 : \tilde{n}}$	$(variable) \frac{}{x : \tilde{n}}$	$(context) \frac{}{(-)_{\tilde{n}} : \tilde{n}' \quad \tilde{n} \subseteq \tilde{n}'}$	$(prefix) \frac{e : \tilde{n}}{\varphi e : \varphi \cup \tilde{n}}$
$(rest) \frac{F : \tilde{n}}{(n)F : \tilde{n} \setminus n}$	$(abs) \frac{p : \tilde{n}}{(x)p : \tilde{n}}$	$(parallel) \frac{F : \tilde{n} \quad p : \tilde{n}'}{F \parallel p : \tilde{n} \cup \tilde{n}'}$	
$(concretion) \frac{p' : \tilde{n}' \quad p : \tilde{n}}{\langle p' : \tilde{n}' \rangle p : \tilde{n} \cup \tilde{n}'}$		$(nesting) \frac{F : \tilde{n}' \quad p : \tilde{n}}{\varphi \langle F : \tilde{n}' \rangle p : \tilde{n} \cup \tilde{n}' \cup \varphi}$	

**Table 1.** Typing rules.

The solution taken in Homer is to extend the scope of the name  $n$  in the reaction (1) *if and only if* the name  $n$  is free in  $p$ , which is consistent with the semantics of Plain CHOCS and HO $\pi$ . This solution contrasts the more eager solution in [14, 4] where the scope of all local names is extended before resources move, i.e. local names will *always* be shared between dynamically created copies of a process.

The vertical scope extension of Homer implies that a context can test if a name is free in a mobile process (see Sec. 3 for a detailed discussion). Consequently, any non-trivial congruence must be *well typed*, meaning that related processes  $r$  and  $r'$  must have the same set of free names. This means that the firewall equation  $(n)(n\langle p \rangle) \approx \mathbf{0}$ , will only hold if the process  $p$  has no free variables. We remedy this problem by explicitly typing processes by a set of names *including* the free names of the process, but allowing additional (unused) names. Interestingly, it turns out that one then also needs to explicitly type all mobile sub resources. We can now state a *well typed firewall equation* as  $(n)(n\langle p : \tilde{n} \rangle) : \tilde{m} \approx \mathbf{0} : \tilde{m}$ , where  $\tilde{n}$  is the type of process  $p$ .

The above constructions are the only primitives in Homer. In particular, process passing is the only means of communication, thus, the issue of name passing is separated from process passing. In [15] we show that the synchronous  $\pi$ -calculus can in fact be encoded in a (simplified) variant of Homer. The simplicity of the calculus helps us to adapt Howe's method to show that both strong and so called delay *input-early* contextual bisimulation are congruences for Homer. This gives a sound and complete characterisation of strong barbed bisimulation, and a sound characterisation of weak barbed bisimulation. It also helps pinpointing the need for typing processes and mobile sub resources, which is studied in more detail in [16].

## 2 The Homer Calculus

We assume an infinite set of *names*  $\mathcal{N}$  ranged over by  $m$  and  $n$ , and let  $\tilde{n}$  range over finite sets of names. We let  $\delta$  range over non-empty finite sequences of names, referred to as *paths* and  $\bar{\delta}$  denotes *co-addresses*. We let  $\varphi$  range over  $\delta$  and  $\bar{\delta}$  and define  $\bar{\bar{\delta}} = \delta$ . We assume an infinite set of *process variables*  $\mathcal{V}$  ranged over by  $x$  and  $y$ . The sets  $\mathbf{p}$  of

$\begin{aligned} \varphi(e : \tilde{n}) &= \varphi e : \tilde{n} \cup \varphi & F : \tilde{n} \parallel p' : \tilde{n}' &= F \parallel p' : \tilde{n} \cup \tilde{n}' & (n)(F : \tilde{n}) &= (n)F : \tilde{n} \setminus n \\ (y)(p : \tilde{n}) &= (y)p : \tilde{n} & \langle p : \tilde{n} \rangle (p' : \tilde{n}') &= \langle p : \tilde{n} \rangle p' : \tilde{n} \cup \tilde{n}' \\ \varphi \langle F : \tilde{n}' \rangle (p : \tilde{n}) &= \varphi \langle F : \tilde{n}' \rangle p : \tilde{n} \cup \tilde{n}' \cup \delta \end{aligned}$
---

**Table 2.** Extension of process constructors to typed terms.

process expressions,  $\mathbf{a}$  of abstractions, and  $\mathbf{c}$  of concretions are defined by the grammar:

$$p ::= \mathbf{0} \mid x \mid \varphi e \mid p \parallel p' \mid (n)p \quad , \quad a ::= (x)p \quad , \quad c ::= b \mid (n)c \quad ,$$

where  $b ::= \langle p' : \tilde{n} \rangle p$  is a basic (unrestricted) concretion and  $e ::= a \mid b$ . We let  $\mathbf{f}$ , ranged over by  $f$ , denote  $\mathbf{p} \cup \mathbf{a} \cup \mathbf{c}$ . Whenever  $e$  denotes a basic abstraction we let  $\bar{e}$  denote a concretion, and vice versa.

The constructors are the standard ones from concurrent process calculi, extended with process variables and by types  $\tilde{n}$ . The processes  $\bar{\delta} \langle p' : \tilde{n} \rangle p$  and  $\delta(x)p$  correspond to sending and receiving processes, except that paths, and not only names, are allowed as addresses. As explained in the introduction, the new constructs  $\bar{\delta} \langle p' : \tilde{n} \rangle p$  and  $\delta(x)p$  add strong mobility to the calculus.

The restriction operator  $(n)$  binds the name  $n$  and  $(x)$  binds the variable  $x$ . The sets  $fn(f)$  and  $fv(f)$  of *free names* (including the types) and *free variables* are defined accordingly as usual. We say that a term with no free variables is *closed* and let  $\mathbf{f}_c$  ( $\mathbf{p}_c$ ) denote the set of closed terms (processes).

As usual, we let prefixing and restriction be right associative and bind stronger than parallel composition. Often we shall write  $\langle p : \tilde{n} \rangle$  instead of  $\langle p : \tilde{n} \rangle \mathbf{0}$ . For a set of names  $\tilde{n} = \{n_1, \dots, n_k\}$  we let  $(\tilde{n})f$  denote  $(n_1) \cdots (n_k)f$ . We will write  $n$  for the set  $\{n\}$  and  $\delta$  for the set of names in  $\delta$  (or  $\bar{\delta}$ ) when no confusion can occur. We write  $f \equiv_\alpha f'$ , if  $f$  and  $f'$  are  $\alpha$ -convertible (wrt. both names and variables), and we let  $\mathbf{f}/\alpha$  (and  $\mathbf{f}_c/\alpha$ ) denote the set of  $\alpha$ -equivalence classes of (closed) terms. Likewise we let  $\mathbf{p}/\alpha$  (and  $\mathbf{p}_c/\alpha$ ) denote the set of  $\alpha$ -equivalence classes of (closed) processes. From now we consider terms up to  $\alpha$ -equivalence.

We define a family of type indexed *evaluation contexts*  $E_{\tilde{n}}$ . They are contexts with no free variables, and whose “hole” is indexed by a type  $\tilde{n}$  and is either not guarded by a prefix or guarded by a prefix  $\delta$  and nested in a concretion, i.e.

$$E_{\tilde{n}} ::= (-)_{\tilde{n}} \mid E_{\tilde{n}} \parallel p \mid (n)E_{\tilde{n}} \mid \delta \langle E_{\tilde{n}} : \tilde{n}' \rangle p \quad , p \in \mathbf{p}_c.$$

The free names of  $E_{\tilde{n}}$  are defined standardly. If the type index of the hole in  $E_{\tilde{n}}$  is not important we may write  $E$  instead of  $E_{\tilde{n}}$ .

We let  $F$  range over processes, abstractions, concretions, and evaluation contexts. If  $F : \tilde{n}$  can be inferred from the rules in Table 1 we say that  $F$  is of type  $\tilde{n}$ . From now on writing  $F : \tilde{n}$  we assume  $F$  is of type  $\tilde{n}$ . Notice that  $F : \tilde{n}$  implies  $fn(F) \subseteq \tilde{n}$ . We can prove a simple kind of subsumption (using  $\alpha$ -conversion if needed).

**Proposition 1.** (*Subsumption*)  $F : \tilde{n}$  implies  $F : \tilde{n}'$  for all  $\tilde{n}'$  where  $\tilde{n} \subseteq \tilde{n}'$ .

By convenience we extend the process constructors to typed terms as defined in Table 2 (following the rules of Table 1). We let  $P, A$ , and  $C$  range over the set of typed processes  $\mathbf{P}/\alpha$ , abstractions  $\mathbf{A}/\alpha$ , and concretions  $\mathbf{C}/\alpha$  up to  $\alpha$ -equivalence respectively, and we let  $T$  range over  $\mathbf{T}/\alpha = \mathbf{P}/\alpha \cup \mathbf{A}/\alpha \cup \mathbf{C}/\alpha$ . The closed variants of  $\mathbf{T}/\alpha$ ,  $\mathbf{P}/\alpha$ ,  $\mathbf{A}/\alpha$ , and  $\mathbf{C}/\alpha$  are denoted by  $\mathbf{T}_{c/\alpha}$ ,  $\mathbf{P}_{c/\alpha}$ ,  $\mathbf{A}_{c/\alpha}$ , and  $\mathbf{C}_{c/\alpha}$ . Finally, we let  $\mathcal{E}_{\tilde{n}}$  range over typed type indexed evaluation contexts and occasionally we leave out the type index writing  $\mathcal{E}$  for  $\mathcal{E}_{\tilde{n}}$  if the type index is not important.

Whenever  $F : \tilde{n}$  then we write  $E_{\tilde{n}}(F)$  (or by convenience  $E_{\tilde{n}} : \tilde{n}'(F : \tilde{n})$ ) for the insertion of  $F$  in the hole of  $E_{\tilde{n}}$ . Note that free names of  $F$  may get bound by insertion of  $F$  in the hole of a context.

**Proposition 2.** *If  $E_{\tilde{n}} : \tilde{n}'$  and  $F : \tilde{n}$  then  $E_{\tilde{n}}(F) : \tilde{n}'$ .*

Substitution of all free occurrences of a variable  $x$  in a typed term  $T$  by a typed process  $P$  is defined inductively by the rules in Table 3, extending types as expected.

$0 : \tilde{n}[p : \tilde{n}'/x] = 0 : \tilde{n} \cup \tilde{n}'$	$(n)f : \tilde{n}[p : \tilde{n}'/x] = (n)(f : \tilde{n}[p : \tilde{n}'/x])$ , if $n \notin \tilde{n}'$
$y : \tilde{n}[p : \tilde{n}'/x] = \begin{cases} y : \tilde{n} \cup \tilde{n}', & \text{if } x \neq y \\ p : \tilde{n} \cup \tilde{n}', & \text{if } x = y \end{cases}$	$(y)p : \tilde{n}[P/x] = (y)(p : \tilde{n}[P/x])$ , if $x \neq y$
$\varphi e : \tilde{n}[P/x] = \varphi(e : \tilde{n}[P/x])$	$\langle P' \rangle p : \tilde{n}[P/x] = \langle P'[P/x] \rangle (p : \tilde{n}[P/x])$
$p \parallel p' : \tilde{n}[P/x] = p : \tilde{n}[P/x] \parallel p' : \tilde{n}[P/x]$	$\delta \langle P' \rangle p : \tilde{n}[P/x] = \delta \langle P'[P/x] \rangle (p : \tilde{n}[P/x])$

**Table 3.** Substitution.

**Proposition 3.** *If  $f : \tilde{n}[p : \tilde{n}'/x] = f' : \tilde{n}''$  then  $\tilde{n}'' = \tilde{n} \cup \tilde{n}'$ .*

In the remaining part of this paper we consider only a restricted form of concretions on the form  $(\tilde{n})\langle p' : \tilde{n}' \rangle p$  where  $\tilde{n} \subseteq \tilde{n}'$ . These concretions we close by convenience under process operators, hence whenever  $c = (\tilde{n})\langle p_1 : \tilde{n}_1 \rangle p$  and assuming  $\tilde{n} \cap (fn(p') \cup n \cup \delta) = \emptyset$  (using  $\alpha$ -conversion if needed) we write  $c \parallel p'$  for  $(\tilde{n})\langle p_1 : \tilde{n}_1 \rangle (p \parallel p')$ , we write  $\delta \langle c : \tilde{n}' \rangle p'$  for  $(\tilde{n})\langle p_1 : \tilde{n}_1 \rangle \delta \langle p : \tilde{n}' \tilde{n} \rangle p'$ , and we let  $(n)c$  denote  $(n\tilde{n})\langle p_1 : \tilde{n}_1 \rangle p$  if  $n \in \tilde{n}_1$  and otherwise it denotes  $(\tilde{n})\langle p_1 : \tilde{n}_1 \rangle (n)p$ . We also allow abstractions to be closed under process constructs, hence whenever  $a = (x)p$  and assuming  $x \notin fv(p')$  (using  $\alpha$ -conversion if needed) we write  $a \parallel p'$  for  $(x)(p \parallel p')$ ,  $\delta \langle a : \tilde{n} \rangle p'$  for  $(x)\delta \langle p : \tilde{n} \rangle p'$ , and  $(n)a$  for  $(x)(n)p$ . Finally, we define the *application* of a typed abstraction  $A = (x)p : \tilde{n}$  to a typed concretion  $C = (\tilde{n}')\langle P \rangle P'$  (assuming that  $\tilde{n} \cap \tilde{n}' = \emptyset$ , using  $\alpha$ -conversion if needed) by  $A \cdot C = C \cdot A = (\tilde{n}')\langle p : \tilde{n}[P/x] \parallel P' \rangle$ .

### 3 Reaction Semantics

We provide Homer with a reaction semantics as usual defined through the use of evaluation contexts, structural congruence, and reaction rules.

$p \parallel \mathbf{0} \equiv p$	$(p \parallel p') \parallel p'' \equiv p \parallel (p' \parallel p'')$	$p \parallel q \equiv q \parallel p$	$(n)(m)p \equiv (m)(n)p$
$(n)p \parallel q \equiv (n)(p \parallel q)$ , if $n \notin \text{fn}(q)$		$(n)p \equiv p$ , if $n \notin \text{fn}(p)$	

**Table 4.** Structural congruence.

As touched upon in the introduction equivalent processes must have the same free names, this we capture using well typedness. A binary relation  $\mathcal{R}$  on  $\mathbf{T}/\alpha$  is *well typed* if  $f : \tilde{n} \mathcal{R} f' : \tilde{n}'$  implies  $\tilde{n} = \tilde{n}'$  and  $f : \tilde{n}'' \mathcal{R} f' : \tilde{n}''$  for all  $\tilde{n}''$  where  $\tilde{n} \subseteq \tilde{n}''$ . In the sequel we always assume binary relations on  $\mathbf{T}/\alpha$  to be well typed.

We say that a well-typed binary relation  $\mathcal{R}$  on  $\mathbf{P}/\alpha$  is *substitutive* if  $P \mathcal{R} P'$  and  $P_1 \mathcal{R} P'_1$  implies  $P[P_1/x] \mathcal{R} P'[P'_1/x]$ . We also say that  $\mathcal{R}$  is *constructor compatible* if  $P \mathcal{R} P'$  and  $P_1 \mathcal{R} P'_1$  implies  $\varphi(x)P \mathcal{R} \varphi(x)P'$ ,  $\varphi\langle P_1 \rangle P \mathcal{R} \varphi\langle P'_1 \rangle P'$ ,  $P \parallel P_1 \mathcal{R} P' \parallel P'_1$ , and  $(n)P \mathcal{R} (n)P'$ . A well typed relation  $\mathcal{R}$  on  $\mathbf{P}/\alpha$  is a *congruence* if it is substitutive and constructor compatible.

*Structural congruence*  $\equiv$  is then the least equivalence relation on  $\mathbf{p}/\alpha$  that is a congruence and that satisfies the (usual) rules in Table 4, which do *not* allow vertical scope extension. We extend structural congruence to typed processes by  $p : \tilde{n} \equiv p' : \tilde{n}$  if  $p \equiv p'$ .

As Homer permits reactions between a process and an arbitrarily deeply nested sub-resource, we define a restricted set of evaluation contexts, i.e. a family of *path contexts*  $D_{\tilde{n}, \gamma}$  indexed by (the type of its hole and) a path address  $\gamma \in \mathcal{N}^*$  which indicates the path under which the context's ‘hole’ is found. We do so conveniently using *multi hole path contexts*. A multi hole path contexts  $D_{\tilde{n}, \gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}$  has  $k+1$  holes and is also indexed by a sequence of types  $\tilde{n}_1, \dots, \tilde{n}_k$  indexing the auxiliary holes in the context. A multi hole path context is defined inductively by  $D_{\tilde{n}, \varepsilon}^\varepsilon = (-)_{\tilde{n}}$  (where epsilon denotes an empty sequence) and

$$D_{\tilde{n}, \delta\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k} ::= \delta\langle (\tilde{m})(D_{\tilde{n}, \gamma}^{\tilde{n}_1, \dots, \tilde{n}_{k-2}} \parallel (-)_{\tilde{n}_{k-1}}) : \tilde{n}' \rangle (-)_{\tilde{n}_k}$$

such that  $\gamma \cap \tilde{m} = \emptyset$  and none of the names in  $\tilde{m}$  are already bound in  $D_{\tilde{n}, \gamma}^{\tilde{n}_1, \dots, \tilde{n}_{k-2}}$ , i.e. we assume all names binding the hole of a context are unique. For closed typed processes  $p_i : \tilde{n}_i$ ,  $i = 1, \dots, k$ , we write  $D_{\tilde{n}, \gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}(p_1, \dots, p_k)$  for the insertion of  $p_i$  in the hole indexed by  $\tilde{n}_i$  in the context, resulting in a (single hole) path context  $D_{\tilde{n}, \gamma}$ .

We let  $\mathcal{D}_{\tilde{n}, \gamma}$  range over well typed (single hole) path contexts. If the indexed path in  $\mathcal{D}_{\tilde{n}, \gamma}$  is not important we write  $\mathcal{D}_{\tilde{n}}$  instead. Finally, we define  $\searrow$  as the least binary relation on  $\mathbf{P}_c/\alpha$  satisfying the (parametrized) reaction rule in Table 5 and closed under structural congruence and all type matching evaluation contexts. By the latter we mean that  $p : \tilde{n} \searrow p' : \tilde{n}$  implies  $\mathcal{E}_{\tilde{n}}(p) \searrow \mathcal{E}_{\tilde{n}}(p')$  for all  $\mathcal{E}_{\tilde{n}}$ . Notice, that the evaluation context  $\delta\langle \mathcal{E}_{\tilde{n}} : \tilde{n}' \rangle p : \tilde{n}''$  enables internal reactions of active resources.

Below we give a few examples illustrating key ideas of Homer, more examples can be found in [17].

$$(react) \quad \overline{\gamma\delta}e \parallel D_{\tilde{n},\gamma}(\delta\bar{e}) : \tilde{n}' \searrow e : \tilde{n}' \cdot D_{\tilde{n},\gamma}(\bar{e}) : \tilde{n}'$$

**Table 5.** Reaction rule

*Example: Recursion and replication* We may encode recursion (up to weak equivalence) using our ability to copy resources. The encoding is similar to the one in [9], except that recursion variables may appear at sub locations, which makes the definition slightly more complicated. Let  $p : \tilde{n}$  and define

$$rec\ x.p =_{def} (a)(rec^a x.p)$$

where  $rec^a x.p = \bar{a}\langle r : a\tilde{n} \rangle \parallel r$  for  $r = a(x)p[(\bar{a}\langle x : \mathbf{0} \rangle \parallel x) : a/x]$ , with  $a \notin \tilde{n}$ . Intuitively,  $r$  places a copy of  $rec^a x.p$  at all occurrences of  $x$  in  $p$ , i.e.  $rec^a x.p \searrow p[rec^a x.p : a\tilde{n}/x]$ . From recursion one may encode replication in the usual way by  $!p =_{def} rec\ x.(p \parallel x)$ .

*Example: Static local name discipline* The following example illustrate the importance of explicitly typing of sub resources. Suppose  $n \notin \tilde{n}$ . Let  $p_1 =_{def} \bar{a}b(x)\bar{a}(y)(y \parallel y)$ , let  $p_2 =_{def} a\langle(n)(b\langle r : n\tilde{n} \rangle \parallel p) : \tilde{n}\rangle$ , and let  $p_3 =_{def} a\langle(n)(b\langle r : \tilde{n} \rangle \parallel p) : \tilde{n}\rangle$ . Then

$$\begin{aligned} p_1 \parallel p_2 &: ab\tilde{n} \searrow (x)\bar{a}(y)(y \parallel y) : ab\tilde{n} \cdot a\langle(n)(\langle r : n\tilde{n} \rangle \parallel p) : \tilde{n}\rangle : ab\tilde{n} \\ &= (n)(\bar{a}(y)(y \parallel y) \parallel a\langle p : n\tilde{n} \rangle) : ab\tilde{n} \searrow (n)(p \parallel p) : ab\tilde{n} \end{aligned}$$

since by convention  $a\langle(n)(\langle r : n\tilde{n} \rangle \parallel p) : \tilde{n}\rangle = (n)\langle r : n\tilde{n} \rangle a\langle(0 \parallel p) : n\tilde{n}\rangle$ , however

$$\begin{aligned} p_1 \parallel p_3 &: ab\tilde{n} \searrow (x)\bar{a}(y)(y \parallel y) : ab\tilde{n} \cdot a\langle(n)(\langle r : \tilde{n} \rangle \parallel p) : \tilde{n}\rangle : ab\tilde{n} \\ &= \bar{a}(y)(y \parallel y) \parallel a\langle(n)p : \tilde{n}\rangle : ab\tilde{n} \searrow (n)p \parallel (n)p : ab\tilde{n} \end{aligned}$$

since by convention  $a\langle(n)(\langle r' : \tilde{n} \rangle \parallel p) : \tilde{n}\rangle = \langle r' : \tilde{n} \rangle a\langle(n)(0 \parallel p) : \tilde{n}\rangle$ . Hence the scope of  $n$  is extended vertically only if  $n$  appear in the type of  $r$ . Thus, if we did not explicitly type sub resources then (for suitable  $p$ ) the context  $a\langle(n)(b\langle(-)_{n\tilde{n}} \rangle \parallel p) : \tilde{n}\rangle \parallel p_1$  would violate the typed perfect firewall equation  $(m)m\langle q \rangle : n\tilde{n} \approx \mathbf{0} : n\tilde{n}$ , for  $m \neq n \in fn(q)$ .

## 4 Transition Semantics

In this section we provide Homer with a labelled transition semantics. We let  $\pi$  range over the set  $\Pi$  of labels, formally defined as

$$\pi ::= \tau \mid \varphi$$

The set of free names in  $\pi$ ,  $fn(\pi)$ , is  $fn(\varphi)$  whenever  $\pi = \varphi$  and  $\emptyset$  otherwise. As for the reduction semantics we define the transitions for  $\alpha$ -equivalence classes of closed processes. The rules in Table 6 then define a labelled transition system

$$(\mathbf{T}_{c/\alpha}, \longrightarrow \subseteq \mathbf{P}_{c/\alpha} \times \Pi \times \mathbf{T}_{c/\alpha}).$$

$(prefix) \frac{}{\varphi e : \tilde{n} \xrightarrow{\varphi} e : \tilde{n}}$	$(sync) \frac{p : \tilde{n} \xrightarrow{\varphi} A \quad p' : \tilde{n} \xrightarrow{\bar{\varphi}} C}{p \parallel p' : \tilde{n} \xrightarrow{\tau} A \cdot C}$
$(par) \frac{p : \tilde{n} \xrightarrow{\pi} T}{p \parallel p' : \tilde{n} \xrightarrow{\pi} T \parallel p' : \tilde{n}}$	$(sym) \frac{p \parallel p' : \tilde{n} \xrightarrow{\pi} T}{p' \parallel p : \tilde{n} \xrightarrow{\pi} T}$
$(rest) \frac{p : \tilde{n}n \xrightarrow{\pi} T}{(n)p : \tilde{n} \xrightarrow{\pi} (n)T}, n \notin fn(\pi)$	$(nesting) \frac{P \xrightarrow{\pi} T}{\delta\langle P \rangle p : \tilde{n} \xrightarrow{\delta \cdot \pi} \delta\langle T \rangle p : \tilde{n}}$

**Table 6.** Transition rules.

The rules are completely standard, except for the handling of typing and interaction with nested resources, which shows in the *(prefix)* and *(nesting)* rules. The *(prefix)* rule expresses that types are preserved by transitions. The *(nesting)* rule takes care of the communication and computation of arbitrarily deeply nested resources. It uses an operation  $\delta \cdot (-)$  formally defined by:

$$\delta \cdot \tau = \tau, \quad \delta \cdot \delta' = \delta\delta'$$

Note that the operation is not defined for  $\bar{\delta}$  since it is directed “downward” and thus not visible outside the resource. Since  $\delta \cdot \tau = \tau$ , the nesting rule implies that  $\delta\langle P \rangle p : \tilde{n} \xrightarrow{\tau} \delta\langle T \rangle p : \tilde{n}$ , if  $P \xrightarrow{\tau} T$ .

As explained in the introduction, the action prefixes allow two kinds of movement of resources. Since they are completely dual, they are both treated by the rule *(prefix)* and the synchronisation rule *(sync)*. For instance, illustrating also the use of the rule *(nesting)*, we have letting  $\tilde{n}' = n\tilde{m}\tilde{n}$  that

$$n\langle m\langle P \rangle p : m\tilde{n} \rangle p' \parallel \bar{n}\bar{m}(x)p'' : \tilde{n}' \xrightarrow{\tau} \langle P \rangle n\langle p : m\tilde{n} \rangle p' : \tilde{n}' \cdot (x)p'' : \tilde{n}'$$

because  $n\langle m\langle P \rangle p : m\tilde{n} \rangle p' : \tilde{n}' \xrightarrow{nm} \langle P \rangle n\langle p : m\tilde{n} \rangle p' : \tilde{n}'$  and  $\bar{n}\bar{m}(x)p'' : \tilde{n}' \xrightarrow{\bar{n}\bar{m}} (x)p'' : \tilde{n}'$ .

## 5 Bisimulation Congruences

In this section, we address the semantic theory of Homer considering how to describe congruences as bisimulations. First, we define *well typed* strong and weak *barbed bisimulation congruences* based on the reduction semantics and next we define late and early variants of strong and delay contextual transition bisimulations. We prove that the early contextual transition bisimulations are congruences using a novel extension of Howe’s method, based on the approach in [18], and that they are sound and complete with respect to barbed bisimulation congruences.

We define weak and strong *barbs* standardly letting  $\searrow^*$  be the transitive and reflexive closure of  $\searrow$ , and choosing barbs similarly to [19, 20]

$$P \Downarrow n \text{ if } P \equiv (\tilde{n})(n\langle P' \rangle p \parallel p') : \tilde{n}' \text{ and } n \notin \tilde{n}, \quad \text{and} \quad P \Downarrow n \text{ if } \exists P' \searrow^* P'. P' \Downarrow n .$$

In a previous version [17] we show (as in [19, 20]) that this choice of barbs is indeed robust. In particular, one could have chosen to observe location paths.

We restrict binary relations  $\mathcal{R}$  on  $\mathbf{P}/\alpha$  to closed terms by  $\mathcal{R}_c = \mathcal{R} \cap \mathbf{P}_{c/\alpha} \times \mathbf{P}_{c/\alpha}$ .

**Definition 1.** A *weak barbed simulation* is a well typed binary relation  $\mathcal{R}$  on  $\mathbf{P}_{c/\alpha}$  such that whenever  $P_1 \mathcal{R} P_2$ ,

$$i) \text{ if } P_1 \downarrow n \text{ then } P'_1 \downarrow n \quad ii) \text{ if } P_1 \searrow P'_1, \text{ then } \exists P_2 \searrow^* P'_2 \text{ such that } P'_1 \mathcal{R} P'_2$$

$\mathcal{R}$  is a *weak barbed bisimulation* if  $\mathcal{R}$  and  $\mathcal{R}^{-1}$  are weak barbed simulations. *Weak barbed bisimulation congruence*  $\approx_b$  is the largest congruence such that  $(\approx_b)_c$  is a weak barbed bisimulation.

We define strong barbed simulation similar to above by replacing  $\searrow^*$  with  $\searrow$  and  $q \downarrow n$  with  $q \downarrow n$ , and define strong barbed bisimulation congruence  $\sim_b$  accordingly.

**Proposition 4.**  $\sim_b$  and  $\approx_b$  are equivalence relations.

We define *delay transitions* [21] by  $P \xrightarrow{\tau} P$  and  $P \xrightarrow{\pi} P'$ , if  $P \xrightarrow{\tau} P'' \xrightarrow{\pi} P'$ , which compared to weak transitions do not allow  $\tau$ -transitions after the visible transition.

We extend well typed binary relations  $\mathcal{R}$  on  $\mathbf{P}_{c/\alpha}$  to open terms the usual way, by defining  $p : \tilde{n} \mathcal{R}^\circ p' : \tilde{n}$  if  $p : \tilde{n}[P_1/x_1] \dots [P_k/x_k] \mathcal{R} p' : \tilde{n}[P_1/x_1] \dots [P_k/x_k]$  for all  $P_1, \dots, P_k \in \mathbf{P}_{c/\alpha}$ , where  $fv(p) = fv(p') = \{x_1, \dots, x_k\}$ , i.e. requiring two open terms to be related after substitution with closed resources. Also we extend  $\mathcal{R}^\circ$  to typed concretions by  $c : \tilde{n} \mathcal{R}^\square c' : \tilde{n}$  if for all  $A, A \cdot c : \tilde{n} \mathcal{R}^\circ A \cdot c' : \tilde{n}$ .

Like in [22] we introduce an (input) *early delay context bisimulation* by

**Definition 2.** An (input) *early delay context simulation* is a well typed binary relation  $\mathcal{R}$  on  $\mathbf{P}_{c/\alpha}$  such that  $p : \tilde{n} \mathcal{R} P$  implies

1. if  $p : \tilde{n} \xrightarrow{\tau} P_1$  then  $\exists P_2. P \xrightarrow{\tau} P_2$  and  $P_1 \mathcal{R} P_2$
2. if  $p : \tilde{n} \xrightarrow{\delta} A$  then  $\forall C \in \mathbf{C}_{c/\alpha}. \exists A'. P \xrightarrow{\delta} A'$  and  $A \cdot C \mathcal{R} A' \cdot C$
3. if  $p : \tilde{n} \xrightarrow{\delta} A$  then  $\forall C \in \mathbf{C}_{c/\alpha}, \mathcal{D}_{\tilde{n}}. \exists A'. P \xrightarrow{\delta} A'$  and  $\mathcal{D}_{\tilde{n}}(A) \cdot C \mathcal{R} \mathcal{D}_{\tilde{n}}(A') \cdot C$
4. if  $p : \tilde{n} \xrightarrow{\delta} C$  then  $\exists C'. P \xrightarrow{\delta} C'$  and  $C \mathcal{R}^\square C'$
5. if  $p : \tilde{n} \xrightarrow{\delta} C$  then  $\exists C'. P \xrightarrow{\delta} C'$  and  $\forall \mathcal{D}_{\tilde{n}}. \mathcal{D}_{\tilde{n}}(C) \mathcal{R}^\square \mathcal{D}_{\tilde{n}}(C')$

$\mathcal{R}$  is an early delay context bisimulation if both  $\mathcal{R}$  and  $\mathcal{R}^{-1}$  are early delay context simulations. Let  $\approx_e$  denote the largest early delay context bisimulation. We define early strong context simulation by replacing  $\xRightarrow{\delta}$  with  $\xrightarrow{\delta}$ , and let  $\sim_e$  denote the largest early strong context bisimulation.

**Proposition 5.**  $\sim_e$  and  $\approx_e$  are equivalence relations.

The following proposition state the correspondence between the reduction semantics and the transition semantics.

**Proposition 6.**  $P \searrow P'$  iff  $P \equiv \xrightarrow{\tau} \equiv P'$ .

$\frac{0 : \tilde{n} \mathcal{R}^\circ P}{0 : \tilde{n} \mathcal{R}^\blacksquare P}$	$\frac{x : \tilde{n} \mathcal{R}^\circ P}{x : \tilde{n} \mathcal{R}^\blacksquare P}$	$\frac{p_1 : \tilde{n} \mathcal{R}^\blacksquare p'_1 : \tilde{n} \quad p_2 : \tilde{n} \mathcal{R}^\blacksquare p'_2 : \tilde{n} \quad p'_1 \parallel p'_2 : \tilde{n} \mathcal{R}^\circ P}{p_1 \parallel p_2 : \tilde{n} \mathcal{R}^\blacksquare P}$
$\frac{P' \mathcal{R}^\blacksquare P'' \quad p' : \tilde{n} \mathcal{R}^\blacksquare p'' : \tilde{n} \quad \varphi \langle P' \rangle p'' : \tilde{n} \mathcal{R}^\circ P}{\varphi \langle P' \rangle p' : \tilde{n} \mathcal{R}^\blacksquare P}$		$\frac{p : \tilde{n} \mathcal{R}^\blacksquare p' : \tilde{n} \quad \varphi(x) p' : \tilde{n} \mathcal{R}^\circ P}{\varphi(x) p : \tilde{n} \mathcal{R}^\blacksquare P}$
$\frac{p : \tilde{n} n \mathcal{R}^\blacksquare p' : \tilde{n} n \quad (n) p' : \tilde{n} \mathcal{R}^\circ P}{(n) p : \tilde{n} \mathcal{R}^\blacksquare P}$		$\frac{P \mathcal{R}^\blacksquare P' \quad p : \tilde{n} \mathcal{R}^\blacksquare p' : \tilde{n} \quad \langle P' \rangle p' : \tilde{n} \mathcal{R}^\square C}{\langle P \rangle p : \tilde{n} \mathcal{R}^\blacksquare C}$
$\frac{c : \tilde{n} n \mathcal{R}^\blacksquare c' : \tilde{n} n \quad (n) c' : \tilde{n} \mathcal{R}^\square C}{(n) c : \tilde{n} \mathcal{R}^\blacksquare C}$		

**Table 7.** Howe relation for typed processes and concretions.

By defining testing contexts for the different kinds of labels, one can prove that early contextual bisimulation, in the strong case, is in fact complete with respect to barbed bisimulation congruence restricted to closed terms, as stated below.

**Proposition 7.**  $(\sim_b)_c \subseteq \sim_e$  .

The result extends to open processes because  $\sim_b$  is substitutive

**Corollary 1.**  $\sim_b \subseteq \sim_e^\circ$  .

We conjecture that early delay contextual bisimulation is not complete with respect to weak barbed bisimulation. We will comment on this discrepancy in the final section.

We now show the main technical result of the paper, that  $\sim_e^\circ$  and  $\approx_e^\circ$  are indeed also congruences, by adapting Howe's method to active mobile processes in nested locations. This is in fact a combination of two new results. It is well-known that late bisimulations usually are not complete with respect to barbed bisimulation congruence for higher-order process calculi. However, it has also proven difficult to extend Howe's method to early bisimulations. The first new result is that Howe's method can indeed be extended to *input-early* strong and delay bisimulations, by extending the relation to also cover concretions. The second result is to extend Howe's method to active process mobility in nested locations.

First we define the *Howe-relation*  $\mathcal{R}^\blacksquare$  on  $\mathbf{P}_{/\alpha} \cup \mathbf{C}_{/\alpha}$ , defined relative to a binary relation  $\mathcal{R}$  on  $\mathbf{P}_{c/\alpha}$  and as the least relation satisfying the rules in Table 7. Let  $\mathcal{R}^\bullet = \mathcal{R}^\blacksquare \cap \mathbf{P}_{/\alpha} \times \mathbf{P}_{/\alpha}$ , i.e. the relation restricted to (possibly open) processes. It is easy to prove that if  $\mathcal{R}$  is well typed then  $\mathcal{R}^\bullet$  and  $\mathcal{R}^\blacksquare$  are also well typed. The novel contribution is the extension of the Howe relation to concretions. As in [18] we first prove the following properties.

**Proposition 8.** Let  $\mathcal{R}$  be an equivalence relation on  $\mathbf{P}_{c/\alpha}$  then

1.  $\mathcal{R}^\blacksquare$  is reflexive.
2.  $\mathcal{R}^\square \subseteq \mathcal{R}^\blacksquare$ .
3.  $\mathcal{R}^\blacksquare \mathcal{R}^\square \subseteq \mathcal{R}^\blacksquare$ .
4.  $\mathcal{R}^\bullet$  is substitutive.
5.  $\mathcal{R}^\bullet$  is constructor compatible
6.  $\mathcal{R}^{\bullet-1} \subseteq \mathcal{R}^{\bullet*}$
7.  $\mathcal{R}^{\bullet*}$  is symmetric

The key is now to show, since  $\sim_e^\bullet$  ( $\approx_e^\bullet$ ) is a congruence, that  $\sim_e^\circ = \sim_e^\bullet$  ( $\approx_e^\circ = \approx_e^\bullet$ ).

Next, as a novel contribution of this paper we extend the Howe-relation to path contexts. Let  $\mathcal{R}$  be a binary relation on  $\mathbf{P}_{c/\alpha}$ . We define the Howe-relation  $\mathcal{R}^\blacktriangle$  on path context by  $D_{\tilde{n},\gamma} : \tilde{n}' \mathcal{R}^\blacktriangle D'_{\tilde{n},\gamma} : \tilde{n}'$  if there exists  $D_{\tilde{n},\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}$  and  $p_i : \tilde{n}_i \mathcal{R}^\bullet p'_i : \tilde{n}_i$ ,  $i = 1, \dots, k$  such that  $D_{\tilde{n},\gamma} : \tilde{n}' = D_{\tilde{n},\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}(p_1, \dots, p_k) : \tilde{n}'$  and  $D'_{\tilde{n},\gamma} : \tilde{n}' = D_{\tilde{n},\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}(p'_1, \dots, p'_k) : \tilde{n}'$ .

Observe that two related path contexts have the same path to their hole which are indexed by the same type, also they restrict their holes by the same names.

In order to establish  $\approx_e^\circ = \approx_e^\bullet$  the method of Howe utilizes that  $\approx_e^\bullet$  satisfies the following bisimulation property.

**Lemma 1.**  $p : \tilde{n}, P \in \mathbf{P}_{c/\alpha}$  and  $p : \tilde{n} \approx_e^\bullet P$  implies that

1. if  $p : \tilde{n} \xrightarrow{\tau} P_1$  then  $\exists P_2. P \xrightarrow{\tau} P_2$  and  $P_1 \equiv \approx_e^\bullet P_2$
2. if  $p : \tilde{n} \xrightarrow{\bar{\delta}} A$  then  $\forall C \approx_e^\bullet C'. \exists A'. P \xrightarrow{\bar{\delta}} A'$  and  $A \cdot C \equiv \approx_e^\bullet A' \cdot C'$
3. if  $p : \tilde{n} \xrightarrow{\delta} A$  then  $\forall C \approx_e^\bullet C'. \forall \mathcal{D}_{\tilde{n}} \approx_e^\bullet \mathcal{D}'_{\tilde{n}}. \exists A'. P \xrightarrow{\delta} A'$  and  $\mathcal{D}_{\tilde{n}}(A) \cdot C \equiv \approx_e^\bullet \mathcal{D}'_{\tilde{n}}(A') \cdot C'$
4. if  $p : \tilde{n} \xrightarrow{\bar{\delta}} C$  then  $\exists C'. P \xrightarrow{\bar{\delta}} C'$  and  $C \approx_e^\bullet C'$
5. if  $p : \tilde{n} \xrightarrow{\delta} C$  then  $\exists C'. P \xrightarrow{\delta} C'$  and  $\forall \mathcal{D}_{\tilde{n}} \approx_e^\bullet \mathcal{D}'_{\tilde{n}}. \mathcal{D}_{\tilde{n}}(C) \approx_e^\bullet \mathcal{D}'_{\tilde{n}}(C')$

A similar bisimulation property for  $\sim_e^\bullet$  is used to prove  $\sim_e^\circ = \sim_e^\bullet$ .

**Proposition 9.**  $\sim_e^\circ$  and  $\approx_e^\circ$  are congruences.

From Prop. 2 and 6 and from Prop. 9 it follows that  $\sim_e^\circ$  and  $\approx_e^\circ$  are sound with respect to barbed and weak barbed bisimulation congruence, respectively.

**Proposition 10.**  $\sim_e^\circ \subseteq \sim_b$  and  $\approx_e^\circ \subseteq \approx_b$ .

From Corol. 1 and Prop. 9 we conclude that

**Theorem 1.**  $\sim_e^\circ = \sim_b$

In [17] we study late delay contextual bisimulation for (an equivalent) variant of Homer. For standard reasons (see e.g.[19]) late delay contextual bisimulation is strictly contained in weak barbed congruence, hence from the theorem above it follows that  $\sim_l \subsetneq \sim_e$ . It turns out that the late delay context bisimulation is in fact *strictly* contained in the early delay context bisimulations.

## 6 Conclusions and Related Work

We presented the calculus Homer, which combines higher-order process passing, local names, nested locations and copyable, active mobile computing resources. We gave reaction and contextual labelled transition semantics as conservative extensions of the standard semantics of a pure process-passing calculi with local names, and presented the first generalisation of Howe's method to a calculus with *active* mobile processes

at nested locations and local names, proving that *early* delay (and strong) contextual bisimulation is a congruence and a sound (and complete) characterisation of weak (strong) barbed bisimulation congruence. We also found that the late bisimulations are *strictly* included in the early bisimulations.

Thomsen [9] and Prasad, Giacalone and Mishra [23] provides so-called higher-order labelled transition bisimulations for respectively Plain CHOCS and Facile, which are early bisimulations where processes in higher-order labels are required to be bisimilar. Sangiorgi shows in [24] that ho-bisimulation is too discriminating and presents an elegant, and now classical, solution to the problem using triggered processes and normal bisimulation for the higher-order  $\pi$ -calculus. Jeffrey and Rathke extend in [25] the result to recursive types. Sound and complete characterisations of barbed bisimulation congruence exist for variants of the Ambient calculus [19, 26–28]. However, none of the variants of the Ambient calculus or  $\text{HO}\pi$  (or the first-order  $\pi$ -calculus) allow for *copyable* active mobile processes.

A series of papers have developed and modified the semantics of the Seal calculus [2, 22, 20]. A sound characterisation of reduction barbed bisimulation congruence for the Seal calculus is provided in [20]. Still, the authors leave important problems open, as e.g. providing a sound and complete characterisation of barbed bisimulation congruence and a deeper understanding of the interrelation between local names and copyable mobile processes. We have contributed to both of these problems by providing a sound and complete characterisation of strong barbed bisimulation congruence and identified the need for typing mobile sub resources if one uses the standard vertical scope extension used in  $\text{HO}\pi$ .

The M-calculus and the Kell calculus of Schmitt and Stefani share with Homer in being based on process passing and process substitution. A sound and complete characterisation of barbed bisimulation congruence has been announced in [4], however, a detailed proof has not been published yet.

So far, all congruence proofs for the above mentioned calculi have been carried out roughly by closing the bisimulation under contexts and prove it to be a bisimulation by a "brute force" case analysis. Hence, none of the proofs adapts Howe's method to active mobile processes at nested locations as carried out in the present paper. Howe's method was originally introduced for lazy functional programming languages. It has later been applied to the Abadi-Cardelli object calculi by Gordon [29], higher-order functions with concurrent communication (a fragment of Reppy's Concurrent ML) by Jeffrey [30], late bisimulations for local names with static scope by Baldamus and Frauenstein in [18].

A problem left open is to give a sound and complete co-inductive characterisation of an appropriate weak barbed congruence. A delay bisimulation seems to be needed to be able to apply Howe's method. However, a standard weak barbed bisimulation is unlikely to be included in a delay bisimulation in the higher-order setting. We are currently investigating a notion of delay barbed bisimulation congruence corresponding to the labelled delay bisimulation. We will also investigate if the trigger, normal and up-to-context bisimulation techniques can be adapted to calculi with non-linear active process mobility and explicit, nested locations. It could provide us with a bisimulation without universal quantification over contexts. Finally, we plan to study more complex

type systems for Homer. This has been initiated in [31], presenting a type system for linear and non-linear mobile resources for an early version of the Homer calculus.

*Acknowledgements* This paper has improved considerably from helpful suggestions from anonymous referees of previous versions.

## References

1. Sangiorgi, D.: A theory of bisimulation for the  $\pi$ -calculus. *Acta Informatica* **33** (1996) 69–97 Earlier version published as Report ECS-LFCS-93-270, University of Edinburgh. An extended abstract appeared in the *Proceedings of CONCUR '93*, LNCS 715.
2. Vitek, J., Castagna, G.: Seal: A framework for secure mobile computations. In: *Internet Programming Languages*. (1999)
3. Schmitt, A., Stefani, J.B.: The M-calculus: A higher-order distributed process calculus. In: *Proceedings of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL'03)*. (2003) 50–61
4. Schmitt, A., Stefani, J.B.: The Kell calculus: A family of higher-order distributed process calculi. In: *Proceedings of the International Workshop on Global Computing 2004 Workshop (GC 2004)*. *Lecture Notes in Computer Science*, Springer Verlag (2004)
5. Howe, D.J.: Proving congruence of bisimulation in functional programming languages. *Information and Computation* **124** (1996) 103–112
6. Howe, D.J.: Equality in lazy computation systems. In: *Proceedings of the 4th IEEE Symposium on Logic in Computer Science*. (1989) 198–203
7. Gordon, A.: *Functional Programming and Input/Output*. *Distinguished dissertations in computer science* (1994)
8. Gordon, A.D.: Bisimilarity as a theory of functional programming. *Theoretical Computer Science* **228** (1999) 5–47
9. Thomsen, B.: Plain CHOCS. A second generation calculus for higher order processes. *Acta Informatica* **30** (1993) 1–59
10. Sangiorgi, D.: *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, LFCS, University of Edinburgh (1993) CST-99-93 (also published as ECS-LFCS-93-266).
11. Boudol, G.: Towards a lambda-calculus for concurrent and communicating systems. In Díaz, J., Orejas, F., eds.: *Proceedings of Theory and Practice of Software Development (TAPSOFT '89)*. Volume 351 of *Lecture Notes in Computer Science*., Springer Verlag (1989) 149–161
12. Godskesen, J.C., Hildebrandt, T., Sassone, V.: A calculus of mobile resources. In: *Proceedings of CONCUR'2002*. LNCS, Springer (2002)
13. Cardelli, L., Gordon, A.D.: Mobile ambients. In Nivat, M., ed.: *Proceedings of FoSSaCS '98*. Volume 1378 of LNCS., Springer (1998) 140–155
14. Jeffrey, A., Rathke, J.: Towards a theory of bisimulation for local names. In: *Proceedings of LICS '99*, IEEE, Computer Society Press (1999) 56–66
15. Bundgaard, M., Hildebrandt, T., Godskesen, J.C.: A CPS encoding of name-passing in higher-order mobile embedded resources. In: *Proceedings of the 11th International Workshop on Expressiveness in Concurrency (EXPRESS'04)*. *Electronic Notes in Theoretical Computer Science*, Elsevier (2004) To appear.
16. Bundgaard, M., Hildebrandt, T.: A bigraphical semantics of higher order mobile embedded resources with local names (2005) submitted for publication.
17. Hildebrandt, T., Godskesen, J.C., Bundgaard, M.: Bisimulation congruences for homer - a calculus of higher order mobile embedded resources. Technical Report ITU TR-2004-52, IT University of Copenhagen, Department of Theoretical Computer Science (2004)

18. Baldamus, M., Frauenstein, T.: Congruence proofs for weak bisimulation equivalences on higher-order process calculi. Technical Report Report 95–21, Berlin University of Technology, Computer Science Department (1995)
19. Merro, M., Hennessy, M.: Bisimulation congruences in safe ambients. Computer Science Report 2001:05, University of Sussex (2001)
20. Castagna, G., Vitek, J., Nardelli, F.Z.: The Seal calculus. accepted for publication in *Information and Computation* (2004)
21. Sangiorgi, D., Walker, D.: The  $\pi$ -calculus: a Theory of Mobile Processes. Cambridge University Press (2001)
22. Castagna, G., Nardelli, F.Z.: The Seal calculus revisited: Contextual equivalence and bisimilarity. In Agrawal, M., Seth, A., eds.: Proceedings of the 22nd Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS'02). Volume 2556 of Lecture Notes in Computer Science., Springer Verlag (2002) 85–96
23. Prasad, S., Giacalone, A., Mishra, P.: Operational and algebraic semantics for Facile: A symmetric integration of concurrent and functional programming. In Paterson, M., ed.: Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90). Volume 443 of Lecture Notes in Computer Science., Springer Verlag (1990) 765–778
24. Sangiorgi, D.: Bisimulation in higher-order process calculi. *Journal of Information and Computation* **131** (1996) 141–178 Available as Rapport de Recherche RR-2508, INRIA Sophia-Antipolis, 1995. An early version appeared in *Proceedings of PROCOMET'94*, pages 207–224. IFIP. North Holland Publisher.
25. Jeffrey, A., Rathke, J.: Contextual equivalence for higher-order  $\pi$ -calculus revisited. In Brookes, S., Panangaden, P., eds.: Proceedings of the 19th Conference on Mathematical Foundations of Programming Semantics (MFPS'04). Volume 83 of Electronic Notes in Theoretical Computer Science., Elsevier (2004)
26. Merro, M., Nardelli, F.Z.: Bisimulation proof techniques for mobile ambients. In: Proceedings of ICALP 2003. (2003) to appear.
27. Merro, M., Nardelli, F.Z.: Behavioural theory for mobile ambients. In: Proceedings of the 3rd International Conference on Theoretical Computer Science (IFIP TCS 2004). (2004) to appear.
28. Bugliesi, M., Crafa, S., Merro, M., Sassone, V.: Communication and mobility control in boxed ambients. *Journal of Information and Computation* (200X)
29. A.D.Gordon: Operational equivalences for untyped and polymorphic object calculi. In: Higher Order Operational Techniques in Semantics, Cambridge University Press (1998)
30. Jeffrey, A.: Semantics for core concurrent ml using computation types. In: Higher Order Operational Techniques in Semantics, Cambridge University Press (1998)
31. Godskesen, J.C., Hildebrandt, T.: Copyability types for mobile computing resources (2004) International Workshop on Formal Methods and Security, Nanjing.

## A Proof of Proposition 6

Proposition 6 follows from:

**Lemma 2.** *If  $P \xrightarrow{\gamma\phi} T$  then there exists  $D_{\tilde{n},\gamma}$ ,  $\tilde{n}$ ,  $e$ , and  $p$  with  $\tilde{n}' \cap \gamma\phi = \emptyset$  such that*

$$P \equiv (\tilde{n}')(D_{\tilde{n},\gamma}(\phi p) \parallel p') : \sigma' \quad \text{and} \quad T \equiv (\tilde{n}')(D_{\sigma,\gamma}(p) \parallel p') : \sigma'$$

## B Proof of Proposition 7

We prove that input-early, strong bisimulation is complete with respect to barbed bisimulation as usual by defining testing contexts for the different kinds of labels. Special care must be taken due to the explicit typing of sub resources. We will exploit the use of nested names.

*Proof.* Let  $p : \tilde{n} \sim_b q : \tilde{n}$  for closed processes  $p$  and  $q$ . We proceed to show the properties of Def. 2 (upto structural congruence).

**Case:**  $p : \tilde{n} \xrightarrow{\tau} P_1$ . Follows from Prop. 6.

**Case:**  $p : \tilde{n} \xrightarrow{\delta} A = (x)p' : \tilde{n}$ . Assume  $C = (\tilde{m})\langle r : \tilde{m}' \rangle r' : \tilde{m}''$  and  $D_{\tilde{n},\gamma} = D_{\tilde{n},\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}(p_1, \dots, p_k)$ . Let  $j$  be the sum of the path lengths of all addresses of resources in  $C$  and  $D_{\tilde{n},\gamma}$ . Let  $\delta^j$  denote the concatenation of  $j$  copies of  $\delta$ . For a fresh name  $m$ , define a context  $C = D_{\tilde{n},\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}(\delta^j(x)\delta\langle x : \emptyset \rangle \delta^j(\mathbf{0})p_1, \dots, p_k) \parallel (\tilde{m})\gamma\delta^j\langle r : \tilde{m}' \rangle \gamma\delta^j(x)(r' \parallel m(\mathbf{0}))$ .

Then  $C(p : \tilde{n}) \searrow \searrow \searrow p'' \parallel m(\mathbf{0}) : \tilde{n}''m$ , where  $p'' : \tilde{n}'' = D_{\tilde{n},\gamma}(A) \cdot C$ . From  $p : \tilde{n} \sim_b q : \tilde{n}$  it follows that  $C(q : \tilde{n}) \searrow \searrow \searrow q'' \parallel m(\mathbf{0}) : \tilde{n}''m$ , where  $q'' : \tilde{n}'' = D_{\tilde{n},\gamma}(A') \cdot C$  and  $q : \tilde{n} \xrightarrow{\delta} A' = (x)q' : \tilde{n}$  and  $p'' \parallel m(\mathbf{0}) : \tilde{n}''m \sim_b q'' \parallel m(\mathbf{0}) : \tilde{n}''m$ . Using the context  $C_m = (-) \parallel m(x)$  it follows that  $p'' : \tilde{n}''m \sim_b q'' : \tilde{n}''m$ . Finally, using structural congruence and the context  $C_{\setminus m} = (m)(-)$  it follows that  $D_{\tilde{n},\gamma}(A) \cdot C \equiv_{\sim_b} \equiv D_{\tilde{n},\gamma}(A') \cdot C$  as desired.

**Case:**  $p : \tilde{n} \xrightarrow{\delta} A$ . As above, but simpler.

**Case:**  $p : \tilde{n} \xrightarrow{\delta} C = (\tilde{m})\langle p'' : \tilde{m}' \rangle p' : \tilde{n}$ . For a fresh name  $m$ , define a context  $C = (-) \parallel \delta(x)(\delta\langle x : \emptyset \rangle \parallel \tilde{m}(\mathbf{0}))$ . Then  $C(p : \tilde{n}) \searrow (\tilde{m})\langle p'' \parallel \delta\langle p'' : \tilde{m}' \rangle \parallel \tilde{m}(\mathbf{0}) \rangle$ . From  $p : \tilde{n} \sim_b q : \tilde{n}$  it follows that  $C(q : \tilde{n}) \searrow (\tilde{m}'')\langle q' \parallel \delta\langle q'' : \tilde{m}''' \rangle \parallel \tilde{m}(\mathbf{0}) \rangle$  and  $q : \tilde{n} \xrightarrow{\delta} C' = (\tilde{m}'')\langle q' : \tilde{m}''' \rangle q' : \tilde{n}$ . Again using the contexts  $C_m$  and  $C_{\setminus m}$  it follows that  $P = (\tilde{m})\langle p'' \parallel \delta\langle p'' : \tilde{m}' \rangle \rangle \equiv_{\sim_b} \equiv Q = (\tilde{m}'')\langle q' \parallel \delta\langle q'' : \tilde{m}''' \rangle \rangle$

For any  $D_{\tilde{n},\gamma} = D_{\tilde{n},\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}(p_1, \dots, p_k)$  and  $A = (x)r$  let  $j$  be the sum of the path lengths of all addresses of resources in  $C, D_{\tilde{n},\gamma}$  and  $A$ . For a fresh name  $m$ , define a context  $C = D_{\tilde{n},\gamma}^{\tilde{n}_1, \dots, \tilde{n}_k}(\delta^j(x)(x \parallel \delta^{2j}(\mathbf{0})p_1, \dots, p_k)(\delta^j\langle (-) \rangle) \parallel \gamma\delta^{j+1}(x)\delta^{2j}(z)(r \parallel \tilde{m}(\mathbf{0})))$ . Then  $C(P) \searrow \searrow \searrow D_{\tilde{n},\gamma}(C) \cdot A \parallel \tilde{m}(\mathbf{0})$ . It follows that  $C(Q) \searrow \searrow \searrow D_{\tilde{n},\gamma}(C') \cdot A \parallel \tilde{m}(\mathbf{0})$ . Again using the contexts  $C_m$  and  $C_{\setminus m}$  it follows that  $D_{\tilde{n},\gamma}(C) \cdot A \equiv_{\sim_b} \equiv D_{\tilde{n},\gamma}(C') \cdot A$  as desired.

**Case:**  $p : \tilde{n} \xrightarrow{\delta} C$ . As above, but simpler.

## C Proof of Lemma 1

**Lemma 3.**  $\equiv_c$  is a strong late context bisimulation.

**Lemma 4.** Let  $\mathcal{R}$  be an equivalence relation on  $\mathbf{P}_{c/\alpha}$ , then  $C \mathcal{R}^\blacksquare C'$  and  $P \mathcal{R}^\bullet P'$  implies  $(x)P \cdot C \mathcal{R}^\bullet (x)P' \cdot C'$ .

*Proof.* The proof is by induction in the derivation of  $C \mathcal{R}^\blacksquare C'$  using the definition of  $\mathcal{R}^\square$  and Proposition 8.3-5.

**Lemma 5.** Let  $\mathcal{R}$  be an equivalence relation on  $\mathbf{P}_{c/\alpha}$ , then  $\mathcal{D}_{\tilde{n}} \mathcal{R}^\blacktriangle \mathcal{D}'_{\tilde{n}}$  and  $p : \tilde{n} \mathcal{R}^\bullet p' : \tilde{n}$  implies  $\mathcal{D}_{\tilde{n}}(p) \mathcal{R}^\bullet \mathcal{D}'_{\tilde{n}}(p')$ .

*Proof.* Follows from the definition of  $\mathcal{R}^\blacktriangle$  because  $\mathcal{R}^\bullet$  is constructor compatible due to Proposition 8.7.

**Lemma 6.** If  $\mathcal{R}$  is an equivalence relation on  $\mathbf{P}_{c/\alpha}$ , then  $C \mathcal{R}^\blacksquare C'$ ,  $p : \tilde{n} \mathcal{R}^\bullet p' : \tilde{n}$ , and  $\mathcal{D}_{\tilde{n}} \mathcal{R}^\blacktriangle \mathcal{D}'_{\tilde{n}}$  implies  $\mathcal{D}_{\tilde{n}}((x)p) \cdot C \mathcal{R}^\bullet \mathcal{D}'_{\tilde{n}}((x)p') \cdot C'$ .

*Proof.* Follows from Lemma 4 and 5.

In the sequel we close typed path contexts and concretions under structural congruence in the obvious way. Notice in particular that  $C \equiv C'$  and  $\mathcal{D}_{\tilde{n},\gamma} \equiv \mathcal{D}'_{\tilde{n},\gamma}$  implies  $\mathcal{D}_{\tilde{n},\gamma}(C) \equiv \mathcal{D}'_{\tilde{n},\gamma}(C')$ .

**Lemma 7.** If  $\mathcal{R}$  is a binary relation on  $\mathbf{P}_{c/\alpha}$  and  $\tilde{n}' \subseteq \tilde{n}$  then  $\mathcal{D}_{\tilde{n}} \mathcal{R}^\blacktriangle \mathcal{D}'_{\tilde{n}}$  and  $p : \tilde{n} \mathcal{R}^\bullet p' : \tilde{n}$  implies

$$\begin{aligned} \mathcal{D}_{\tilde{n}}(\delta\langle(-)_{\tilde{n}'} : \tilde{n}'\rangle p) &\equiv \mathcal{R}^\blacktriangle \equiv \mathcal{D}'_{\tilde{n}}(\delta\langle(-)_{\tilde{n}'} : \tilde{n}'\rangle p'), \text{ if } \delta \subseteq \tilde{n} \\ \mathcal{D}_{\tilde{n}}((-)_{\tilde{n}'} \parallel p) &\equiv \mathcal{R}^\blacktriangle \equiv \mathcal{D}'_{\tilde{n}}((-)_{\tilde{n}'} \parallel p') \text{ if } \mathcal{D}_{\tilde{n}} \neq (-)_{\tilde{n}} : \tilde{n}'' \\ \mathcal{D}_{\tilde{n}}((n)(-)_{\tilde{n}'n}) &\equiv \mathcal{R}^\blacktriangle \equiv \mathcal{D}'_{\tilde{n}}((n)(-)_{\tilde{n}'n}), \text{ if } n \notin \tilde{n} \text{ and } \mathcal{D}_{\tilde{n}} \neq (-)_{\tilde{n}} : \tilde{n}'' \end{aligned}$$

*Proof.* Follows from the definition of  $\mathcal{R}^\blacktriangle$ .

In the proof of Lemma 1 we make use of the following definition where restrictions guarding the hole in a typed path context are removed.

Define  $\mathcal{D}_{\tilde{n},\gamma} : \tilde{n}' \setminus m = \mathcal{D}_{\tilde{n},\gamma} \setminus m : \tilde{n}'$  where  $\mathcal{D}_{\tilde{n},\gamma} \setminus m$  is defined inductively by

$$\mathcal{D}_{\tilde{n},\gamma} \setminus m = \begin{cases} (-)_{\tilde{n}}, & \text{if } \mathcal{D}_{\tilde{n},\gamma} = (-)_{\tilde{n}} \\ \delta\langle(\tilde{m} \setminus m)(\mathcal{D}_{\tilde{n},\gamma'} \setminus m \parallel p') : \tilde{n}'\rangle p, & \text{if } \mathcal{D}_{\tilde{n},\gamma} = \delta\langle(\tilde{m})(\mathcal{D}_{\tilde{n},\gamma'} \parallel p') : \tilde{n}'\rangle p \end{cases}$$

We write  $\mathcal{D}_{\tilde{n}} \setminus \tilde{m}$  for  $\mathcal{D}_{\tilde{n}} \setminus m_1 \dots \setminus m_k$  whenever  $\tilde{m} = \{m_1, \dots, m_k\}$ .

**Lemma 8.**  $\mathcal{D}_{\tilde{n}} \mathcal{R}^\blacktriangle \mathcal{D}'_{\tilde{n}}$  implies  $\mathcal{D}_{\tilde{n}} \setminus \tilde{m} \mathcal{R}^\blacktriangle \mathcal{D}'_{\tilde{n}} \setminus \tilde{m}$ .

*Proof.* The proof follows immediately from the definition of  $\mathcal{R}^\blacktriangle$ .

We define the set of names,  $bn(\mathcal{D}_{\sigma,\gamma})$ , binding the hole of a path context,  $\mathcal{D}_{\sigma,\gamma}$ , inductively by  $bn(\mathcal{D}_{\sigma,\gamma} : \sigma') = bn(\mathcal{D}_{\sigma,\gamma})$ ,  $bn((-)_{\sigma}) = \emptyset$ , and  $bn(\delta\langle(\tilde{n})(\mathcal{D}_{\sigma,\gamma'} \parallel p') : \sigma'\rangle p) = \tilde{n} \cup bn(\mathcal{D}_{\sigma,\gamma'})$ .

### C.1 Proof of Lemma 1

The proof of Lemma 1 follows closely the one in [18], we provide only a few key cases.

*Proof.* Suppose  $P, P' \in \mathbf{P}_{c/\alpha}$ ,  $P \approx_e^\bullet P'$  and  $P \xrightarrow{\pi} T$ . We proceed by induction on the derivation of  $P \xrightarrow{\pi} T$ .

**Case (prefix):**  $P = \varphi e : \sigma$  and  $T = e : \sigma$ .

**Case** Suppose  $e = (x)p$ . Then since  $P \approx_e^\bullet P'$  there exists  $p'$  such that  $p : \delta n \approx_e^\bullet p' : \delta n$  and  $\varphi(x)p' : \delta n \approx_e^\circ P'$ . We may assume  $p'$  is closed because  $\approx_e^\bullet$  is substitutive (Proposition 8.4), hence  $\varphi(x)p' : \delta n \xrightarrow{\varphi} (x)p' : \delta n$ .

If  $\varphi = \delta$  then, since  $\varphi(x)p' : \delta n \approx_e P'$ , for all closed  $C'$  and  $\mathcal{D}'_{\tilde{n}}$  there exists  $P' \xrightarrow{\delta} A'$  such that  $\mathcal{D}'_{\tilde{n}}((x)p' : \tilde{n}) \cdot C' \approx_e^\circ \mathcal{D}'_{\tilde{n}}(A') \cdot C'$ .

Let  $C, C' \in \mathbf{C}_{c/\alpha}$  such that  $C \approx_e^\blacksquare C'$  and let  $\mathcal{D}_{\tilde{n}} \approx_e^\blacktriangle \mathcal{D}'_{\tilde{n}}$ . Because  $p : \tilde{n} \approx_e^\bullet p' : \tilde{n}$  we infer from Lemma 6 that

$$\mathcal{D}_{\tilde{n}}((x)p : \tilde{n}) \cdot C \approx_e^\bullet \mathcal{D}'_{\tilde{n}}((x)p' : \tilde{n}) \cdot C'$$

Hence, since  $\approx_e^\bullet \approx_e^\circ \subseteq \approx_e^\bullet$  (Proposition 8.3 restricted to processes) we get

$$\mathcal{D}_{\tilde{n}}((x)p : \tilde{n}) \cdot C \approx_e^\bullet \mathcal{D}'_{\tilde{n}}(A') \cdot C'$$

The case where  $\varphi = \bar{\delta}$  is similar.

**Case:** Suppose  $e = \langle p_1 : \tilde{n}' \rangle p_2$ . Then since  $P \approx_e^\bullet P'$  there exists  $p'_1$  and  $p'_2$  such that  $p_1 : \tilde{n}' \approx_e^\bullet p'_1 : \tilde{n}'$ ,  $p_2 : \tilde{n} \approx_e^\bullet p'_2 : \tilde{n}$ , and  $\varphi \langle p'_1 : \tilde{n}' \rangle p'_2 : \tilde{n} \approx_e^\circ P'$ . We may assume  $p'_1$  and  $p'_2$  are closed because  $\approx_e^\bullet$  is substitutive (Proposition 8.4), hence  $\varphi \langle p'_1 : \tilde{n}' \rangle p'_2 : \tilde{n} \xrightarrow{\varphi} \langle p'_1 : \tilde{n}' \rangle p'_2 : \tilde{n}$ .

Let  $\varphi = \delta$ , then because  $\varphi \langle p'_1 : \tilde{n}' \rangle p'_2 : \tilde{n} \approx_e^\circ P'$ , there exists  $P' \xrightarrow{\delta} C'$  such that for all  $\mathcal{D}'_{\tilde{n}}$ :

$$\mathcal{D}'_{\tilde{n}}(\langle p'_1 : \tilde{n}' \rangle p'_2 : \tilde{n}) \approx_e^\square \mathcal{D}'_{\tilde{n}}(C')$$

Next, let  $\mathcal{D}_{\tilde{n}} \approx_e^\blacktriangle \mathcal{D}'_{\tilde{n}}$ . Then  $bn(\mathcal{D}_{\tilde{n}}) = bn(\mathcal{D}'_{\tilde{n}})$ . Due to the notational convenience, observe that

$$\mathcal{D}_{\tilde{n}}(\langle p_1 : \tilde{n}' \rangle p_2 : \tilde{n}) = (\tilde{m}) \langle p_1 : \tilde{n}' \rangle (\mathcal{D}_{\tilde{n}} \setminus \tilde{m})(p_2 : \tilde{n})$$

and

$$\mathcal{D}'_{\tilde{n}}(\langle p'_1 : \tilde{n}' \rangle p'_2 : \tilde{n}) = (\tilde{m}) \langle p'_1 : \tilde{n}' \rangle (\mathcal{D}'_{\tilde{n}} \setminus \tilde{m})(p'_2 : \tilde{n})$$

where  $\tilde{m} = bn(\mathcal{D}_{\tilde{n}}) \cap \tilde{n}'$ . Then, since  $(\mathcal{D}_{\tilde{n}} \setminus \tilde{m}) \approx_e^\blacktriangle (\mathcal{D}'_{\tilde{n}} \setminus \tilde{m})$  due to Lemma 8 we get from Lemma 5 that  $(\mathcal{D}_{\tilde{n}} \setminus \tilde{m})(p_2 : \tilde{n}) \approx_e^\bullet (\mathcal{D}'_{\tilde{n}} \setminus \tilde{m})(p'_2 : \tilde{n})$ . Hence, because  $\approx_e^\square$  is reflexive we obtain

$$(\tilde{m}) \langle p_1 : \tilde{n}' \rangle (\mathcal{D}_{\tilde{n}} \setminus \tilde{m})(p_2) \approx_e^\blacksquare (\tilde{m}) \langle p'_1 : \tilde{n}' \rangle (\mathcal{D}'_{\tilde{n}} \setminus \tilde{m})(p'_2)$$

Finally, since  $\approx_e^\blacksquare \approx_e^\square \subseteq \approx_e^\blacksquare$  (Proposition 8.3) we have  $\mathcal{D}_{\tilde{n}}(\langle p_1 : \tilde{n}' \rangle p_2 : \tilde{n}) \approx_e^\blacksquare \mathcal{D}'_{\tilde{n}}(C')$ .

The case where  $\varphi = \bar{\delta}$  is similar.

**Case (sync):** Assume  $P \xrightarrow{\tau} T$  because  $P = p_1 \parallel p_2 : \tilde{n}$ ,  $p_1 : \tilde{n} \xrightarrow{\varphi} A$ ,  $p_2 : \tilde{n} \xrightarrow{\bar{\varphi}} C$ , and  $T = A \cdot C$ .

Then from the definition of  $\approx_e^\bullet$  there exists  $p'_1$  and  $p'_2$  such that  $p_1 : \tilde{n} \approx_e^\bullet p'_1 : \tilde{n}$ ,  $p_2 : \tilde{n} \approx_e^\bullet p'_2 : \tilde{n}$ , and  $p'_1 \parallel p'_2 : \tilde{n} \approx_e^\circ P'$ .

We may assume  $p'_1$  and  $p'_2$  are closed because  $\approx_e^\bullet$  is substitutive (Proposition 8.4). Hence by induction hypothesis, since  $p_2 : \tilde{n} \approx_e^\bullet p'_2 : \tilde{n}$ , there exists  $p'_2 : \tilde{n} \xrightarrow{\bar{\varphi}} C'$  such that  $C \approx_e^\blacksquare C'$ . Also by induction hypothesis, since  $p_1 : \tilde{n} \approx_e^\bullet p'_1 : \tilde{n}$ , there exists  $p'_1 : \tilde{n} \xrightarrow{\varphi} A'$  such that  $A \cdot C \equiv \approx_e^\bullet A' \cdot C'$ .

Therefore,  $p'_1 \parallel p'_2 : \tilde{n} \xrightarrow{\tau} A' \cdot C'$ . Then, since  $p'_1 \parallel p'_2 : \tilde{n} \approx_e P'$  there exists  $P' \xrightarrow{\tau} P''$  such that  $A' \cdot C' \approx_e P''$  and therefore  $A \cdot C \equiv \approx_e^\bullet P''$  because  $\approx_e^\bullet \approx_e^\circ \subseteq \approx_e^\bullet$  (Proposition 8.3 restricted to processes).

**Case (nesting):** Suppose  $P \xrightarrow{\delta \cdot \pi} T$  because  $P = \delta \langle P_1 \rangle p_1 : \tilde{n}$  and  $P_1 \xrightarrow{\pi} f : \tilde{n}'$ , it follows that  $T = \delta \langle f : \tilde{n}' \rangle p_1 : \tilde{n}$ .

Then it follows from the definition of  $\approx_e^\bullet$  there exists  $P'_1$  and  $p'_1$  such that  $P_1 \approx_e^\bullet P'_1$ ,  $p_1 : \tilde{n} \approx_e^\bullet p'_1 : \tilde{n}$ , and  $\delta \langle P'_1 \rangle p'_1 \approx_e^\circ P'$ . We may assume  $P'_1$  and  $p'_1$  are closed because  $\approx_e^\bullet$  is substitutive (Proposition 8.4).

**case** Suppose  $\pi = \delta'$  and  $f = a$ . Let  $C, C' \in \mathbf{C}_{c/\alpha}$  such that  $C \approx_e^\blacksquare C'$  and let  $\mathcal{D}_{\tilde{n}} \approx_e^\blacktriangle \mathcal{D}'_{\tilde{n}}$ . Then from Lemma 7,  $\mathcal{D}_{\tilde{n}}(\delta \langle (-)_{\tilde{n}'} \rangle p_1 : \tilde{n}) \equiv \approx_e^\blacktriangle \equiv \mathcal{D}'_{\tilde{n}}(\delta \langle (-)_{\tilde{n}'} \rangle p'_1 : \tilde{n})$ .

By induction hypothesis there exists  $P'_1 \xrightarrow{\delta'} a' : \tilde{n}'$  such that

$$\mathcal{D}_{\tilde{n}}(\delta \langle a : \tilde{n}' \rangle p_1 : \tilde{n}) \cdot C \equiv \approx_e^\bullet \equiv \mathcal{D}'_{\tilde{n}}(\delta \langle a' : \tilde{n}' \rangle p'_1 : \tilde{n}) \cdot C'$$

Hence,  $\delta \langle P'_1 \rangle p'_1 : \tilde{n} \xrightarrow{\delta \delta'} \delta \langle a' : \tilde{n}' \rangle p'_1 : \tilde{n}$ . Because  $\delta \langle P'_1 \rangle p'_1 : \tilde{n} \approx_e P'$  there exists  $P' \xrightarrow{\delta \delta'} A'$  such that

$$\mathcal{D}'_{\tilde{n}}(\delta \langle a' : \tilde{n}' \rangle p'_1 : \tilde{n}) \cdot C' \approx_e \mathcal{D}'_{\tilde{n}}(A') \cdot C'$$

Finally, since  $\equiv \cap \mathbf{P}_{c/\alpha} \times \mathbf{P}_{c/\alpha} \subseteq \approx_e$ ,  $\approx_e$  is transitive, and  $\approx_e^\bullet \approx_e^\circ \subseteq \approx_e^\bullet$  (Proposition 8.3 restricted to processes) we obtain

$$\mathcal{D}_{\tilde{n}}(\delta \langle a : \tilde{n}' \rangle p_1 : \tilde{n}) \cdot C \equiv \approx_e^\bullet \mathcal{D}'_{\tilde{n}}(A') \cdot C'$$

**case** Suppose  $\pi = \delta'$  and  $f = c$ . By induction hypothesis there exists  $P'_1 \xrightarrow{\delta'} c' : \tilde{n}'$  such that for all  $\mathcal{D}_{\tilde{n}'} \approx_e^\blacktriangle \mathcal{D}'_{\tilde{n}'}$

$$\mathcal{D}_{\tilde{n}'}(c : \tilde{n}') \approx_e^\blacksquare \mathcal{D}'_{\tilde{n}'}(c' : \tilde{n}') \quad (2)$$

Also,  $\delta \langle P'_1 \rangle p'_1 : \tilde{n} \xrightarrow{\delta \delta'} \delta \langle c' : \tilde{n}' \rangle p'_1 : \tilde{n}$ . Because  $\delta \langle P'_1 \rangle p'_1 : \tilde{n} \approx_e P'$  there exists  $P' \xrightarrow{\delta \delta'} C'$  such that for all  $\mathcal{D}'_{\tilde{n}'}$

$$\mathcal{D}'_{\tilde{n}'}(\delta \langle c' : \tilde{n}' \rangle p'_1 : \tilde{n}) \approx_e^\square \mathcal{D}'_{\tilde{n}'}(C')$$

Let  $\mathcal{D}_{\tilde{n}} \approx_e^\blacktriangle \mathcal{D}'_{\tilde{n}}$ . Then by Lemma 7,

$$\mathcal{D}_{\tilde{n}}(\delta \langle (-)_{\tilde{n}'} \rangle p_1 : \tilde{n}) \equiv \approx_e^\blacktriangle \equiv \mathcal{D}'_{\tilde{n}}(\delta \langle (-)_{\tilde{n}'} \rangle p'_1 : \tilde{n})$$

Hence from (2),

$$\mathcal{D}_{\tilde{n}}(\delta \langle c : \tilde{n}' \rangle p_1 : \tilde{n}) \equiv \approx_e^\blacksquare \equiv \mathcal{D}'_{\tilde{n}}(\delta \langle c' : \tilde{n}' \rangle p'_1 : \tilde{n})$$

Finally, since  $\equiv \cap \mathbf{C}_{c/\alpha} \times \mathbf{C}_{c/\alpha} \subseteq \approx_e^\square$ ,  $\approx_e^\square$  is transitive, and  $\approx_e^\blacksquare \approx_e^\square \subseteq \approx_e^\blacksquare$  (Proposition 8.3) we obtain

$$\mathcal{D}_{\tilde{n}}(\delta \langle c : \tilde{n}' \rangle p_1 : \tilde{n}) \approx_e^\blacksquare \mathcal{D}'_{\tilde{n}}(C')$$

**case** The case where  $\pi = \tau$  is immediate.

**Case (restriction):** Suppose  $P \xrightarrow{\pi} T$  because  $P = (n)p : \tilde{n}$ ,  $p : \tilde{n}n \xrightarrow{\pi} f : \tilde{n}n$ ,  $n \notin \text{fn}(\pi)$ , and  $T = (n)f : \tilde{n}$ . Then from the definition of  $\approx_e^\bullet$  there exists  $p'$  such that  $p : \tilde{n}n \approx_e^\bullet p' : \tilde{n}n$  and  $(n)p' : \tilde{n} \approx_e^\circ P'$ . We may assume  $p'$  is closed because  $\approx_e^\bullet$  is substitutive (Proposition 8.4).

**case** Suppose  $\pi = \delta$  and  $f = a$ . Let  $C, C' \in \mathbf{C}_{c/\alpha}$  such that  $C \approx_e^\bullet C'$  and let  $\mathcal{D}_{\tilde{n}} \approx_e^\bullet \mathcal{D}_{\tilde{n}}'$ . We proceed on the structure of  $\mathcal{D}_{\tilde{n}}$ .

Let  $\mathcal{D}_{\tilde{n}} = (-)_{\tilde{n}} : \tilde{n}'$ . Since  $(-)_{\tilde{n}n} : \tilde{n}'n \approx_e^\bullet (-)_{\tilde{n}n} : \tilde{n}'n$  it follows by induction hypothesis that there exists  $p' : \tilde{n}n \xrightarrow{\delta} a' : \tilde{n}n$  such that

$$a : \tilde{n}'n \cdot C \equiv \approx_e^\bullet a' : \tilde{n}'n \cdot C' \quad (3)$$

Hence,  $(n)p' : \tilde{n} \xrightarrow{\delta} (n)a' : \tilde{n}$  and because  $(n)p' : \tilde{n} \approx_e^\circ P'$  there exists  $P' \xrightarrow{\delta} A'$  such that

$$\mathcal{D}_{\tilde{n}}((n)a' : \tilde{n}) \cdot C' = (n)a' : \tilde{n}' \cdot C' \approx_e^\circ \mathcal{D}_{\tilde{n}}(A') \cdot C'$$

Using  $\alpha$ -conversion we may assume that  $n \notin \text{fn}(C) \cup \text{fn}(C')$  and hence  $(n)a : \tilde{n}' \cdot C \equiv (n)(a : \tilde{n}'n \cdot C)$  and  $(n)a' : \tilde{n}' \cdot C' \equiv (n)(a' : \tilde{n}'n \cdot C')$ .

Moreover since  $\approx_e^\bullet$  (Proposition 8.5) and also  $\equiv$  are constructor compatible we get from (3) that  $(n)(a : \tilde{n}'n \cdot C) \equiv \approx_e^\bullet (n)(a' : \tilde{n}'n \cdot C')$ . Then because  $\equiv$  and  $\approx_e^\circ$  are transitive,  $\equiv \subseteq \approx_e^\circ$ , and  $\approx_e^\bullet \approx_e^\circ \subseteq \approx_e^\bullet$  (Proposition 8.3 restricted to processes) we obtain:

$$\mathcal{D}_{\tilde{n}}((n)a : \tilde{n}) \cdot C = (n)a : \tilde{n}' \cdot C \equiv \approx_e^\bullet \mathcal{D}_{\tilde{n}}(A') \cdot C'$$

Suppose  $\mathcal{D}_{\tilde{n}} \neq (-)_{\tilde{n}} : \tilde{n}'$ . Then from Lemma 7,  $\mathcal{D}_{\tilde{n}}((n)(-)_{\tilde{n}n}) \equiv \approx_e^\bullet \equiv \mathcal{D}_{\tilde{n}}'((n)(-)_{\tilde{n}n})$  and by induction hypothesis there exists  $p' : \tilde{n}n \xrightarrow{\delta} a' : \tilde{n}n$  such that

$$\mathcal{D}_{\tilde{n}}((n)a) \cdot C \equiv \approx_e^\bullet \equiv \mathcal{D}_{\tilde{n}}'((n)a') \cdot C'$$

Hence,  $(n)p' : \tilde{n} \xrightarrow{\delta} (n)a' : \tilde{n}$  and because  $(n)p' : \tilde{n} \approx_e P'$  there exists  $P' \xrightarrow{\delta} A'$  such that

$$\mathcal{D}_{\tilde{n}}'((n)a') \cdot C' \approx_e \mathcal{D}_{\tilde{n}}'(A') \cdot C'$$

Therefore since  $\equiv \cap \mathbf{P}_{c/\alpha} \times \mathbf{P}_{c/\alpha} \subseteq \approx_e$ ,  $\approx_e$  is transitive, and  $\approx_e^\bullet \approx_e^\circ \subseteq \approx_e^\bullet$  (Proposition 8.3 restricted to processes) we obtain:

$$\mathcal{D}_{\tilde{n}}((n)a) \cdot C \equiv \approx_e^\bullet \mathcal{D}_{\tilde{n}}'((n)a') \cdot C'$$

**case** Suppose  $\pi = \bar{\delta}$  and  $f = a$ . Similar to the case above.

**case** Suppose  $\pi = \bar{\delta}$  and  $f = c$ . Similar to the corresponding case for nesting.

**case** Suppose  $\pi = \bar{\delta}$  and  $f = c$ . Similar to the case where  $\pi = \delta$  and  $f = c$ .

**case** Suppose  $\pi = \tau$  and  $f = p$ . Immediate

## D Proof of Proposition 9

**Lemma 9.**  $((\sim_e^\bullet)_c)^*$  and  $((\approx_e^\bullet)_c)^*$  are respectively early and early delay context bisimulations.

*Proof.* We only show that  $((\approx_e^\bullet)_c)^*$  is an early delay context bisimulation. The proof of  $((\sim_e^\bullet)_c)^*$  being an early context bisimulation is similar. It is enough to show that  $((\approx_e^\bullet)_c)^*$  is an early delay context simulation because due to Proposition 8.7  $(\approx_e^\bullet)^*$  is symmetric and hence so is  $((\approx_e^\bullet)_c)^*$ .

Let  $P ((\approx_e^\bullet)_c)^* P'$ . Then for some  $k \geq 0$ ,  $P ((\approx_e^\bullet)_c)^k P'$ . The rest of the proof is by induction in  $k$ .

**Base:**  $k = 0$ . Immediate because  $((\approx_e^\bullet)_c)^*$  is reflexive.

**Step:**  $k > 0$ . Let

$$P = P_0 (\approx_e^\bullet)_c P_1 \dots P_{k-1} (\approx_e^\bullet)_c P_k = P'$$

and suppose  $P \xrightarrow{\pi} T$ .

**Case:**  $\pi = \delta$  and  $T = a : \tilde{n}$ . By induction, for all  $C$  and  $\mathcal{D}_{\tilde{n}}$  there exists  $P_{k-1} \xrightarrow{\delta} a_{k-1} : \tilde{n}$  such that:

$$\mathcal{D}_{\tilde{n}}(a) \cdot C (((\approx_e^\bullet)_c)^*)^\circ \mathcal{D}_{\tilde{n}}(a_{k-1}) \cdot C$$

By repeated applications of Lemma 1 using that  $\equiv$  is transitive, that  $\equiv_c$  is a strong late context bisimulation (Lemma 3), and that  $\approx_e^\bullet$  is reflexive we obtain the existence of some  $P' \xrightarrow{\delta} A'$  such that:

$$\mathcal{D}_{\tilde{n}}(a_{k-1} : \tilde{n}) \cdot C \equiv \approx_e^\bullet \mathcal{D}_{\tilde{n}}(A') \cdot C$$

Hence because  $\equiv \subseteq \approx_e^\circ \subseteq \approx_e^\bullet$  (Proposition 8.3 restricted to processes) and since  $\approx_e^\bullet$  is constructor compatible (Proposition 8.5) we have that:

$$\mathcal{D}_{\tilde{n}}(a : \tilde{n}) \cdot C (((\approx_e^\bullet)_c)^*)^\circ \mathcal{D}_{\tilde{n}}(A') \cdot C$$

**Case:**  $\pi = \bar{\delta}$  and  $T = A$ . Similar to the case above.

**Case:**  $\pi = \varphi$  and  $T = C$ . Similar to the case above using that  $\equiv^\square \subseteq \approx_e^\square \subseteq \approx_e^\blacksquare$ , Lemma 4, and that  $\approx_e^\bullet$  is substitution closed.

**Proposition 11.**  $((\sim_e^\bullet)_c)^* \subseteq \sim_e^\circ$  and  $((\approx_e^\bullet)_c)^* \subseteq \approx_e^\circ$ .

*Proof.* We only show that  $((\approx_e^\bullet)_c)^* \subseteq \approx_e^\circ$ . The proof of  $((\sim_e^\bullet)_c)^* \subseteq \sim_e^\circ$  is similar.

It is sufficient to prove that  $((\approx_e^\bullet)_c)^* \subseteq \approx_e$  because  $(\cdot)^\circ$  is monotonic. Since  $\approx_e$  is the largest early delay context bisimulation it is enough to show that  $((\approx_e^\bullet)_c)^*$  is an early delay context bisimulation. We refer the reader to Lemma 9.

The actual proof of Proposition 9 goes as follows:

*Proof.* We only show that  $\approx_e^\circ$  is a congruence, the proof of  $\sim_e^\circ$  being a congruence is similar. Because  $\approx_e^\bullet$  is a congruence due to Proposition 8.4 and Proposition 8.5 it suffices to show that  $\approx_e^\circ = \approx_e^\bullet$ . From Proposition 8.2 (restricted to processes) we have  $\approx_e^\circ \subseteq \approx_e^\bullet$  so it only remains to prove that  $\approx_e^\bullet \subseteq \approx_e^\circ$ .

First observe that since  $\approx_e^\bullet$  is substitutive due to Proposition 8 then  $\approx_e^\bullet \subseteq ((\approx_e^\bullet)_c)^\circ$ . Moreover, by definition  $((\approx_e^\bullet)_c)^\circ \subseteq (((\approx_e^\bullet)_c)^*)^\circ$  and since we from Proposition 11 have  $((\approx_e^\bullet)_c)^* \subseteq \approx_e^\circ$  we get  $\approx_e^\bullet \subseteq \approx_e^\circ$ .