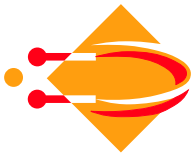

Bisimulation Congruences for Homer

a calculus of Higher-order mobile embedded resources

Thomas Hildebrandt
Jens Chr. Godskesen
Mikkel Bundgaard

IT University Technical Report Series **TR-2004-52**

ISSN 1600–6100 **October 2004**

ISSN 1600–6100

ISBN 87-7949-074-3

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: +45 72 18 50 00
Telefax: +45 72 18 50 01
Web `www.itu.dk`

Bisimulation Congruences for Homer

a calculus of Higher-order mobile embedded resources

Thomas Hildebrandt, Jens Chr. Godskesen, and Mikkel Bundgaard

Department of Theoretical Computer Science
IT University of Copenhagen
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark
{hilde, jcg, mikkelbu}@itu.dk

Abstract. We extend Howe's method for proving that late labelled transition bisimulations are congruences to a core process passing calculus with local names, extended with *non-linear active process mobility* and *nested locations*, as found in the Seal calculus, M-calculus, and Kell-calculus. The calculus we consider, called Homer for Higher-order Mobile Embedded Resources, has a very simple syntax and semantics, which conservatively extend the standard syntax and semantics for process passing calculi. The extension of Howe's method gives a sound characterisation of barbed bisimulation congruence in terms of a late contextual bisimulation. We show that early contextual bisimulation is complete with respect to barbed bisimulation congruence, but that the late bisimulation is in fact *strictly* included in the early bisimulation. We discuss the issue of scope extension through location boundaries in detail, in particular the difference between *fresh name generation* and *static local names*. We propose *free name extension* as a primitive construct in calculi with non-linear process mobility, explicit locations and local names.

1 Introduction

Process passing calculi, such as Plain CHOCS [1], provide an *explicit* kind of process mobility, by allowing a process to be *passed* on a named channel and substituted for a process variable in the receiving process. This can be expressed by the reaction rule

$$\overline{n}\langle r\rangle \,.\, p \parallel n(x) \,.\, q \searrow p \parallel q[r/x] \ , \tag{1}$$

where r is a process and x is a process variable binding any number of occurrences of x in q. The process r is usually assumed to be *passive*, meaning that it can neither compute internally nor interact with other processes before it has been sent. The receiver may activate 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 *weak* mobility, as opposed to (active) *process* mobility, or *strong* mobility, allowing processes to move between locations while they compute and interact with other processes.

The first-order π-calculus [2] allows *implicit* representations of code mobility as well as active process mobility. Code mobility is encoded by sending a reference to a process instead of the actual process, and active process mobility is encoded by representing the location of a (computing) process by the set of names at which it can communicate (see e.g. [3]). First-order calculi are usually considered simpler to reason about than higher-order calculi. However, the fact that location and mobility are derived notions in the π-calculus, makes it less direct to express properties and interrelations of locations, such as e.g. nesting of locations. Moreover, the encoding of active process mobility in the π-calculus is intrinsically *linear*: copyability of processes must be coded explicitly, such that a process at any step can provide a copy of itself.

Several recent calculi for mobility, such as the Ambient calculus [4] and the Seal calculus [5], have introduced an *explicit* notion of, possibly nested, named locations (called ambients or seals, respectively). In both calculi $n[p]$ denotes an active mobile process p computing

at a location named n. The semantics of these process calculi are as usual first defined by a set of reaction rules and a notion of barbed congruence. A standard problem is then to provide a labelled transition bisimulation congruence, which is desirable for co-inductive compositional reasoning about processes, and preferably prove that it is a sound and complete characterisation of the barbed congruence defined for the reaction semantics. This problem is well-known to be hard for higher-order calculi, in particular if one allows local names and active copyable mobile processes at explicit locations, as in the Seal calculus (for which only sound characterisations have been provided so far).

A classical method for proving that a higher-order labelled transition bisimulation is a congruence is the method of Howe [6, 7]. The method is to define a congruence inductively in the structure of the process terms that includes a late labelled bisimulation, and then prove it to be included in the labelled bisimulation as well. Howe's method was originally introduced for lazy functional programming languages, but has been applied to a calculus similar to Plain CHOCS (with local names and code mobility) by Baldamus and Frauenstein in [8]. The main technical contribution of the present paper is to extend Howe's method to local names, code mobility *and* non-linear active process mobility and explicit, nested locations. To this end, we introduce a core process passing calculus with the standard notion of local names, extended with *nested* names and a notion of (active) process *pulling*. These two extensions allow passive processes to be sent to (sub) locations and active processes to be pulled/moved from (sub) locations. We call the calculus Homer as short for *Higher-Order Mobile Embedded Resources* [1].

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

$$n[r] \,.\, p \parallel \overline{n}(x) \,.\, q \searrow p \parallel q[r/x] \ , \tag{2}$$

similar to rule (1) for passive process-passing. However, there are two important differences compared to the passive process passing: The first is that the process r in the prefix $n[r]$ may perform internal computations, i.e.

$$r \searrow r' \text{ implies } n[r] \,.\, p \searrow n[r'] \,.\, p \ . \tag{3}$$

This kind of rule was also considered by Boudol in [9]. It allows mobile resources to compute, but not to *interact* with other processes before movement. The second difference compared to passive process passing is that we allow interaction with active mobile resources. It is introduced by allowing *nested* names as location addresses, as found in nested name spaces of distributed systems and introduced in the calculus of Mobile Resources (MR) [10]. For instance, a resource r may be sent to the subaddress $server : 80$ by the reaction

$$\overline{server : 80}\langle r \rangle \,.\, p \parallel server[80(x) \,.\, q' \parallel q''] \searrow p \parallel server[q'[r/x] \parallel q''] \ . \tag{4}$$

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

$$server[80[r] \,.\, q' \parallel q''] \parallel \overline{server : 80}(x) \,.\, p \searrow server[q' \parallel q''] \parallel p[r/x] \ . \tag{5}$$

[1] and as a reference to the legend of Troy and the dangerous potential of mobile embedded resources.

In general, we allow interaction with arbitrarily deeply nested sub resources. However, note that two processes that are neither locally parallel nor in the sub/super process relation need a common super process to act as a router that mediates communication. For instance, the processes at locations A and B in the process

$$\begin{array}{l} A[toBat80[r] \,.\, q' \parallel q''] \parallel \overline{A : toBat80}(x) \,.\, \overline{B : 80}\langle x\rangle \parallel B[80(x) \,.\, p' \parallel p''] \searrow \\ A[q' \parallel q''] \parallel \overline{B : 80}\langle r\rangle \parallel B[80(x) \,.\, p' \parallel p''] \searrow \\ A[q' \parallel q''] \parallel B[p'[r/x] \parallel p''] \end{array} \tag{6}$$

cannot communicate synchronously with each other. Thus, Homer respects the recommended distinction between local synchronous and global asynchronous communication. Only are sub processes also considered as local processes.

The second feature of Homer, local names, is introduced in the usual way, letting $(n)p$ 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* [4], which in the Ambient calculus is written

$$(n)(n[p]) \approx \mathbf{0} \ , \tag{7}$$

expressing that a resource computing at a *local* location 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. For instance, if the resource r below contains the name n, we will expect the reaction

$$m[(n)(m'[r] \parallel p)] \parallel \overline{m : m'}(x) \,.\, q \searrow (n)(m[p] \parallel q[r/x]) \tag{8}$$

in which the scope of n is extended appropriately. We will refer to this kind of scope extension as *vertical* scope extension, since it extends the scope of a name vertically in the location tree, as opposed to the usual scope extension that extends the scope between parallel processes at the same location.

In the Ambient calculus vertical scope extension is dealt with in the structural congruence by introducing the equation

$$m[(n)p] \equiv (n)m[p] \ , \text{ if } n \neq m \ . \tag{9}$$

However, when combining local names and *non-linear* process mobility this equation is unsound, as also identified in [5, 11, 12]. The vertical scope extension rule identifies the two processes at the left hand of the reactions

$$m[(n)p] \parallel \overline{m}(x) \,.\, (x \parallel x) \searrow (n)p \parallel (n)p \tag{10}$$

and

$$(n)(m[p]) \parallel \overline{m}(x) \,.\, (x \parallel x) \searrow (n)(p \parallel p) \tag{11}$$

but the right hand sides will in general not be equivalent. The solution taken in [11, 12] is to exclude (9) and (eagerly) perform the vertical scope extension in the reaction rules when a process is moved. This means that copies of a computing process will always share local names, and in particular that the *left* hand process of the reaction (10) will perform a reaction

to the *right* hand process of the reaction (11) above. This choice can be seen as considering the local name constructor as *fresh name generation* (with priority over mobility).

In [5] they also exclude (9), but choose to extend the scope of the name n in the reaction (8) *if and only if* the name n is free in r. In Homer we take the same view of vertical scope extension. It means that local names are not necessarily shared between dynamically created copies of the same running process as demonstrated by (10), that only happens if the scope of the name is first extended (by (8) for instance) to encompass the process that performs the copy (as in e.g. (11)). This choice can be seen as considering the local name constructor as a *static local name* (e.g. an internal channel or address) of the resource. We believe that both the fresh and the static local name disciplines are interesting for active process mobility.

The static local name discipline implies that a context can test if a name is free in a mobile process. This can be illustrated by the following example, assuming n is free in p and combining the reductions (8), (10) and (11) above:

$$m[(n)(m'[r] \parallel p)] \parallel \overline{m : m'}(z) \,.\, \overline{m}(x) \,.\, (x \parallel x) \tag{12}$$

Since the scope of n is extended if and only if n is free in r, the process reduces to either the process in (10) or the process in (11), depending on whether or not n is free in r. Consequently, for any non-trivial congruence related processes r and r' must have the same set of free names. We refer to bisimulations between processes with the same set of free names as *well typed* relations and refer to the set of free names as the *interface* of a process. In particular, the firewall equation (7) will only hold if the process p has no free names. This is remedied by including a novel process operator $\{n\}$ in the calculus for extending the interface of a process with an *idle* name n. It allows us to state a *well typed firewall equation* as

$$(n)(n[p]) \approx \{m_1, m_2, \ldots, m_k\}\mathbf{0} \qquad \mathit{fn}(p)\backslash\{n\} = \{m_1, m_2, \ldots, m_k\} \tag{13}$$

where $\mathit{fn}(p)$ is the set of free names in p.

Note that the example (being similar to (12) above)

$$m[(n)(m'[\{n\}r] \parallel p)] \parallel \overline{m : m'}(z) \,.\, \overline{m}(x) \,.\, (x \parallel x) \ \ ,$$

where n is not free in r can be used to show that interface extension $\{n\}$ cannot be extended vertically either.

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 [13] we show that the synchronous π-calculus can in fact be encoded in a (simplified) variant of Homer. The major design criteria behind Homer is to provide a simple setting in which to study the interrelation between higher-order[2], non-linear active process mobility, nested locations and local names, and the problem of characterising a notion of barbed congruence by a labelled transition bisimulation congruence. In particular, the syntax and semantics conservatively extend the standard syntax and semantics for local names and process passing as found in e.g. Plain CHOCS.

[2] We use the term higher-order as synonym for process-passing, although Homer in the sense of [14] is only second-order.

Our adaptation of Howe's method allows us to show that both strong and weak *late* contextual bisimulation are congruences for Homer. This gives sound characterisations of the respective barbed bisimulation congruences. To get a complete characterisations one needs to show that the barbed bisimulation congruences are included in the late labelled bisimulation congruence as well. If the calculus is expressive enough, i.e. it allows to express the appropriate "testing" contexts for all labels in the labelled semantics, it is usually possible to show that barbed bisimulation congruence is included in an *early* labelled bisimulation. We prove that this is indeed the case for strong [3] barbed bisimulation. One may then proceed to show that early and late labelled bisimulations coincide. However, we show that this is not the case for Homer. Alternatively, one may try to prove directly that the early bisimulation is a congruence. We conjecture that this is the case, and are currently working on extending the results of the present paper to early labelled bisimulations for Homer.

Related Work Homer shares with the Seal calculus [5, 15], the M-calculus [16] (and its recent successor the Kell calculus [12]), the ability to represent copyable, objectively mobile anonymous resources in nested named locations. That the mobile resources are moved objectively means that they are moved by an outside process. That they are anonymous means that the local location boundary and its name disappears when the resource is moved. This is opposed to the Ambient calculus, where ambients move subjectively (by themselves) together with their named location boundary, and cannot be copied[4]. The M-calculus and the Kell calculus of Schmitt and Stefani share with Homer in being based on process passing and process substitution. The Kell calculus introduces, independently of our work, an active process prefix as found in Homer.

The syntax and transition semantics of Homer is considerably simpler than for the above calculi. In particular, seals are moved by an atomic three-party interaction (as in [10]), which introduces three *partial* synchronisations for mobility in the labelled transition semantics. Besides complicating the labelled transition semantics, three-party interactions also abstract from a locking scheme that might be non-trivial to implement in a global, distributed system.

The treatment of vertical scope extension has varied for the various versions of the Seal calculus. Originally [5], as described above, it was dealt with as in Homer, but formulated using an auxiliary heating relation. In [15] the authors take a more restrictive stand that bound names in a process have to be explicit communicated *by name passing* to the surrounding environment before the process can be moved. Hence the mobility of a process depends on name-passing, which means that one needs the primitives for name-passing to be explicit in the calculus. Moreover, mobile resources are in fact blocked from moving if they require vertical scope extension, which, as also remarked in [12], seems quite counterintuitive. As in Homer, contexts can test the free names of a process, which implies that non-trivial congruences must require related processes to have the same set of free names. This has been added to the definition in the later version of the paper [17]. It seems to suggest that support for interface extension as introduced in Homer should be added to the Seal calculus, e.g. to allow proving the firewall property for non-trivial processes.

[3] it also holds for the weak case, if the calculus is extended with a notion of *passive* process pulling as in [13].

[4] The M-calculus in fact supports both subjective and objective mobility.

Thomsen [1] and Prasad, Giacalone, and Mishra [18] provide so-called higher-order labelled transition bisimulations for respectively Plain CHOCS and Facile. The ho-bisimulations are early bisimulations where processes in higher-order labels are required to be bisimilar. As described nicely by Sangiorgi in [14] ho-bisimulations are too discriminating. He presents an elegant, and now classical, solution to the problem of giving a fully abstract characterisation by using triggered processes and normal bisimulation for the higher-order π-calculus. Jeffrey and Rathke extend in [19] the result to recursive types.

Fully abstract characterisations exist for variants of the Ambient calculus, which all allow active, *linear* (in contrast to the more general non-linear mobility of Homer), process mobility and nested locations. In the seminal paper [20] Merro and Hennessy examines a version of Safe Ambients, where they augment the synchronous mobility with passwords. For this calculus they give an early labelled transition system and associated contextual bisimulation equivalence, which is then proven to coincide with reduction barbed congruence. Merro and Nardelli have later extended this result to Ambient systems [21] and later to the full Ambient calculus [22]. Bugliesi *et al* present in [23] a sound and complete characterisation of reduction barbed congruence in Boxed Ambients in terms of an early contextual bisimulation equivalence.

In [17] Castagna, Vitek, and Nardelli considers the semantic theory of a revised version of the Seal calculus [5]. They introduce a reduction barbed congruence, an early labelled transition semantics, and an early contextual delay bisimilarity (called Hoe bisimilarity). They prove that the contextual delay bisimulation is a sound characterisation of reduction barbed congruence, but also argue that it is easily shown to be strictly contained in the barbed congruence and thus fails to be complete. A more delicate reason for incompleteness pointed out by the authors is a kind of asynchrony of seal movement, implying that it is unobservable if a process is moved from a seal in the context and placed immediately in a seal with the same name, analogously to input prefixes being unobservable in the asynchronous π-calculus.

For all of the labelled bisimulation congruences mentioned above, the congruence proof is roughly carried out 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 this paper.

We are not aware of any published sound and complete characterisation of barbed bisimulation congruence for a calculus with non-linear active process mobility and local names. Schmitt and Stefani do announce one in [12], again using a "brute force" proof technique. However, at the time of writing the details of the proof are still in the making.

Structure of the paper In Sec. 2 and Sec. 3 we present the syntax and reaction semantics of Homer. In Sec. 4 we provide examples of the expressiveness of the calculus, giving a short comparison with some related calculi. In Sec. 5 we present the labelled transition semantics, and then in Sec. 6 we show how to adapt Howe's method as presented in [8], proving a late contextual bisimulation to be a congruence, sound with respect to weak barbed bisimulation congruence. We also show that barbed bisimulation congruence is included in early contextual bisimulation, but that the late bisimulations are strictly included in the early contextual bisimulations.

2 The 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 γ range over (possibly empty) sequences of names, and δ range over non-empty sequences of names, referred to as *addresses*. We assume an infinite set of *process variables* $\mathcal{V}$ ranged over by x and y.

The set $\mathcal{P}$ of *process expressions* is then defined by the grammar:

$$\begin{array}{rll} p, q, r ::= & \mathbf{0} & \text{inactive process} \\ & \mid \lambda \,.\, p & \text{action prefixing} \\ & \mid p \parallel q & \text{parallel composition} \\ & \mid (n)p & \text{let } n \text{ be local in } p \\ & \mid \{n\}p & \text{let } n \text{ be free in } p \\ & \mid x & \text{process variable} \end{array}$$

and the set Λ of prefixes is defined by the grammar:

$$\begin{array}{rll} \lambda ::= & \delta(x) & \text{receive a resource at } \delta \text{ and bind it to } x \\ & \mid \overline{\delta}(x) & \text{take computing resource from } \delta \text{ and bind it to } x \\ & \mid \overline{\delta}\langle r \rangle & \text{send a resource } r \text{ to } \delta \\ & \mid \delta[r] & \text{computing resource } r \text{ at } \delta \end{array}$$

We let $\varphi ::= \delta \mid \overline{\delta}$ and let $\varphi[\![r]\!]$ range over $\overline{\delta}\langle r \rangle$ and $\delta[r]$.

The process constructors are the standard constructors from concurrent process calculi, extended with process variables as in Plain CHOCS and a constructor $\{n\}p$ for extending the *interface* of a process p with an idle name n. The restriction operator (n) binds the name n and the prefixes $\varphi(x)$ bind the variable x. The sets $bn(p)$ and $bv(p)$ of *bound names* and *bound variables*, and the sets $fn(\lambda)$, $fn(p)$ and $fv(p)$ of *free names* and *free variables* are defined accordingly as usual, with the addition that $fn(\{n\}p) = \{n\} \cup fn(p)$.

The prefixes $\overline{\delta}\langle r \rangle$ and $\delta(x)$ correspond to the send and receive prefixes in CHOCS, except that paths, and not only names, are allowed as addresses. As explained in the introduction, the new prefixes $\delta[r]$ and $\overline{\delta}(x)$ add strong mobility to the calculus. We say that a process with no free variables is *closed* and let $\mathcal{P}_c$ denote the set of closed processes. We write $p \equiv_\alpha q$, if p and q are α-convertible (wrt. both names and variables), and we let $\mathcal{P}_{/\alpha}$ (and $\mathcal{P}_{c/\alpha}$) denote the set of α-equivalence classes of (closed) process expressions. We consider processes up to α-equivalence. We define the process $p[q/x]$ to be p with all free occurrences of x replaced by q, if necessary α-converting p such that no free names and variables in q are bound.

By convenience, we omit trailing $\mathbf{0}$s and hence write λ instead of $\lambda \,.\, \mathbf{0}$. As usual, we let prefixing and restriction be right associative and bind stronger than parallel composition. For a set of names $\tilde{n} = \{n_1, \ldots, n_k\}$ we let $(\tilde{n})p$ denote $(n_1) \cdots (n_k)p$ and $\{\tilde{n}\}p$ denote $\{n_1\} \cdots \{n_k\}p$. We write $\tilde{m}\tilde{n}$ for $\tilde{m} \cup \tilde{n}$, always implicitly assuming $\tilde{m} \cap \tilde{n} = \emptyset$. We will write n for the set $\{n\}$ and δ for the set $fn(\delta)$, when no confusion can occur.

3 Reaction Semantics

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

$$p \parallel \mathbf{0} \equiv p \quad (p \parallel p') \parallel p'' \equiv p \parallel (p' \parallel p'') \quad p \parallel q \equiv q \parallel p$$

$$(n)p \parallel q \equiv (n)(p \parallel q), \text{ if } n \notin fn(q) \quad \lambda \,.\, (n)p \equiv (n)\lambda \,.\, p, \text{ if } n \notin fn(\lambda)$$

$$(n)(m)p \equiv (m)(n)p \quad (n)p \equiv p, \text{ if } n \notin fn(p)$$

$$\{n\}p \parallel q \equiv \{n\}(p \parallel q), \quad \lambda \,.\, \{n\}p \equiv \{n\}\lambda \,.\, p \quad \{n\}\{m\}p \equiv \{m\}\{n\}p \quad \{n\}p \equiv p, \text{ if } n \in fn(p)$$

$$(n)\{n\}p \equiv (n)p \quad (n)\{m\}p \equiv \{m\}(n)p, \text{ if } n \neq m$$

Table 1. Structural congruence.

A binary relation $\mathcal{R}$ on $\mathcal{P}_{/\alpha}$ is *well typed* if $p \,\mathcal{R}\, q$ implies $fn(p) = fn(q)$. In the sequel we always assume binary relations on $\mathcal{P}_{/\alpha}$ to be well typed. $\mathcal{R}$ is *substitutive* if $p \,\mathcal{R}\, q$ and $p' \,\mathcal{R}\, q'$ implies $p[p'/x] \,\mathcal{R}\, q[q'/x]$. $\mathcal{R}$ is *constructor compatible* if $p \,\mathcal{R}\, q$ and $p' \,\mathcal{R}\, q'$ implies

- $\varphi(x) \,.\, p \,\mathcal{R}\, \varphi(x) \,.\, q$,
- $\varphi\langle p\rangle \,.\, p' \,\mathcal{R}\, \varphi\langle q\rangle \,.\, q'$,
- $p \parallel p' \,\mathcal{R}\, q \parallel q'$,
- $\{n\}p \,\mathcal{R}\, \{n\}q$, and
- $(n)p \,\mathcal{R}\, (n)q$.

A well typed equivalence relation $\mathcal{R}$ on $\mathcal{P}_{/\alpha}$ is a *congruence* if it is substitutive and constructor compatible.

Structural congruence $\equiv$ is then the least congruence on $\mathcal{P}_{/\alpha}$ satisfying the rules in Table 1. The first row of the equations express that $(P, \parallel, \mathbf{0})$ is a commutative monoid, the next two rows enforce the rules for scope of name restriction. Hereafter follows a row with the similar rules for the scope of interface extension. Finally, the last row describes the relationship between the name restriction operator and the operator for extending the interface of a process.

Contexts C are, as usual, terms with a hole $(-)$. We write $C(p)$ for the insertion of p in the hole of context C. Note that free names (and variables) of p may get bound by insertion of p in the hole of a context. We extend $fn()$ to contexts by $fn(C) = fn(C(\mathbf{0}))$. *Evaluation contexts* E are contexts with no free variables, whose hole is neither guarded by a prefix, nor appear in the value of a send prefix, i.e.

$$E ::= (-) \mid E \parallel p \mid (n)E \mid \{n\}E \mid \delta[E] \,.\, p \ , \text{ for } p \in \mathcal{P}_c.$$

The evaluation context $\delta[E] \,.\, p$ enables internal reactions of active resources.

As Homer permits reactions between a process and an arbitrarily deeply nested subresource, we define a family of *path contexts* $C^{\tilde{n}}_{\gamma}$ indexed by a path address $\gamma \in \mathcal{N}^*$ and a set of names $\tilde{n}$. For a context $C^{\tilde{n}}_{\gamma}$, the path address γ indicates the path under which the context's 'hole' is found and the set $\tilde{n}$ of names indicates the bound names of the hole. We define the path contexts inductively in $\tilde{n}$ and γ by $C^{\emptyset}_{\epsilon} ::= (-)$ and whenever $p, q \in \mathcal{P}_c$,

$$C^{\tilde{n}\tilde{m}}_{\delta\gamma} ::= \delta[(\tilde{n})(C^{\tilde{m}}_{\gamma} \parallel p)] \,.\, q \ ,$$

(*send*)	$\overline{\gamma\delta}\langle r\rangle \,.\, q \parallel C^{\tilde{m}}_{\gamma}(\delta(x) \,.\, p) \searrow \{\tilde{n}\gamma\delta\}q \parallel C^{\tilde{m}}_{\gamma}(\{\delta\}p[r/x])$, if $\tilde{n} = \mathit{fn}(r)$ and $\tilde{m} \cap (\delta \cup \tilde{n}) = \emptyset$
(*take*)	$C^{\tilde{m}}_{\gamma}(\delta[r].q) \parallel \overline{\gamma\delta}(x) \,.\, p \searrow (\tilde{n} \cap \tilde{m})(C^{\tilde{m}}_{\gamma}\backslash\tilde{n}(\{\tilde{n}\delta\}q) \parallel \{\gamma\delta\}p[r/x])$, if $\tilde{n} = \mathit{fn}(r)$ and $\tilde{m} \cap (\delta \cup \mathit{fn}(p)) = \emptyset$

Table 2. Reaction rules

where $\tilde{n} \cap \gamma = \emptyset$. The side condition ensures that none of the names in the path address are bound. The bound names $(\tilde{n})$ are needed since the structural congruence does not permit vertical scope extension, as described in the introduction. Note that for any evaluation context E we have $E(q) \equiv (\tilde{n})(C^{\tilde{m}}_{\gamma}(q) \parallel p)$ for some $\tilde{n}$, p, and $C^{\tilde{m}}_{\gamma}$.

We define $\searrow$ as the least binary relation on $\mathcal{P}_{c/\alpha}$ satisfying the rules in Table 2 and closed under all evaluation contexts E and structural congruence.

The (*send*) rule expresses how a passive resource r is sent (down) to the (sub) location γ, where it is received at the address δ, and substituted into the receiving process p, possibly in several copies. Notice, no free names of r get bound during movement. The (*take*) rule captures that a computing resource r is taken from the sublocation γ, where it is running at the address δ, and substituted into the process p, again possibly in several copies.

We handle the vertical scope extension using an *open* operator, defined on path contexts $C^{\tilde{n}}_{\gamma}\backslash\tilde{m}$ inductively by: $C^{\emptyset}_{\epsilon}\backslash\tilde{m} = C^{\emptyset}_{\epsilon}$ and

$$C^{\tilde{n}_1\tilde{n}_2}_{\delta\gamma}\backslash\tilde{m} = \delta[(\tilde{n}_1\backslash\tilde{m})(C^{\tilde{n}_2}_{\gamma}\backslash\tilde{m} \parallel p)] \,.\, q \ ,$$

if $C^{\tilde{n}_1\tilde{n}_2}_{\delta\gamma} = \delta[(\tilde{n}_1)(C^{\tilde{n}_2}_{\gamma} \parallel p)] \,.\, q$ and $\tilde{n}_1\tilde{n}_2 \cap \mathit{fn}(C^{\tilde{n}_1\tilde{n}_2}_{\delta\gamma}) = \emptyset$. When applied in the reaction rule (*take*), the latter condition can always be met by α-conversion, and ensures that we can extend the scope by using the open operator and place the restriction at top level, without any name captures.

The interface extensions $\{\tilde{n}\gamma\delta\}$, $\{\delta\}$, $\{\tilde{n}\delta\}$, and $\{\gamma\delta\}$ ensure that no names can disappear from the interface of a process or any of its sub resources during reaction. However, note that the process that receives the moved resource r gets its interface extended by the free names of r that do not already appear in its interface.

The use of path contexts allows us to treat movement at the same location between two parallel processes as just special cases of moving up or down. In this special case $\gamma = \epsilon$ and the two rules reduce to the single rule schema:

$$\varphi[\![r]\!] \,.\, q \parallel \overline{\varphi}(x) \,.\, p \searrow \{\tilde{n}\}(q \parallel p[r/x]) \ , \text{ for } \tilde{n} = \mathit{fn}(\varphi) \cup \mathit{fn}(r),$$

(using the structural congruence to extend the scope of the interface extension).

4 Examples

In this section we give examples of the expressiveness of Homer related to similar calculi for mobility.

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 [1], except that recursion variables may appear at sub locations, which makes the definition slightly more complicated. Define

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

where $rec^a x \,.\, p = \overline{a}\langle r\rangle \parallel r$ for $r = a(x) \,.\, p[(\overline{a}\langle x\rangle \parallel x)/x]$, for $a \notin fn(p)$. Intuitively, r places a copy of $rec^a x \,.\, p$ at all occurrences of x in p, i.e. $rec^a x \,.\, p \searrow p[rec^a x \,.\, p/x]$. From recursion one may encode replication (again up to weak equivalence) in the usual way, defining $!p =_{def} rec\, x \,.\, (p \parallel x)$.

Persistent locations (Slots) As described in the introduction, the process $\delta[p] \,.\, q$ denotes a mobile computing resource p at a location named δ, where q is the residual process that remains when the resource p is moved. Depending on q it is possible to represent different kinds of locations. Setting q to the nil process $\mathbf{0}$, one represent an asynchronous, Seal-like location as described below.

One may expect some locations to be persistent, that is, if a resource is moved from a location n, this (part of the) location persists, and in particular, one may place a resource at that (part of the) location again, as e.g. the physical slot of a smart card reader. This was the idea of the *slots* introduced in the MR calculus [10], where $n\lfloor\bullet\rfloor$ denoted an empty (space of a) location named n at which exactly one resource could be put and $n\lfloor p\rfloor$ denoted a space of location n occupied by the resource p. This can be modelled in Homer (using recursion) by

$$\begin{aligned} n\lfloor p\rfloor &=_{def} n[p] \,.\, n\lfloor\bullet\rfloor \\ n\lfloor\bullet\rfloor &=_{def} rec\, x \,.\, n(y) \,.\, n[y] \,.\, x \end{aligned}$$

A resource r can then be moved from a slot named n to a slot named m by the reactions:

$$n\lfloor r\rfloor \parallel \overline{n}(x) \,.\, \overline{m}\langle x\rangle \parallel m\lfloor\bullet\rfloor \searrow n\lfloor\bullet\rfloor \parallel \overline{m}\langle r\rangle \parallel m\lfloor\bullet\rfloor \searrow\searrow n\lfloor\bullet\rfloor \parallel \mathbf{0} \parallel m\lfloor r\rfloor$$

(using an extra reaction to unfold the recursion).

Seal-like mobility The basic mobility in Seal allows a computing resource to be moved from an address m and started again immediately in a number of copies at addresses $m_1, \ldots, m_k$. This is expressed by a three-party reaction rule

$$\overline{n}\{m\}.p \mid m[r] \mid n\{m_1, m_2, \ldots, m_k\}.q \searrow p \mid m_1[r] \mid m_2[r] \mid \ldots \mid m_k[r] \mid q \ ,$$

where $m[r]$ is a seal, i.e. a process r computing at location m, and the action prefixes $\overline{n}\{m\}$ and $n\{m_1, m_2, \ldots, m_k\}$ denotes respectively send the content of a seal named m at channel n, and receive the content on channel n and run it in k copies at locations $\{m_1, m_2, \ldots, m_k\}$. We may simulate the three-party interaction as two actions in Homer:

$$\begin{aligned} &\overline{n}\{m\}.p =_{def} \overline{m}(x) \,.\, \overline{n}\langle x\rangle \,.\, p \ , \text{ for } x \notin fv(p) \\ &n\{m_1, m_2, \ldots, m_k\}.q =_{def} n(x) \,.\, (m_1[x] \parallel m_2[x] \parallel \ldots \parallel m_k[x] \parallel q) \ , \text{ for } x \notin fv(q) \end{aligned}$$

giving reductions

$$\overline{n}\{m\}.p \parallel m[r] \parallel n\{m_1, m_2, \ldots, m_k\}.q \searrow \overline{n}\langle r\rangle.p \parallel n\{m_1, m_2, \ldots, m_k\}.q$$
$$\searrow \{n\}(p \parallel m_1[r] \parallel m_2[r] \parallel \ldots \parallel m_k[r] \parallel q) \ .$$

This is of course not a faithful encoding, since the atomicity is broken. To simulate the atomic three-party movement one needs to encode a locking (and backtracking) scheme, which is beyond the scope of this paper.

Ambient-like mobility As mentioned in the introduction, the mobility in the ambient calculus is subjective and ambients preserve their name, as opposed to the anonymous objective movement of Homer. To simulate subjective mobility in Homer one needs external routers (appropriately dispersed at any location) that are able to move (and open) ambients at their request. A naive input router could be modelled by

$$In_{n,m} = \overline{n : inm}(x) \,.\, \overline{n}(y) \,.\, \overline{m}(z) \,.\, m[n[y] \parallel z] \ .$$

The ambient calculus process $n[in\ m.p \parallel p'] \parallel m[q]$, which can reduce to $m[n[p \parallel p'] \parallel q]$, could then be simulated by the process

$$n[inm[0] \,.\, p \parallel p'] \parallel m[q] \parallel In_{n,m} \ ,$$

which can reduce in the following way

$$n[inm[0] \,.\, p \parallel p'] \parallel m[q] \parallel In_{n,m} \searrow$$
$$n[p \parallel p'] \parallel m[q] \parallel \overline{n}(y) \,.\, \overline{m}(z) \,.\, m[n[y] \parallel z] \searrow$$
$$m[q] \parallel \overline{m}(z) \,.\, m[n[p \parallel p'] \parallel z] \searrow$$
$$m[n[p \parallel p'] \parallel q] \ .$$

In general, one could use replicated routers at any location and for any possible combination of ambient names. As for the Seal calculus, one should also implement a locking (and backtracking) scheme to ensure atomicity of the movement.

5 Transition Semantics

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

$$\pi ::= \tau \mid \varphi(x) \mid (\tilde{n})\varphi[\![r]\!] \ ,$$

defining $(\emptyset)\lambda = \lambda$. The set of free and bound names in π, $fn(\pi)$, $bn(\pi)$, is respectively $fn(\lambda)\backslash\tilde{n}$ and $\tilde{n}$ whenever $\pi = (\tilde{n})\lambda$. As for the reduction semantics we define the transitions for α-equivalence classes of closed processes. The rules in Table 3 then define a labelled transition system

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

$$(\textit{prefix})\ \frac{}{\lambda \,.\, p \xrightarrow{\lambda} \{\tilde{n}\}p}\ \tilde{n} = \mathit{fn}(\lambda) \qquad (\textit{sync})\ \frac{p_1 \xrightarrow{(\tilde{n})\overline{\varphi}[\![r]\!]} p_1' \qquad p_2 \xrightarrow{\varphi(x)} p_2'}{p_1 \parallel p_2 \xrightarrow{\tau} (\tilde{n})(p_1' \parallel p_2'[r/x])}\ \tilde{n} \cap \mathit{fn}(p_2) = \emptyset$$

$$(\textit{par})\ \frac{p \xrightarrow{\pi} p'}{p \parallel q \xrightarrow{\pi} p' \parallel q},\ \mathit{fn}(q) \cap \mathit{bn}(\pi) = \emptyset \qquad (\textit{sym})\ \frac{p \parallel q \xrightarrow{\pi} p'}{q \parallel p \xrightarrow{\pi} p'}$$

$$(\textit{rest})\ \frac{p \xrightarrow{\pi} p'}{(n)p \xrightarrow{\pi} (n)p'},\ n \notin \mathit{fn}(\pi) \qquad (\textit{open})\ \frac{p \xrightarrow{(\tilde{n})\varphi[\![r]\!]} p'}{(n)p \xrightarrow{(n\tilde{n})\varphi[\![r]\!]} p'},\ n \in \mathit{fn}(r)\backslash(\mathit{fn}(\varphi) \cup \tilde{n})$$

$$(\textit{free})\ \frac{p \xrightarrow{\pi} p'}{\{n\}p \xrightarrow{\pi} \{n\}p'},\ n \notin \mathit{bn}(\pi) \qquad (\textit{nesting})\ \frac{p \xrightarrow{\pi} p'}{\delta[p] \,.\, q \xrightarrow{\delta \cdot \pi} \delta[p'] \,.\, q},\ \mathit{fn}(q) \cap \mathit{bn}(\pi) = \emptyset$$

Table 3. Transition rules.

The rules are completely standard, except for the handling of free names and interaction with nested resources, which shows in the (*prefix*), (*free*), and (*nesting*) rules. The (*prefix*) rule is changed to ensure that interfaces are preserved. The (*free*) rule expresses that interface extension does not change the behaviour of processes. The (*nesting*) rule takes care of the communication and computation of arbitrarily deeply nested resources. It uses an operation $\delta \cdot (\_)$ formally defined by:

- $\delta \cdot \tau = \tau$,
- $\delta \cdot \delta'(x) = \delta\delta'(x)$, and
- $\delta \cdot (\tilde{n})\delta'[r] = (\tilde{n})\delta\delta'[r]$, for $\delta \cap \tilde{n} = \emptyset$.

Note that the operation is not defined for take and send labels, since these actions are directed "downward" and thus not visible outside the resource. When we write $\delta \cdot \pi$, we implicitly assume that π is a label for which the operation is defined. Since $\delta \cdot \tau = \tau$, the nesting rule implies that

$$\delta[p] \,.\, q \xrightarrow{\tau} \delta[p'] \,.\, q, \text{ if } p \xrightarrow{\tau} p' \ .$$

We also have e.g.

$$n[m[r] \,.\, p] \,.\, p' \xrightarrow{nm[r]} n[\{m\tilde{n}\}p] \,.\, p' \quad \text{and} \quad n[m(x) \,.\, p] \,.\, p' \xrightarrow{nm(x)} n[\{m\}p] \,.\, p' \ ,$$

where $\tilde{n} = \mathit{fn}(r)$.

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*), defining $\overline{\overline{\delta}} = \delta$.

As for Plain CHOCS and the π-calculus, scope extension is handled by the (*open*) rule, which moves the binding to the output label. The binding is then closed again by the (*sync*) rule. The side condition of the (*open*) rule ensures, as usual, that the bound name is indeed free in the resource r and neither part of the path nor already bound in the label.

6 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 late contextual transition bisimulations are congruences using a novel extension of Howe's method, based on the approach in [8], and that they are sound with respect to barbed bisimulation congruences.

6.1 Barbed Bisimulation Congruence

We define weak and strong *barbs* standardly letting $\searrow^*$ be the transitive and reflexive closure of $\searrow$, and choosing barbs similarly to [20, 17], e.g. the possibility to observe an unrestricted location n at top level

$$p \downarrow n \quad \text{if } p \equiv (\tilde{n})(n[r] \,.\, p' \parallel q) \text{ where } n \notin \tilde{n}, \text{ and}$$
$$p \Downarrow n \quad \text{if } \exists p'.\, p \searrow^* p' \text{ and } p' \downarrow n \ .$$

The following two propositions state the standard correspondence between the reduction semantics and the transition semantics.

Proposition 1. *$p \searrow q$ iff $p \xrightarrow{\tau} \equiv q$.*

Proposition 2. *$p \downarrow n$ iff $p \xrightarrow{(\tilde{n})n[r]} q$ for some $\tilde{n}$, r, and q.*

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

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

$i)$ if $p \downarrow n$ then $q \Downarrow n$

$ii)$ if $p \searrow p'$, then $\exists q'.\, q \searrow^* q'$ and $p' \mathcal{R} q'$

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

Definition 2. A *strong barbed simulation* is a well typed binary relation $\mathcal{R}$ on $\mathcal{P}_{c/\alpha}$, such that whenever $p \,\mathcal{R}\, q$,

$i)$ if $p \downarrow n$ then $q \downarrow n$

$ii)$ if $p \searrow p'$, then $\exists q'.\, q \searrow q'$ and $p' \mathcal{R} q'$

$\mathcal{R}$ is a *strong barbed bisimulation* if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong barbed simulations. We let $\simeq_b$ denote the largest strong barbed bisimulation. *Strong barbed bisimulation congruence* $\sim_b$ is the largest congruence such that $(\sim_b)_c$ is a strong barbed bisimulation.

We may define barbs more systematically from the reduction semantics as follows:

$$
\begin{array}{ll}
p \downarrow \gamma\delta & \text{if } p \equiv (\tilde{n})(C_{\gamma}^{\tilde{m}}(\delta[r].p') \parallel q) \text{ and } \delta \cap \tilde{n}\tilde{m} = \emptyset \text{ and } \gamma \cap \tilde{n} = \emptyset, \\
p \downarrow \gamma\delta(x) & \text{if } p \equiv (\tilde{n})(C_{\gamma}^{\tilde{m}}(\delta(x).p') \parallel q) \text{ and } \delta \cap \tilde{n}\tilde{m} = \emptyset \text{ and } \gamma \cap \tilde{n} = \emptyset, \\
p \downarrow \overline{\delta} & \text{if } p \equiv (\tilde{n})(\overline{\delta}\langle r\rangle \,.\, p' \parallel q) \text{ and } \delta \cap \tilde{n} = \emptyset, \\
p \downarrow \overline{\delta}(x) & \text{if } p \equiv (\tilde{n})(\overline{\delta}(x) \,.\, p' \parallel q) \text{ and } \delta \cap \tilde{n} = \emptyset,
\end{array}
$$

The barb $p \downarrow \gamma\delta$ express the existence of a mobile resource at the address $\gamma\delta$, hence our choice of barbs at the beginning of this section is the special case of $p \downarrow \gamma\delta$, where the path has length one. Define the set of all barbs $\mathcal{B}$ by the grammar

$$b ::= \varphi \mid \varphi(x) \ .$$

Letting $b[n/m]$ denote the barb where we have substituted the name n for the name m, we say that a subset of barbs $B \subseteq \mathcal{B}$ is *substitution closed*, if $b \in B$ implies $b[n/m] \ \in B$.

Definition 3. For any *non-empty*, substitution closed subset of barbs B we define weak B-*barbed simulation* and subsequently weak B-*barbed bisimulation* $\approxeq_B$ by replacing condition $i)$ in Def. 1 by

$$i) \text{ if } p \downarrow b \text{ for } b \in B \text{ then } q \Downarrow b$$

and define weak B-*barbed bisimulation congruence* $\approx_B$ to be the largest congruence such that $(\approx_B)_c$ is a weak B-barbed bisimulation. We define strong B-barbed bisimulation congruence accordingly.

The proposition below shows (as in [20]) that our choice of barbs is indeed robust. In particular, one could have chosen to observe the existence of a location path δ. But first we state a lemma relating the different kinds of barbs.

Lemma 1. *For any pair of barbs $b, b' \in \mathcal{B}$ there exists a context $C_{b,b'}$ such that for all $r \in \mathcal{P}_c$ where $fn(b') \cap fn(r) = \emptyset$, we have $r \downarrow b$ if and only if $C_{b,b'}(r) \searrow r'$ and $r' \downarrow b'$.*

Proof. (sketch) Due to the duality of our prefixes and we can define all the contexts $C_{b,b'}$ using the same schema. Letting $\widehat{(-)}$ denote the operator that maps a barb to its corresponding prefix: $\widehat{\varphi} = \varphi[\langle\mathbf{0}\rangle]$ and $\widehat{\varphi(x)} = \varphi(x)$ we define a family of contexts as

$$C_{b,b'} = (-) \parallel \widehat{b^{op}} \,.\, \widehat{b'} \ ,$$

where $\varphi(x)^{op} = \overline{\varphi}$ and $\varphi^{op} = \overline{\varphi}(x)$ Then we have $r \downarrow b$ if and only if $C_{b,b'}(r) \searrow r'$ and $r' \downarrow b'$.

Compared with [20] these contexts seem almost trivial. There are several reasons for the simplicity of these contexts, most important is that we have locations as a prefix in the calculus, hence enabling us to observe the movement of a resource. The objective mobility also aid to the simplicity.

Proposition 3. *For any two non-empty, substitution closed subsets of barbs B and B' it holds that $\sim_B = \sim_{B'}$ and $\approx_B = \approx_{B'}$.*

Proof. We only present the proof of $\sim_B = \sim_{B'}$, the proof of $\approx_B = \approx_{B'}$ is similar.

We show that $(\sim_{B'})_c$ is a strong barbed B-bisimulation, by symmetry it follows that $(\sim_B)_c$ is a strong B'-barbed bisimulation. It is clear that both $(\sim_B)_c$ and $(\sim_{B'})_c$ are reduction closed, so the only remaining task is to show that the two different choices of barbs imply each other:

– if $p(\sim_{B'})_c q$ and $p \downarrow b \in B$ then we must be able to show that $q \downarrow b$. Since $(\sim_B)_c$ is a congruence, we have that $p(\sim_B)_c q$ implies $C(p)\ (\sim_B)_c\ C(q)$, for all contexts C. So we choose a $b' \in B'$ such that $fn(b') \cap (fn(p) \cup fn(q)) = \emptyset$. By Lemma 1 we know that there exists an context $C_{b,b'}$. $C_{b,b'}(p) \searrow p'$ and $p' \downarrow b'$, hence we know that there exists a q' such that $C_{b,b'}(q) \searrow q'$ and $q' \downarrow b'$, which in turn implies $q \downarrow b$ as desired.

6.2 Labelled Bisimulation Congruence

As standard in context bisimulations, we employ *abstractions* and *concretions* for the higher order output transitions in the definition of our bisimulation. Normally, an abstraction is a *receiving* process, with a free variable for which the send process is substituted. As resources may be moved from arbitrarily deep sublocations, we define abstractions as path contexts in parallel with a receiving process.

$$(x)A^{\tilde{n}}_{\gamma} ::= C^{\tilde{n}}_{\gamma} \parallel p \ , \qquad \text{for } fv(p) \subseteq \{x\},$$

and we write $(x)A_\gamma$ for $(x)A^{\tilde{n}}_{\gamma}$, if $\tilde{n} = \emptyset$. The extra generality is used for instance when the resource r in the transition

$$n[r].q \xrightarrow{n[r]} \{n\tilde{m}\}q$$

is received by the context

$$m[(-)] \parallel \overline{m : n}(x).p$$

corresponding to the abstraction $m[(-)] \parallel p$.

As in [8] we consider concretions of the form $(\tilde{n}, r, q)$, and write $(\tilde{m}, r, q) \equiv_\alpha (\tilde{m}', r', q')$ whenever $(\tilde{m})\overline{\delta}\langle r\rangle\ .\ q \equiv_\alpha (\tilde{m}')\overline{\delta}\langle r'\rangle\ .\ q'$, for some δ. We then define the application of an abstraction $(x)A^{\tilde{n}}_{\gamma} = C^{\tilde{n}}_{\gamma} \parallel p$ to a concretion $(\tilde{m}, r, q)$ by

$$(x)A^{\tilde{n}}_{\gamma} \cdot (\tilde{m}, r, q) = (\tilde{m}'\tilde{n}')(C^{\tilde{n}}_{\gamma} \backslash \tilde{n}'(q') \parallel p[r'/x]),$$

if $(\tilde{m}, r, q) \equiv_\alpha (\tilde{m}', r', q')$, $\tilde{m}' \cap (fn(A^{\tilde{n}}_{\gamma}) \cup \tilde{n}) = \emptyset$, and $\tilde{n}' = fn(r') \cap \tilde{n}$.

We define *delay* transitions [24] by $p \stackrel{\tau}{\Longrightarrow} p$ and $p \stackrel{(\tilde{m})\lambda}{\Longrightarrow} q$, if $p \stackrel{\tau}{\Longrightarrow} p' \stackrel{(\tilde{m})\lambda}{\longrightarrow} q$, which compared to weak transitions do not allow τ-transitions after the visible transition.

We extend well typed binary relations $\mathcal{R}$ on $\mathcal{P}_{c/\alpha}$ to open terms the usual way, by defining $p\ \mathcal{R}^\circ\ q$ if for all $r_1, \ldots, r_k \in \mathcal{P}_{c/\alpha}$, $p[r_1/x_1] \ldots [r_k/x_k]\ \mathcal{R}\ q[r_1/x_1] \ldots [r_k/x_k]$, where $fv(p) = fv(q) = \{x_1, \ldots, x_k\}$, i.e. requiring two open terms to be related after substitution with closed resources.

Definition 4. A *late delay context simulation* is a well typed binary relation $\mathcal{R}$ on $\mathcal{P}_{c/\alpha}$ such that $p\ \mathcal{R}\ q$ implies

1. if $p \xrightarrow{\tau} p'$ then $\exists q'.\ q \stackrel{\tau}{\Longrightarrow} q'$ and $p'\ \mathcal{R}\ q'$
2. if $p \xrightarrow{\varphi(x)} p'$ then $\exists q'.\ q \stackrel{\varphi(x)}{\Longrightarrow} q'$ and $p'\ \mathcal{R}^{\circ}\ q'$
3. if $p \xrightarrow{(\tilde{n})\overline{\delta}\langle r\rangle} p'$ then $\exists q'.\ q \stackrel{(\tilde{n}')\overline{\delta}\langle r'\rangle}{\Longrightarrow} q'$ and $\forall (x)A_{\epsilon}.\ (x)A_{\epsilon}\cdot(\tilde{n},r,p')\ \mathcal{R}\ (x)A_{\epsilon}\cdot(\tilde{n}',r',q')$
4. if $p \xrightarrow{(\tilde{n})\delta[r]} p'$ then $\exists q'.\ q \stackrel{(\tilde{n}')\delta[r']}{\Longrightarrow} q'$ and $\forall (x)A^{\tilde{m}}_{\gamma}.\ (x)A^{\tilde{m}}_{\gamma}\cdot(\tilde{n},r,p')\ \mathcal{R}\ (x)A^{\tilde{m}}_{\gamma}\cdot(\tilde{n}',r',q')$

$\mathcal{R}$ is a late delay context bisimulation if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are late delay context simulations. Let $\approx_l$ denote the largest late delay context bisimulation. We define late strong context simulation by replacing $\Longrightarrow$ with $\longrightarrow$, and let $\sim_l$ denote the largest late strong context bisimulation.

Like in [15] we may introduce an *early* delay context bisimulation simply by replacing the second condition in Def. 4 by

$$2.\ \text{if } p \xrightarrow{\varphi(x)} p' \text{ then } \forall r \in \mathcal{P}_{c/\alpha} \exists q'.\ q \stackrel{\varphi(x)}{\Longrightarrow} q' \text{ and } p'[r/x]\ \mathcal{R}\ q'[r/x]$$

We let $\approx_e$ denote the largest early context delay bisimulation and we let $\sim_e$ denote the largest strong early context bisimulation.

Note that the first three conditions in Def. 4 correspond exactly to the conditions for the *late weak context simulation* in [8], since $(x)A_{\epsilon} = (-) \parallel p$ and $(-) \parallel p \cdot (\tilde{n},r,q) = (\tilde{n})(q \parallel p[r/x])$ (assuming $fn(p) \cap \tilde{n} = \emptyset$). Thus, the bisimulation is a *conservative* extension of the bisimulation in [8] to nested locations.

From Prop. 1 and Prop. 2 it follows standardly that the early contextual bisimulations are included in the corresponding barbed bisimulations.

Proposition 4. *$\sim_e \subseteq \simeq_b$, and $\approx_e \subseteq \cong_b$* .

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 5. $(\sim_b)_c \subseteq \sim_e$.

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

Theorem 1. $\sim_b \subseteq \sim_e^{\circ}$.

For standard reasons (see e.g.[20]) late delay contextual bisimulation is strictly contained in weak barbed congruence, and so cannot be complete with respect to weak barbed congruence. We conjecture that it do hold for a proper weak contextual bisimulation.

A perhaps more surprising result is that the late bisimulations are strictly contained in the early bisimulations. The proof exploits the ability to place, and replicate resources in locally named locations.

Proposition 6. *We have the following inclusions*

$$(\textit{nil})\ \frac{\mathbf{0}\ \mathcal{R}^\circ\ p}{\mathbf{0}\ \mathcal{R}^\bullet\ p} \qquad (\textit{var})\ \frac{x\ \mathcal{R}^\circ\ p}{x\ \mathcal{R}^\bullet\ p} \qquad (\textit{rest})\ \frac{r\ \mathcal{R}^\bullet\ q \qquad (n)q\ \mathcal{R}^\circ\ p}{(n)r\ \mathcal{R}^\bullet\ p}$$

$$(\textit{extend})\ \frac{r\ \mathcal{R}^\bullet\ q \qquad \{n\}q\ \mathcal{R}^\circ\ p}{\{n\}r\ \mathcal{R}^\bullet\ p} \qquad (\textit{par})\ \frac{r_1\ \mathcal{R}^\bullet\ q_1 \qquad r_2\ \mathcal{R}^\bullet\ q_2 \qquad q_1 \parallel q_2\ \mathcal{R}^\circ\ p}{r_1 \parallel r_2\ \mathcal{R}^\bullet\ p}$$

$$(\textit{in})\ \frac{r\ \mathcal{R}^\bullet\ q \qquad \varphi(x)\,.\,q\ \mathcal{R}^\circ\ p}{\varphi(x)\,.\,r\ \mathcal{R}^\bullet\ p} \qquad (\textit{out})\ \frac{r_1\ \mathcal{R}^\bullet\ q_1 \qquad r_2\ \mathcal{R}^\bullet\ q_2 \qquad \varphi(\!|q_1|\!)\,.\,q_2\ \mathcal{R}^\circ\ p}{\varphi(\!|r_1|\!)\,.\,r_2\ \mathcal{R}^\bullet\ p}$$

Table 4. The congruence relation $\mathcal{R}^\bullet$.

1. $\approx_l \subsetneq \approx_e$.
2. $\sim_l \subsetneq \sim_e$ *(if replication is added to the calculus).*

Proof. (sketch) We use replication of processes in the proof. In the weak case we can use the encoding of replication given in Sec. 4 (assuming that it is indeed an encoding). In the strong case we assume replication is added to the calculus in the usual way.

Define $s = !m[x]$, $p_1 = (m)(s \parallel n'[\mathbf{0}])$, $p_2 = (m)(s)$, and $p_3 = (m)(s \parallel \overline{m : m'}(z).n'[\mathbf{0}])$, for $n' \neq m$.

To prove 1. we define $p = !n(x)\,.\,p_1 \parallel !n(x)\,.\,p_2$ and $q = !n(x)\,.\,p_1 \parallel !n(x)\,.\,p_2 \parallel !n(x)\,.\,p_3$. We then have that $p \approx_e q$ but $p \not\approx_l q$.

The crucial step is if $q \xrightarrow{\tau} n(x)\,.\,p_3 \parallel q \xrightarrow{n(x)} p_3 \parallel q$.

These steps can be matched by p in an early semantics but not in a late semantics. For a resource r such that $r \Downarrow m'$, the process p can match with $p \xrightarrow{\tau} n(x)\,.\,p_1 \parallel p \xrightarrow{n(x)} p_1 \parallel p$. We prove that $p_1[r/x] \approx_e p_3[r/x] \approx_e \{\tilde{n}\}n'[\mathbf{0}]$, for $\tilde{n} = fn(r)$ and using that if $r \Downarrow m'$ then for all $!m[r] \Longrightarrow r'$ we have $r' \Downarrow m : m'$.

For a resource r such that $r \not\Downarrow m'$, one can match with the transition $p \xrightarrow{\tau} n(x)\,.\,p_2 \parallel p \xrightarrow{n(x)} (m)(s) \parallel p$. Here we use that if $r \not\Downarrow m'$ then $!m[r] \not\Downarrow m : m'$, to prove that $p_2[r/x] \approx_e p_3[r/x] \approx_e \{\tilde{n}\}\mathbf{0}$, for $\tilde{n} = fn(r)$.

In the late case, p has either to take the p_1 path or the p_2 path, which will not fit for all r. If the input for p_1 is chosen, it will not match for a resource r such that $r \not\Downarrow m'$, if the input p_2 is chosen it will not match for a resource r such that $r \Downarrow m'$.

In proving 2., we assume replication is added to the calculus (since otherwise the unfolding can be observed). One may then prove using similar reasoning as above that $p \parallel !\tau \sim_e q \parallel !\tau$ but $p \parallel !\tau \not\sim_l q \parallel !\tau$. The replicated τ process is used in the case where p_3 in q performs an interaction with the received resource.

We now show the main technical result of the paper, that ${\sim_l}^\circ$ and ${\approx_l}^\circ$ are indeed also congruences, by adapting Howe's method to active mobile processes in nested locations. Define the usual *congruence* $\mathcal{R}^\bullet$ on $\mathcal{P}_{/\alpha}$, of a binary relation $\mathcal{R}$ on $\mathcal{P}_{c/\alpha}$ by the rules in Table 4.

It is easy to prove that if $\mathcal{R}$ is well typed then $\mathcal{R}^\bullet$ is also well typed.

The key is now to show, since $\sim_l{}^\bullet$ ($\approx_l{}^\bullet$) is a congruence, that $\sim_l{}^\circ = \sim_l{}^\bullet$ ($\approx_l{}^\circ = \approx_l{}^\bullet$). As in [8] we first prove the following properties.

Proposition 7. *Let $\mathcal{R}$ be an equivalence relation on $\mathcal{P}_{c/\alpha}$ then*

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

We extend binary relations $\mathcal{R}$ on $\mathcal{P}_{c/\alpha}$ to abstractions by defining $(x)A^{\tilde{n}}_\gamma \; \mathcal{R}^\blacktriangle \; (x)B^{\tilde{n}}_\gamma$ if $p \, \mathcal{R}^\bullet \, q$ and $p' \, \mathcal{R}^\bullet \, q'$ implies $(x)A^{\tilde{n}}_\gamma \cdot (\tilde{m}, p, p') \equiv \mathcal{R}^\bullet \equiv (x)B^{\tilde{n}}_\gamma \cdot (\tilde{m}, q, q')$. The use of $\mathcal{R}^\blacktriangle$ is a new contribution of the present paper, it is required since in contrast to [8] we do not only use process terms (with a single free variable) as abstractions, we use abstractions being the parallel composition of a process and a path context.

From (4) and (7) of Prop. 7 we get

Lemma 2. *Let $\mathcal{R}$ be an equivalence relation on $\mathcal{P}_{c/\alpha}$ then $\mathcal{R}^\blacktriangle$ is reflexive.*

We will also use substitutiveness for abstractions.

Proposition 8. *Let $\mathcal{R}$ be an equivalence relation and let $p, q \in \mathcal{R}_{c/\alpha}$. Then $p\mathcal{R}^\bullet q$ and $(x)A^{\tilde{n}}_\gamma \mathcal{R}^\blacktriangle (x)B^{\tilde{n}}_\gamma$ implies*

$$(x)A^{\tilde{n}}_\gamma((-) \parallel p) \;\; \mathcal{R}^\blacktriangle \;\; (x)B^{\tilde{n}}_\gamma((-) \parallel q), \quad (x)A^{\tilde{n}}_\gamma(\delta[(-)] \,.\, p) \;\; \mathcal{R}^\blacktriangle \;\; (x)B^{\tilde{n}}_\gamma(\delta[(-)] \,.\, q),$$
$$(x)A^{\tilde{n}}_\gamma\big((n)(-)\big) \;\; \mathcal{R}^\blacktriangle \;\; (x)B^{\tilde{n}}_\gamma\big((n)(-)\big), \quad (x)A^{\tilde{n}}_\gamma\big(\{n\}(-)\big) \;\; \mathcal{R}^\blacktriangle \;\; (x)B^{\tilde{n}}_\gamma\big(\{n\}(-)\big),$$

whenever the left and right hand sides are well defined abstractions, possibly put on the normal form for abstractions by use of structural congruence.

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

Lemma 3. *Let $p, q \in \mathcal{P}_{c/\alpha}$. Then $p \approx_l{}^\bullet q$ implies:*

1. *if $p \xrightarrow{\tau} p'$ then $\exists q'.\; q \stackrel{\tau}{\Longrightarrow} q'$ and $p' \equiv \approx_l{}^\bullet q'$*
2. *if $p \xrightarrow{\varphi(x)} p'$ then $\exists q'.\; q \stackrel{\varphi(x)}{\Longrightarrow} q'$ and $p' \approx_l{}^\bullet q'$*
3. *if $p \xrightarrow{(\tilde{n})\overline{\delta}\langle r\rangle} p'$ then $\exists q'.\; q \stackrel{(\tilde{n}')\overline{\delta}\langle r'\rangle}{\Longrightarrow} q'$ and $\forall (x)A_\epsilon \approx_l{}^\blacktriangle (x)B_\epsilon.\; (x)A_\epsilon \cdot (\tilde{n}, r, p') \equiv \approx_l{}^\bullet (x)B_\epsilon \cdot (\tilde{n}', r', q')$*
4. *if $p \xrightarrow{(\tilde{n})\delta[r]} p'$ then $\exists q'.\; q \stackrel{(\tilde{n}')\delta[r']}{\Longrightarrow} q'$ and $\forall (x)A^{\tilde{m}}_\gamma \approx_l{}^\blacktriangle (x)B^{\tilde{m}}_\gamma.\; (x)A^{\tilde{m}}_\gamma \cdot (\tilde{n}, r, p') \equiv \approx_l{}^\bullet (x)B^{\tilde{m}}_\gamma \cdot (\tilde{n}', r', q')$*

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

Theorem 2. *$\sim_l{}^\circ$ and $\approx_l{}^\circ$ are congruences.*

From Prop. 1 and Thm. 2 it follows that $\sim_l{}^\circ$ and $\approx_l{}^\circ$ are sound with respect to barbed and weak barbed bisimulation congruence, respectively.

Theorem 3. *$\sim_l{}^\circ \subseteq \sim_b$ and $\approx_l{}^\circ \subseteq \approx_b$.*

Even though we do not have a complete characterisation, a sound characterisation is useful to prove concrete processes to be barbed bisimulation congruent. For instance, we can prove the well typed perfect firewall equation (13) from the introduction, because $\{\big((n)n[r], \{\tilde{n}\}\mathbf{0}\big) \mid r \in \mathcal{P}_c,\ \tilde{n} = fn(p)\backslash n\}$ is a late delay context bisimulation.

7 Conclusions and Future Work

We presented the calculus Homer, which combines higher-order process passing, local names, nested locations and 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. We presented the first generalisation of Howe's method to a calculus with *active* mobile processes at nested locations and local names, proving that *late* (strong and delay) contextual bisimulation is a congruence and a sound characterisation of (strong and weak) barbed bisimulation congruence. We showed that *early* contextual bisimulation is complete with respect to barbed bisimulation congruence in the strong case. However, we also found that the late bisimulations are *strictly* included in the early bisimulations. We conjecture that the example also shows that late contextual bisimulation is strictly included in the barbed bisimulation congruence, which excludes that our late contextual bisimulation characterisation is complete.

The semantics of Homer highlights the importance of keeping track of the free names of processes. We therefore introduce an explicit free name operator $\{n\}$ and consider only bisimulations between processes with the same set of free names.

We conjecture that early contextual bisimulation is indeed a characterisation of barbed bisimulation congruence, and are working on extending the congruence results of the present paper to early labelled bisimulations. One direction we explore is to use the theory of bigraphical reactive systems [25]. We also investigate if the trigger and normal 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. However this is complicated by the fact that triggered processes cannot be accessed from remote locations.

We are also considering type systems for Homer, in particular a type system for linear and non-linear mobile resources, which was presented for an early version of Homer in [26]. In this setting one can reason about systems containing both copyable as well as non-copyable mobile resources, which is crucial for systems containing both mobile hardware and mobile software.

Acknowledgements We acknowledge the anonymous referees of previous versions of the present paper for their helpful reviews and suggestions.

References

1. Thomsen, B.: Plain CHOCS. A second generation calculus for higher order processes. Acta Informatica **30** (1993) 1–59
2. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, part I/II. Journal of Information and Computation **100** (1992) 1–77
3. Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press (1999)
4. Cardelli, L., Gordon, A.D.: Mobile ambients. In Nivat, M., ed.: Proceedings of FoSSaCS '98. Volume 1378 of LNCS., Springer (1998) 140–155
5. Vitek, J., Castagna, G.: Seal: A framework for secure mobile computations. In: Internet Programming Languages. (1999)
6. Howe, D.J.: Proving congruence of bisimulation in functional programming languages. Information and Computation **124** (1996) 103–112
7. Howe, D.J.: Equality in lazy computation systems. In: Proceedings of the 4th IEEE Symposium on Logic in Computer Science. (1989) 198–203
8. 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)
9. 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
10. Godskesen, J.C., Hildebrandt, T., Sassone, V.: A calculus of mobile resources. In: Proceedings of CONCUR'2002. LNCS, Springer (2002)
11. Jeffrey, A., Rathke, J.: Towards a theory of bisimulation for local names. In: Proceedings of LICS '99, IEEE, Computer Society Press (1999) 56–66
12. 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)
13. 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.
14. 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.
15. 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
16. 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), ACM Press (2003) 50–61
17. Castagna, G., Vitek, J., Nardelli, F.Z.: The Seal calculus. accepted for publication in *Information and Computation* (2004)
18. 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
19. Jeffrey, A., Rathke, J.: Contextual equivalence for higher-order π-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)
20. Merro, M., Hennessy, M.: Bisimulation congruences in safe ambients. Computer Science Report 2001:05, University of Sussex (2001)
21. Merro, M., Nardelli, F.Z.: Bisimulation proof techniques for mobile ambients. In: Proceedings of ICALP 2003. (2003) to appear.
22. 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.
23. Bugliesi, M., Crafa, S., Merro, M., Sassone, V.: Communication and mobility control in boxed ambients. Journal of Information and Computation (200X)
24. Sangiorgi, D., Walker, D.: The π-calculus: a Theory of Mobile Processes. Cambridge University Press (2001)
25. Jensen, O.H., Milner, R.: Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, University of Cambridge, Computer Laboratory (2004)
26. Godskesen, J.C., Hildebrandt, T.T.: Copyability types for mobile computing resources (2003) International Workshop on Formal Methods and Security, Nanjing.

A Proof of Lemma 3

The proof of Lemma 3 follows closely [8], we provide only a few key cases.

Proof. Suppose $p, q \in \mathcal{P}_{c/\alpha}$, $p \approx^{\bullet} q$ and $p \xrightarrow{\pi} t$. We proceed by induction in the derivation of $p \xrightarrow{\pi} t$.

Case (prefix): We only consider the case $p \xrightarrow{\delta[r_1]} t$. Then $p = \delta[r_1].p_1$, $t = \{\tilde{n}\}p_1$ where $\tilde{n} = fn(r_1) \cup \delta$. By the definition of $\approx^{\bullet}$ we get $r_1 \approx^{\bullet} r_2$, $p_1 \approx^{\bullet} p_2$, and $\delta[r_2].p_2 \approx^{\circ} q$. (We may assume all processes are closed because $\approx^{\bullet}$ is substitutive.) Because $\delta[r_2].p_2 \xrightarrow{\delta[r_2]} \{\tilde{n}\}p_2$ ($fn(r_1) = fn(r_2)$ because $\approx^{\bullet}$ is well typed) and $\delta[r_2].p_2 \approx^{\circ} q$ there exists $q \xrightarrow{(\tilde{m})\delta[r]} q'$ such that for all $(x)A^{\tilde{n}'}_{\gamma}$, $(x)A^{\tilde{n}'}_{\gamma} \cdot (\emptyset, r_2, \{\tilde{n}\}p_2) \approx^{\circ} (x)A^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, r, q')$. Now, $\forall (x)A^{\tilde{n}'}_{\gamma} \approx^{\blacktriangle} (x)B^{\tilde{n}'}_{\gamma}.(x)A^{\tilde{n}'}_{\gamma} \cdot (\emptyset, r_1, t) \equiv \approx^{\bullet} \equiv (x)B^{\tilde{n}'}_{\gamma} \cdot (\emptyset, r_2, \{\tilde{n}\}p_2)$. Hence, $\forall (x)A^{\tilde{n}'}_{\gamma} \approx^{\blacktriangle} (x)B^{\tilde{n}'}_{\gamma}.(x)A^{\tilde{n}'}_{\gamma} \cdot (\emptyset, r_1, t) \equiv \approx^{\blacktriangle} (x)B^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, r, q')$ because $\equiv \subseteq \approx$ and $\approx^{\bullet} \approx^{\circ} \subseteq \approx^{\bullet}$.

Case (sync): Assume $p \xrightarrow{\tau} t = (\tilde{n})(p'_1 \parallel p'_2[u/x])$ because $p = p_1 \parallel p_2$ where $p_1 \xrightarrow{(\tilde{n})\overline{\varphi}\langle\!\langle u \rangle\!\rangle} p'_1, p_2 \xrightarrow{\varphi(x)} p'_2$, and $fn(p_2) \cap \tilde{n} = \emptyset$. By the definition of $\approx^{\bullet}$ we get $p_1 \approx^{\bullet} t_1$, $p_2 \approx^{\bullet} t_2$, and $t_1 \parallel t_2 \approx^{\circ} q$. By induction we get $\exists t_1 \stackrel{(\tilde{m})\overline{\varphi}\langle\!\langle u' \rangle\!\rangle}{\Longrightarrow} t'_1$ such that $\forall (y)A^{\tilde{n}'}_{\gamma} \approx^{\blacktriangle} (y)B^{\tilde{n}'}_{\gamma}.(y)A^{\tilde{n}'}_{\gamma} \cdot (\tilde{n}, u, p'_1) \equiv \approx^{\bullet} (y)B^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, u', t'_1)$. Also, by induction we get $\exists t_2 \stackrel{\varphi(x)}{\Longrightarrow} t'_2$ such that $p'_2 \approx^{\bullet} t'_2$. By use of (sync) (and (par) and (sym) to handle the τ steps of the weak transitions) of Table 3 (and using α conversion to get $(fn(t_2) \cap \tilde{m}) = \emptyset$ if necessary), we then get $t_1 \parallel t_2 \stackrel{\tau}{\Longrightarrow} (m)(t'_1 \parallel t'_2[u'/x])$. Since $t_1 \parallel t_2 \approx^{\circ} q$ it then follows that $\exists q \stackrel{\tau}{\Longrightarrow} q'$ such that $(\tilde{m})(t'_1 \parallel t'_2[u'/x]) \approx^{\circ} q'$.

Letting $(x)A_{\epsilon} = (-) \parallel p'_2$ and $(x)B_{\epsilon} = (-) \parallel t'_2$ we get that $(x)A_{\epsilon} \approx^{\blacktriangle} (x)B_{\epsilon}$. It then follows that $t = (\tilde{n})(p'_1 \parallel p'_2[u/x]) \equiv \approx^{\bullet} (\tilde{m})(t'_1 \parallel t'_2[u'/x]) \approx^{\circ} q'$. Finally, since $\approx^{\bullet} \approx^{\circ} \subseteq \approx^{\bullet}$ we get $t \equiv \approx^{\bullet} q'$ as required.

Case (nesting): Assume $p \xrightarrow{\pi} t = \delta[t'_1] \,.\, p_1$ because $p = \delta[t_1] \,.\, p_1$, $\pi = \delta \cdot \pi'$, $t_1 \xrightarrow{\pi'} t'_1$, $fn(p_1) \cap bn(\pi') = \emptyset$. We only consider the case where $\pi' = (\tilde{n})\delta'[v]$. By the definition of $\approx^{\bullet}$ we get $t_1 \approx^{\bullet} t_2$, $p_1 \approx^{\bullet} p_2$ and $\delta[t_2] \,.\, p_2 \approx^{\circ} q$. Hence, by induction we get $\exists t_2 \stackrel{\pi''}{\Longrightarrow} t'_2$ such that $\pi'' = (\tilde{m})\delta'[v']$ and $\forall (x)A^{\tilde{n}'}_{\gamma} \approx^{\blacktriangle} (x)B^{\tilde{n}'}_{\gamma}.(x)A^{\tilde{n}'}_{\gamma} \cdot (\tilde{n}, v, t'_1) \equiv \approx^{\bullet} (x)B^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, v', t'_2)$.

By (nesting) of Table 3 (possibly repeatedly to handle the τ steps of the weak transition) we then get $\delta[t_2] \,.\, p_2 \stackrel{\delta \cdot \pi''}{\Longrightarrow} \delta[t'_2] \,.\, p_2$ (using α-conversion if needed to ensure $\tilde{m} \cap fn(p_2) = \emptyset$). From $\delta[t_2] \,.\, p_2 \approx^{\circ} q$ we then get $q \stackrel{\delta \cdot \pi'''}{\Longrightarrow} q'$ such that $\pi''' = (\tilde{m}')\delta'[v'']$ and $\forall (x)B^{\tilde{n}'}_{\gamma}.(x)B^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, v', \delta[t'_2] \,.\, p_2) \approx^{\circ} (x)B_{\gamma} \cdot (\tilde{m}', v', q')$.

Now, since for all $(x)A^{\tilde{n}'}_{\gamma} \approx^{\blacktriangle} (x)B^{\tilde{n}'}_{\gamma}$ we have $(x)A^{\tilde{n}'}_{\gamma}(\delta[(-)] \,.\, p_1) \approx^{\blacktriangle} (x)B^{\tilde{n}'}_{\gamma}(\delta[(-)] \,.\, p_2)$ due to Prop. 8 we obtain $(x)A^{\tilde{n}'}_{\gamma}(\delta[(-)] \,.\, p_1) \cdot (\tilde{n}, v, t'_1) \equiv \approx^{\bullet} (x)B^{\tilde{n}'}_{\gamma}(\delta[(-)] \,.\, p_2) \cdot (\tilde{m}, v', t'_2)$. Moreover, $(x)A^{\tilde{n}'}_{\gamma} \cdot (\tilde{n}, v, t) \equiv (x)A^{\tilde{n}'}_{\gamma}(\delta[(-)] \,.\, p_1) \cdot (\tilde{n}, v, t'_1)$ and $(x)B^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, v', \delta[t'_2] \,.\, p_2) \equiv (x)B^{\tilde{n}'}_{\gamma}(\delta[(-)] \,.\, p_2) \cdot (\tilde{m}, v', t'_2)$ so $(x)A^{\tilde{n}'}_{\gamma} \cdot (\tilde{n}, v, t) \equiv \approx^{\bullet} \equiv (x)B^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, v', \delta[t'_2] \,.\, p_2)$. Then since $(x)B^{\tilde{n}'}_{\gamma} \cdot (\tilde{m}, v', \delta[t'_2] \,.\, p_2) \approx^{\circ} (x)B_{\gamma} \cdot (\tilde{m}', v', q')$ we have $(x)A^{\tilde{n}'}_{\gamma} \cdot (\tilde{n}, v, t) \equiv \approx^{\bullet} \equiv \approx^{\circ} (x)B_{\gamma} \cdot (\tilde{m}', v', q')$. Finally, since $\approx^{\bullet} \approx^{\circ} \subseteq \approx^{\bullet}$ and $\equiv \subseteq \approx^{\circ}$ we get $(x)A^{\tilde{n}'}_{\gamma} \cdot (\tilde{n}, v, t) \equiv \approx^{\bullet} (x)B_{\gamma} \cdot (\tilde{m}', v', q')$ as required.

B Proof of Theorem 2

Lemma 4. *$((\sim^\bullet)_c)^*$ and $((\approx^\bullet)_c)^*$ are respectively late and weak late context bisimulations.*

Proof. We only show that $((\approx^\bullet)_c)^*$ is a weak late context bisimulation. The proof of $((\sim^\bullet)_c)^*$ being a late context bisimulation is similar. It is enough to show that $((\approx^\bullet)_c)^*$ is a weak late context simulation because due to Proposition 7 $(\approx^\bullet)^*$ is symmetric and hence so is $((\approx^\bullet)_c)^*$.

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

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

Step: $k > 0$. Let

$$p = p_0\ (\approx^\bullet)_c\ p_1 \ldots p_{k-1}\ (\approx^\bullet)_c\ p_k = q$$

and suppose $p \xrightarrow{\pi} p'$.

Case: $\pi = (\tilde{n})\delta[r]$. By induction, there exists $p_{k-1} \overset{(\tilde{n}_{k-1})\delta[r_{k-1}]}{\Longrightarrow} p'_{k-1}$ such that for all $(x)A^{\tilde{n}'}_\gamma$

$$(x)A^{\tilde{n}'}_\gamma \cdot (\tilde{n}, r, p')\ ((\approx^\bullet)_c)^*\ (x)A^{\tilde{n}'}_\gamma \cdot (\tilde{n}_{k-1}, r_{k-1}, p'_{k-1})$$

By repeated applications of Lemma 3 using the fact that $\equiv$ is transitive, a late context bisimulation, and $\equiv \subseteq \approx^\circ$ and that $\approx^\blacktriangle$ is reflexive we obtain the existence of some $q \overset{(\tilde{m})\delta[r']}{\Longrightarrow} q'$ such that for all $(x)A^{\tilde{n}'}_\gamma$

$$(x)A^{\tilde{n}'}_\gamma \cdot (\tilde{n_k}, r_k, p_k) \equiv (\approx^\bullet)_c\ (x)A^{\tilde{n}'}_\gamma \cdot (\tilde{m}, r', q')$$

and hence for every $(x)A^{\tilde{n}'}_\gamma$

$$(x)A^{\tilde{n}'}_\gamma \cdot (\tilde{n}, u, t)\ ((\approx^\bullet)_c)^*\ (x)A^{\tilde{n}'}_\gamma \cdot (\tilde{m}, u', t')$$

because $\equiv \subseteq \approx^\circ \subseteq \approx^\bullet$.

Case: $\pi = \tau$, $\pi = \varphi(x)$, and $\pi = \overline{\delta}\langle r\rangle$. Similar to the case above.

Proposition 9. *$(((\sim^\bullet)_c)^*)^\circ \subseteq \sim^\circ$ and $(((\approx^\bullet)_c)^*)^\circ \subseteq \approx^\circ$.*

Proof. It is sufficient to prove that $((\sim^\bullet)_c)^* \subseteq \sim$ because $(\cdot)^\circ$ is monotone. Since $\sim$ is the largest late context bisimulation it is enough to show that $((\sim^\bullet)_c)^*$ is a late context bisimulation. We refer the reader to Lemma 4. The proof is similar in case of $(((\approx^\bullet)_c)^*)^\circ \subseteq \approx^\circ$.

The actual proof of Theorem 2 goes as follows:

Proof. We only show that $\approx^\circ$ is a congruence, the proof of $\sim^\circ$ being a congruence is similar. Because $\approx^\bullet$ is a congruence due to Proposition 7 it suffices to show that $\approx^\circ = \approx^\bullet$. From Proposition 7 we have $\approx^\circ \subseteq \approx^\bullet$. $\approx^\bullet \subseteq \approx^\circ$ follows from Proposition 9 since, due to monotonicity of $(\cdot)^\circ$ and $(\cdot)_c$, $\approx^\bullet \subseteq ((\approx^\bullet)_c)^\circ$ and $((\approx^\bullet)_c)^\circ \subseteq (((\approx^\bullet)_c)^*)^\circ$.