A CPS Encoding of Name-Passing in Higher-Order Mobile Embedded Resources

Mikkel Bundgaard Thomas Hildebrandt Jens Chr. Godskesen

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

Abstract

We present an encoding of the synchronous π-calculus in the calculus of Higher-Order Mobile Embedded Resources (Homer), a pure higher-order calculus with mobile processes in nested locations, defined as a simple, conservative extension of the core process-passing subset of Thomsen's Plain CHOCS. We prove that our encoding is fully abstract with respect to barbed bisimulation and sound with respect to barbed congruence. Our encoding demonstrates that higher-order process-passing together with mobile resources in (local) named locations are sufficient to express π-calculus name-passing. The encoding uses a novel continuation passing style to facilitate the encoding of synchronous communication.

Key words: π-calculus, Name-passing encoding, Process-passing, Nested locations, Continuation-passing, Explicit Substitutions

1 Introduction

The π-calculus [8,7] is, by most people, considered the classic process calculus for modelling mobile systems. Its most prominent features, compared to its predecessor CCS, are the communication of names as expressed by the reduction rule

$$n(m) \,.\, P \mid \overline{n}\langle o\rangle \,.\, Q \rightarrow_{\pi} \{o/m\}P \mid Q$$

and the creation of local names with static scope. Combined these concepts provide the π-calculus with most of its expressive power. Notably, by representing the location of a process by its links, the ability to dynamically change the communication links between processes makes it possible to model mobile computing processes.

This account of mobility has been very successful for a decade, but it has its limitations. Recently, a number of calculi have been proposed, e.g. the Ambient calculus [1] and the Seal calculus [2], with an explicit representation

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

of mobile computing resources in nested locations which is not easy to model in the π-calculus. Many of the proposed calculi include the name-passing capability of the π-calculus as well, which increases the complexity of the calculi. A natural question is if name-passing can be expressed using mobile computing resources in nested locations alone.

In this paper we present a compositional encoding of the synchronous π-calculus, and thus name passing, in a pure higher-order calculus with nested locations, obtained as a simple, conservative extension of the core process-passing subset of Thomsen's Plain CHOCS [11]. Thomsen demonstrated that the π-calculus could be encoded in Plain CHOCS by making crucial use of explicit name substitution to encode the dynamic linking. The calculus is a subcalculus of the Homer calculus of Higher-Order Mobile Embedded Resources presented in [5]. The Homer calculus does *not* have explicit name substitution. Thus the encoding of [11] cannot be applied in Homer. Homer introduce mobile computing resources in (local) named locations, and our encoding demonstrates that this, together with higher-order process-passing is sufficient to express π-calculus name-passing without relying on name substitution. The encoding uses a novel continuation passing style to facilitate the encoding of synchronous communication.

To briefly recall, mobility of processes in Plain CHOCS is introduced by replacing the *name*-passing of the π-calculus with *process*-passing. We will represent this kind of interaction with the prefixes $\overline{n}\langle q\rangle$ (*send*) and $n(x)$ (*receive*), respectively. Here x is a *process variable* for which the received process is substituted, as expressed formally by the reduction rule

$$\overline{n}\langle q\rangle \,.\, p_1 \parallel n(x) \,.\, p_2 \searrow p_1 \parallel p_2[q/x] \quad . \tag{1}$$

As usual, there may be any number of occurrences of x in p_2 meaning that processes may both be discarded and copied, making Plain CHOCS a *non-linear* higher-order calculus. However, as also remarked by Thomsen, the process q cannot start computing before it is moved, and once it has started computing, it can not be moved again. This is known as *code* mobility or *weak* mobility, as opposed to *process* mobility or *strong* mobility, where processes may move *during* their computation.

In Homer strongly mobile computing resources in nested locations are introduced by allowing an *additional* kind of interaction, given by two new complementary prefixes $n\langle q\rangle$ (*resource*) and $\overline{n}(x)$ (*take*). The process $n\langle q\rangle \,.\, p_1$ denotes a resource q residing at the location (or address) n which may be *moved* or *taken* by the complementary action prefix, $\overline{n}(x) \,.\, p_2$. Just as for the previous interaction the synchronisation is expressed by the reduction

$$n\langle q\rangle \,.\, p_1 \parallel \overline{n}(x) \,.\, p_2 \searrow p_1 \parallel p_2[q/x] \quad . \tag{2}$$

The important difference between the two types of interactions is that the resource q in $n\langle q\rangle$ is able to interact with processes outside its location by

allowing resources to be send down to and taken up from q. In other words, the state of q may be modified by processes outside the location n. We introduce this kind of interaction, as in the Mobile Resources (MR) calculus [4], by allowing *sequences* as addresses in the downward prefixes *take* and *send*. For instance, using the sequence $n_1 n_2$ in the address of the take prefix, a resource q may be taken from the address n_2 in a resource running at address n_1, as in

$$n_1\langle n_2\langle q\rangle \,.\, q' \parallel q''\rangle \,.\, p_1 \parallel \overline{n_1 n_2}(x) \,.\, p_2 \searrow n_1\langle q' \parallel q''\rangle \,.\, p_1 \parallel p_2[q/x] \ .$$

We allow sequences of names in addresses of the *receive* and *resource* prefixes as well. This permits the physical nested structure of the address space to be different from the abstract structure

$$\overline{n_1 n_2}\langle q\rangle \,.\, p_1 \parallel n_1 n_2(x) \,.\, p_2 \searrow p_1 \parallel p_2[q/x] \ .$$

To summarise, the two dual kinds of process movement allow us to express mobile resources in a nested location structure that may be moved (and copied) locally or upwards, and to send passive resources that may be received (and copied) by a local process or a sub-resource.

The interaction presented above is the only kind of interaction we need for the results in the present paper. The full Homer calculus [5] also allows for mobile resources that can make internal reactions. This however requires a more careful treatment of free names in the semantics. The only other feature is that of local names as found in the π-calculus and Plain CHOCS. We let $(n)p$ denote a process p in which the name n is local.

Related Work

Thomsen demonstrated that the recursion and the name-passing of the π-calculus can be encoded in Plain CHOCS [11] by passing wires instead of names. An *a-wire* representing the π-calculus name a is defined as

$$i? \,.\, a?x \,.\, c!x \,.\, nil + o? \,.\, c?x \,.\, a!x \,.\, nil \ ,$$

where i and o are used to indicate whether the wire is used for input or output, and c is used as an auxiliary forwarder. Thomsen used an encoding scheme in two levels, resembling the encoding presented in this paper: a structurally defined encoding translating free names and names bound by an input prefix into process variables, and names bound by restriction into wires, and on top-level, an instantiation of the process variables representing free names with wires.

The most complex part of the encoding is the encoding of prefixes defined

as

$$[\![x(y)\,.\,P]\!]_1 = \big(x[c \;\mapsto c'][i \mapsto i'][o \mapsto o'] \mid i'!.c'?y.[\![P]\!]_1\big)\backslash c'\backslash i'\backslash o'$$
$$[\![\overline{x}\langle y\rangle\,.\,P]\!]_1 = \big(x[c \;\mapsto c'][i \mapsto i'][o \mapsto o'] \mid o'!.c'!y.[\![P]\!]_1\big)\backslash c'\backslash i'\backslash o' \;\;,$$

where $(x[c \;\mapsto c'][i \mapsto i'][o \mapsto o'] \mid \ldots)\backslash c'\backslash i'\backslash o'$ uses a explicit substitution $[c \;\mapsto c'][i \mapsto i'][o \mapsto o']$ to localise the wire input for x. This localisation is essential for the encoding, since the main problem of encoding a first-order calculi, like the π-calculus, is dynamic linking (the name substitution). The encoding presented in this paper differs in a crucial way from the encoding by Thomsen, since Homer does not include the explicit name substitution of Plain CHOCS. Instead we obtain dynamic linking from the possibility to communicate with strongly mobile resources in (local) locations.

Zimmer presented in [12] an encoding of the synchronous π-calculus into a restricted Ambient calculus containing only the mobility primitives and the hierarchical structure of the ambients, and therefore neither communication nor name substitution. Much like the presentation in this paper, Zimmer designed an intermediate calculus π_{esc} (π-calculus with Explicit Substitutions and Channels), but whereas π_{esc} has explicit variables and channels as part of the syntax, we follow the path of the $\pi\xi$ calculus [3], also a π-calculus with explicit substitutions, and consider our processes with respect to *one* global environment. Another clear difference between the approach of Zimmer and ours is that the mobility of Mobile ambients is subjective, and the mobility in Homer is objective.

The Seal calculus [2] is a calculus with name-passing, non-linear process-passing and named, nested locations as Homer. But it is not immediately clear, though, if one can reduce name-passing to processes-passing in Seal, since scope extension for mobile processes in the Seal semantics depends on name-passing.

The connection between first-order and higher-order calculi has been examined in several contexts, most notably in [9], where Sangiorgi shows how Higher-Order π-calculus, containing both first- and higher-order communication primitives, can be represented in first-order π-calculus.

We prove the correspondence between the π-calculus and its encoding in Homer following the same approach as [10], by proving an operational correspondence between π-calculus processes and their encoding in Homer, from which we can infer full abstractness with respect to barbed bisimulation and soundness with respect to barbed congruence.

Outline

In Section 2 we present the syntax and reduction semantics of Homer. In Section 3 we do the same for the monadic synchronous π-calculus and introduce a π-calculus with explicit substitutions. We present the encoding and give an

example of the encoding in Section 4. In Section 5 we prove the operational correspondence between π-calculus processes and their encoding in Homer, full abstractness with respect to barbed bisimulation, and soundness with respect to barbed congruence.

2 Homer

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 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:

$$p, q, r ::= \mathbf{0} \quad | \quad p \parallel q \quad | \quad (n)p \quad | \quad \lambda \,.\, p \quad | \quad x$$

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

$$\begin{aligned} \lambda ::= \quad & | \quad \delta(x) \text{ receive a resource at } \delta \text{ and bind it to } x \\ & | \quad \overline{\delta}(x) \text{ take resource from } \delta \text{ and bind it to } x \\ & | \quad \overline{\delta}\langle r \rangle \text{ send a resource } r \text{ to } \delta \\ & | \quad \delta\langle r \rangle \text{ resource } r \text{ at } \delta \end{aligned}$$

The process constructors are the standard constructors from concurrent process calculi, extended with process variables as in the λ-calculus and Plain CHOCS. For an introduction to Homer, its semantics, and examples, see [5].

The prefixes $\overline{\delta}\langle r \rangle$ and $\delta(x)$ correspond to the send and receive prefixes in CHOCS, except from paths and not only names being allowed as addresses. The new prefixes allowing strong mobility are the prefixes $\delta\langle r \rangle$ and $\overline{\delta}(x)$. We define $\overline{\overline{\delta}} = \delta$, and let the restriction operator (n) bind the name n and the prefixes $\delta(x)$ and $\overline{\delta}(x)$ bind the variable x. The sets $fn(\lambda)$, $fn(p)$ and $fv(\lambda)$, $fv(p)$ of *free names* and *free variables* are defined accordingly as usual.

We say that a process with no free variables is *closed* and let $\mathcal{P}_c$ denote the set of closed processes. For any subset $\mathcal{P}'$ of $\mathcal{P}$ we let $\mathcal{P}'_{/\alpha}$ denote the set of α-equivalence classes (with respect to both names and variables) of process expressions. 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 $\lambda \,.\, \mathbf{0}$. 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 $(n_1) \cdots (n_k)p$. We write $\tilde{m}\tilde{n}$ for $\tilde{m} \cup \tilde{n}$, always implicitly assuming $\tilde{m} \cap \tilde{n} = \emptyset$. Finally, we will write n for the singleton set $\{n\}$ when no confusion can occur.

2.1 Reductions

We provide Homer with a reduction semantics defined through the use of contexts, structural congruence, and reduction rules.

Structural congruence $\equiv$ is the least congruence on $\mathcal{P}_{/\alpha}$ satisfying the following rules.

E_1. $p \parallel \mathbf{0} \equiv p$ $\quad$ E_2. $(n)\mathbf{0} \equiv \mathbf{0}$

E_3. $p \parallel q \equiv q \parallel p$ $\quad$ E_4. $(n)(m)p \equiv (m)(n)p$

E_5. $(p \parallel p') \parallel p'' \equiv p \parallel (p' \parallel p'')$ $\quad$ E_6. $(n)p \parallel q \equiv (n)(p \parallel q)$, if $n \notin fn(q)$

Contexts $\mathcal{C}$ are, as usual, terms with a hole $(-)$, and we write $\mathcal{C}(p)$ for the insertion of p in the hole of context $\mathcal{C}$. Note that free names (and variables) of p may get bound by insertion of p in the hole of a context. We define $fn(\mathcal{C})$ as $fn(\mathcal{C}(\mathbf{0}))$ and $fv(\mathcal{C})$ as $fv(\mathcal{C}(\mathbf{0}))$. *Evaluation contexts* $\mathcal{E}$ are contexts with no free variables, and whose hole is not guarded by a prefix, nor appear as the object of a send or resource prefix, i.e.

$$\mathcal{E} ::= (-) \quad | \quad \mathcal{E} \parallel p \quad | \quad p \parallel \mathcal{E} \quad | \quad (n)\mathcal{E} \ , \text{ for } p \in \mathcal{P}_c.$$

As our calculus allows actions involving terms at depths arbitrarily far apart, we define a family of *path contexts* $\mathcal{C}^{\tilde{n}}_{\gamma}$ indexed by a path address $\gamma \in \mathcal{N}^*$ and a set of names $\tilde{n}$. The path address γ indicates the path under which the hole of the context is found, and the set $\tilde{n}$ indicates the bound names of the hole. We define the path contexts inductively in $\tilde{n}$ and γ by $\mathcal{C}^{\emptyset}_{\epsilon} ::= (-)$ and whenever $p, q \in \mathcal{P}_c$,

$$\mathcal{C}^{\tilde{m}\tilde{n}}_{\gamma\delta} ::= \mathcal{C}^{\tilde{m}}_{\gamma}(\delta\langle(\tilde{n})(- \parallel p)\rangle \,.\, q), \text{where } \tilde{m} \cap \delta = \emptyset.$$

The side condition ensures that none of the names in the path address are bound. We need to handle scope extension when resources are taken up from sub-locations. For this purpose we define an *open* operator on path contexts $\mathcal{C}^{\tilde{n}}_{\gamma}\backslash\tilde{m}$ inductively by: $(-)\backslash\tilde{m} = (-)$ and

$$\mathcal{C}^{\tilde{n}'}_{\gamma}(\delta\langle(\tilde{n})(- \parallel p)\rangle \,.\, q)\backslash\tilde{m} = \mathcal{C}^{\tilde{n}'}_{\gamma}\backslash\tilde{m}(\delta\langle(\tilde{n}\backslash\tilde{m})(- \parallel p)\rangle \,.\, q) \ ,$$

if $\tilde{m} \cap (\delta \cup \tilde{n}' \cup fn(\mathcal{C}^{\tilde{n}'}_{\gamma}) \cup fn(q)) = \emptyset$. The side condition ensures that the opened names do not equal any names outside the scope. When applied in the reduction rule, this 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 the top level.

We finally define $\searrow$ as the least binary relation on $\mathcal{P}_{c/\alpha}$ satisfying the following rules and closed under all evaluation contexts $\mathcal{E}$ and structural con-

gruence.

$$(send) \quad \overline{\gamma\delta}\langle r\rangle \,.\, q \parallel \mathcal{C}^{\tilde{m}}_{\gamma}(\delta(x) \,.\, p) \searrow q \parallel \mathcal{C}^{\tilde{m}}_{\gamma}(p[r/x]) \ , \text{ if } \tilde{m} \cap (fn(r) \cup \delta) = \emptyset$$

$$(take) \quad \mathcal{C}^{\tilde{m}}_{\gamma}(\delta\langle r\rangle.q) \parallel \overline{\gamma\delta}(x) \,.\, p \searrow (\tilde{m}')(\mathcal{C}^{\tilde{m}}_{\gamma} \backslash \tilde{m}'(q) \parallel p[r/x]) \ ,$$
$$\text{if } \tilde{m}' = fn(r) \cap \tilde{m} \text{ and } \tilde{m} \cap \delta = \emptyset$$

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. The side condition guarantees that no free names of r may be bound by the path context. This can always be guaranteed by α-conversion, and thus will never block mobility. The (*take*) rule captures that a resource r is taken from the sub-location γ, where it is running at the address δ and substituted into the process p, possibly in several copies. The open operator is used to extend the scope of the local names defined in $\mathcal{C}^{\tilde{m}}_{\gamma}$ that occur free in r. Note that for $\gamma = \epsilon$ the two rules reduce to the two reduction rules (1) and (2) given in the introduction.

We may encode general recursion in Homer (up to weak equivalence) using copyability of resources. The encoding is similar to the one in [11], except that recursion variables may appear at sublocations which makes the definition slightly more complicated. Define

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

where $rec^a x \,.\, p = \overline{a}\langle r\rangle \parallel r$ and $r = a(x) \,.\, p[(\overline{a}\langle x\rangle \parallel x)/x]$, for $a \notin fv(p)$. Intuitively, r places a copy of $rec^a x.p$ at all occurrences of x in p, i.e. $rec^a x.p \searrow p[rec^a x \,.\, p/x]$.

3 The Pi-calculus

We present the monadic synchronous π-calculus without summations and replication. We present its syntax, structural congruence relation, and the reaction rule. For a much more thorough introduction to and description of the π-calculus, see e.g. [8,7].

We will in this paper only consider a π-calculus without replication in order to make the presentation of the encoding and, in particular, the proof of the encoding succinct. But since we can encode general recursion in Homer and thereby replication, we can also encode the replication operator.

Even though some of the process constructors of Homer collide with the constructors of the π-calculus, we will nonetheless use the same symbols, since any ambiguity can easily be resolved from the context. We let N denote an infinite set of names and let m, n range over N. The set $\mathcal{P}_{\pi}$ of *process*

expressions is then defined by the grammar

$$P, Q ::= \mathbf{0} \quad | \quad P \mid Q \quad | \quad (\nu n)P \quad | \quad \overline{n}\langle m\rangle \,.\, P \quad | \quad n(m) \,.\, P$$

We consider π-calculus terms up to α-conversion and define *structural congruence* $\equiv_\pi$ in the π-calculus as for Homer. We define the reaction relation $\rightarrow_\pi$ in terms of evaluation contexts

$$\mathcal{E}_\pi ::= (-) \quad | \quad \mathcal{E}_\pi \mid P \quad | \quad P \mid \mathcal{E}_\pi \quad | \quad (\nu n)\mathcal{E}_\pi \ , \text{ for } P \in \mathcal{P}_\pi.$$

$\rightarrow_\pi$ is then the least binary relation over $\mathcal{P}_\pi$ satisfying the following rule and closed under all evaluation contexts $\mathcal{E}_\pi$ and structural congruence.

$$\text{(React)} \ \frac{}{n(m) \,.\, P \mid \overline{n}\langle m'\rangle \,.\, Q \rightarrow_\pi \{m'/m\}P \mid Q}$$

As usual, we let $\{n/m\}P$ denote the process P with all free occurrences of m replaced by n, using α-conversion to avoid that n becomes bound in P.

3.1 Pi-calculus with Explicit Substitutions

We introduce an intermediate calculus: π-calculus with *explicit substitutions* to ease the proof of our encoding and to make the intuition of our encoding clearer. The only way this π-calculus differs from traditional π-calculus is that we record the substitution occurring in a reaction in a global environment. An *environment* σ is associated with a process P, giving rise to the judgement $\sigma \vdash P$. The substitution is a partial function from names to names, and we let $\operatorname{dom}\sigma$ ($\operatorname{codom}\sigma$) denote the domain (codomain) of the function σ. Furthermore, for a judgement $\sigma \vdash P$ we require that $\operatorname{dom}\sigma \supseteq fn(P)$. We write id_A for the substitution that is the identity on A.

We let $P\sigma$ denote the process, where all free names in P have been simultaneously substituted according to σ. We let $\sigma[m \mapsto n]$ denote the substitution σ extended such that m maps to n. Finally, we let $\mathcal{P}_{\pi\sigma}$ denote the set of process judgements.

We define the reaction relation $\rightarrow_{\pi\sigma}$ as before, except with respect to the following rule, where the first side-condition always can be satisfied by α-conversion. It guarantees that local names differ from the names already present in the substitution, in particular the free names of the process.

$$\text{(React}\sigma\text{)} \ \frac{}{\sigma \vdash n(m) \,.\, P \mid \overline{n'}\langle m'\rangle \,.\, Q \rightarrow_{\pi\sigma} \sigma[m \mapsto \sigma m'] \vdash P \mid Q} \ ,$$
$$\text{if } m \notin \operatorname{dom}\sigma \cup \operatorname{codom}\ \sigma \text{ and } \sigma n = \sigma n'$$

We note that there is an operational correspondence between a traditional π-calculus term and the corresponding term with explicit substitutions.

Lemma 3.1 *If* $P = Q\sigma$ *then* $P \rightarrow_{\pi} P'$ *iff* $\sigma \vdash Q \rightarrow_{\pi\sigma} \sigma' \vdash Q'$ *such that* $P' = Q'\sigma'$.

4 The Encoding

We adopt the following shorthand for taking a copy of the computing resource at address n and placing the copy in the variable x as the prefix $n \triangleright x$, defined as

$$n \triangleright x \,.\, p =_{def} \overline{n}(x) \,.\, (n\langle x\rangle \parallel p)$$

We encode π-calculus processes with names in $\mathcal{N} \setminus \{v, c, s, r\}$ as Homer processes with names in $\mathcal{N} \uplus \mathcal{N}' \uplus \{v, c, s, r\}$, where $\mathcal{N}'$ is ranged over by n', m' and we assume the mapping of $n \in \mathcal{N}$ to $n' \in \mathcal{N}'$ is a bijection. The names $\{v, c, s, r\}$ are used as auxiliary names in the encoding [1].

We then encode a π-calculus name n as a mobile resource $[\![n]\!]$ that can perform two tasks: sending and receiving along the name n.

$$\begin{aligned} Send_n &= v(x) \,.\, c(y) \,.\, \overline{n}\langle x\rangle \,.\, y \\ Receive_n &= v(x) \,.\, c(y) \,.\, n(z) \,.\, (a)(a\langle x\rangle \parallel \overline{av}\langle z\rangle \,.\, \overline{a}(z') \,.\, (y \parallel z')) \\ [\![n]\!] &= s\langle Send_n\rangle \parallel r\langle Receive_n\rangle \end{aligned}$$

The $Send_n$ process can be seen as taking two parameters on the locations v and c, respectively. On location v it takes the encoding of the name $[\![m]\!]$ to send, and on location c the encoding of the continuation $[\![P]\!]$, resulting in a process of the following form $\overline{n}\langle[\![m]\!]\rangle \,.\, [\![P]\!]$.

$Receive_n$ also takes two parameters: on location v it takes a process $Bind_b = v(x) \,.\, b'\langle x\rangle$ responsible for binding the received name, and on location c the encoding $[\![Q]\!]$ of the continuation, resulting in a process of the following form $n(z) \,.\, (a)(a\langle Bind_b\rangle \parallel \overline{av}\langle z\rangle \,.\, \overline{a}(z') \,.\, ([\![Q]\!] \parallel z'))$. In parallel these two processes can perform a synchronisation on the location n, corresponding to the actual π-calculus synchronisation. After the synchronisation the received name $[\![m]\!]$ is sent to the $Bind_b$ process and finally placed in parallel with the continuations, resulting in $[\![P]\!] \parallel [\![Q]\!] \parallel b'\langle[\![m]\!]\rangle$.

The encoding of a π-calculus process $\sigma \vdash P$ is done at two levels: at top-level, we translate all names in σ. At the next, we give a compositional encoding of P. Let $[\![\sigma \vdash P]\!]$ denote the following

$$(\tilde{m})\big([\![P]\!] \parallel \Pi_{n \in \mathrm{dom}\,\sigma}\; n'\langle[\![\sigma n]\!]\rangle\big) \;,$$

where $\tilde{m} = \{n' \mid n \in \mathrm{dom}\,\sigma \text{ and } n \neq \sigma n\}$. Note that the encoding of a name n (or more precisely, its image under the substitution) is kept as a resource at

[1] The set $\mathcal{N}' \cup \{v, c, s, r\}$ is used for readability, one could use the same name for v and c, and exploit nested names to distinguish between the send and receive methods.

the address n'. The restriction $\tilde{m}$ restricts the locations of substituted names, which intuitively are bound to local names. Then the encoding of a π-calculus process P is defined to be $[\![id_{fn(P)} \vdash P]\!]$. The encoding $[\![-]\!]$ of local names, parallel composition, and the inactive process is given inductively as below.

$$\begin{aligned}
[\![(\nu n)P]\!] &= (n)(n')([\![P]\!] \parallel n'\langle [\![n]\!] \rangle) \\
[\![P \mid Q]\!] &= [\![P]\!] \parallel [\![Q]\!] \\
[\![\mathbf{0}]\!] &= \mathbf{0}
\end{aligned}$$

We encode the output and input prefixes of the π-calculus as follows

$$[\![\bar{n}\langle m\rangle \,.\, P]\!] = (a)(n' \triangleright x \,.\, (a\langle x\rangle \parallel m' \triangleright y \,.\, \overline{asv}\langle y\rangle \,.\, \overline{asc}\langle [\![P]\!]\rangle \,.\, \overline{as}(x'') \,.\, \overline{a}(z) \,.\, x''))$$

$$[\![n(m) \,.\, P]\!] = (a)(n' \triangleright x \,.\, (m')(a\langle x\rangle \parallel \overline{arv}\langle Bind_m\rangle \,.\, \overline{arc}\langle [\![P]\!]\rangle \,.\, \overline{ar}(x'') \,.\, \overline{a}(z) \,.\, x''))$$

where $Bind_m = v(x) \,.\, m'\langle x\rangle$. As described above, the process $Bind_m$ is responsible for binding the received name to the local name m. Intuitively, $[\![\bar{n}\langle m\rangle \,.\, P]\!]$ first takes a copy of the encoded name at location n' and keeps it at the local address a. Then a copy of the encoded name at location m' is taken which, together with the continuation $[\![P]\!]$, is sent to the *Send* part of the encoded name at a. The *Send* part is then retrieved, and finally the rest of the encoded name at location a is discarded.

$[\![n(m) \,.\, P]\!]$ is encoded using the same template, but it is a bit more complicated due to the binding. First and foremost, we restrict the location of the formal parameter of the input prefix to ensure that this is only available to the continuation, and that it can be α-converted. Secondly, we use the $Bind_m$ process to create a new location to contain the received, encoded name.

There are several relevant observations to be made from the encoding. First, our encoding of name substitution relies heavily on the ability to communicate with resources in local, named locations in Homer and our ability to copy resources. We use the ability to communicate with resources in local, named locations to localise the communication instead of using name substitution as in the CHOCS encoding. We make use of the non-linearity of Homer to take a copy of the encoding of a name each time we use it. Secondly, strongly mobile resources are utilised in several places in the encoding: in the encoding of $Receive_n$ and in the two prefixes where we send resources to local addresses and let them compute, before taking the resources up again.

Also note that there is a subtle difference between reactions of π-calculus processes and reactions in our π-calculus with explicit substitutions (and in the encoding). Reaction in the π-calculus ($P \rightarrow_\pi P'$) can reduce the set of free names, while this set is preserved in the substitutions and hence remains fixed in our encoding. Actually, the same situation occurs in Milner and Jensen's encoding of the asynchronous π-calculus (without replication and summation) as a bigraphical reactive system [6].

Finally, note how the continuation passing style of the encoding facilitates the encoding of synchronous communication.

4.1 An Example

As an example of how our encoding properly simulates reactions in the π-calculus with explicit substitutions, we look at the process

$$id_A \vdash \bar{n}\langle m\rangle \,.\, m(d) \,.\, P \mid n(f) \,.\, \bar{f}\langle e\rangle \,.\, Q \rightarrow_{\pi\sigma}$$
$$id_A[f \mapsto m] \vdash m(d) \,.\, P \mid \overline{f}\langle e\rangle \,.\, Q \rightarrow_{\pi\sigma}$$
$$id_A[f \mapsto m][d \mapsto e] \vdash P \mid Q \ ,$$

where A is the set of free names in $\bar{n}\langle m\rangle \,.\, m(d) \,.\, P \mid n(f) \,.\, \bar{f}\langle e\rangle \,.\, Q$. Letting $r = \Pi_{n\in A}\ n'\langle[\![n]\!]\rangle$, we get the following reductions

$$\begin{aligned}
&[\![\bar{n}\langle m\rangle \,.\, m(d) \,.\, P]\!] \parallel [\![n(f) \,.\, \bar{f}\langle e\rangle \,.\, Q]\!] \parallel r && \searrow^* \\
&\overline{n}\langle[\![m]\!]\rangle \,.\, [\![m(d) \,.\, P]\!] \parallel (f')(n(z) \,.\, (a)([\![\overline{f}\langle e\rangle \,.\, Q]\!]_{z/f})) \parallel r && \searrow^* \\
&(f')\big([\![m(d) \,.\, P]\!] \parallel [\![\bar{f}\langle e\rangle \,.\, Q]\!] \parallel f'\langle[\![m]\!]\rangle \parallel r\big) && \searrow^* \\
&(f')(d')\big([\![P]\!] \parallel [\![Q]\!] \parallel d'\langle[\![e]\!]\rangle \parallel f'\langle[\![m]\!]\rangle \parallel r\big) = [\![id_A[f \mapsto m][d \mapsto e] \vdash P \mid Q]\!] \ ,
\end{aligned}$$

where $[\![\overline{f}\langle e\rangle \,.\, Q]\!]_{z/f} = a\langle Bind_f\rangle \parallel \overline{av}\langle z\rangle \,.\, \overline{a}(z') \,.\, ([\![\bar{f}\langle e\rangle \,.\, Q]\!] \parallel z')$ is the part of the $Receive_n$ process that with help from $Bind_f$ binds the received name to the local name f and then runs the continuation $[\![\bar{f}\langle e\rangle \,.\, Q]\!]$.

5 Proof of Correspondence

In this section we prove the soundness of the encoding. Following the approach in [10], we first show that there is an operational correspondence between a π-calculus process $\sigma \vdash P$ and its encoding $[\![\sigma \vdash P]\!]$ in Homer.

We define strong barbs in a π-calculus term P as usual $\sigma \vdash P \downarrow \sigma n$, if $\sigma \vdash P$ can perform an input action with subject n, we define weak barbs, $\sigma \vdash P \ \Downarrow n$, if $\sigma \vdash P \rightarrow^*_{\pi\sigma}\downarrow n$. Correspondingly in Homer, we write $p \downarrow n$ if p can perform a receive action $n(m)$ and $p \ \Downarrow n$, if $p \searrow^* \downarrow n$. We then define a *matching barbed bisimilarity* between π-calculus and Homer terms.

Definition 5.1 *Matching barbed bisimilarity* is the largest relation $\dot{\approx} \subseteq \mathcal{P}_{\pi\sigma/\alpha} \times \mathcal{P}_{c/\alpha}$, if whenever $(\sigma \vdash P \,,\, q) \ \in \dot{\approx}$,

- if $\sigma \vdash P \rightarrow_{\pi\sigma} \sigma' \vdash P'$ then there exists q' such that $q \searrow^* q'$ and $\sigma' \vdash P' \dot{\approx} q'$
- if $q \searrow q'$ then there exists P' and σ' such that $\sigma \vdash P \rightarrow^*_{\pi\sigma} \sigma' \vdash P'$ and $\sigma' \vdash P' \dot{\approx} q'$
- if $\sigma \vdash P \downarrow n$ then $q \Downarrow n$

- if $q \downarrow n$ then $\sigma \vdash P \Downarrow n$

The main result of this paper is that there is a matching barbed bisimulation between the encoding and the encoded process.

Theorem 5.2 *For all π-calculus processes P and substitutions σ, we have $\sigma \vdash P \dot{\approx} [\![\sigma \vdash P]\!]$.*

Definition 5.3 *Barbed bisimilarity* in Homer is the largest symmetric relation $\dot{\approx}_H$ on $\mathcal{P}_{c/\alpha}$, such that whenever $p \dot{\approx}_H q$,

- if $p \downarrow n$ then $q \Downarrow n$
- if $p \searrow p'$, then there exists q' such that $q \searrow^* q'$ and $p' \dot{\approx}_H q'$

From Thm. 5.2 it follows that the encoding is fully abstract with respect to barbed bisimulation.

Corollary 5.4 *Let $\dot{\approx}_\pi$ denote the standard barbed bisimilarity in π-calculus, then we have $\sigma \vdash P \dot{\approx}_\pi \sigma \vdash Q$ iff $[\![\sigma \vdash P]\!] \dot{\approx}_H [\![\sigma \vdash Q]\!]$.*

From the compositionality of the encoding we can prove soundness with respect to barbed congruence.

Theorem 5.5 *Letting $\mathcal{C}_H$ denote a Homer context, $\mathcal{C}_\pi$ a π-calculus context, we have that $\forall \mathcal{C}_H.\ \mathcal{C}_H([\![\sigma \vdash P]\!]) \dot{\approx}_H \mathcal{C}_H([\![\sigma \vdash Q]\!])$ implies $\forall \mathcal{C}_\pi \forall \sigma'.\ \sigma\sigma' \vdash \mathcal{C}_\pi(P) \dot{\approx}_\pi \sigma\sigma' \vdash \mathcal{C}_\pi(Q)$, such that $\operatorname{dom}\sigma' \supseteq fn(\mathcal{C}_\pi) \setminus \operatorname{dom}\sigma$ and $\operatorname{dom}\sigma' \cap \operatorname{dom}\sigma = \emptyset$.*

6 Conclusions and Future Work

In this paper we have presented a novel encoding of the name-passing, and hence name substitution, of the π-calculus in Homer using process-passing, strongly mobile resources, named nested locations, and local names. We have used a continuation passing style to give an elegant encoding of synchronous communication. We introduced a π-calculus with explicit substitutions to maintain the set of free names under reaction, and for the purpose of making the correspondence intuitive. In this paper we described only an encoding of the finite π-calculus without matching, we will describe how to encode the full π-calculus, including replication and matching, in a forthcoming paper.

Several interesting questions arise from the work done in this paper. First and foremost, a logical next step would be to see if it is possible to encode a version of Homer extended with name-passing in Homer. It is not clear at this point how to make an encoding like this, or if it is possible at all. Secondly, it would be interesting to look for a completeness result for the encoding with respect to barbed congruence. As mentioned in [10], this is a difficult problem for synchronous calculi. A possible solution could be to use a labelled bisimulation characterisation of barbed congruence in both π-calculus and Homer. We explore labelled bisimulation congruence in Homer in [5].

References

[1] Cardelli, L. and A. D. Gordon, *Mobile ambients*, in: M. Nivat, editor, *Proceedings of the First International Conference of Foundations of Software Science and Computation Structures (FOSSACS'98)*, Lecture Notes in Computer Science **1378** (1998), pp. 140–155.

[2] Castagna, G. and F. Z. Nardelli, *The Seal calculus revisited: Contextual equivalence and bisimilarity*, in: M. Agrawal and A. Seth, editors, *Proceedings of the 22nd Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS'02)*, Lecture Notes in Computer Science **2556** (2002), pp. 85–96.

[3] Ferrari, G., U. Montanari and P. Quaglia, *A π-calculus with explicit substitutions*, Theoretical Computer Science **168** (1996), pp. 53–103.

[4] Godskesen, J. C., T. Hildebrandt and V. Sassone, *A calculus of mobile resources*, in: L. Brim, P. Jancar, M. Kretínský and A. Kucera, editors, *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR'02)*, Lecture Notes in Computer Science **2421** (2002), pp. 272–287.

[5] Hildebrandt, T., J. C. Godskesen and M. Bundgaard, *Bisimulation congruences for higher-order mobile embedded resources with local names* (2004), in preparation.

[6] Jensen, O. H. and R. Milner, *Bigraphs and mobile processes (revised)*, Technical Report UCAM-CL-TR-580, University of Cambridge, Computer Laboratory (2004).

[7] Milner, R., "Communicating and Mobile Systems: the π-calculus," Cambridge University Press, 1999.

[8] Milner, R., J. Parrow and D. Walker, *A calculus of mobile processes, parts I and II*, Journal of Information and Computation **100** (1992), pp. 1–40 and 41–77.

[9] Sangiorgi, D., "Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms," Ph.D. thesis, Department of Computer Science, University of Edinburgh (1992).

[10] Sangiorgi, D. and D. Walker, "The π-calculus," Cambridge University Press, 2001.

[11] Thomsen, B., *Plain CHOCS: A second generation calculus for higher order processes*, Acta Informatica **30** (1993), pp. 1–59.

[12] Zimmer, P., *On the expressiveness of pure mobile ambients*, in: L. Aceto and B. Victor, editors, *Proceedings of the 7th International Workshop on Expressiveness in Concurrency (EXPRESS'00)*, Electronic Notes in Theoretical Computer Science **39** (2003), pp. 81–104.

A Reductions of Prefixes

As explained in section 4, our encoding requires several communications in Homer in order to mimic a single reaction in the π-calculus. We call the step that corresponds to a reaction in the π-calculus the *reaction step*, and the additional steps *bookkeeping steps*. From the encoding we see that for an output prefix there are only bookkeeping steps before the reaction step, whereas for an input prefix there are also bookkeeping steps after the reaction step.

We index our encoding of a π-calculus prefix to capture the intuition that a π-calculus prefix corresponds to a sequence of Homer processes. We take index 0 to be the original translation of the prefix (before any bookkeeping reductions have occurred in the encoding), i.e. $[\![\overline{n}\langle m\rangle . P]\!]^0_\sigma = [\![\overline{n}\langle m\rangle . P]\!]$ and then define $[\![\overline{n}\langle m\rangle . P]\!]^k$ such that

$$[\![\overline{n}\langle m\rangle . P]\!]^0_\sigma \searrow \ldots \searrow [\![\overline{n}\langle m\rangle . P]\!]^6_\sigma = \overline{\sigma n}\langle [\![\sigma m]\!]\rangle . [\![P]\!] \ .$$

More precisely, given a substitution σ, we define $[\![\overline{n}\langle m\rangle . P]\!]^k_\sigma$, for $0 \leq k \leq 6$, as

$$\begin{aligned}
[\![\overline{n}\langle m\rangle . P]\!]^0_\sigma &= [\![\overline{n}\langle m\rangle . P]\!] \\
[\![\overline{n}\langle m\rangle . P]\!]^1_\sigma &= (a)(a\langle [\![\sigma n]\!]\rangle \parallel \underline{m' \triangleright y} . \overline{asv}\langle y\rangle . \overline{asc}\langle [\![P]\!]\rangle . \overline{as}(x'') . \overline{a}(z) . x'') \\
[\![\overline{n}\langle m\rangle . P]\!]^2_\sigma &= (a)(a\langle \underline{[\![\sigma n]\!]}\rangle \parallel \underline{\overline{asv}\langle [\![\sigma m]\!]\rangle} . \overline{asc}\langle [\![P]\!]\rangle . \overline{as}(x'') . \overline{a}(z) . x'') \\
[\![\overline{n}\langle m\rangle . P]\!]^3_\sigma &= (a)(a\Big\langle s\langle \underline{c(y)} . \overline{\sigma n}\langle [\![\sigma m]\!]\rangle . y\rangle \parallel r\langle Receive_{\sigma n}\rangle\Big\rangle \\
&\qquad \parallel \underline{\overline{asc}\langle [\![P]\!]\rangle} . \overline{as}(x'') . \overline{a}(z) . x'') \\
[\![\overline{n}\langle m\rangle . P]\!]^4_\sigma &= (a)(a\Big\langle \underline{s\langle \overline{\sigma n}\langle [\![\sigma m]\!]\rangle . [\![P]\!]\rangle} \parallel r\langle Receive_{\sigma n}\rangle\Big\rangle \parallel \underline{\overline{as}(x'')} . \overline{a}(z) . x'') \\
[\![\overline{n}\langle m\rangle . P]\!]^5_\sigma &= (a)(\underline{a\Big\langle r\langle Receive_{\sigma n}\rangle\Big\rangle} \parallel \underline{\overline{a}(z)} . \overline{\sigma n}\langle [\![\sigma m]\!]\rangle . [\![P]\!]) \\
[\![\overline{n}\langle m\rangle . P]\!]^6_\sigma &= \underline{\overline{\sigma n}\langle [\![\sigma m]\!]\rangle} . [\![P]\!] \ ,
\end{aligned}$$

where we have underlined the synchronising prefixes and emphasised some of the resource boundaries for readability. Notice that in index 0 we need a location n' in the environment, in index 1 a location m', and in index 6 we need a matching co-action in the environment in order to be able to perform the reaction.

Similarly, we define $[\![n(m)\,.\,P]\!]^k_\sigma$, for $0 \leq k \leq 5$, as

$$
\begin{aligned}
[\![n(m)\,.\,P]\!]^0_\sigma &= [\![n(m)\,.\,P]\!] \\
[\![n(m)\,.\,P]\!]^1_\sigma &= (a)(m')(a\langle \underline{[\![\sigma n]\!]} \rangle \parallel \underline{\overline{arv}\langle Bind_m \rangle}\,.\,\overline{arc}\langle [\![P]\!] \rangle\,.\,\overline{ar}(x'')\,.\,\overline{a}(z)\,.\,x'') \\
[\![n(m)\,.\,P]\!]^2_\sigma &= (a)(m')(a\Big\langle s\langle Send_{\sigma n}\rangle \parallel r\langle \underline{c(y)}\,.\,\sigma n(z)\,.\,(a)(a\langle Bind_m\rangle \parallel \\
&\qquad \overline{av}\langle z\rangle\,.\,\overline{a}(z')\,.\,(y \parallel z'))\rangle\Big\rangle \parallel \underline{\overline{arc}\langle [\![P]\!] \rangle}\,.\,\overline{ar}(x'')\,.\,\overline{a}(z)\,.\,x'') \\
[\![n(m)\,.\,P]\!]^3_\sigma &= (a)(m')(a\Big\langle s\langle Send_{\sigma n}\rangle \parallel \\
&\qquad \underline{r\langle \sigma n(z)\,.\,(a)(a\langle Bind_m\rangle \parallel \overline{av}\langle z\rangle\,.\,\overline{a}(z')\,.\,([\![P]\!] \parallel z'))\rangle}\Big\rangle \parallel \\
&\qquad \underline{\overline{ar}(x'')}\,.\,\overline{a}(z)\,.\,x'') \\
[\![n(m)\,.\,P]\!]^4_\sigma &= (a)(m')(\underline{a\Big\langle s\langle Send_{\sigma n}\rangle\Big\rangle} \parallel \underline{\overline{a}(z)}\,.\,\sigma n(z)\,.\,(a)(a\langle Bind_m\rangle \parallel \\
&\qquad \overline{av}\langle z\rangle\,.\,\overline{a}(z')\,.\,([\![P]\!] \parallel z'))) \\
[\![n(m)\,.\,P]\!]^5_\sigma &= (m')(\underline{\sigma n(z)}\,.\,(a)([\![P]\!]_{z/m})) \\
[\![P]\!]^0_{e/m} &= (m')(a)(a\langle \underline{Bind_m}\rangle \parallel \underline{\overline{av}\langle [\![e]\!] \rangle}\,.\,\overline{a}(z')\,.\,([\![P]\!] \parallel z')) \\
[\![P]\!]^1_{e/m} &= (m')(a)(\underline{a\langle m'\langle [\![e]\!]\rangle\rangle} \parallel \underline{\overline{a}(z')}\,.\,([\![P]\!] \parallel z'))\ ,
\end{aligned}
$$

where $[\![P]\!]_{z/m} = a\langle Bind_m\rangle \parallel \overline{av}\langle z\rangle\,.\,\overline{a}(z')\,.\,([\![P]\!] \parallel z')$. Again, notice that in index 0 we need a location n' in the environment, and in index 5 we need a matching co-action in the surrounding environment. $[\![P]\!]^0_{e/m}$ and $[\![P]\!]^1_{e/m}$ represent the binding steps (to the local name m) after a name $[\![e]\!]$ has been input.

B Proof of Correspondence

We consider π-calculus processes in a (pre) normal form in order to ease the proof.

Definition B.1 A *normal form* for a π-calculus process P is defined as follows

$$P = (\nu\tilde{n})\big(I_1 \mid \cdots \mid I_k \mid O_1 \mid \cdots \mid O_m\big)\ ,$$

where each I_i is on the form $x(y)\,.\,P_i$ and each O_i is on the form $\overline{x}\langle y\rangle\,.\,P'_i$, both for some x, y, and where $P_1, \ldots, P_k$ and $P'_1, \ldots, P'_m$ are in normal form.

Proposition B.2 *Every π-calculus term P is structurally congruent to a term P' in normal form.*

Definition B.3 A *prenormal form* for a π-calculus process P is defined as follows

$$P = (\nu\tilde{n})\big(I_1 \mid \cdots \mid I_k \mid O_1 \mid \cdots \mid O_m \mid P_1 \mid \cdots \mid P_l\big)\ , \qquad (\star)$$

where each I_i is on the form $x(y) \,.\, P_i'$, and each O_i is on the form $\overline{x}\langle y\rangle \,.\, P_i''$, for some names x, y, and where $P_1', \ldots, P_k'$ and $P_1'', \ldots, P_m''$ are in normal form and $P_1, \ldots, P_l$ are in normal form.

It is easy to prove that if P is a π-calculus process on prenormal form and $\sigma \vdash P \;\rightarrow_{\pi\sigma} P'$, then there exists a $P'' \equiv_\pi P'$, and P'' is on prenormal form. Hence, we need only to consider processes on prenormal form, both in π-calculus with explicit substitutions and their encodings in Homer, and we do not lose any behaviour because of this.

The processes $P_1, \ldots, P_l$ will correspond to input prefixes in the encoding, which have received a value, but have not completed the following bookkeeping. In more detail, we translate processes on prenormal form ($\star$) into the following Homer processes

$$\begin{aligned}[\![\sigma \vdash P]\!]^{(\overline{i},\overline{o},\overline{p},\overline{e},\overline{m})} = (\tilde{m})\big((\tilde{n}')\big([\![I_1]\!]_\sigma^{i_1} \parallel \cdots \parallel [\![I_k]\!]_\sigma^{i_k} \parallel [\![O_1]\!]_\sigma^{o_1} \parallel \cdots \parallel [\![O_m]\!]_\sigma^{o_m} \\ \parallel [\![P_1]\!]^{p_1}_{e_1/m_1} \parallel \cdots \parallel [\![P_l]\!]^{p_l}_{e_l/m_l} \parallel \Pi_{n\in\tilde{n}}\; n'\langle [\![n]\!]\rangle\big) \parallel r\big) \;,\end{aligned} \tag{$\star\star$}$$

where $\tilde{m} = \{m' \mid m \in \operatorname{dom}\sigma \text{ and } m \neq \sigma m\}$, $\tilde{n}' = \{n, n' \mid n \in \tilde{n}\}$ and $r = \Pi_{n\in\operatorname{dom}\sigma}\; n'\langle [\![\sigma n]\!]\rangle$. We introduced the notation $[\![-]\!]^{(\overline{i},\overline{o},\overline{p},\overline{e},\overline{m})}$, where

- $\overline{i}$ is the list of indexes of the input prefixes $\langle i_1, \ldots, i_k\rangle$ (e.g. for each i_j, $0 \leq i_j \leq 6$),
- $\overline{o}$ is the list of indexes of the output prefixes $\langle o_1, \ldots, o_m\rangle$ (e.g. for each o_j, $0 \leq o_j \;\leq 5$),
- $\overline{p}$ is the list of indexes of the preforms $\langle p_1, \ldots, p_l\rangle$ (e.g. for each p_j, $0 \leq p_j \leq 1$), and the lists $\overline{e}$ and $\overline{m}$ are lists of names, consisting of the names received in the input and the local names, respectively. Notice that the length of the lists $\overline{p}$, $\overline{e}$, and $\overline{m}$ must be the same, l.

Proof of Theorem 5.2 (Sketch) Define a relation $\mathcal{R}$, such that for all π-calculus processes P on the form of ($\star$), all substitutions σ, and for all possible lists $\overline{i}, \overline{o}, \overline{p}, \overline{e}, \overline{m}$ (with respect to P)

$$\sigma[\overline{m} \;\mapsto \overline{e}] \vdash P \;\; \mathcal{R} \;\; [\![\sigma \vdash P]\!]^{(\overline{i},\overline{o},\overline{p},\overline{e},\overline{m})} \;.$$

We show that $\mathcal{R}$ is a matching barbed bisimulation. We prove each of the four conditions of Definition 5.1 separately. In the following cases assume that we have taken an arbitrary pair from $\mathcal{R}$ (e.g. $\sigma[\overline{m} \;\mapsto \overline{e}] \vdash P \;\; \mathcal{R} \;\; [\![\sigma \vdash P]\!]^{(\overline{i},\overline{o},\overline{p},\overline{e},\overline{m})}$).
First condition: There are four possible cases to consider here. The only type of synchronisation that can occur is between an output and an input prefix, and either of these can be in preform. We consider only one of the cases where neither of the prefixes is in preform.

- We have an input prefix I_i, $1 \leq i \leq k$, of the form $n(m) \,.\, P'$ and an output prefix O_j, $1 \leq j \leq m$, of the form $\overline{h}\langle f\rangle \,.\, Q'$. Without loss of generality, assume that the form of Q' is $(\nu\tilde{n_Q})\big(I_1^Q \mid \cdots \mid I_{k'}^Q \mid O_1^Q \mid \cdots \mid O_{m'}^Q\big)$ and the

form of P' is $(\nu \tilde{n_P})\big(I_1^P \mid \cdots \mid I_{k''}^P \mid O_1^P \mid \cdots \mid O_{m''}^P\big)$, and that $\sigma n = \sigma h, \sigma f = g$. After the synchronisation, we have the following expression on prenormal form

$$\begin{aligned}\sigma[\overline{m} \mapsto \overline{e}][m \mapsto g] \ \vdash (\tilde{n} n \tilde{n_P} \tilde{n_Q})\big(&I_1 \mid \cdots \mid I_{i-1} \mid I_{i+1} \mid \cdots \mid I_k \mid I_1^Q \mid \cdots \mid I_{k'}^Q \\ &\mid I_1^P \mid \cdots \mid I_{k''}^P \mid O_1 \mid \cdots \mid O_{j-1} \mid O_{j+1} \mid \cdots \mid O_m \mid O_1^P \mid \cdots \mid O_{m'}^P \\ &\mid O_1^Q \mid \cdots \mid O_{m''}^Q \mid P_1 \mid \cdots \mid P_l\big) \ .\end{aligned} \tag{B.1}$$

In Homer, our output prefix is on the form $[\![\overline{h}\langle f\rangle \,.\, Q']\!]_\sigma^{i'}$, so after $6 - i'$ reductions we have $[\![\overline{h}\langle f\rangle \,.\, Q']\!]_\sigma^6 = \overline{\sigma h}\langle [\![\sigma f]\!]\rangle \,.\, [\![Q']\!]$. For our input prefix we have $[\![n(m) \,.\, P']\!]_\sigma^{j'}$, so after $5 - j'$ reductions we have $[\![n(m) \,.\, P']\!]_\sigma^5 = (m')(\sigma n(z) \,.\, (a)([\![P']\!]_{z/m}))$, and hence we have the reductions

$$\begin{aligned}&[\![\overline{h}\langle f\rangle \,.\, Q']\!]_\sigma^6 \parallel [\![n(m) \,.\, P']\!]_\sigma^5 \searrow [\![Q']\!] \parallel [\![P']\!]_{g/m}^0 \searrow \\ &[\![Q']\!] \parallel [\![P']\!]_{g/m}^1 \searrow (m')([\![Q']\!] \parallel [\![P']\!] \ \parallel m'\langle [\![g]\!]\rangle) \ .\end{aligned}$$

So we end up with the following Homer process

$$\begin{aligned}(\tilde{m})\big((\tilde{n'} \tilde{n_P}' \tilde{n_Q}')\big([\![I_1]\!]_\sigma \parallel \cdots \parallel [\![I_{i-1}]\!]_\sigma \parallel [\![I_{i+1}]\!]_\sigma \parallel \cdots \parallel [\![I_k]\!]_\sigma \parallel [\![I_1^Q]\!]_\sigma^0 \parallel \cdots \parallel \\ [\![I_{k'}^Q]\!]_\sigma^0 \parallel [\![I_1^P]\!]_\sigma^0 \ \parallel \cdots \parallel [\![I_{k''}^P]\!]_\sigma^0 \parallel [\![O_1]\!]_\sigma \parallel \cdots \parallel [\![O_{j-1}]\!]_\sigma \parallel [\![O_{j+1}]\!]_\sigma \parallel \cdots \parallel [\![O_m]\!]_\sigma \\ \parallel [\![O_1^P]\!]_\sigma^0 \parallel \cdots \parallel [\![O_{m'}^P]\!]_\sigma^0 \parallel [\![O_1^Q]\!]_\sigma^0 \parallel \cdots \parallel [\![O_{m''}^Q]\!]_\sigma^0 \parallel [\![P_1]\!]_{e_1/m_1} \parallel \cdots \parallel \\ [\![P_l]\!]_{e_l/m_l} \parallel \Pi_{n \in \tilde{n}} \ n'\langle [\![n]\!]\rangle\big) \parallel r\big) \ ,\end{aligned} \tag{B.2}$$

where $\tilde{n_P}' = \{n, n' \mid n \in \tilde{n_P}\}$ and $\tilde{n_Q}' = \{n, n' \mid n \in \tilde{n_Q}\}$, and $\tilde{m}$ and r are defined as in ($\star\star$), since we have just extended σ with $[m \mapsto g]$. Notice that we have left out the indices on the untouched prefixes, since these are unchanged. The π-calculus process in (B.1) and the Homer process in (B.2) are related by $\mathcal{R}$.

Second condition : For the second condition there are again several cases to consider: either the reduction can come from an intern step of one of the components of the parallel composition, or a synchronisation between an input on index 5 and an output on index 6. We present only the last of the cases.

- If the reduction came from a synchronisation between an input I_i on index 5 ($[\![l(m) \,.\, P']\!]_\sigma^5$) and an output O_j on index 6 ($[\![\overline{l'}\langle n\rangle \,.\, Q']\!]_\sigma^6$), assuming $\sigma l = \sigma l'$ and $\sigma n = g$. Then these can synchronise to become $[\![P']\!]_{g/m}^0$ and $[\![Q']\!]$, respectively. Assume without loss of generality, that the form of Q' is $(\nu \tilde{n_Q})\big(I_1^Q \mid \cdots \mid I_{k'}^Q \mid O_1^Q \mid \cdots \mid O_{m'}^Q\big)$. Then the entire expression after the

synchronisation in Homer have this form

$$(\tilde{m})\big((\tilde{n}'\tilde{n_Q}')\big(\llbracket I_1\rrbracket_\sigma \parallel \cdots \parallel \llbracket I_{i-1}\rrbracket_\sigma \parallel \llbracket I_{i+1}\rrbracket_\sigma \parallel \cdots \parallel \llbracket I_k\rrbracket_\sigma \parallel \llbracket I_1^Q\rrbracket_\sigma^0 \parallel \cdots \parallel \llbracket I_{k'}^Q\rrbracket_\sigma^0 \parallel \llbracket O_1\rrbracket_\sigma \parallel \cdots \parallel \llbracket O_{j-1}\rrbracket_\sigma \parallel \llbracket O_{j+1}\rrbracket_\sigma \parallel \cdots \parallel \llbracket O_m\rrbracket_\sigma \parallel \llbracket O_1^Q\rrbracket_\sigma^0 \parallel \cdots \parallel \llbracket O_{m'}^Q\rrbracket_\sigma^0 \parallel \llbracket P_1\rrbracket_{e_1/m_1} \parallel \cdots \parallel \llbracket P_l\rrbracket_{e_l/m_l} \parallel \llbracket P'\rrbracket_{g/m}^0 \parallel \Pi_{n\in\tilde{n}}\ n'\langle\llbracket n\rrbracket\rangle\big) \parallel r\big)\ , \tag{B.3}$$

where $\tilde{n_Q}' = \{n, n' \mid n \in \tilde{n_Q}\}$, and $\tilde{m}$ and r are unchanged by the synchronisation. Again, we have left out the indices on the untouched prefixes, since these are unchanged.

We can match this synchronisation in the π-calculus by performing a synchronisation between I_i $(l(m)\,.\,P')$ and O_j $(\overline{l'}\langle n\rangle\,.\,Q')$ producing the following prenormal form

$$\sigma[\overline{m}\mapsto\overline{e}][m\mapsto g]\ \vdash (\tilde{n}\tilde{n_Q})\big(I_1 \mid \cdots \mid I_{i-1} \mid I_{i+1} \mid \cdots \mid I_k \mid I_1^Q \mid \cdots \mid I_{k'}^Q \mid O_1 \mid \cdots \mid O_{j-1} \mid O_{j+1} \mid \cdots \mid O_m \mid O_1^Q \mid \cdots \mid O_{m'}^Q \mid P_1 \mid \cdots \mid P_l \mid P'\big)\ . \tag{B.4}$$

Notice how the substitution $[m \mapsto g]$ corresponds to the process in prenormal form $\llbracket P'\rrbracket_{g/m}^0$, as required by $\mathcal{R}$. Again, the π-calculus process in (B.4) is related to the Homer process in (B.3) by $\mathcal{R}$.

Third condition: Assume that we have an unrestricted input prefix $\sigma \vdash P \downarrow n$, for some n. From the form of our π-calculus expressions, we know that this can either come from some input prefix I_i, for $1 \le i \le k$, or from one of the processes in preform P_i, for $1 \le i \le l$. Here we only consider the first case.

- If one of the input prefixes I_i is on the form $l(m).P'$, and assuming that $\sigma l = n$, then it gives rise to an unrestricted input on n. From the correspondence, we have $\llbracket l(m)\,.\,P'\rrbracket_\sigma^j$ for some j, where $0 \le j \le 5$. Hence $\llbracket l(m)\,.\,P'\rrbracket_\sigma^j$ can make $5-j$ reductions and become $\llbracket l(m)\,.\,P'\rrbracket_\sigma^5$, and $\llbracket l(m)\,.\,P'\rrbracket_\sigma^5 \downarrow n$.

Fourth condition: We assume that $\llbracket\sigma \vdash P\rrbracket^{(\overline{i},\overline{o},\overline{p},\overline{e},\overline{m})} \downarrow n$. Considering the induced prenormal form ($\star\star$) and the encoding of prefixes, this can only occur, if an unrestricted receive prefix exists in index 5 ($\llbracket I_i\rrbracket_\sigma^5$, for some $1 \le i \le k$). Without loss of generality, we assume that I_i is of the form $l(m)\,.\,P'$ and that $\sigma l = n$). Hence, we have that $\sigma \vdash P \downarrow n$. □