A Calculus of Mobile Resources

Theory Department, IT University of Copenhagen

Jens Christian Godskesen (jcg@it.edu)
Thomas Hildebrandt (hilde@it.edu)
Vladimiro Sassone (vs@susx.ac.uk)

IT University Technical Report Series **TR–2002-16**

ISSN 1600–6100

ISSN 1600–6100

ISBN 87-7949-021-2

Copies may be obtained by contacting:

IT University of Copenhagen
Glentevej 67
DK-2400 Copenhagen NV
Denmark

Telephone: +45 38 16 88 88
Telefax: +45 38 16 88 99
Web `www.it-c.dk`

A Calculus of Mobile Resources

Jens Christian Godskesen Thomas Hildebrandt

Theory Department, IT University of Copenhagen
Glentevej 67, 24000 Copenhagen NV, Denmark
{jcg,hilde}@itu.dk

Vladimiro Sassone*

COGS, University of Sussex
Brighton BN1 9QH
vs@susx.ac.uk

Abstract

We introduce a calculus of *Mobile Resources* (MR) tailored for the design and analysis of systems containing mobile, possibly nested, computing devices that may have resource and access constraints, and which are not copyable nor modifiable per se. We provide a reduction as well as a labelled transition semantics and prove a correspondence between barbed bisimulation congruence and a higher-order bisimulation. We provide examples of the expressiveness of the calculus, and apply the theory to prove one of its characteristic properties. This report is the full version of [11].

Introduction

Mobile computing resources moving in and out of other computing resources abound in our daily life. Prime examples are smart cards [12] used e.g. in Subscriber Identity Module (SIM) cards or next generation credit cards, moving from card issuers to card holders and in and out of mobile phones or automatic teller machines (ATMs). Accordingly, the ability to reason about correctness of the behavior of concurrent systems containing such resources, as well as the need of design and implementation tools, will raise to an increasingly prominent role. We propose a calculus of *mobile resources* (MR) aimed at designing and analysing systems containing nested, mobile computing resources residing in named locations that have *capacity* constraints. Our goals include to devise a formal framework to express and prove properties that may depend on the assumption that such resources are neither *copyable* nor arbitrarily *modifiable* per se. These assumptions are crucial for the security of systems based on smart cards as trusted computing bases, such as e-cash and SIMs.

The calculus MR is inspired by the Mobile Ambient calculus [5, 16], bears relationships to Boxed Ambients [3] and the Seal calculus [24], and to distributed process algebras [13, 23, 10], but differs from all these in important ways, motivated by our specific goals. Building upon a CCS-like calculus [19] with prefix, restriction, parallel composition, replication, no summation nor recursion, we introduce *named slots*, i.e., if p is a process, then $n\lfloor\, p \,\rfloor$ represents a resource p in a slot named by n. In general, we allow slot aliasing, that is slots to be named by more than one name, writing $\tilde{n}\lfloor\, p \,\rfloor$ for a resource p in a slot named by a set of names $\tilde{n}$. Finally, we assume that slots may carry names of the form $\natural n$ used for deletion, e.g. a slot $\{m, \natural n\}\lfloor\, p \,\rfloor$ can be accessed via the name n and deleted by processes knowing the name n.

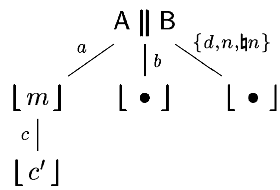
*Supported by 'MyThS: Models and Types for Security in Mobile Distributed Systems', EU FET-GC IST-2001-32617.

We postulate that a resource can move from a location to another only if an empty slot can be found at the target location. This makes $n\lfloor \bullet \rfloor$ very different from a slot containing a terminated process, and allows us to model locations that can only contain a bounded number of resources, thus capturing a very relevant aspect of real-world devices carrying embedded processors. To abstract away from this, replication in the style of the π calculus can be used to recover the usual semantics of locations by generating unboundedly many slots at a location, as, e.g., in $!n\lfloor \bullet \rfloor$. Since resources are processes, they might themselves contain slots, giving rise to a nested spatial structure. By allowing restriction of location names, we can represent restricted access to a location.

To help focusing our ideas, let us consider the processes

$$\begin{array}{rcl} \mathsf{Alice} & \triangleq & (m)(a\lfloor \mathsf{C} \rfloor \parallel \mathsf{A}) \\ \mathsf{Bob} & \triangleq & (b)(n)(b\lfloor \bullet \rfloor \parallel \mathsf{B} \parallel \{d, n, \natural n\}\lfloor \bullet \rfloor) \\ \mathsf{C} & \triangleq & c\lfloor c' \rfloor \parallel m \\ P & \triangleq & \mathsf{Alice} \parallel \mathsf{Bob} \end{array}$$

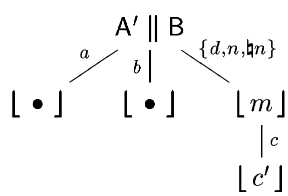
consisting of a process Alice with a resource C in a public slot named a and a process Bob having an empty, private slot named b, and an empty slot with a public name d and a private name n that may also be removed using the name n. For the sake of this discussion, the spatial structure of P can be depicted as in the labelled tree below, where edges represent slots, labels slot names, and nodes processes other than slots.



Mobility of resources in MR is '*objective*,' as opposed to 'subjective,' i.e. the migration of a resource is initiated and controlled not by the resource itself, but by an external process. More precisely, a resource is controlled by a process *outside* the slot where the resource is placed, that is a process residing at a super location. We introduce this notion by means of *move actions* of the form $n \triangleright \overline{m}$, a capability that should be read 'move a resource from a slot at the location n to a slot at the location m.' We use a notation reminiscent of action/co-action pairs to stress the dual roles of n and m that, respectively, give and take a resource, and we will adopt consistent conventions throughout the paper. If for instance $\mathsf{A} \triangleq a \triangleright \overline{d} \,.\, \mathsf{A}'$, we would have

$$\mathsf{P} \searrow (m)(b)(n)(a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor \bullet \rfloor \parallel \mathsf{B} \parallel \{d, n, \natural n\}\lfloor \mathsf{C} \rfloor)$$

whose spatial structure can be drawn as follows.



Observe that the movement of C from a to d causes a *scope extension* for m.

Carrying on with our example, supposing $\mathsf{B} \triangleq d \triangleright \overline{b} \,.\, \mathsf{B}'$ we have the reduction

$$\begin{array}{l} (m)(b)(n)(a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor \bullet \rfloor \parallel d \triangleright \overline{b} \,.\, \mathsf{B}' \parallel \{d, n, \natural n\}\lfloor \mathsf{C} \rfloor) \searrow \\ \qquad (m)(b)(n)(a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor \mathsf{C} \rfloor \parallel \mathsf{B}' \parallel \{d, n, \natural n\}\lfloor \bullet \rfloor). \end{array}$$

Observe that the last two reductions illustrate the passage of resource C from Alice to the private slot of Bob, without Alice's knowing the name of Bob's private slot.

As for mobility, deletion of slots is objective, and is controlled by *delete actions* of the form $\natural n$. Supposing $\mathsf{B}' \triangleq \natural n \,.\, \mathsf{D}$ we have the reduction

$$(m)(b)(n)(a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor \mathsf{C} \rfloor \parallel \natural n \,.\, \mathsf{D} \parallel \{d, n, \natural n\}\lfloor \bullet \rfloor) \searrow$$
$$(m)(b)(n)(a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor \mathsf{C} \rfloor \parallel \mathsf{D}\ ).$$

The posibility of slot deletion may be viewed as observable and controlable failure of slots. The remarkable features are that a slot can only be removed by processes knowing a name for deletion, and if the slot has no names for deletion it can not fail. [1]

While explicit mobility captures asynchronous communication via resource passing, *synchronous communication* is the second central concept of MR, covering several different aspects of process interaction in our application domain. As in CCS, co-located parallel processes can communicate synchronously by performing respectively an a-action and a $\overline{a}$-action. In addition, we allow a process to communicate with any of its descendants, by performing a *directed action* of the form δa, where δ is a sequence of slot names. For example, if $\mathsf{D} = bc\overline{c}' \,.\, \mathsf{D}'$ in our running example, we have the reduction

$$(m)(b)(n)(\ a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor c\lfloor c' \rfloor \parallel m \rfloor \parallel bc\overline{c}' \,.\, \mathsf{D}'\ ) \searrow$$
$$(a')(b)(b')(\ a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor c\lfloor 0 \rfloor \parallel m \rfloor \parallel \mathsf{D}'\ ),$$

where the co-action from D synchronises with the corresponding action from slot c inside slot b. In this way, the actions of the resource C (and its sub-resources) are *dynamically bound* to the directed actions of Bob. Unlike e.g. the Seal calculus, we do not distinguish between undirected actions and actions that may synchronise with ascendants.

Using sequences of names in move actions as for the synchronisation, we can move a resource (subtree) from a slot at an arbitrarily deep sub-location to an empty slot (a black leaf) at another arbitrarily deep sub-location. For instance, if $\mathsf{D}' = bc \triangleright \overline{a} \,.\, \mathsf{D}''$ we have the reduction

$$(m)(b)(n)(a\lfloor \bullet \rfloor \parallel \mathsf{A}' \parallel b\lfloor c\lfloor 0 \rfloor \parallel m \rfloor \parallel bc \triangleright \overline{a} \,.\, \mathsf{D}''\ ) \searrow$$
$$(m)(b)(n)(\ a\lfloor 0 \rfloor \parallel \mathsf{A}' \parallel b\lfloor c'\lfloor \bullet \rfloor \parallel m \rfloor \parallel \mathsf{D}''\ ).$$

The reductions presented above constitute the primary mechanisms of MR.

Structure of the paper & Results. After introducing the syntax of MR in §1, in §2 we lay the foundations of its semantic theory by giving a reduction semantics formalising the different ways of interaction discussed above; §3 discusses several small examples aimed at illustrating some particularities of MR. We then proceed in §4 to give a labelled transition semantics to MR equivalent to the reduction one. This is well known to be a non-trivial task for calculi allowing (higher-order) process mobility and scope extension, as in MR when resources containing restricted names are moved. In §5 we provide a characterisation of the barbed congruence in terms of a higher-order labelled transition bisimulation. Predictably, the main difficulty in proving the transition bisimulation to be a congruence is the insertion of processes into slots. One of the examples in §3 will point out one of the reasons for that. The detailed proofs of our results can be found in the appendix. As usual with higher-order bisimulations, the characterisation here uses a selected set of contexts that play the role of destructors for the higher-order values, namely *receiving contexts* dealing with the reception of resources into slots. We will return on this later on. In §6 we give an application of the characterisation, proving a linearity property of the calculus by giving a bisimulation of two processes.

Design issues & Related work. As already mentioned, MR shares ideas with the Mobile Ambients (MA) [5]. In both calculi, in fact, processes are equipped with nested, named locations – the ambients – containing processes, and the spatial structure can be dynamically extended or change

[1] The distinguished names for deletion of slots is new compared to the version of MR in [11].

due to movement. However, likewise the Seal calculus [24], it is the *anonymous contents* of locations to be moved in MR, and it is moved by a process *external* to the location. On the contrary, in MA it is the *named location* to be moved by a process *within* it. Another departure point with MA, where ambients communicate only asynchronously, processes in MR may communicate both *synchronously*, as in CCS and the π-calculus, and *asynchronously*, by exchanging resources. Resource movement is a three-party interaction in both MR and the Seal calculus. However, slots in Seal are pure references that disappear after interaction, while in MR they remain as empty slots until explicitly removed. Moreover, the reception of a resource is via a pair action/co-action in Seal.

To the best of our knowledge, the boundedness of resources is unique, among process algebras, to the calculus proposed in this paper. Similar ideas may of course be found in related areas, most notably bounded places in Petri nets, but, besides the obvious analogies, there seem to be no formal relationships with our notion here.

Our calculus shares with Safe Ambients (SA) [16] – and with several other proposals that space does not allow us to survey upon – the wish to put a stricter control on mobility and access to locations, taking the objective mobility viewpoint. While this is realised by an action/co-action synchronisation between mover and movee in the SA approach, MR relies on move actions performed by the mover. Also, the idea of direct actions across location boundaries is reminiscent of the forms of communications found in the Seal calculus, Boxed Ambients, in Dπ [13] and in the distributed Join calculus[10], though in the latter communication is asynchronous and locations distributed.

Observe that, differently from all these, MR does not allow explicit communication of names. This design choice seems consistent with our application, in that it confines information inside resources and allows network topology evolution only by means of extensions and replacement of substructures, maintaining a strictly hierarchical network structure. We leave to future work the investigation of a capability-passing version of MR, as well as the impact of asynchronous communication between remote, non directly nested sites, and the expressiveness of movements that refer to sibling slots.

Concerning the choice of moves and communication that span multiple slot boundaries our hypothesis is that, slots do not necessarily represent physical location boundaries that enforce a notion of communication distance. Distance may be enforced by use of restricted slot names akin to private fields in Java. For this reason we prefer to develop the theory in full generality. After all, that a location may not be accessed from "grand parent nodes" is an issue that can easily be demanded to the control of a type system. Of course, this choice makes the calculus more complex; its price is a more complex semantic theory, yet – we believe – still manageable. For future reference, let us call MR_2 the calculus restricted to paths of length at most two, that is with only directed communication across at most one boundary and short moves of the form $a\triangleright\overline{c}$ (flat), $ab\triangleright\overline{c}$ (up), and $a\triangleright\overline{cb}$ (down). All the results in the paper carry naturally over for this sub-calculus.

1 The Calculus

We assume an infinite set of *names* $\mathcal{N}$ ranged over by n and m. Let $\overline{\mathcal{N}} = \{\overline{n} \,|\, n \in \mathcal{N}\}$ be the set of *co-names*. We let α range over $\mathcal{A} = \mathcal{N} \cup \overline{\mathcal{N}}$ and γ over the set $\mathcal{N}^*$ of sequences of names, referred to as *direction paths*, with ϵ denoting the empty sequence. We use δ to denote elements of $\mathcal{N}^+$, the set of non-empty direction paths. The set $\mathcal{L}$ of *prefix labels* is then defined by:

$$\lambda \quad ::= \quad \gamma\alpha \mid \delta\triangleright\overline{\delta'} \mid \natural n.$$

The actions α play the same role as in CCS. However, as explained in the introduction, we allow actions to be extended with a sequence of slot names, so that $n\alpha$ is an action directed to a resource in a slot identified by n. An action $n\alpha$ synchronises with the corresponding co-action $\overline{\alpha}$ performed by a resource in a slot at n.

Let $\tilde{n}$ range over sets of names and deletion names, that is subsets of $\mathcal{N} \cup \{\natural n \mid n \in \mathcal{N}\}$. The

$E_1.$	$p \parallel \mathbf{0} \equiv p$	$E_4.$	$(n)\mathbf{0} \equiv \mathbf{0}$	$E_7.$	$!p \equiv p \parallel\ !p$
$E_2.$	$p \parallel q \equiv q \parallel p$	$E_5.$	$(n)p \parallel p' \equiv (n)(p \parallel p')$, if $n \notin fn(p')$		
$E_3.$	$(p \parallel p') \parallel p'' \equiv p \parallel (p' \parallel p'')$	$E_6.$	$(n)\tilde{n}\lfloor p \rfloor \equiv \tilde{n}\lfloor (n)p \rfloor$, if $\{n, \natural n\} \cap \tilde{n} = \emptyset$		

Table 1: Structural equivalence.

sets $\mathcal{P}$ of *process expressions* is then defined by:

$$\begin{array}{lll} p, q & ::= \mathbf{0} \mid \lambda \,.\, p \mid p \parallel q \mid !p \mid (n)p \mid \tilde{n}\lfloor r \rfloor & \qquad (\mathcal{P}) \\ r & ::= \bullet \mid p & \end{array}$$

Processes $\mathbf{0}$, $\lambda \,.\, p$, and $p \parallel q$ are the ordinary CCS-like constructs, representing respectively the inactive process, the prefixed process, and the parallel composition of processes. The replicated process $!p$ provides as many parallel instances of p as required and adds to the calculus the power of recursive definitions. The restriction $(n)p$ makes name n local to p. The novelty of the calculus resides in the slot processes already described in the introduction: $\tilde{n}\lfloor \bullet \rfloor$, an empty *slot*, and $\tilde{n}\lfloor p \rfloor$ a slot containing a process p, both carrying the names (and deletion names) in $\tilde{n}$. We will write $n\lfloor r \rfloor$ for a slot $\{n\}\lfloor r \rfloor$ possessing only a single name. We refer to processes within slots as *resources*.

The restriction operator (n) is the only binding construct; the set $fn(p)$ of free names of p is defined accordingly as usual. By convenience, we omit trailing $\mathbf{0}$s and hence write λ instead of $\lambda \,.\, \mathbf{0}$. As usual, we let prefixing, replication, and restriction be right associative and bind stronger than parallel composition hence writing e.g. $!(n)n \,.\, p \parallel q$ instead of $(!((n)(n \,.\, p))) \parallel q$. For at set of names $\tilde{n} = \{n_1, \ldots, n_k\}$ we let $(\tilde{n})p$ denote $(n_1) \cdots (n_k)p$. We write $n\tilde{n}$ for $\{n\} \cup \tilde{n}$.

2 Reduction Semantics

Contexts $\mathscr{C}$ are, as usual, terms with a hole $(-)$. We write $\mathscr{C}(p)$ for the insertion of p in the hole of context $\mathscr{C}$. An equivalence relation $\mathscr{S}$ on $\mathcal{P}$ is a *congruence* if it is preserved by all contexts. As our calculus allows actions involving terms at depths arbitrarily far apart, in order to express its notions with formal precision, yet in succinct terms, we need to make an essential use of a particular kind of contexts throughout the paper. We define an $\mathcal{N}^*$-indexed family of *path contexts*, $\mathscr{C}_\gamma$, inductively as:

$$\mathscr{C}_\epsilon ::= (-) \qquad \mathscr{C}_{n\gamma} ::= \tilde{n}\lfloor \mathscr{C}_\gamma \parallel p \rfloor \ , \qquad n \in \tilde{n}.$$

Observe that the direction path γ for a context $\mathscr{C}_\gamma$ indicates a path under which the context's 'hole' is found. We extend $fn()$ to path contexts by $fn(\mathscr{C}_\gamma) = fn(\mathscr{C}_\gamma(\mathbf{0}))$. We also define a family of path contexts

$$\mathscr{D}_{\gamma n} ::= \mathscr{C}_\gamma\big(\tilde{n}\lfloor (-) \rfloor\big) \ , \qquad n \in \tilde{n},$$

for the special case where the hole is the only content of a slot.

The structural congruence relation $\equiv$ is the least congruence on $\mathcal{P}$ satisfying alpha-conversion and the rules in Table 1. The equations express that $(P, \|, \mathbf{0})$ is a commutative monoid (E_1–E_3) and enforce the usual rules for scope (E_4–E_6) and replication (E_7). We write $p \equiv_\alpha q$ if p and q are alpha-convertible.

Evaluation contexts $\mathscr{E}$ are contexts whose hole does not appear under prefix or replication, i.e.

$$\mathscr{E} ::= (-) \mid \tilde{n}\lfloor \mathscr{E} \rfloor \mid \mathscr{E} \parallel p \mid (n)\mathscr{E}$$

Define $\searrow$ as the least binary relation on $\mathcal{P}$ satisfying the rules in Table 2 and closed under $\equiv$ and under all evaluation contexts $\mathscr{E}$.

The first rule captures both the standard CCS synchronous communication and a synchronisation reminiscent of the one found in the Seal calculus and in Boxed ambients, in that communication in

$$\gamma\alpha\,.\,p \parallel \mathscr{C}_\gamma(\,\overline{\alpha}\,.\,q\,) \searrow p \parallel \mathscr{C}_\gamma(\,q\,)$$
$$\gamma\delta_1 \triangleright \overline{\gamma\delta_2}\,.\,p \parallel \mathscr{C}_\gamma\big(\,\mathscr{D}_{\delta_1}(q) \parallel \mathscr{D}_{\delta_2}(\bullet)\,\big) \searrow p \parallel \mathscr{C}_\gamma\big(\,\mathscr{D}_{\delta_1}(\bullet) \parallel \mathscr{D}_{\delta_2}(q)\,\big)$$
$$\natural m\,.\,p \parallel \tilde{n}\lfloor\,r\,\rfloor \searrow p \qquad \natural m \in \tilde{n}$$

Table 2: Reduction rules

$n\alpha$ is directed downward and may synchronize with a local communication on α inside n. The purpose of context $\mathscr{C}_\gamma$ there is to express that $\gamma\alpha$ synchronises with an $\overline{\alpha}$ found under path γ. The second rule defines movement of resources. This movement is '*objective*,' meaning that resources are moved from the outside and not by the resource itself, as in the Ambient calculus. The third rule defines deletion of slots. The process that performs it must hold all the names of the slot.

As already remarked, moves across multiple boundaries make the calculus formally more complex. The price we pay is a pervasive use of contexts, starting here in the reduction rules and with effects reaching – as we will see – our bisimulation congruence. We remark that in MR_2, where paths have length at most two, the movement rules above specialise to

$$a \triangleright \overline{cb}\,.\,p \;\parallel\; a\lfloor\,s\,\rfloor \;\parallel\; c\lfloor\,q \parallel b\lfloor\,\bullet\,\rfloor\,\rfloor \quad\searrow\quad p \;\parallel\; a\lfloor\,\bullet\,\rfloor \;\parallel\; c\lfloor\,q \parallel b\lfloor\,s\,\rfloor\,\rfloor,$$
$$ab \triangleright \overline{c}\,.\,p \;\parallel\; a\lfloor\,b\lfloor\,s\,\rfloor \parallel q\,\rfloor \;\parallel\; c\lfloor\,\bullet\,\rfloor \quad\searrow\quad p \;\parallel\; a\lfloor\,b\lfloor\,\bullet\,\rfloor \parallel q\,\rfloor \;\parallel\; c\lfloor\,s\,\rfloor,$$
$$a \triangleright \overline{b}\,.\,p \;\parallel\; a\lfloor\,s\,\rfloor \;\parallel\; b\lfloor\,\bullet\,\rfloor \quad\searrow\quad p \;\parallel\; a\lfloor\,\bullet\,\rfloor \;\parallel\; b\lfloor\,s\,\rfloor,$$

and similarly for the communication rules.

We move now to study the semantic theory of MR. We start by discussing the notion of observation. It seems fair to observe the communication actions processes offer to the environment. We then define barbs as:

$$p \downarrow n \quad \text{if} \quad p \equiv (\tilde{n})(\alpha\,.\,p' \parallel q), \quad \alpha \in \{n, \overline{n}\}, \quad n \notin \tilde{n}$$

This excludes observing restricted actions, as well as directed actions and move actions. Several alternative choices of observation appear natural, as e.g. observing at top level – i.e., not inside any slots – free slot names, empty slots, path actions, or movements. Our choice is robust, as none of these alternatives would actually give rise to a different semantic theory from the one we develop below.

Definition 1 A *barbed bisimulation* is a symmetric relation $\mathscr{S}$ on $\mathcal{P}$ such that whenever $p \mathrel{\mathscr{S}} q$

$$p \downarrow n \text{ implies } q \downarrow n$$
$$p \searrow p', \text{ then } \exists q \searrow q' \text{ with } p' \mathrel{\mathscr{S}} q'$$

Barbed bisimulation congruence $\sim_b$ is the largest congruence that is a barbed bisimulation.

The definition above is in principle stricter than the classical notion of barbed congruence, defined as the largest congruence *contained* in the barbed bisimulation. It is however gaining credit in process algebra theory for its good properties (cf. e.g [9, 1]).

3 Examples

Faulty domain. As already remarked in the introduction, slots may be given multiple names and separate names for deletion. Consider the process

$$(m)\big(\,\{n, \natural m\}\lfloor\,\bullet\,\rfloor \;\parallel\; !\natural m\,.\,\{n, \natural m\}\lfloor\,\bullet\,\rfloor\,\big).$$

The process illustrates both slot deletion and dynamic creation of slots. The environment can insert/take resources into/from the empty slot. However, at any time, the slot may be deleted (with or without contents) and replaced by an identical, but empty slot.

Linearity. It is a fundamental property of MR that resources cannot be copied. This interacts with the usual scoping rules enforced by restriction to yield an interesting '*linearity*' property. Consider the term $P \triangleq (b)\big(a\lfloor b\rfloor \parallel !a\overline{b}\,.\,c\big)$. Because of the restriction (b) only the resource inside slot a will ever be able to use b to interact with the replicated term on its side. However, it may in principle be possible that exporting it to some external context, such a resource may be able to 'copy' itself, replicating the reference to b. This would be possible in several (higher-order) calculi. Yet, it is not possible in MR. In particular, if a resource containing a reference to a local name is given out, only (a residual of) *that* resource may in future refer that name. This is stated by the following equation, that we prove in §6.

$$(b)\big(a\lfloor b\rfloor \parallel !a\overline{b}\,.\,c\big) \sim_b (b)\big(a\lfloor b\rfloor \parallel a\overline{b}\,.\,c\big)$$

Suggestively, the process $(b)\big(a\lfloor b\rfloor \parallel !a\overline{b}\,.\,c\big)$ can be regarded as a model of a pre-paid cash card (the resource b) in the slot a of a vending machine $a\lfloor \bullet \rfloor \parallel !a\overline{b}\,.\,c$ that delivers a cup of coffee (action c) for each cash card of the right kind, b, inserted in a. The $\sim_b$-equation above then states that if there exists only *one* card of the 'right' type, then there will ever be only *one* cup of coffee; in other words, the cash card cannot be copied. This is the kind of properties relevant to our intended application area.

Scope extension and mobility The interplay of upward and downward moves and scope extension gives rise to interactions unexpected at first, and is a major challenge for the theory of the observational congruence presented in the next section. Here, we exemplify it as follows. Consider the contexts

$$\mathscr{C}_1 \triangleq c\lfloor (-) \rfloor \parallel a, \qquad \mathscr{C}_2 \triangleq d\lfloor (-) \rfloor \parallel d\overline{a}\,.\,b,$$

and the process $p \triangleq (a)\mathscr{C}_1(\mathscr{C}_2(\bullet)) = (a)\big(c\lfloor d\lfloor \bullet \rfloor \parallel d\overline{a}\,.\,b\rfloor \parallel a\big)$. Since name a is private, it would appear that no resource inserted into the empty slot d can synchronise with the $d\overline{a}$-action and, similarly, that no process can ever synchronise with the a action at top-level. It would then follow that the b action can never be revealed and, in particular, that p behaves like $q \triangleq (a)(c\lfloor d\lfloor \bullet \rfloor\rfloor)$, no matter the context. Yet, this is not the case. Under a suitable context, it is possible for the process a to change its role from being the parent of $d\overline{a}\,.\,b$ to being its child in the slot named d. Suppose in fact that p and q are inserted into the context

$$\mathscr{C} = x\lfloor (-) \rfloor \parallel y\lfloor \bullet \rfloor \parallel xc\triangleright\overline{y}\,.\,x\triangleright\overline{yd}.$$

Then $\mathscr{C}(p)$ reduces (in two steps) to

$$(a)\big(x\lfloor \bullet \rfloor \parallel y\lfloor \mathscr{C}_2(\mathscr{C}_1(\bullet)) \rfloor\big) = (a)\big(x\lfloor \bullet \rfloor \parallel y\lfloor d\lfloor c\lfloor \bullet \rfloor \parallel a\rfloor \parallel d\overline{a}\,.\,b\rfloor\big),$$

where $\mathscr{C}_1$ and $\mathscr{C}_2$ have swapped place. Now the b-action may be unleashed upon synchronisation on a. Since $b \notin fn(\mathscr{C}(q))$, clearly $\mathscr{C}(q)$ cannot reduce to a process with a b-action, so $p \not\sim_b q$.

Digital Signature Card. The following example models a digital signature card. For readability we use the names *reg*, *in* and *out* for slots representing respectively a *register*, an *in-buffer* and a *out-buffer*. We then give a model of a process Enc_k parametrized by a name k (the key) that (repeatedly) encrypts resources received in its *in*-buffer with key k, and returns the encrypted resource via its *out*-buffer.

$$Enc_k \quad \triangleq \quad !(reg)(\,in\triangleright\overline{reg\ k}\,.\,reg\triangleright\overline{out} \parallel reg\lfloor k\lfloor \bullet \rfloor\rfloor) \parallel in\lfloor \bullet \rfloor \parallel out\lfloor \bullet \rfloor$$

Dually, we can define a process Dec_k that (repeatedly) decrypts resources received in its *in*-buffer with key k and returns the decrypted resource via its *out*-buffer.

$$Dec_k \quad \triangleq \quad !(reg)(in\triangleright\overline{reg}\,.\,reg\ k\triangleright\overline{out}. \parallel reg\lfloor \bullet \rfloor) \parallel in\lfloor \bullet \rfloor \parallel out\lfloor \bullet \rfloor$$

If the name k is globally known, anyone can perform encryption and decryption. On the other hand, if k is a shared secret between two processes, e.g. Alice and Bob, and Alice (resp. Bob) possesses an encryption (resp. decryption) process as a private resource, then Alice can send messages secretly to Bob.

$$\begin{aligned}
Alice_{k,M} &\triangleq (m)(a)\big(a\lfloor Enc_k \rfloor \parallel m\lfloor M \rfloor \parallel m \triangleright \overline{a\ in} \,.\, a\ out \triangleright \overline{network}\big) \\
Bob_k &\triangleq (m)(b)(b\lfloor Dec_k \rfloor \parallel m\lfloor \bullet \rfloor \parallel network \triangleright \overline{b\ in} \,.\, b\ out \triangleright \overline{m}) \\
SecretCom_M &\triangleq (k)\big(Alice_{k,M} \parallel Bob_k\big) \parallel network\lfloor \bullet \rfloor
\end{aligned}$$

We may prove the encryption property by showing that for any processes (messages) of the form $M = a_1 \,.\, a_2 \,.\, \ldots \,.\, a_i$ and $M' = a'_1 \,.\, a'_2 \,.\, \ldots \,.\, a'_j$, we have

$$SecretCom_M \sim_b SecretCom_{M'} \,.$$

We then may model a digital signature card which generates the key and exports the decryption resource (as many times as needed) but keeps the encryption resource private.

$$SignatureCard \triangleq (k)\big(!export\lfloor Dec_k \rfloor \parallel Enc_k\big)$$

4 Transition Semantics

In this section we set out to provide MR with a labelled transition semantics. The interplay of mobility and local names as illustrated by examples in the previous section has interesting consequences in this respect. One example, *Linearity*, in fact, shows that a certain amount of information must be retained about resources given out to the context. This is not similar to the transition semantics for the π calculus (cf. [19, 21]). In the π calculus the relevant information concerns the extruded names; in MR things may in principle be more complex, since interaction involves passing around resources, i.e. higher-order, evolving entities. The example *Scope extension and mobility* also points out that we must consider that exported resources may be received in arbitrarily complex contexts.

We focus on explaining the transition rules for the characteristic features of our calculus, i.e. slots, directed communication and objective mobility, which are shown in Table 4. Also, we explain the interplay between labels.

To capture directed communication, we introduce labels of the form $\overline{\delta}\alpha$ that may synchronise with the directed actions of the form $\delta\alpha$ that appear as prefixes in the calculus. We do this by defining $\overline{\overline{n}a} = n\overline{a}$. For example, we have

$$n\lfloor a.p \rfloor \xrightarrow{\overline{n}a} n\lfloor p \rfloor \qquad (nesting)$$

and the usual synchronisation rule yields

$$n\overline{a} \,.\, p \parallel n\lfloor a \,.\, q \rfloor \xrightarrow{\tau} p \parallel n\lfloor q \rfloor \ ,$$

The three-party interaction required for the movement of resources is modelled by means of higher-order labels. We introduce

$$\overline{\delta} \triangleright \langle p \rangle \quad (p \text{ exits from } \delta) \qquad \text{and} \qquad (p) \triangleright \delta \quad (p \text{ enters in } \delta).$$

The corresponding co-labels will be indicated by $\delta \triangleright (p)$ and $\langle p \rangle \triangleright \overline{\delta}$, respectively.

The move action $\delta_1 \triangleright \overline{\delta_2}$ – whose co-action we denote by $\overline{\delta_1} \triangleright \delta_2$ – and the two higher-order labels will partially match each other in pairs, so as to give rise to *co-label* corresponding to the third party, the one missing to perfection the three-way synchronisation. That is,

$$\begin{array}{llll}
\delta_1 \triangleright \overline{\delta_2} & \text{coalesces with} & \overline{\delta_1} \triangleright \langle p \rangle & \text{yielding} & \langle p \rangle \triangleright \overline{\delta_2}, \\
\delta_1 \triangleright \overline{\delta_2} & \text{coalesces with} & (p) \triangleright \delta_2 & \text{yielding} & \delta_1 \triangleright (p), \\
\overline{\delta_1} \triangleright \langle p \rangle & \text{coalesces with} & (p) \triangleright \delta_2 & \text{yielding} & \overline{\delta_1} \triangleright \delta_2.
\end{array}$$

$$(\textit{prefix})\ \frac{}{\lambda\,.\,p \xrightarrow{\lambda} p} \qquad (\textit{rest})\ \frac{p \xrightarrow{\pi} p'}{(n)p \xrightarrow{\pi} (n)p'},\ n \notin \mathit{fn}(\pi) \cup \mathit{bn}(\pi)$$

$$(\textit{sync})\ \frac{p_1 \xrightarrow{(\tilde{n})\overline{\pi}} p_1' \qquad p_2 \xrightarrow{\pi} p_2'}{p_1 \parallel p_2 \xrightarrow{\tau} (\tilde{n})(p_1' \parallel p_2')},\ \mathit{fn}(p_2) \cap \tilde{n} = \emptyset$$

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

Table 3: Transition rules, standard.

Hence, the three labels match in any order and annihilate their matching action/co-action particles to yield, at last, a τ. For instance, a resource that exits a slot produces a transition

$$n\lfloor p \rfloor \xrightarrow{\overline{n} \triangleright \langle p \rangle} n\lfloor \bullet \rfloor \qquad \textit{(exit)},$$

and similarly, an empty slot that receives a resource, gives rise to a higher-order transition

$$m\lfloor \bullet \rfloor \xrightarrow{(p) \triangleright m} m\lfloor p \rfloor \qquad \textit{(enter)}.$$

These 'exit' and 'enter' transitions may synchronise, yielding a *co-move* transition

$$n\lfloor p \rfloor \parallel m\lfloor \bullet \rfloor \xrightarrow{\overline{n} \triangleright m} n\lfloor \bullet \rfloor \parallel m\lfloor p \rfloor \qquad \textit{(co-move)}$$

which, in turn, can synchronise with the corresponding move action to yield a τ-action that represent the completed interaction:

$$n\lfloor p \rfloor \parallel m\lfloor \bullet \rfloor \parallel n \triangleright \overline{m}\,.\,q \xrightarrow{\tau} n\lfloor \bullet \rfloor \parallel m\lfloor p \rfloor \parallel q.$$

Symmetrically, 'exit' and 'move' transitions may synchronise, resulting in a 'give' transition

$$n\lfloor p \rfloor \parallel n \triangleright \overline{m}\,.\,q \xrightarrow{\langle p \rangle \triangleright \overline{m}} n\lfloor \bullet \rfloor \parallel q \qquad \textit{(give)}$$

which may synchronise with the dual 'enter' transition. Dually, 'enter' and 'move' transitions may synchronise, resulting in a 'take' transition

$$m\lfloor \bullet \rfloor \parallel n \triangleright \overline{m}\,.\,q \xrightarrow{n \triangleright (p)} m\lfloor p \rfloor \parallel q \qquad \textit{(take)},$$

which is ready to synchronise with the dual 'exit' transition.

Rules (*exit*) and, in particular, (*enter*) may at first appear to be 'spontaneous' rules. A closer analysis though reveals that they are akin to 'output' and 'input' transitions in the 'early' labelled transitions semantics of the (higher-order) π calculus, rather than transition that may fire autonomously an unbounded number of times.

The last issue involved in the movement of resources is the treatment of scope extension when resources are moved. The phenomenon is totally analogous to that in the (higher-order) pi-calculus, and we handle it as usual (cf. [21]) by restrictions on the labels of (*exit*) and (*give*) transitions, e.g.,

$$(m)n\lfloor m \rfloor \xrightarrow{(m)\overline{n} \triangleright \langle m \rangle} n\lfloor \bullet \rfloor \qquad \textit{(O1)}$$

and by explicit scope extension in the synchronisation rule.

$$(exit)\ \frac{}{\tilde{n}\lfloor p\rfloor \xrightarrow{\overline{n}\triangleright\langle p\rangle} \tilde{n}\lfloor \bullet \rfloor},\ n\in\tilde{n} \qquad (enter)\ \frac{}{\tilde{n}\lfloor \bullet \rfloor \xrightarrow{(p)\triangleright n} \tilde{n}\lfloor p\rfloor},\ n\in\tilde{n}$$

$$(give)\ \frac{p_1 \xrightarrow{(\tilde{n})\overline{\delta_1}\triangleright\langle q\rangle} p_1' \qquad p_2 \xrightarrow{\delta_1\triangleright\overline{\delta_2}} p_2'}{p_1 \parallel p_2 \xrightarrow{(\tilde{n})\langle q\rangle\triangleright\overline{\delta_2}} p_1' \parallel p_2'},\ fn(p_2)\cap\tilde{n}=\emptyset \qquad (take)\ \frac{p_2 \xrightarrow{\delta_1\triangleright\overline{\delta_2}} p_2' \qquad p_1 \xrightarrow{(q)\triangleright\delta_2} p_1'}{p_1 \parallel p_2 \xrightarrow{\delta_1\triangleright(q)} p_1' \parallel p_2'}$$

$$(co\text{-}move)\ \frac{p_1 \xrightarrow{(\tilde{n})\overline{\delta_1}\triangleright\langle q\rangle} p_1' \qquad p_2 \xrightarrow{(q)\triangleright\delta_2} p_2'}{p_1 \parallel p_2 \xrightarrow{\overline{\delta_1}\triangleright\delta_2} (\tilde{n})(p_1' \parallel p_2')},\ fn(p_2)\cap\tilde{n}=\emptyset$$

$$(nesting)\ \frac{p \xrightarrow{\pi} p'}{\tilde{n}\lfloor p\rfloor \xrightarrow{n\cdot(\pi)} \tilde{n}\lfloor p'\rfloor},\ n\in\tilde{n}, \qquad (delete)\ \frac{}{\tilde{n}\lfloor r\rfloor \xrightarrow{\overline{\natural m}} \mathbf{0}} \quad \natural m\in\tilde{n}$$

$$(O1)\ \frac{p \xrightarrow{(\tilde{n})\overline{\delta}\triangleright\langle q\rangle} p'}{(n)p \xrightarrow{(n\tilde{n})\overline{\delta}\triangleright\langle q\rangle} p'},\ n\in fn(q)\backslash(fn(\delta)\cup\tilde{n}) \quad (O2)\ \frac{p \xrightarrow{(\tilde{n})\langle q\rangle\triangleright\overline{\delta}} p'}{(n)p \xrightarrow{(n\tilde{n})\langle q\rangle\triangleright\overline{\delta}} p'},\ n\in fn(q)\backslash(fn(\delta)\cup\tilde{n})$$

Table 4: Transition rules for resources and mobility.

The directed communication and movement actions are generalised to actions spanning several levels by the (*nesting*) rule. This uses an operation $n\cdot(\_)$ to prepend n to labels coming from processes enclosed into slot n defined as follows:

$$\begin{array}{lll} n\cdot(\tau)=\tau, & n\cdot(\overline{\gamma}\alpha)=\overline{n\gamma}\alpha, & n\cdot(\overline{\delta_1}\triangleright\delta_2)=\overline{n\delta_1}\triangleright n\delta_2, \\ n\cdot((p)\triangleright\delta)=(p)\triangleright n\delta, & n\cdot\big((\tilde{n})\overline{\delta}\triangleright\langle p\rangle\big)=(\tilde{n})\overline{n\delta}\triangleright\langle p\rangle, & \end{array}$$

where $n\notin\tilde{n}$. By using $n\cdot(\pi)$, we implicitly assume that π is a label of one of the kinds above. Finally, the rule *(delete)* allows a slot to be deleted.

We let π range over the complete set Π of labels used in our transition semantics, defined formally as follows.

$$\pi ::= \beta \mid (\tilde{n})\overline{\delta}\triangleright\langle p\rangle \mid (\tilde{n})\langle p\rangle\triangleright\overline{\delta} \quad \text{where} \quad \beta ::= \tau \mid \lambda \mid \overline{\natural n} \mid \overline{\delta}\alpha \mid \overline{\delta}\triangleright\delta \mid (p)\triangleright\delta \mid \delta\triangleright(p).$$

The set of *bound names* in a label π, $bn(\pi)$, is $\tilde{n}$ if π is $(\tilde{n})\overline{\delta}\triangleright\langle p\rangle$ or $(\tilde{n})\langle p\rangle\triangleright\overline{\delta}$, and $\emptyset$ if π is a β-label. The set of *free names* in π, $fn(\pi)$, are those that are not bound. We let $fn(\delta)$ denote the set of names in δ.

The rules in Table 3 and 4 then define a labelled transition system

$$(\mathcal{P}, \longrightarrow\ \subseteq \mathcal{P}\times\Pi\times\mathcal{P}).$$

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

Proposition 1 $p \xrightarrow{\tau} p'$ *if and only if* $p \searrow p'$.

Let $p \xrightarrow{\pi}$ denote $p \xrightarrow{\pi} p'$ for some p' then

Proposition 2 $p \xrightarrow{n}$ *or* $p \xrightarrow{\overline{n}}$ *if and only if* $p\downarrow n$.

5 Bisimulation Congruence

In this section we provide a labelled transition bisimulation, and prove that it coincides with the barbed bisimulation congruence.

As remarked in §4, we need to take into account that resources may be moved at arbitrary depth. As often happens in higher-order bisimulations (cf. e.g. [20, 8, 17]), we need to use an appropriate selection of destructors in order to test and assess the higher-order values exchanged by interaction. Analogously to what is done in [17] for the ambient calculus, we embody such contexts – resource receptors in our case – in the label. That is, we replace the higher-order actions $(\tilde{n})\langle p\rangle \triangleright \overline{\delta}$ and $(\tilde{n})\overline{\delta} \triangleright \langle p\rangle$ with the family of actions $\triangleright\overline{\delta}(\mathscr{D}_\delta)$ and $(\mathscr{C}_\gamma)\overline{\delta'}\triangleright(\mathscr{D}_\delta)$, respectively. The path contexts $\mathscr{D}_\delta$ and $\mathscr{C}_\gamma$ represent the surrounding slots that the resource crosses during its movement.

Definition 1 *For $\mathscr{D}_\delta$ and $\mathscr{C}_\gamma$ path contexts, we define:*

- $p \xrightarrow{\triangleright\overline{\delta}(\mathscr{D}_\delta)} (\tilde{n})\big(p' \parallel \mathscr{D}_\delta(q)\big)$ *if* $p \xrightarrow{(\tilde{n})\langle q\rangle\triangleright\overline{\delta}} p'$, *and* $fn(\mathscr{D}_\delta) \cap \tilde{n} = \emptyset$.
- $p \xrightarrow{(\mathscr{C}_\gamma)\overline{\delta'}\triangleright(\mathscr{D}_\delta)} (\tilde{n})\big(\mathscr{C}_\gamma(p') \parallel \mathscr{D}_\delta(q)\big)$ *if* $p \xrightarrow{(\tilde{n})\overline{\delta'}\triangleright\langle q\rangle} p'$, *and* $(fn(\mathscr{C}_\gamma) \cup fn(\mathscr{D}_\delta)) \cap \tilde{n} = \emptyset$.

The set of actions considered in the bisimulation game below is thus:

$$\psi ::= \beta \mid \triangleright\overline{\delta}(\mathscr{D}_\delta) \mid (\mathscr{C}_\gamma)\overline{\delta'}\triangleright(\mathscr{D}_\delta).$$

Definition 2 A *simulation* is a binary relation $\mathcal{S}$ over $\mathcal{P}$ such that whenever $p\ \mathcal{S}\ q$

$$\text{if } p \xrightarrow{\psi} p' \text{ then } \exists q \xrightarrow{\psi} q' \text{ such that } p'\ \mathcal{S}\ q'$$

$\mathcal{S}$ is a *bisimulation* if $\mathcal{S}$ and $\mathcal{S}^{-1}$ are simulations. We write $p \sim q$ if there exists a bisimulation $\mathcal{S}$ such that $p\ \mathcal{S}\ q$.

It follows immediately from the definition of bisimulation that it is an *equivalence relation*. Also, $\sim$ can be proved to be a congruence.

Theorem 1 $\sim$ *is a congruence.*

From the correspondence between τ transitions and reductions (Prop. 1 and 2), and from the fact that $\sim$ is a congruence (Thm. 1), it follows easily that $\sim$ is sound with respect to the barbed bisimulation congruence. On the other hand, the proof that $\sim_b$ is a bisimulation can be found in the appendix.

Theorem 2 $\sim_b = \sim$

It can be argued that the use of 'receptor embodying' labels and their employment in the bisimulation make the latter a form of contextual equivalence, so that proving processes bisimilarity becomes overly hard. We claim that establishing $\sim$ is still much easier than proving barbed congruence, since the needed contexts have a very simple structure. The next section aims at supporting this claim by analysing an example.

In §7 we show path contexts with paths at length most two are enough for MR *altogether*.

6 An application

In this section we prove the $\sim_b$-equivalence illustrated in §3 about the vending machine and the 'linear' behaviour of resources, taking up the opportunity to put $\sim$ at work. Exploiting Thm. 2, we prove that

$$(b)\big(a\lfloor b\rfloor \parallel !a\overline{b}\,.\,c\big) \sim_b (b)\big(a\lfloor b\rfloor \parallel a\overline{b}\,.\,c\big)$$

by showing that the two processes are $\sim$-bisimilar that, by co-induction, can be proved by proving that $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \{(p,p)\}$ is a bisimulation where

$$\begin{aligned}
\mathcal{S}_1 &= \Big\{ \big\langle (b)\big(\mathscr{C}_\gamma(\mathscr{C}_{\gamma'}(!a\bar{b}\,.\,c) \parallel \mathscr{D}_\delta(b)) \parallel p\big), (b)\big(\mathscr{C}_\gamma(\mathscr{C}_{\gamma'}(a\bar{b}\,.\,c) \parallel \mathscr{D}_\delta(b)) \parallel p\big) \big\rangle \\
&\qquad \big| b \notin fn(\mathscr{C}_\gamma) \cup fn(\mathscr{C}_{\gamma'}) \cup fn(\mathscr{D}_\delta) \cup fn(p) \Big\} \\
\mathcal{S}_2 &= \Big\{ \big\langle (b)\big(\mathscr{C}_\gamma(!a\bar{b}\,.\,c) \parallel p\big), (b)\big(\mathscr{C}_\gamma(q) \parallel p\big) \big\rangle \big| b \notin fn(\mathscr{C}_\gamma) \cup fn(p),\ q \in \{\mathbf{0}, a\bar{b}\,.\,c\} \Big\}
\end{aligned}$$

It is then relatively easy to verify that $\mathcal{S}$ is a bisimulation that contains the pair of processes under analysis. Note that it would have been considerably more difficult to prove barbed congruence directly, since that would have required considering *all* contexts, in particular arbitrary replication.

7 Bisimulation with contexts with depth at most two

In §5 we defined a bisimulation with path contexts of any depth in the labels of the lts semantics, and as already mentioned this may imply that proving bisimilarity becomes overly hard. To remedy this we provide an alternative bisimulation, $\dot{\sim}$, in which it is sufficient to operate with contexts with depth at most two. We show $\dot{\sim}$ coincides with $\sim$.

First we define at new transition relation $\mapsto \subseteq \mathcal{P} \times \Pi \times \mathcal{P}$ as the least set defined by rules similar to the ones in Tabel 3 and 4 and the rules defined by

Definition 3 *For $\mathscr{D}_m$ and $\mathscr{C}_m$ path contexts define*

- $p \overset{\triangleright\overline{m}(\mathscr{D}_m)}{\longmapsto} (\tilde{n})(p' \parallel \mathscr{D}_m(q))$ *if* $p \overset{(\tilde{n})\langle q\rangle\triangleright\overline{m}}{\longrightarrow} p', \tilde{n} \cap fn(\mathscr{D}_m) = \emptyset$
- $p \overset{\triangleright\overline{m\delta}(\mathscr{C}_m(\{n,\natural n\}\lfloor(-)\rfloor))}{\longmapsto} (\tilde{n})(p' \parallel \mathscr{C}_m(\{n,\natural n\}\lfloor q\rfloor))$
if $p \overset{(\tilde{n})\langle q\rangle\triangleright\overline{m\delta}}{\longrightarrow} p', \tilde{n} \cap fn(\mathscr{C}_m) = \emptyset, n \notin \tilde{n} \cup fn(p) \cup fn(\mathscr{C}_m) \cup fn(m\delta)$

For $\mathscr{C}_\gamma$ and $\mathscr{D}_\delta$ path contexts with γ and δ of length at most two define

- $p \overset{(\mathscr{C}_\gamma)\overline{\delta'}\triangleright(\mathscr{D}_\delta)}{\longmapsto} (\tilde{n})(\mathscr{C}_\gamma(p') \parallel \mathscr{D}_\delta(q))$ *if* $p \overset{(\tilde{n})\overline{\delta'}\triangleright\langle q\rangle}{\longrightarrow} p', \tilde{n} \cap (fn(\mathscr{C}_\gamma) \cup fn(\mathscr{D}_\delta)) = \emptyset$

Then let $\dot{\sim}$ be a standard bisimulation (as defined in Def. 2) defined over the actions φ:

$$\varphi ::= \beta \mid \triangleright\overline{m}(\mathscr{D}_m) \mid \triangleright\overline{m\delta}(\mathscr{C}_m(\{n,\natural n\}\lfloor(-)\rfloor)) \mid (\mathscr{C}_\gamma)\overline{\delta'}\triangleright(\mathscr{D}_\delta)$$

where γ and δ have length at most two. In the appendix we then prove

Theorem 3 $\sim = \dot{\sim}$.

Conclusions and Further Work

We have presented MR, a calculus of nested mobile resources designed to provide fine control on the migration and duplication of resources, as relevant for application in the analysis of mobile embedded devices. Its key properties are: the enforcement of bounded capacity for locations; the synchronous communication between co-located processes and toward children location; and the objective mobility provided by move actions. We have studied a semantic theory for MR based on a reduction semantics and a labelled transition systems, culminating in a bisimulation congruence that coincides with the barbed bisimulation. We provided examples of the expressiveness of MR and put the theory at work by exploiting the correspondence between semantics to prove a characteristic ‘linearity’ property of the calculus.

Among the open issues we plan to address in future work, we mention the study of spatial logics in the style of [7], the provision of suitable type theories, as e.g. [6, 4], to enforce communication and migration safety as well as access control. We plan to extend MR with name-passing, while maintaining the orthogonality of communication by mobility and by exchange of messages. We also plan to explore expressiveness issues by considering alternative design choice and reduced versions of MR, including the absence of communication primitives, move actions that only span single slot boundaries, disallowing scope extension through slot-boundaries, asynchronous messaging, and the cohabitation of slots with '*soft*' slots that allow copying resources. Also, we are working on an encoding of (a form of linear) capability-passing in MR, and studying the formal relationships with MR_2. We think the theory we have developed carries on smoothly to weak bisimulation; the details are under investigation.

We have applied Sewell's [22] to derive a transition semantics for a finitary fragment of MR_2 and proved that the bisimulation that so arises is included in ours. We conjecture that they coincide. It would then be interesting to carry on and recast (a larger fragment of) MR in a framework where to understand the relationship with Leifer-Milner's RPOs based bisimulation [15].

References

[1] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of POPL '01*. ACM, January 2001.

[2] ACM. *27th Annual Symposium on Principles of Programming Languages (POPL) (Boston, MA)*, January 2000.

[3] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Boxed ambients. In Benjamin Pierce, editor, *TACS'01*, volume 2215 of *Lecture Notes in Computer Science*, pages 38–63. Springer-Verlag, 2001.

[4] Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Ambient groups and mobility types. In J. van Leeuwen, O. Watanabe, M. Hagiya, P. D. Mosses, and T. Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics, Proceedings of the International IFIP Conference TCS 2000 (Sendai, Japan)*, volume 1872 of *LNCS*, pages 333–347. IFIP, Springer, August 2000.

[5] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In Maurice Nivat, editor, *Proceedings of FoSSaCS '98*, volume 1378 of *LNCS*, pages 140–155. Springer, 1998.

[6] Luca Cardelli and Andrew D. Gordon. Types for mobile ambients. In *Proceedings of POPL '99*, pages 79–92. ACM, 1999.

[7] Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proceedings of POPL '00* [2], pages 365–377.

[8] William Ferreira, Matthew Hennessy, and Alan Jeffrey. A theory of weak bisimulation for core CML. *Journal of Functional Programming*, 8(5):447–491, 1998.

[9] Cédric Fournet and Georges Gonthier. A hierarchy of equivalences for asynchronous calculi. In Larsen et al. [14], pages 844–855.

[10] Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile agents. In Ugo Montanari and Vladimiro Sassone, editors, *Proceedings of CONCUR '96*, volume 1119 of *LNCS*, pages 406–421. Springer, 1996.

[11] Jens Chr. Godskesen, Thomas Hildebrandt, and Vladimiro Sassone. A calculus of mobile resources. In *Proc. CONCUR'2002*, LNCS. Springer, 2002.

[12] Uwe Hansmann, Martin S. Nicklous, Thomas Schäck, and Frank Seliger. *Smart Card Application Development Using Java*. Springer, 200.

[13] Matthew Hennessy and James Riely. Resource access control in systems of mobile agents. In Uwe Nestmann and Benjamin C. Pierce, editors, *Proceedings of HLCL '98*, volume 16.3 of *ENTCS*, pages 3–17. Elsevier Science Publishers, 1998.

[14] Kim G. Larsen, Sven Skyum, and Glynn Winskel, editors. *25th Colloquium on Automata, Languages and Programming (ICALP) (Aalborg, Denmark)*, volume 1443 of *LNCS*. Springer, July 1998.

[15] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *Proceedings of CONCUR 2000*, volume 1877 of *LNCS*, pages 243–258. Springer, 2000.

[16] Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In *Proceedings of POPL '00* [2], pages 352–364.

[17] Massimo Merro and Matthew Hennessy. Bisimulation congruences in safe ambients. *ACM SIGPLAN Notices*, 31(1):71–80, January 2002.

[18] R. Milner. *Communication and Concurrency*. Series in Computer Science. Prentice–Hall International, 1989.

[19] Robin Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, May 1999.

[20] Davide Sangiorgi. Bisimulation for Higher-Order Process Calculi. *Information and Computation*, 131(2):141–178, 1996.

[21] Davide Sangiorgi and David Walker. *The pi-calculus: a Theory of Mobile Processes*. Cambridge Universtity Press, 2001.

[22] Peter Sewell. From rewrite rules to bisimulation congruences. In D. Sangiorgi and R. de Simone, editors, *Proceedings CONCUR'98*, volume 1466 of *LNCS*, pages 269–284. Springer, 1998.

[23] Peter Sewell. Global/local subtyping and capability inference for a distributed pi-calculus. In Larsen et al. [14], pages 695–706.

[24] Jan Vitek and Giuseppe Castagna. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, 1999.

A Proof of Proposition 1 and 2

We now establish the correspondance between the reduction semantics and the transition semantics as expressed by Proposition 1 and 2.

Lemma 1 $p \xrightarrow{\lambda} p'$ iff for some $\tilde{n}$, q, and q' where $\tilde{n} \cap fn(\lambda) = \emptyset$,

$$p \equiv (\tilde{n})(\lambda.q \parallel q') \quad \text{and} \quad p' \equiv (\tilde{n})(q \parallel q')$$

Proof: The "if" direction follows from the rule *struct* in Table 3. The "only if" direction is shown on induction on the size of the derivation of $p \xrightarrow{\lambda} p'$.

Proposition 2 follows directly from Lemma 1.

The proofs for Lemma 2 to 8 are similar to the proof of Lemma 1.

Lemma 2 $p \xrightarrow{\overline{\delta}\alpha} p'$ iff for some $\tilde{n}$, $\mathscr{C}_\delta$, q, and q' where $\tilde{n} \cap fn(\delta\alpha) = \emptyset$,

$$p \equiv (\tilde{n})(\mathscr{C}_\delta(\alpha.q) \parallel q') \quad \text{and} \quad p' \equiv (\tilde{n})(\mathscr{C}_\delta(q) \parallel q')$$

Lemma 3 $p \xrightarrow{\overline{\gamma\delta}\triangleright\gamma\delta'} p'$ iff for some $\tilde{n}$, $\mathscr{C}_\gamma$, $\mathscr{D}_\delta$, $\mathscr{D}_{\delta'}$, q, and q' where $\tilde{n} \cap fn(\overline{\gamma\delta}\triangleright\gamma\delta') = \emptyset$,

$$p \equiv (\tilde{n})(\mathscr{C}_\gamma(\mathscr{D}_\delta(q) \parallel \mathscr{D}_{\delta'}(\bullet)) \parallel q') \quad \text{and} \quad p' \equiv (\tilde{n})(\mathscr{C}_\gamma(\mathscr{D}_\delta(\bullet) \parallel \mathscr{D}_{\delta'}(q)) \parallel q')$$

Lemma 4 $p \xrightarrow{\overline{\natural n}} p'$ iff for some $\tilde{n}$, $\tilde{m}$, r, and q where $n \notin \tilde{n}$, $\natural n \in \tilde{m}$, and

$$p \equiv (\tilde{n})(\tilde{m}\lfloor r \rfloor \parallel q) \quad \text{and} \quad p \equiv (\tilde{n})q$$

Lemma 5 $p \xrightarrow{(q)\triangleright\delta} p'$ iff for some $\tilde{n}$, $\mathscr{D}_\delta$ and q' where $\tilde{n} \cap fn((q)\triangleright\delta) = \emptyset$,

$$p \equiv (\tilde{n})(\mathscr{D}_\delta(\bullet) \parallel q') \quad \text{and} \quad p' \equiv (\tilde{n})(\mathscr{D}_\delta(q) \parallel q')$$

Lemma 6 $p \xrightarrow{\delta\triangleright(q)} p'$ iff for some $\tilde{n}$, δ', $\mathscr{D}_{\delta'}$, q', and q'' where $\tilde{n} \cap fn(\delta\triangleright(q)) = \emptyset$,

$$p \equiv (\tilde{n})(\mathscr{D}_{\delta'}(\bullet) \parallel \delta\triangleright\overline{\delta'}.q' \parallel q'') \quad \text{and} \quad p' \equiv (\tilde{n})(\mathscr{D}_{\delta'}(q) \parallel q' \parallel q'')$$

Lemma 7 $p \xrightarrow{(\tilde{n})\overline{\delta}\triangleright\langle q\rangle} p'$ iff for some $\tilde{m}$, $\mathscr{D}_\delta$, and q' where $\tilde{n} \cap fn(\delta) = \tilde{m} \cap (fn(q) \cup fn(\delta)) = \emptyset$, and $\tilde{n} \subseteq fn(q)$,

$$p \equiv (\tilde{n})(\tilde{m})(\mathscr{D}_\delta(q) \parallel q') \quad \text{and} \quad p' \equiv (\tilde{m})(\mathscr{D}_\delta(\bullet) \parallel q')$$

Lemma 8 $p \xrightarrow{(\tilde{n})\langle q\rangle\triangleright\overline{\delta}} p'$ iff for some $\tilde{m}$, δ', $\mathscr{D}_{\delta'}$, q' and q'' where $\tilde{n} \cap fn(\delta) = \tilde{m} \cap (fn(q) \cup fn(\delta)) = \emptyset$, and $\tilde{n} \subseteq fn(q)$,

$$p \equiv (\tilde{n})(\tilde{m})(\mathscr{D}_{\delta'}(q) \parallel \delta'\triangleright\overline{\delta}.q' \parallel q'') \quad \text{and} \quad p' \equiv (\tilde{m})(\mathscr{D}_{\delta'}(\bullet) \parallel q' \parallel q'')$$

Lemma 9 $p \xrightarrow{\tau} p'$ implies $p \searrow p'$.

Proof: On induction on the size of the derivation of $p \xrightarrow{\tau} p'$.

Lemma 10 $p \searrow p'$ implies $p \xrightarrow{\tau} p'$.

Proof: All the rules in Table 2 implies a τ–transition and τ–transitions are preserved by evaluation contexts and structural congruense.

B Proof of Theorem 1

In this section we show that $\sim$ is a congruence. The result follows from Lemma 11 and 17 and Corollary 1 to 4.

Lemma 11 *If $p \sim q$ then $\lambda.p \sim \lambda.q$.*

Proof: Immediate.

Lemma 12 If $p \equiv (\tilde{n})(\mathscr{D}_\delta(\bullet) \parallel p')$ then $p \xrightarrow{(q)\triangleright\delta} (\tilde{n})(\mathscr{D}_\delta(q) \parallel p')$ for all q where $\tilde{n} \cap fn((q)\triangleright\delta) = \emptyset$.

Proof: By induction on the structure of p.

Lemma 13 If $p \equiv (\tilde{n})(\mathscr{D}_{\delta'}(\bullet) \parallel \delta\triangleright\overline{\delta'}.p' \parallel p'')$ then $p \xrightarrow{\delta\triangleright(q)} (\tilde{n})(\mathscr{D}_{\delta'}(q) \parallel p' \parallel p'')$ for all δ and q where $\tilde{n} \cap fn(\delta\triangleright(q)) = \emptyset$.

Proof: By induction on the structure of p.

Lemma 14 *If $p \xrightarrow{(\tilde{m})\overline{\delta}\triangleright\langle q\rangle} p'$ then $\tilde{m} \cap fn(\delta) = \emptyset$, $\tilde{m} \subseteq fn(q)$ and $fn(q) \setminus \tilde{m} \subseteq fn(p)$.*

Proof: By induction on the size of the derivation of $p \xrightarrow{(\tilde{m})\overline{\delta}\triangleright\langle q\rangle} p'$.

Lemma 15 *$p \xrightarrow{(\tilde{m})\overline{\delta}\triangleright\langle q\rangle} p'$ and $\tilde{m} \cap fn(\mathscr{C}_\gamma) = \emptyset$ iff $\mathscr{C}_\gamma(p) \xrightarrow{(\tilde{m})\overline{\gamma\delta}\triangleright\langle q\rangle} \mathscr{C}_\gamma(p')$.*

Proof: By induction on the structure of $\mathscr{C}_\gamma$.

Lemma 16 *If $p \sim p'$ and $\tilde{n} \cap (fn(p) \cup fn(p')) = \emptyset$ then $(\tilde{n})(\mathscr{C}_\gamma(p) \parallel q) \sim (\tilde{n})(\mathscr{C}_\gamma(p') \parallel q)$.*

Proof: Define

$$\mathcal{S} = \{((\tilde{n})(\mathscr{C}_\gamma(p_1) \parallel p), (\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p)) \,|\, p_1 \sim p_2, \tilde{n} \cap (fn(p_1) \cup fn(p_2)) = \emptyset\}$$

and let $\mathcal{S}$ be closed wrt. $\equiv$. We show $\mathcal{S}$ is a bisimulation. Below we only prove that $\mathcal{S}$ is a simulation, to prove that $\mathcal{S}^{-1}$ is a simulation is similar.

Let $r_1 \; \mathcal{S} \; r_2$. Then by definition of $\mathcal{S}$, $r_i \equiv (\tilde{n})(\mathscr{C}_\gamma(p_i) \parallel p)$, for some $\tilde{n}$, $\mathscr{C}_\gamma$, p_i, and p, $i = 1, 2$ where $p_1 \sim p_2$ and $\tilde{n} \cap (fn(p_1) \cup fn(p_2)) = \emptyset$. Suppose $r_1 \xrightarrow{\psi} p_1'$. Then by the rule *struct*,

$$(\tilde{n})(\mathscr{C}_\gamma(p_1) \parallel p) \xrightarrow{\psi} p_1' \tag{1}$$

In order to show that $\mathcal{S}$ is a simulation it is sufficient (because of *struct*) to show the existence of some transition $(\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p) \xrightarrow{\psi} p_2'$ such that $p_1' \; \mathcal{S} \; p_2'$. If p_1 does not contribute to the transition (1) then the result follows from the definition of $\mathcal{S}$, so in the remaining part of the proof we suppose p_1 does contribute.

Case: $\psi = \lambda$, $\psi = \overline{\delta}\alpha$, $\psi = \overline{\natural}n$, $\psi = (q)\triangleright\delta$, or $\psi = \delta\triangleright(q)$. Easy.

Case: (*co-move*) $\psi = \overline{\delta_1}\triangleright\delta_2$. If only p_1 and not $\mathscr{C}_\gamma$ nor p contributes to (1) then the result follows from $p_1 \sim p_2$. If a resource from the context $\mathscr{C}_\gamma$ or from p enters p_1 again the result follows from $p_1 \sim p_2$. The interesting case is when a resource is exited from p_1. Hence, suppose $\mathscr{C}_\gamma(p_1) \xrightarrow{\overline{\delta_1}\triangleright\delta_2} p_1''$ because of some

$$p_1 \xrightarrow{(\tilde{m}_1)\overline{\delta_1'}\triangleright\langle q_1\rangle} p_1''' \tag{2}$$

Then $\mathscr{C}_\gamma(p_1) \equiv \mathscr{C}_{\gamma_1}(\mathscr{C}_{\gamma_2}(p_1) \parallel q)$ for some $\mathscr{C}_{\gamma_2}(p_1) \parallel q \xrightarrow{\overline{\gamma_2\delta_1'}\triangleright\delta_2'} (\tilde{m}_1)(\mathscr{C}_{\gamma_2}(p_1''') \parallel q')$ where $\mathscr{C}_{\gamma_2}(p_1) \xrightarrow{(\tilde{m}_1)\overline{\gamma_2\delta_1'}\triangleright\langle q_1\rangle} \mathscr{C}_{\gamma_2}(p_1''')$, $q \xrightarrow{(q_1)\triangleright\delta_2'} q'$, $\delta_1 = \gamma_1\gamma_2\delta_1'$, $\delta_2 = \gamma_1\delta_2'$. Due to Lemma 14 and 15 and the rule *co-move* it follows that, $(fn(\mathscr{C}_{\gamma_2}) \cup fn(q)) \cap \tilde{m}_1 = \emptyset$ and $\tilde{m}_1 \subseteq fn(q_1)$.

According to Lemma 5 we have $q \equiv (\tilde{n}')(\mathscr{D}_{\delta'_2}(\bullet) \parallel q'')$ for some $\tilde{n}'$, $\mathscr{D}_{\delta'_2}$, and q'' where $fn((q_1)\triangleright\delta'_2)\cap\tilde{n}' = \emptyset$. Since $(fn(\mathscr{C}_{\gamma_2})\cup fn(\mathscr{D}_{\delta'_2}))\cap\tilde{m}_1 = \emptyset$ then from (2) we obtain $p_1 \stackrel{(\mathscr{C}_{\gamma_2})\overline{\delta'_1}\triangleright(\mathscr{D}_{\delta'_2})}{\longrightarrow} (\tilde{m}_1)(\mathscr{C}_{\gamma_2}(p'''_1) \parallel \mathscr{D}_{\delta'_2}(q_1))$.

Now, because $p_1 \sim p_2$ we have $p_2 \stackrel{(\mathscr{C}_{\gamma_2})\overline{\delta'_1}\triangleright(\mathscr{D}_{\delta'_2})}{\longrightarrow} (\tilde{m}_2)(\mathscr{C}_{\gamma_2}(p'''_2) \parallel \mathscr{D}_{\delta'_2}(q_2))$ due to some $p_2 \stackrel{(\tilde{m}_2)\overline{\delta'_1}\triangleright\langle q_2\rangle}{\longrightarrow} p'''_2$ where $(fn(\mathscr{C}_{\gamma_2}) \cup fn(\mathscr{D}_{\delta'_2})) \cap \tilde{m}_2 = \emptyset$ and $(\tilde{m}_1)(\mathscr{C}_{\gamma_2}(p'''_1) \parallel \mathscr{D}_{\delta'_2}(q_1)) \sim (\tilde{m}_2)(\mathscr{C}_{\gamma_2}(p'''_2) \parallel \mathscr{D}_{\delta'_2}(q_2))$.

Then, since $\mathscr{C}_{\gamma_2}(p_2) \stackrel{(\tilde{m}_2)\overline{\gamma_2\delta'_1}\triangleright\langle q_2\rangle}{\longrightarrow} \mathscr{C}_{\gamma_2}(p'''_2)$, due to Lemma 15, and because $q \stackrel{(q_2)\triangleright\delta'_2}{\longrightarrow} (\tilde{n}')(\mathscr{D}_{\delta'_2}(q_2) \parallel q'')$ due to Lemma 12 (assuming $\tilde{n}' \cap fn(q_2) = \emptyset$ using alpha-conversion if needed) we obtain $(\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p) \stackrel{\overline{\delta_1}\triangleright\delta_2}{\longrightarrow} p'_2$ and that $p'_1 \mathrel{\mathcal{S}} p'_2$ where $p'_i \equiv (\tilde{n})(\tilde{n}')(\mathscr{C}_{\gamma_1}((\tilde{m}_1)(\mathscr{C}_{\gamma_2}(p'''_i) \parallel \mathscr{D}_{\delta'_2}(q_1)) \parallel q'') \parallel p)$, $i = 1, 2$ (assuming $\tilde{n} \cap (fn(\mathscr{C}_{\gamma_1}) \cup fn(\mathscr{C}_{\gamma_2})) = \emptyset$ using alpha-conversion if needed).

The case where $p \stackrel{(q)\triangleright\delta_2}{\longrightarrow} p'$ and $\mathscr{C}_\gamma(p_1) \stackrel{(\tilde{m})\overline{\delta_1}\triangleright\langle q\rangle}{\longrightarrow} p''_1$ for some q and $\tilde{m}$ because $p_1 \stackrel{(\tilde{m})\overline{\delta'_1}\triangleright\langle q\rangle}{\longrightarrow} p'''_1$ for some p'''_1 and δ'_1 such that $\delta_1 = \delta'_1\gamma$ is similar to the one above.

case: $\psi = (\mathscr{C}_{\gamma'})\overline{\delta'}\triangleright(\mathscr{D}_\delta)$. We only consider the case where $\mathscr{C}_{\gamma'} \neq \mathscr{C}_\epsilon$.

By definition there exists $(\tilde{n})(\mathscr{C}_\gamma(p_1) \parallel p) \stackrel{(\tilde{m})\overline{\delta'}\triangleright\langle q\rangle}{\longrightarrow} p''_1$ such that $(fn(\mathscr{C}_{\gamma'}) \cup fn(\mathscr{D}_\delta)) \cap \tilde{m} = \emptyset$ and $p'_1 \equiv (\tilde{m})(\mathscr{C}_{\gamma'}(p''_1) \parallel \mathscr{D}_\delta(q))$ Since we are only interested in cases where p_1 contributes to the transition (1) we infer $\mathscr{C}_\gamma(p_1) \stackrel{(\tilde{m})\overline{\delta'}\triangleright\langle q\rangle}{\longrightarrow} \mathscr{C}_\gamma(p'''_1)$ for some p'''_1 such that $p''_1 \equiv (\tilde{n})(\mathscr{C}_\gamma(p'''_1) \parallel p)$ and $fn(p) \cap \tilde{m} = \emptyset$.

Because of Lemma 15 we have $p_1 \stackrel{(\tilde{m})\overline{\delta''}\triangleright\langle q\rangle}{\longrightarrow} p'''_1$ where $\delta' = \gamma\delta''$ and $\tilde{m} \cap fn(\mathscr{C}_\gamma) = \emptyset$. It then follows that $p_1 \stackrel{(\mathscr{C}_{\gamma'}(\mathscr{C}_\gamma\parallel p))\overline{\delta''}\triangleright(\mathscr{D}_\delta)}{\longrightarrow} (\tilde{m})(\mathscr{C}_{\gamma'}(\mathscr{C}_\gamma(p'''_1) \parallel p) \parallel \mathscr{D}_\delta(q))$. Now, because $p_1 \sim p_2$ we have

$$p_2 \stackrel{(\mathscr{C}_{\gamma'}(\mathscr{C}_\gamma\parallel p))\overline{\delta''}\triangleright(\mathscr{D}_\delta)}{\longrightarrow} (\tilde{m}')(\mathscr{C}_{\gamma'}(\mathscr{C}_\gamma(p'''_2) \parallel p) \parallel \mathscr{D}_\delta(q'))$$

for some $p_2 \stackrel{(\tilde{m}')\overline{\delta''}\triangleright\langle q'\rangle}{\longrightarrow} p'''_2$ where $(fn(\mathscr{C}_{\gamma'}(\mathscr{C}_\gamma \parallel p)) \cup fn(\mathscr{D}_\delta)) \cap \tilde{m}' = \emptyset$ and

$$(\tilde{m})(\mathscr{C}_{\gamma'}(\mathscr{C}_\gamma(p'''_1) \parallel p) \parallel \mathscr{D}_\delta(q)) \sim (\tilde{m}')(\mathscr{C}_{\gamma'}(\mathscr{C}_\gamma(p'''_2) \parallel p) \parallel \mathscr{D}_\delta(q'))$$

It then follows from Lemma 14 and 15 that $(\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p) \stackrel{(\tilde{m}')\overline{\delta'}\triangleright\langle q'\rangle}{\longrightarrow} p''_2$ with $p''_2 \equiv (\tilde{n})(\mathscr{C}_\gamma(p'''_2) \parallel p)$ (using alpha-conversion if needed to obtain $\tilde{n}\cap\tilde{m}' = \emptyset$). Hence we infer $(\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p) \stackrel{(\mathscr{C}_{\gamma'})\overline{\delta'}\triangleright(\mathscr{D}_\delta)}{\longrightarrow} p'_2$ where $p'_2 \equiv (\tilde{m}')(\mathscr{C}_{\gamma'}(p''_2)) \parallel \mathscr{D}_\delta(q'))$ and $p'_1 \mathrel{\mathcal{S}} p'_2$ (using alpha-conversion if needed to obtain $\tilde{n} \cap (fn(\mathscr{C}_{\gamma'}) \cup fn(\mathscr{D}_\delta)) = \emptyset$).

case: $\psi = \triangleright\overline{\delta}(\mathscr{D}_\delta)$. The case where $\mathscr{C}_\gamma = \mathscr{C}_\epsilon$ follows directly from $p_1 \sim p_2$. Hence, suppose $\mathscr{C}_\gamma \neq \mathscr{C}_\epsilon$. By definition there exists $(\tilde{n})(\mathscr{C}_\gamma(p_1) \parallel p) \stackrel{(\tilde{m})\langle q\rangle\triangleright\overline{\delta}}{\longrightarrow} p''_1$ such that $\tilde{m} \cap fn(\mathscr{D}_\delta) = \emptyset$ and $p'_1 \equiv (\tilde{m})(p''_1 \parallel \mathscr{D}_\delta(q))$.

Since by assumption $\mathscr{C}_\gamma \neq \mathscr{C}_\epsilon$ and because we are only interested in transitions (1) in which p_1 contributes it follows that $p \stackrel{\delta'\triangleright\overline{\delta}}{\longrightarrow} p'$ and $\mathscr{C}_\gamma(p_1) \stackrel{(\tilde{m})\overline{\delta'}\triangleright\langle q\rangle}{\longrightarrow} \mathscr{C}_\gamma(p'''_1)$ for some δ' and p'''_1 (using alpha-conversion if needed to make sure $\tilde{m} \cap fn(p) = \emptyset$). Hence $p''_1 \equiv (\tilde{n})(\mathscr{C}_\gamma(p'''_1) \parallel p')$.

From Lemma 15 we obtain $p_1 \stackrel{(\tilde{m})\overline{\delta''}\triangleright\langle q\rangle}{\longrightarrow} p'''_1$ where $\delta' = \gamma\delta''$ and $\tilde{m} \cap fn(\mathscr{C}_\gamma) = \emptyset$. Then by definition, $p_1 \stackrel{(\mathscr{C}_\gamma)\overline{\delta''}\triangleright(\mathscr{D}_\delta)}{\longrightarrow} (\tilde{m})(\mathscr{C}_\gamma(p'''_1) \parallel \mathscr{D}_\delta(q))$.

Now, because $p_1 \sim p_2$ it follows that $p_2 \stackrel{(\mathscr{C}_\gamma)\overline{\delta''}\triangleright(\mathscr{D}_\delta)}{\longrightarrow} (\tilde{m}')(\mathscr{C}_\gamma(p'''_2) \parallel \mathscr{D}_\delta(q'))$ for some $p_2 \stackrel{(\tilde{m}')\overline{\delta''}\triangleright\langle q'\rangle}{\longrightarrow} p'''_2$ where $(fn(\mathscr{C}_\gamma) \cup fn(\mathscr{D}_\delta)) \cap \tilde{m}' = \emptyset$ and

$$(\tilde{m})(\mathscr{C}_\gamma(p'''_1) \parallel \mathscr{D}_\delta(q)) \sim (\tilde{m}')(\mathscr{C}_\gamma(p'''_2) \parallel \mathscr{D}_\delta(q'))$$

Also, $p_2 \stackrel{(\tilde{m}')\overline{\delta''}\triangleright\langle q'\rangle}{\longrightarrow} p_2'''$ implies $(\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p) \stackrel{(\tilde{m}')\langle q'\rangle\triangleright\overline{\delta}}{\longrightarrow} p_2''$ because of Lemma 15 and 14 (and using alpha-conversion if needed to obtain $\tilde{m}' \cap \tilde{n} = \emptyset$) where $p_2'' \equiv (\tilde{n})(\mathscr{C}_\gamma(p_2''') \parallel p')$. Hence $(\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p) \stackrel{\triangleright\overline{\delta}(\mathscr{D}_\delta)}{\longrightarrow} p_2'$ where $p_2' \equiv (\tilde{m})(p_2'' \parallel \mathscr{D}_\delta(q'))$ and $p_1' \ \mathcal{S} \ p_2'$ (using alpha-conversion if needed to ensure $\tilde{n} \cap fn(\mathscr{D}_\delta) = \emptyset$).

case: $\psi = \tau$. Since we are only interested in the cases where p_1 contributes to (1) it must be that either

$$\mathscr{C}_\gamma(p_1) \parallel p \stackrel{\tau}{\longrightarrow} (\tilde{m})(\mathscr{C}_\gamma(p_1'') \parallel p') \tag{3}$$

for some $\mathscr{C}_\gamma(p_1) \stackrel{(\tilde{m})\overline{\pi}}{\longrightarrow} \mathscr{C}_\gamma(p_1'')$ and $p \stackrel{\pi}{\longrightarrow} p'$ (or dually, $p \stackrel{(\tilde{m})\overline{\pi}}{\longrightarrow} p'$ and $\mathscr{C}_\gamma(p_1) \stackrel{\pi}{\longrightarrow} \mathscr{C}_\gamma'(p_1'')$), or for some $\mathscr{C}_\gamma'$ and p_1''

$$\mathscr{C}_\gamma(p_1) \stackrel{\tau}{\longrightarrow} \mathscr{C}_\gamma'(p_1'') \tag{4}$$

The proof in case of (3) is contained in the proof for (4). Hence, suppose (4) is the case.

If $\mathscr{C}_\gamma = \mathscr{C}_\epsilon$ then the result follows directly from $p_1 \sim p_2$.

If $\mathscr{C}_\gamma \neq \mathscr{C}_\epsilon$ then it must be that $\mathscr{C}_\gamma = \mathscr{C}_{\gamma_1}(\mathscr{C}_{\gamma_2} \parallel p')$ for some $\mathscr{C}_{\gamma_2}(p_1) \parallel p' \stackrel{\tau}{\longrightarrow} (\tilde{m})(\mathscr{C}_{\gamma_2}(p_1'') \parallel p'')$ where $\mathscr{C}_{\gamma_2}(p_1) \stackrel{(\tilde{m})\overline{\pi}}{\longrightarrow} \mathscr{C}_{\gamma_2}(p_1'')$ and $p' \stackrel{\pi}{\longrightarrow} p''$ [2] for some π and $fn(p') \cap \tilde{m} = \emptyset$.

It follows that $p_1' \equiv (\tilde{n})(\mathscr{C}_{\gamma_1}(p_1''') \parallel p)$ where $p_1''' \equiv (\tilde{m})(\mathscr{C}_{\gamma_2}(p_1'') \parallel p'')$.

All cases for π except $\pi = \delta\triangleright(q)$, and when $\mathscr{C}_{\gamma_2} = \mathscr{C}_\epsilon$ also $\pi = (q)\triangleright\delta$, follows directly from $p_1 \sim p_2$. We only give the proof for $\pi = \delta\triangleright(q)$, the proof for $\pi = (q)\triangleright\delta$ is similar.

Hence, suppose $p' \stackrel{\delta\triangleright(q)}{\longrightarrow} p''$ and $\mathscr{C}_{\gamma_2}(p_1) \stackrel{(\tilde{m})\overline{\delta}\triangleright\langle q\rangle}{\longrightarrow} \mathscr{C}_{\gamma_2}(p_1'')$. From Lemma 6 we infer $p' \equiv (\tilde{n}')(\mathscr{D}_{\delta'}(\bullet) \parallel q'')$ and $p'' \equiv (\tilde{n}')(\mathscr{D}_{\delta'}(q) \parallel q''')$ for some $\tilde{n}'$, $\mathscr{D}_{\delta'}$, q'', and q''' where $fn(q) \cap \tilde{n}' = \emptyset$.

From Lemma 15 we obtain $p_1 \stackrel{(\tilde{m})\overline{\delta''}\triangleright\langle q\rangle}{\longrightarrow} p_1''$ such that $\delta = \gamma_2\delta''$ and $fn(\mathscr{C}_{\gamma_2}) \cap \tilde{m} = \emptyset$. Lemma 14 implies $\tilde{m} \subseteq fn(q)$, hence $\tilde{n}' \cap \tilde{m} = \emptyset$. Then by definition, since $(fn(\mathscr{C}_{\gamma_2}) \cup fn(\mathscr{D}_{\delta'})) \cap \tilde{m} = \emptyset$, we have

$$p_1 \stackrel{(\mathscr{C}_{\gamma_2})\overline{\delta''}\triangleright(\mathscr{D}_{\delta'})}{\longrightarrow} (\tilde{m})(\mathscr{C}_{\gamma_2}(p_1'') \parallel \mathscr{D}_{\delta'}(q))$$

Now, because $p_1 \sim p_2$ we infer, $p_2 \stackrel{(\mathscr{C}_{\gamma_2})\overline{\delta''}\triangleright(\mathscr{D}_{\delta'})}{\longrightarrow} (\tilde{m}')(\mathscr{C}_{\gamma_2}(p_2'') \parallel \mathscr{D}_{\delta'}(q'))$ for some $p_2 \stackrel{(\tilde{m}')\overline{\delta''}\triangleright\langle q'\rangle}{\longrightarrow} p_2''$ where $(fn(\mathscr{C}_{\gamma_2}) \cup fn(\mathscr{D}_{\delta'})) \cap \tilde{m}' = \emptyset$, such that

$$(\tilde{m})(\mathscr{C}_{\gamma_2}(p_1'') \parallel \mathscr{D}_{\delta'}(q)) \sim (\tilde{m}')(\mathscr{C}_{\gamma_2}(p_2'') \parallel \mathscr{D}_{\delta'}(q'))$$

Due to Lemma 15, $\mathscr{C}_{\gamma_2}(p_2) \stackrel{(\tilde{m}')\overline{\delta}\triangleright\langle q'\rangle}{\longrightarrow} \mathscr{C}_{\gamma_2}(p_2'')$. Let (using alpha-conversion if needed) $\tilde{n}' \cap fn(q') = \emptyset$. Then by Lemma 13, $p' \stackrel{\delta\triangleright(q')}{\longrightarrow} p'''$ where $p''' \equiv (\tilde{n}')(\mathscr{D}_{\delta'}(q') \parallel q''')$. Let (using alpha-conversion if needed) $\tilde{m}' \cap fn(p') = \emptyset$. We then have $\mathscr{C}_{\gamma_2}(p_2) \parallel p' \stackrel{\tau}{\longrightarrow} p_2'''$ where $p_2''' \equiv (\tilde{m}')(\mathscr{C}_{\gamma_2}(p_2'') \parallel p''')$. Hence, $(\tilde{n})(\mathscr{C}_\gamma(p_2) \parallel p) \stackrel{\tau}{\longrightarrow} p_2'$ with $p_2' \equiv (\tilde{n})(\mathscr{C}_{\gamma_1}(p_2''') \parallel p)$.

Finally, $p_1' \ \mathcal{S} \ p_2'$, using alpha-conversion if needed to obtain $\tilde{n}' \cap (fn(\mathscr{C}_i) \cup fn(p_i'')) = \emptyset$, $i = 1, 2$ and $\tilde{n}' \cap fn(p)) = \emptyset$.

Corollary 1 *If $p \sim p'$ then $p \parallel q \sim p' \parallel q$.*

Corollary 2 *If $p_1 \sim p_2$ and $q_1 \sim q_2$ then $p_1 \parallel q_1 \sim p_2 \parallel q_2$.*

Corollary 3 *If $p \sim p'$ then $(n)p \sim (n)p'$.*

Corollary 4 *If $p \sim q$ then $\tilde{n}\lfloor p \rfloor \sim \tilde{n}\lfloor q \rfloor$.*

Lemma 17 *If $p \sim q$ then $!p \sim !q$.*

[2] Or symmetrically $\mathscr{C}_{\gamma_2}(p_1) \stackrel{\pi}{\longrightarrow} \mathscr{C}_{\gamma_2}(p_1'')$ and $p' \stackrel{(\tilde{m})\overline{\pi}}{\longrightarrow} p''$. We shall not pursue this case futher, it is similar to the one we prove.

Proof: Let $p \sim q$. Observe that $p' \parallel !p \sim q' \parallel !p$ for all $p' \sim q'$.

Let $\mathcal{S}_0$ be some bisimulation such that $p' \parallel \langle !p \rangle \ \mathcal{S} \ q' \parallel \langle !p \rangle$ for all $p' \sim q'$, where we have placed a parenthesis $\langle \ \rangle$ around the two occurences of $!p$ in order to be able to identify $!p$ in any pair of the bisimulation. The syntactic labelling with $\langle \ \rangle$ has no influence on the process semantics. Formally, we may define $\langle !p \rangle \equiv p \parallel \langle !p \rangle$.

Next, define $\mathcal{S} = \{(p', q'[\langle !q \rangle / \langle !p \rangle] \mid p' \sim q'\}$ where $q'[\langle !q \rangle / \langle !p \rangle]$ is q' with $\langle !p \rangle$ replaced by $\langle !q \rangle$. We show $\mathcal{S}$ to be a bisimulation from which it follows that $\mathcal{S}' = \{(p'[!p/\langle !p \rangle], q'[!q/\langle !q \rangle] \mid p' \ \mathcal{S} \ q'\}$ is a bisimulation. Then since $p \parallel !p \ \mathcal{S}' \ q \parallel !q$ we are done considering processes up to $\equiv$.

It is sufficient to show that $\mathcal{S}$ is a bisimulation up to $\sim$ (see e.g. [18]).

Let $p' \ \mathcal{S} \ q'$. If $\langle !q \rangle$ does not occur in q' we are done since then $p' \sim q'$. Hence we assume $\langle !q \rangle$ does occur in q'. It then follows that $q' \equiv (\tilde{n})(\mathscr{C}_\gamma(\langle !q \rangle) \parallel q'')$ for some $\tilde{n}$, $\mathscr{C}_\gamma$, and q''. The reason why is that either $\langle !q \rangle$ stay put at the top level or it is moved around in which case it is always inserted into a path context. We assume $\tilde{n} \cap (fn(p) \cup fn(q)) = \emptyset$.

Suppose $p' \xrightarrow{\psi} p''$. By definition of $\mathcal{S}$, $p' \sim (\tilde{n})(\mathscr{C}_\gamma(\langle !p \rangle) \parallel q'')$ so there exists $(\tilde{n})(\mathscr{C}_\gamma(\langle !p \rangle) \parallel q'') \xrightarrow{\psi} p'''$ such that $p'' \sim p'''$.

From the semantics of replication we infer, $(\tilde{n})(\mathscr{C}_\gamma(p \parallel \ldots \parallel p \parallel \langle !p \rangle) \parallel q'') \xrightarrow{\psi} p'''$ for some number k of p's such that $(\tilde{n})(\mathscr{C}_\gamma(p \parallel \ldots \parallel p \parallel \langle !q \rangle) \parallel q'') \xrightarrow{\psi} p'''[\langle !q \rangle / \langle !p \rangle]$.

From Corollary 2 and Lemma 16 is follows that

$$(\tilde{n})(\mathscr{C}_\gamma(p \parallel \ldots \parallel p \parallel \langle !q \rangle) \parallel q'') \sim (\tilde{n})(\mathscr{C}_\gamma(q \parallel \ldots \parallel q \parallel \langle !q \rangle) \parallel q'')$$

where each of the k p's in $(\tilde{n})(\mathscr{C}_\gamma(p \parallel \ldots \parallel p \parallel \langle !q \rangle) \parallel q'')$ has been replaced by a q in $(\tilde{n})(\mathscr{C}_\gamma(q \parallel \ldots \parallel q \parallel \langle !q \rangle) \parallel q'')$. Also, $(\tilde{n})(\mathscr{C}_\gamma(\langle !q \rangle) \parallel q'') \equiv (\tilde{n})(\mathscr{C}_\gamma(q \parallel \ldots \parallel q \parallel \langle !q \rangle) \parallel q'')$.

Hence, for some $(\tilde{n})(\mathscr{C}_\gamma(\langle !q \rangle) \parallel q'') \xrightarrow{\psi} q'''$, $p'''[\langle !q \rangle / \langle !p \rangle] \sim q'''$. Therefore we conclude, $p'' \mathcal{S} p'''[\langle !q \rangle / \langle !p \rangle] \sim q'''$ and we are done.

C Proof of Theorem 2

Lemma 18 $\tilde{n}\lfloor r \rfloor \sim_b (n')\{n'\} \cup \tilde{n}\lfloor r \rfloor$ *if* $n' \notin \tilde{n} \cup fn(r)$.

Proposition 3 $\sim_b \subseteq \sim$.

Proof:We use the shorthands $\epsilon\lfloor r \rfloor$ for r and $n\gamma\lfloor r \rfloor$ for $n\lfloor \gamma\lfloor r \rfloor \rfloor$. Assume $p \sim_b q$ and $p \xrightarrow{\psi} p'$. We show by cases on ψ that there exists q' such that $q \xrightarrow{\psi} q'$ and $p' \sim_b q'$.

Case: $\psi = \tau$. Follows from Prop. 1.

Case: $\psi = \overline{\gamma}\alpha$, $\overline{\natural n}$, or $\psi = \overline{\delta_1} \triangleright \delta_2$. Consider the context $C = \overline{\psi}.a \parallel (-)$ for $a \notin fn(p) \cup fn(q)$. From $p \sim_b q$ it follows that $C(p) \sim_b C(q)$. Now, $C(p) \searrow a \parallel p'$ so there exists q'' such that $C(q) \searrow q''$ such that $a \parallel p' \sim_b q''$ and in particular $q'' \downarrow a$. Since $a \notin fn(p) \cup fn(q)$ it follows that $q'' = a \parallel q'$ and $q \xrightarrow{\psi} q'$. Now, using the context $(-) \parallel \overline{a}$ and a similar argument, it follows that $p' \sim_b q'$.

Case: $\psi = n\gamma\alpha$. Consider the context

$$C = (-) \parallel (n')(\{n, n', \natural m\}\lfloor \gamma\lfloor \overline{\alpha}.a \rfloor \rfloor \parallel n'\gamma\overline{a}.a'.\natural m.a'')$$

for $a, a', a'', n', m \notin fn(p) \cup fn(q)$. From $C(p) \sim_b C(q)$ and $C(p) \searrow p'' \searrow p'''$ for $p''' = (n')(p' \parallel \{n, n', \natural m\}\lfloor \gamma\lfloor 0 \rfloor \rfloor \parallel a'.\natural m.a'')$ it follows that $C(q) \searrow q'' \searrow q'''$ such that $p''' \sim_b q'''$ $q'' \downarrow a'$. Since $a' \notin fn(p) \cup fn(q)$ it follows that

$$q''' = (n')(q' \parallel \{n, n', \natural m\}\lfloor \gamma\lfloor z \rfloor \rfloor \parallel a'.\natural m.a'')$$

for some z, and since $a \notin fn(p) \cup fn(q)$ it follows that

$$q'' = (n')(q' \parallel \{n, n', \natural m\}\lfloor \gamma\lfloor a \rfloor \rfloor \parallel n'\gamma\overline{a}.a'.a'.\natural m.a'')$$

and $q \xrightarrow{\psi} q'$. By first using the context $(-) \parallel \overline{a'}$ and then the context $(-) \parallel \overline{a''}$ it follows as above that $(n')p' \sim_b (n')q'$ and since $(n')p' \equiv p'$ and $(n')q' \equiv q'$ we are done.

Case: $\psi = n_1\gamma_1 \triangleright \overline{n_2\gamma_2}$. Consider the context $(-) \parallel \{n_1, \natural m_1\}\lfloor \gamma_1 \lfloor a \rfloor \rfloor \parallel \{n, n_2, \natural m_2\}\lfloor \gamma_2 \lfloor \bullet \rfloor \rfloor \parallel n\gamma_2\overline{a}.\natural m_1.\natural m_2.a$ for $n, m_1, m_2, a \notin fn(p) \cup fn(q)$. From a similar argument as above it follows that $\exists q \xrightarrow{\psi} q'$ and $p' \sim_b q'$.

Case: $\psi = \natural m$. Consider the context $C = (-) \parallel \{n, \natural m\}\lfloor a \rfloor \parallel n\overline{a}.a$ for $a, n \notin fn(p) \cup fn(q)$. Now $C(p) \searrow p' \parallel n\overline{a}.a$ and $C(q) \searrow q'$ such that $p' \parallel n\overline{a}.a \sim_b q'$. The reduction $C(q) = q \parallel \{n, \natural m\}\lfloor a \rfloor \parallel n\overline{a}.a \searrow q'$ can be the result of 1) a reduction in q, 2) a reduction in C or 3) a joint reduction between q and C. We show that the first two cases lead to a contradiction with $p' \parallel n\overline{a}.a \sim_b q'$. In case 1), we get $q' = q''' \parallel \{n, \natural m\}\lfloor a \rfloor \parallel n\overline{a}.a \searrow q''' \parallel \{n, \natural m\}\lfloor 0 \rfloor \parallel a \downarrow a$, but since $n, a \notin fn(p) \cup fn(q)$ there can not exists p'' such that $p' \parallel n\overline{a}.a \searrow p''$ and $p'' \downarrow a$. In case 2) the reduction can only be $C(q) \searrow q \parallel \{n, \natural m\}\lfloor 0 \rfloor \parallel a = q'$, but then $q' \downarrow a$ and it is not the case that $p' \parallel n\overline{a}.a \downarrow a$. So we can conclude that the reduction $C(q) \searrow q'$ is a joint reduction between q and C, which can only be caused by $q \xrightarrow{\natural m} q''$ and $q' = q'' \parallel n\overline{a}.a$. Proceeding as in the above cases it then follows that $\exists q \xrightarrow{\psi} q'$ and $p' \sim_b q'$.

Case: $\psi = (p'') \triangleright \delta$. Using the context $(-) \parallel (m)(m \triangleright \overline{\delta} \,.\, \natural m \,.\, a \parallel \{m, \natural m\}\lfloor p'' \rfloor)$, for $a, m \notin fn(p) \cup fn(q)$, it follows that $\exists q \xrightarrow{\psi} q'$ and $p' \sim_b q'$.

Case: $\psi = n\gamma \triangleright (p'')$. Using the context

$$C = (-) \parallel \{n, n', m\}\lfloor \gamma \lfloor p'' \rfloor \rfloor \parallel n'' \triangleright \overline{n'\gamma}.\natural m.\natural m'.a \parallel \{n'', \natural m'\}\lfloor 0 \rfloor$$

for $a, n', n'', m, m' \notin fn(p) \cup fn(p'') \cup fn(q)$, it follows that $\exists q \xrightarrow{\psi} q'$ and $p' \sim_b q'$.

Case: $\psi = \triangleright \overline{\delta}(\mathscr{D}_\delta)$ for $\mathscr{D}_\delta = \mathscr{C}_\gamma(\tilde{n}\lfloor (-) \rfloor)$. Then $p \xrightarrow{(\tilde{n})\langle p'' \rangle \triangleright \overline{\delta}} p'''$, $fn(\mathscr{D}_\delta) \cap \tilde{n} = \emptyset$ and $p' = (\tilde{n})(p''' \parallel \mathscr{D}_\delta(p''))$. Using Lem. 18 and the context

$$C = (-) \parallel (n'')(\mathscr{C}_\gamma(\{n', \natural m'\} \cup \tilde{n}\lfloor \bullet \rfloor \parallel n' \triangleright \overline{n''}.\natural m'.a \parallel \{n''\} \cup \tilde{n}\lfloor \bullet \rfloor))$$

for $a, n, n', n'', m' \notin fn(p) \cup fn(q)$ it follows that $\exists q \xrightarrow{(\tilde{m})\langle q'' \rangle \triangleright \overline{\delta}} q'''$ such that $fn(\mathscr{D}_\delta) \cap \tilde{m} = \emptyset$ and $(\tilde{n})(p''' \parallel \mathscr{D}_\delta(p'')) \sim_b (\tilde{m})(q''' \parallel \mathscr{D}_\delta(q''))$, and so $\exists q \xrightarrow{\triangleright \overline{\delta}(\mathscr{D}_\delta)} q' = (\tilde{m})(q''' \parallel \mathscr{D}_\delta(q''))$ such that $p' \sim_b q'$.

Case: $\psi = (\mathscr{C}_\gamma)\overline{\delta} \triangleright (\mathscr{D}_{\delta'})$, for $\mathscr{D}_{\delta'} = \mathscr{C}_{\gamma'}(\tilde{n}\lfloor (-) \rfloor)$. Then $p \xrightarrow{(\tilde{n})\overline{\delta} \triangleright \langle p'' \rangle} p'''$ such that $(fn(\mathscr{C}_\gamma) \cup fn(\mathscr{D}_{\delta'})) \cap \tilde{n} = \emptyset$ and $p' = (\tilde{n})(\mathscr{C}_\gamma(p''') \parallel \mathscr{D}_\delta(p''))$.

Using Lem. 18 and the context

$$(n')(n)\big(\mathscr{C}_\gamma((-) \parallel \delta \triangleright \overline{n}.a.\natural m'.a' \parallel \{n, \natural m'\}\lfloor \bullet \rfloor) \parallel \gamma n \triangleright \overline{\gamma' n'}.a'' \parallel \mathscr{C}_{\gamma'}(\{n'\} \cup \tilde{n}\lfloor \bullet \rfloor)\big)$$

for $a, a', a'', n, n', m' \notin fn(p) \cup fn(q)$, it follows that $\exists q \xrightarrow{(\tilde{m})\overline{\delta} \triangleright \langle q'' \rangle} q'''$ such that $(fn(\mathscr{C}_\gamma) \cup fn(\mathscr{D}_{\delta'})) \cap \tilde{m} = \emptyset$ and $(\tilde{n})(\mathscr{C}_\gamma(p''') \parallel \mathscr{D}_\delta(p'')) \sim_b (\tilde{m})(\mathscr{C}_\gamma(q''') \parallel \mathscr{D}_\delta(q''))$, so $\exists q \xrightarrow{(\mathscr{C}_\gamma)\overline{\delta} \triangleright (\mathscr{D}_{\delta'})} q' = (\tilde{m})(\mathscr{C}_\gamma(q''') \parallel \mathscr{D}_\delta(q''))$ such that $p' \sim_b q'$.

D Proof of Theorem 3

Lemma 19 $\sim \subseteq \dot{\sim}$

Proof: Suppose $p_1 \sim p_2$. We only show the case where $p_1 \xmapsto{\varphi} p_1'$ implies $p_2 \xmapsto{\varphi} p_2'$ for some p_2' such that $p_1' \sim p_2'$. The symmetric case is similar.

The case where $\varphi = \beta$ follows immediately from $p_1 \sim p_2$, the same does the case $\varphi = \triangleright \overline{m}(\mathscr{D}_m)$ and the case $\varphi = (\mathscr{C}_\gamma)\overline{\delta'} \triangleright (\mathscr{D}_\delta)$ where γ and δ have length at most two.

Suppose $p_1 \stackrel{\triangleright\overline{m\delta}(\mathscr{C}_m(\{n,\natural n\}\lfloor(-)\rfloor))}{\longmapsto} (\tilde{n})(p_1' \parallel \mathscr{C}_m(\{n,\natural n\}\lfloor q \rfloor))$ because $p_1 \stackrel{(\tilde{n})\langle q\rangle\triangleright\overline{m\delta}}{\longrightarrow} p_1'$ and where $\tilde{n} \cap fn(\mathscr{C}_m) = \emptyset$ and $n \notin \tilde{n} \cup fn(p_1) \cup fn(\mathscr{C}_m) \cup fn(m\delta)$. From Lemma 8 it follows that $\tilde{n} \cap fn(m\delta) = \emptyset$.

Let $\mathscr{C}_m(\{n,\natural n\}\lfloor(-)\rfloor) = \tilde{m}'\lfloor\{n,\natural n\}\lfloor(-)\rfloor \parallel p\rfloor$ and let $\delta = a_1 \ldots a_k$. Define

$$\mathscr{D}_\delta = \{a_1,\natural a_1\}\lfloor\{a_2,\natural a_2\}\lfloor \ldots \{a_k,\natural a_k\}\lfloor(-)\rfloor \parallel \natural a_k.b_k \ldots \rfloor \parallel \natural a_2.b_2\rfloor$$

and $p' = \delta\triangleright\overline{n}.\natural a_1.b_1$ where all b_i, $i = 1,\ldots,k$, are fresh.

Now, let $\mathscr{D}_{m\delta} = \tilde{m}'\lfloor\{n,\natural n\}\lfloor\bullet\rfloor \parallel p \parallel \mathscr{D}_\delta \parallel p'\rfloor$. Then, because $p_1 \stackrel{(\tilde{n})\langle q\rangle\triangleright\overline{m\delta}}{\longrightarrow} p_1'$ and since $fn(\mathscr{D}_{m\delta}) \cap \tilde{n} = \emptyset$ it follows that $p_1 \stackrel{\triangleright\overline{m\delta}(\mathscr{D}_{m\delta})}{\longrightarrow} (\tilde{n})(p_1' \parallel \mathscr{D}_{m\delta}(q))$. From $p_1 \sim p_2$ we have $p_2 \stackrel{\triangleright\overline{m\delta}(\mathscr{D}_{m\delta})}{\longrightarrow} (\tilde{m})(p_2' \parallel \mathscr{D}_{m\delta}(q'))$ for some $p_2 \stackrel{(\tilde{m})\langle q'\rangle\triangleright\overline{m\delta}}{\longrightarrow} p_2'$ where $fn(\mathscr{D}_{m\delta}) \cap \tilde{m} = \emptyset$ and

$$(\tilde{n})(p_1' \parallel \mathscr{D}_{m\delta}(q)) \sim (\tilde{m})(p_2' \parallel \mathscr{D}_{m\delta}(q')) \tag{5}$$

Hence we conclude that $p_2 \stackrel{\triangleright\overline{m\delta}(\mathscr{C}_m(\{n,\natural n\}\lfloor(-)\rfloor))}{\longmapsto} (\tilde{m})(p_2' \parallel \mathscr{C}_m(\{n,\natural n\}\lfloor q'\rfloor))$, because $\tilde{m} \cap fn(\mathscr{C}_m(\{n,\natural n\}\lfloor(-)\rfloor)) = \emptyset$, and from (5) we infer $(\tilde{n})(p_1' \parallel \mathscr{C}_m(\{n,\natural n\}\lfloor q\rfloor)) \sim (\tilde{m})(p_2' \parallel \mathscr{C}_m(\{n,\natural n\}\lfloor q'\rfloor))$.

Lemma 20 $\dot{\sim} \subseteq \sim$

Proof: Suppose $p_1 \dot{\sim} p_2$. We only show the case where $p_1 \stackrel{\psi}{\longrightarrow} p_1'$ implies $p_2 \stackrel{\psi}{\longrightarrow} p_2'$ for some p_2' such that $p_1' \dot{\sim} p_2'$. The symmetric case is similar.

The case where $\psi = \beta$ follows immediately from $p_1 \dot{\sim} p_2$, and so does the case $\psi = \triangleright\overline{m}(\mathscr{D}_m)$.

Suppose $p_1 \stackrel{\triangleright\overline{m\delta}(\mathscr{D}_{m\delta})}{\longrightarrow} (\tilde{n})(p_1' \parallel \mathscr{D}_{m\delta}(q))$ because $p_1 \stackrel{(\tilde{n})\langle q\rangle\triangleright\overline{m\delta}}{\longrightarrow} p_1'$ and $\tilde{n} \cap fn(\mathscr{D}_{m\delta}) = \emptyset$. From Lemma 8 it follows that $\tilde{n} \cap fn(m\delta) = \emptyset$.

Let $\mathscr{D}_{m\delta} = \tilde{m}'\lfloor\mathscr{D}_\delta \parallel p\rfloor$ and define $\mathscr{C}_m(\{a,\natural a\}\lfloor(-)\rfloor) = \tilde{m}'\lfloor\{a,\natural a\}\lfloor(-)\rfloor \parallel p'\rfloor$ where

$$p' = \mathscr{D}_\delta(\bullet) \parallel a\triangleright\overline{\delta}.\natural a.b \parallel p$$

and where a and b are fresh.

Now, because $p_1 \stackrel{(\tilde{n})\langle q\rangle\triangleright\overline{m\delta}}{\longrightarrow} p_1'$ and since $\tilde{n} \cap fn(\mathscr{C}_m(\{a,\natural a\}\lfloor(-)\rfloor)) = \emptyset$ it follows that $p_1 \stackrel{\triangleright\overline{m\delta}(\mathscr{C}_m(\{a,\natural a\}\lfloor(-)\rfloor))}{\longmapsto} (\tilde{n})(p_1' \parallel \mathscr{C}_m(\{a,\natural a\}\lfloor q\rfloor))$. From the assumption $p_1 \dot{\sim} p_2$ we obtain that $p_2 \stackrel{\triangleright\overline{m\delta}(\mathscr{C}_m(\{a,\natural a\}\lfloor(-)\rfloor))}{\longmapsto} (\tilde{m})(p_2' \parallel \mathscr{C}_m(\{a,\natural a\}\lfloor q'\rfloor))$ for some $p_2 \stackrel{(\tilde{m})\langle q'\rangle\triangleright\overline{m\delta}}{\longrightarrow} p_2'$ where $\tilde{m} \cap fn(\mathscr{C}_m(\{a,\natural a\}\lfloor(-)\rfloor)) = \emptyset$ and

$$(\tilde{n})(p_1' \parallel \mathscr{C}_m(\{a,\natural a\}\lfloor q\rfloor)) \dot{\sim} (\tilde{m})(p_2' \parallel \mathscr{C}_m(\{a,\natural a\}\lfloor q'\rfloor)) \tag{6}$$

Clearly, $\tilde{m} \cap fn(\mathscr{D}_{m\delta}) = \emptyset$ so we conclude that $p_2 \stackrel{\triangleright\overline{m\delta}(\mathscr{D}_{m\delta})}{\longrightarrow} (\tilde{m})(p_2' \parallel \mathscr{D}_{m\delta}(q'))$ and from (6) we infer that $(\tilde{n})(p_1' \parallel \mathscr{D}_{m\delta}(q)) \dot{\sim} (\tilde{m})(p_2' \parallel \mathscr{D}_{m\delta}(q'))$.

The case where $\psi = (\mathscr{C}_\gamma)\overline{\delta'}\triangleright(\mathscr{D}_\delta)$ follows from $p_1 \dot{\sim} p_2$ if γ and δ has length at most two. Otherwise, the case is proven using the same technique as above.