Mobility control via passports (Extended abstract)

Samuel Hym

Mobility Control via Passports
(Extended Abstract)

Samuel Hym

PPS, Université Paris Diderot (Paris 7) & CNRS

Abstract. Dπ is a simple distributed extension of the π-calculus in which agents are explicitly located, and may use an explicit migration construct to move between locations.
We introduce passports to control those migrations; in order to gain access to a location agents are now expected to show some credentials, granted by the destination location. Passports are tied to specific locations, from which migration is permitted. We describe a type system for these passports, which includes a novel use of dependent types, and prove that well-typing enforces the desired behaviour in migrating processes.
Passports allow locations to control incoming processes. This induces major modifications to the observations which can be made of agent-based systems. Using the type system we describe these observations, and use them to build a *loyal* notion of observational equivalence. Finally we provide a complete proof technique in the form of a bisimilarity for establishing equivalences between systems.
Key words: process calculus; control of agent migrations; distributed computation; observational equivalence

1 Introduction

Dπ [1] is a process calculus designed to reason about distribution of computation. It is built as a simple extension of the π-calculus in which agents are explicitly located without nesting so that a system might look like:

$$l_1[\![c\,!\,\langle b\rangle\,P_1]\!] \mid l_2[\![P_2]\!] \mid (\mathsf{new}\,a : \mathsf{E})(l_3[\![P_3]\!] \mid l_1[\![c\,?\,(x : \mathsf{T})\,P_4]\!])$$

where the l_i are location names and the P_i are processes located in one of those locations. Here, P_1 and P_4 are placed in the same location l_1, even if they are scattered in the term. Channels also are distributed: one channel is anchored in exactly one location: two processes must be in the same location to communicate. In our example, the system can evolve into

$$l_1[\![P_1]\!] \mid l_2[\![P_2]\!] \mid (\mathsf{new}\,a : \mathsf{E})(l_3[\![P_3]\!] \mid l_1[\![P_4\{^b/_x\}]\!])$$

when P_1 and P_4 communicate. This makes Dπ a streamlined distributed version of the π-calculus, which allows to concentrate our attention on agent migrations.

Dπ agents can trigger their migration from their current location, say k, to the location l via the primitive

$$\mathsf{goto}_p\,l$$

The p, added by the present work, is a *passport* which must match the actual migration attempted, from k to l. Those passports are permits, requested whenever trying to enter a location and therefore allowing that location to control which processes should be granted access.

Some other approaches to control migrations have been investigated in process calculi. In Ambients-related calculi, the migrations are particularly hard to control so many works tried to address this problem: in Safe Ambients ([2]), the destination location must grant access to incoming ambients by using a *co-capability*. These co-capabilities have been enriched in [3] with *passwords*: the password used to migrate is syntactically checked at *runtime* when the migration is to be granted. This idea of passwords was pursued in the NBA calculus ([4]) which combines it with another choice to control behaviours of ambients: communications across boundaries are allowed so that the troublesome open primitive from the original Mobile Ambients can be removed without impeding the expressivity of the calculus. This second approach was also used in different hierarchical calculi like Seal [5] or Kell [6].

In non-hierarchical calculi, we have a better handle over migrating behaviours so that more powerful techniques can be employed, for instance leveraging type systems. The present work is inspired in that direction by [7]. In that work, access to a location is a capability tied to that location via its type: access is either always granted or always denied depending on the type used when the location is generated. Of course, even when access is granted, the location name can then be transmitted without giving access; nevertheless, this setting lacks flexibility. In the present work, we refine that approach to be able to grant access selectively, depending on the origin location and to authorise such access migrations dynamically, namely after the generation of the location itself. That is why *passport* names are added to the calculus to bear those authorisations. We chose to use regular names to preserve the *homogeneity* of the calculus: in particular, they can be exchanged over channels and their scopes are dealt with in precisely the same way as any other name, including for their extrusions. Types are then used to tie rights to the names of the passports: for instance, the type $l \mapsto k$ is attached to some passport granting access to k from l. The typing system will therefore have to include dependent types to describe the link between passports and the locations they are attached to[1]. Fortunately, those dependent types bring little extra complexity to the type system itself and to the proofs of its properties, including subject reduction. What is more, this approach to tie rights to types provides type-based tools and techniques to reason about security properties. We also argue that relying on names to bear access rights gives a good handle to control those rights.

Other type systems have been used to control mobility in Dπ-based calculi. In [9], access requires the knowledge of a *port* which also governs subsequent re-

[1] Since a passport must grant access to only *one* location, the one which delivered that passport, using "groups" ([8]) to try and avoid dependent types would fall back on defining one group per location. So it would only reduce the expressivity of the language.

source accesses by typing the migrating processes, using for this complex process types developed in particular in [10]. This approach is strongly constraining processes and requires higher order actions. The present work provides a first-order theory that aims at becoming a foundation for a fine-grained control of comparable power to [9]: while the passport types developed hereafter correspond to a simple mobility control, they should leave room to extensions to control resource accesses.

In [11], access to locations and resources is conditioned by policies based on the history of migrations of the agent. In the present work, the only location of the history taken into account to grant access is the origin of the migrating process: we will define a simple setting in which it is possible to describe "trust sub-networks" such as an intranet. Furthermore, the origin of a process seems easier to assert realistically than its full history. The setting we propose here relies on a simple view of trust: when a location l expresses its trust into another one k (through a passport valid from k), it also decides to trust k not to relay any dangerous process from another location.

In the following, we will investigate the notion of typed observational equivalence inherited from [12]. The founding intuition of observational equivalences is to distinguish two systems only when it is possible to observe a difference between them through a series of interactions. In a typed observational equivalence where types represents permissions, the *barbs* the observer is allowed to see are conditioned by the permissions he managed to get access to. Since permissions are represented by types, a normal type environment is used to describe the observer's rights.

Control of migrations has a great impact on the set of possible observations: since all interactions are performed over located channels, permissions to access these locations, *i.e.* passports, are mandatory to observe anything if the observer abides by the rules. We will therefore introduce an intuitive typed congruence that takes into account the migration rights of such a *loyal* observer. We argue that relying on names to bear access rights also gives a clean equivalence theory, in which the rights granted to the observer are easily expressed. As usual, the closure of the equivalence over all admissible contexts makes this equivalence intractable. So we will provide an alternative coinductive definition for this equivalence as a bisimilarity based on *actions* which identify the possible interactions between the system and its observer. This alternative definition reveals a difficulty arising from dependent types: as an artefact of dependencies, some name scopes must be opened even when the name itself is not revealed to the observer.

2 Typed Dπ with Passports

We present here a stripped-down version of the Dπ-calculus to focus on migration control. A more complete description of the specificities of the calculus we use here can be found in the long version of this work [13] or in [14]. Most of it is inherited from previous works, like [15], so we will insist mostly on the differences.

Fig. 1 Syntax for the Dπ-calculus

M ::=		*Systems*
	$l[\![P]\!]$	Located process
	$M_1 \mid M_2$	Parallel composition
	$(\mathsf{new}\, a : \mathsf{E})\, M$	Name scope
	$\mathbf{0}$	Inactive system
P ::=		*Processes*
	$u\,!\,\langle V \rangle\, P$	Writing on channel
	$u\,?\,(X : \mathsf{T})\, P$	Reading on channel
	$\mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$	Condition
	$\mathsf{goto}_v\ u.\ P$	Migration
	$\mathsf{newchan}\ c : \mathsf{C}\ \mathsf{in}\ P$	Channel generation
	$\mathsf{newloc}\ l, (\vec{c}), (\vec{p}), (\vec{q}) : \mathsf{L}\ \mathsf{with}\ P_l\ \mathsf{in}\ P$	Location generation
	$\mathsf{newpass}\ p\ \mathsf{from}\ \tilde{u}^\star\ \mathsf{in}\ P$	Passport generation
	$P_1 \mid P_2$	Parallel composition
	$*P$	Replication
	stop	Termination

Processes are described using *names* (usually written $a, b, \ldots$, reserving c, d for *channel* names, k, l for *locations* and p, q for *passports*) and *variables* (usually written $x, y, \ldots$). When both names and variables can be used, we will talk of *identifiers* and write them $u, v, \ldots$. We will write $\tilde{u}$ for a set of identifiers and $\vec{u}$ for a tuple. We will also write $\tilde{u}^\star$ when either $\tilde{u}$ or $\star$ is expected (the meaning of $\star$ will be explained shortly). Finally, we will use capital letters when tuples are allowed so V can represent $(v_1, (v_2, v_3), v_4)$ or any other *value*, composed of identifiers and X any *pattern*, composed of variables.

The syntax of Dπ is given in Figure 1. Our contributions are:

- The migration construct $\mathsf{goto}_v\ u$ now mentions the *passport* v to get access to the location u.
- The new construct to generate passports, newpass, provides two kinds of origin control:
  - passports that allow migration from a given set of originating locations $\tilde{u}$ are created by $\mathsf{newpass}\ p\ \mathsf{from}\ \tilde{u}$; thus a location can express its trust in the sub-network $\tilde{u}$: every process using p will be granted access from any location in $\tilde{u}$;
  - *universal passports*, that allow migration from any location (for instance when describing the behaviour of a public server accepting requests from anywhere) are created by $\mathsf{newpass}\ p\ \mathsf{from}\ \star$.

  Of course, the location a passport grants access to is the location where the passport is generated: that is the only way to allow locations to control incoming processes.
- The construct to generate new locations, newloc, is enriched: passports to access the new location (*child*) or the location where the construct is called (*mother*) can be generated on the fly. This is the only way to model all the

Fig. 2 Reduction semantics, extracts

$$\text{(R-GOTO)}\quad l[\![\mathsf{goto}_p\ k.\ P]\!] \longrightarrow k[\![P]\!]$$

$$\text{(R-NEWLOC)}\quad l[\![\mathsf{newloc}\ k, (\vec{c}), (\vec{p}), (\vec{q}) : \textstyle\sum x : \text{LOC}.\ \mathsf{T}\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!] \longrightarrow (\mathsf{new}\langle k, ((\vec{c}), (\vec{p}), (\vec{q})) : \mathsf{T}\{{}^{l}/_{x}\}\rangle)(k[\![P_k]\!] \mid l[\![P]\!])$$

$$\text{(R-NEWPASS)}\quad l[\![\mathsf{newpass}\ p\ \mathsf{from}\ \tilde{k}^{\star}\ \mathsf{in}\ P]\!] \longrightarrow (\mathsf{new}\ p : \tilde{k}^{\star} \mapsto l)\, l[\![P]\!]$$

$$\text{(R-COMM)}\quad l[\![a\,!\,\langle V\rangle\ P_1]\!] \mid l[\![a\,?\,(X : \mathsf{T})\ P_2]\!] \longrightarrow l[\![P_1]\!] \mid l[\![P_2\{{}^{V}/_{X}\}]\!]$$

possible situations (the child location granting access to processes from the mother location; or vice versa; and any other variation). Indeed, if passports to access the child were always created from inside the child itself, some passports granting access from the child would be needed to export them...

Since passports allow a location to accept processes depending on their origin, they can be delivered for specific communications, for instance the response awaited from a server: in

$$cl[\![\mathsf{newpass}\ pass\ \mathsf{from}\ sv\ \mathsf{in} \\ \mathsf{goto}_{p_{sv}}\ sv.\ req\,!\,\langle(sv, cl), (quest, res, pass)\rangle \mid \ldots res\,?\,(x)\ P]\!]$$

the client cl generates a passport specific to the server sv before going there and requesting some computation while waiting for the result in cl. The corresponding server might look like:

$$sv[\![*req\,?\,((x_{sv}, x_{cl}), (x_{quest}, x_{res}, x_{pass}) : \mathsf{T}) \cdots \mathsf{goto}_{x_{pass}}\ x_{cl}.\ x_{res}\,!\,\langle r\rangle]\!]$$

Let us consider now the semantics associated with the calculus: the most interesting rules concerning this work are given in Figure 2 (the full set is provided in appendix). Those rules are fairly unsurprising since passports are homogeneously added to the calculus. In the reduction rule for the migration (R-GOTO), the passport involved is simply ignored: the verification of the passport will be performed using *types*. In the two rules for generation, types are instantiated in a similar way to what is usually done for channels: when passports are actually generated in (R-NEWPASS), they are tied to the location to which they will grant access, by getting the type $\tilde{k} \mapsto l$ (from $\tilde{k}$ to l).

The types we can associate with identifiers or values are summed up in Figure 3. Two major modifications are made here. Firstly, we introduce a new type for passport: $\tilde{u} \mapsto v$ will be the type of a passport to access v from one of the locations in $\tilde{u}$ and $\star \mapsto v$ of a universal passport to v. Secondly, we add a dependent sum type for values that are transmitted over channels: since the type for a passport mentions the names of the source and target locations, the dependent sum provides a way to send those names (locations and passport), packed together. They are also used to describe the tie between the locations and the passports in the newloc construct. So that the system

$$l[\![\mathsf{newloc}\ k, p, q : \textstyle\sum x : \text{LOC}.\ \sum y : \text{LOC}.\ x \mapsto y, y \mapsto x\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!]$$

Fig. 3 Syntax of types

$\mathsf{E} ::=$	*Identifiers types*
$\textsc{loc}$	Location
$\textsc{r}\langle \mathsf{T}_1 \rangle_{@} u$	Channel in location u: right to read values of type T_1
$\textsc{w}\langle \mathsf{T}_2 \rangle_{@} u$	Channel in location u: right to write values of type T_2
$\textsc{rw}\langle \mathsf{T}_1, \mathsf{T}_2 \rangle_{@} u$	Intersection of the two previous types
$\tilde{u}^\star \mapsto v$	Passport
$\mathsf{T} ::=$	*Transmissible values types*
E	Identifier
$(\mathsf{T}_1, \ldots, \mathsf{T}_n)$	Tuple
$\sum \vec{x} : \vec{\textsc{loc}}.\ \mathsf{T}$	Dependent sum
$\mathsf{L} ::=$	*Types to declare locations*
$\sum x : \textsc{loc}.\ \sum y : \textsc{loc}.\ (\mathsf{C}_1{}_{@} y, \ldots), (\tilde{u}_1^\star \mapsto y, \ldots), (\tilde{v}_1^\star \mapsto x, \ldots)$	

reduces into

$$(\mathsf{new}\, k : \textsc{loc})\,(\mathsf{new}\, p : l \mapsto k)\,(\mathsf{new}\, q : k \mapsto l)\, k[\![P_k]\!] \mid l[\![P]\!]$$

For space reasons we refer the reader to the long version [13] for the full explanations of the L types, in particular their unwinding into simple types for every identifiers, as they bring little insight on the actual passports.

Again, we provide here only a simple presentation of the set of types to focus on passports (in particular, we got rid of the recursive types which are completely orthogonal to passports types; see [16] for a detailed account of recursive types). The main property of interest about passport types is subtyping: for instance, a universal passport to access l allows to come from anywhere so should be a subtype of any passport to l. The following inference rules sum up subtyping for passports:

$$(\textsc{sr-pass}) \quad \frac{\tilde{u}' \subseteq \tilde{u}}{\tilde{u} \mapsto v <: \tilde{u}' \mapsto v} \qquad (\textsc{sr-pass-}^*) \quad \star \mapsto v <: \tilde{u}^\star \mapsto v$$

We refer the reader to previous works (in particular [16]) for a complete presentation of subtyping in Dπ. Let us simply state here that the property of partial meets is preserved in this setting:

Theorem 1 (Partial meets). *Any two types sharing a subtype have a meet.*

As usual, the type system relies on *type environments*, written Γ, Φ, Ω, which are lists of hypotheses, *i.e.* associations of types to identifiers, for instance $l : \textsc{loc}, k : \textsc{loc}, p : \star \mapsto k, \ldots$ Those environments are used to prove typing judgements like $\Gamma \vdash p : l \mapsto k$, which states that p can be used to migrate from l to k according to Γ, or $\Gamma \vdash_l P$, which states that running the process P in location l will require at most the permissions contained in Γ. These judgments are

derived using inference rules: we give here only some rules relevant to passports.

$$\text{(T-GOTO)}\quad \frac{\Gamma \vdash u : w \mapsto v \qquad \Gamma \vdash_v P}{\Gamma \vdash_w \mathsf{goto}_u\, v.\ P} \qquad\qquad \text{(T-NEWPASS)}\quad \frac{\Gamma \vdash \tilde{u} : \text{L}\tilde{\text{O}}\text{C} \qquad \Gamma; p : \tilde{u}^\star \mapsto w \vdash_w P}{\Gamma \vdash_w \mathsf{newpass}\, p\, \mathsf{from}\, \tilde{u}^\star\, \mathsf{in}\, P}$$

Those rules are fairly straightforward and provide the two expected theorems about the type system: subject reduction and type safety. Let us state here only the important part for passports using an *erroneous reduction* of a system M, written $M \xrightarrow{\text{err}}_\Gamma$, defined by the reduction $l[\![\mathsf{goto}_p\, k.\ P]\!] \xrightarrow{\text{err}}_\Gamma$ in any context whenever $\Gamma \nvdash p : l \mapsto k$. A really simple case of erroneous reduction might look like this (this reduction is erroneous in any well-formed environment, in particular the empty one):

$$(\mathsf{new}\, l_1, l_2, l_3 : \text{LOC})\,(\mathsf{new}\, p : l_1 \mapsto l_2)\, l_1[\![\mathsf{goto}_p\, l_3.\ \mathbf{0}]\!] \xrightarrow{\text{err}}_\emptyset$$

Theorem 2. *$\Gamma \vdash M$ and $M \longrightarrow^* N$ imply $N \not\xrightarrow{\text{err}}_\Gamma$.*

3 Loyal Observational Equivalence

The main goal of passports is to allow a location to control the processes it accepts. Naturally, this implies that the *observable* behaviour of a system depends on the actual authorisations the *observer* is granted. Let us then define an equivalence that takes passports into account drawing inspiration from [17].

For this, we will describe explicitly the knowledge of the observer, including his passports, using a type environment written Ω. This type environment thus describes the observations that can be performed, in a similar way to the knowledge-indexed relations defined in [7]. The basic observations must be interactions with the studied system, *i.e.* communications over some channels. Since channels are located, this will be possible only when the observer is granted access to their location. To actually allow the system to "choose" which locations should be reachable, we decided to place the observer into a *fresh* location. This implies that the only *directly reachable* locations are the destinations of the *universal passports* in Ω. So we define *barbs* thus:

Definition 1 (Barbs). *M shows a barb on c to Ω, written $\Omega \rhd M \Downarrow c$, whenever there exist a location l and a passport p such that: $\Omega \vdash p : \star \mapsto l$; $\Omega \vdash c : \text{R}\langle \mathsf{T} \rangle_{@l}$, for some type T; and there exist some P, M' and $(\vec{a} : \vec{\mathsf{E}})$ with $c, l \notin \vec{a}$ and such that $M \longrightarrow^* \equiv (\mathsf{new}\, \vec{a} : \vec{\mathsf{E}})(M' \mid l[\![c\,!\,\langle V \rangle\ P]\!])$.*

Note that the only control performed in this definition is whether the observer is able to reach the location where the interaction takes place: since our mobility control happens only when *entering* a location, it will always be possible to report the observation in the observer's home location.

Some observer knowing Ω will be able to distinguish two systems as soon as they show different sets of barbs. To get an equivalence out of this simple property, the observer is usually allowed to test the system by putting it in any context in order to eventually obtain a distinguishing barb. In our setting, we should consider only *loyal* contexts, *i.e.* contexts which use only rights available to the observer: they should not try to launch code in *unreachable* locations and access channels without the corresponding permissions. We formally define a location l as *reachable* knowing Ω when there exist $p : \star \mapsto l_1$, $p_1 : l_1 \mapsto l_2$, $\ldots$, $p_n : l_n \mapsto l$ in Ω. We will write $\mathcal{R}_\Omega$ for the set of such reachable locations. Then a context of the form $[\cdot] \mid l[\![P]\!]$ is *loyal* only when l is reachable and P is well-typed in Ω. The observer must also be *loyal* when introducing new names (for instance to be used in P):

Definition 2 (Loyal extension). *Γ' is a* loyal extension *of Γ when:*

- *$\Gamma; \Gamma'$ is a well-formed environment;*
- *for every $u : \mathsf{C}_@ w$ when $w : \textsc{loc} \in \Gamma$, we must have $w \in \mathcal{R}_\Gamma$;*
- *for every $u : \tilde{v}^\star \mapsto w \in \Gamma'$ when $w : \textsc{loc} \in \Gamma$, we must have $w \in \mathcal{R}_\Gamma$.*

Finally, we define the loyal contextuality of a relation $\mathcal{S}$. writing $\Omega \vDash M \; \mathcal{S} \; N$ when M and N are in $\mathcal{S}$ for an observer knowing Ω. For this we need to extend subtyping to environments: we will say that Γ is a subtype of Γ' as soon as every typing judgment that can be inferred in Γ' can also be inferred in Γ.

Definition 3 (Loyally contextual relation). *A relation $\mathcal{S}$ is said* loyally contextual *only when:*

- *If $\Omega \vDash M \; \mathcal{S} \; N$ and Ω' is a loyal extension of Ω such that for every $a : \mathsf{E}$ in Ω', a is fresh, then $\Omega; \Omega' \vDash M \; \mathcal{S} \; N$.*
- *If $\Omega \vDash M \; \mathcal{S} \; N$, $k \in \mathcal{R}_\Omega$ and $\Omega \vdash k[\![P]\!]$ then $\Omega \vDash M \mid k[\![P]\!] \; \mathcal{S} \; N \mid k[\![P]\!]$.*
- *If $\Omega; a : \mathsf{E} \vDash M \; \mathcal{S} \; N$ and both $(\mathsf{new}\, a : \mathsf{E})\, M$ and $(\mathsf{new}\, a : \mathsf{E})\, N$ are well-typed in some subtype environment of Ω, then $\Omega \vDash (\mathsf{new}\, a : \mathsf{E})\, M \; \mathcal{S} \; (\mathsf{new}\, a : \mathsf{E})\, N$.*

The loyal barbed congruence follows from the notion of contextuality.

Definition 4 (Loyal barbed congruence). *We call* loyal barbed congruence*, written $\cong^l$, the biggest symmetric loyally contextual relation that preserves barbs and is closed over reductions.*

The contexts considered in this congruence can launch processes in every *reachable* location (to allow more contexts to be used) while barbs can only be observed in *directly reachable* (to get the simplest notion of observations). Note though that the congruence obtained does not depend on this choice because it is closed: it is simple to see that *reachable barbs* or *directly reachable contexts* would end up defining the same equivalence.

Fig. 4 Labelled transition system, most significant rules

$$\text{(LTS-GOTO)}\quad \Omega \rhd l[\![\mathsf{goto}_p\, k.\ P]\!] \xrightarrow{\tau} \Omega \rhd k[\![P]\!]$$

$$\text{(LTS-W)}\quad \frac{l \in \mathcal{R}_\Omega \qquad \Omega \vdash_l a : \mathrm{R}\langle \mathsf{T} \rangle \quad \text{where } \mathsf{T} = \Omega^r(a)}{\Omega \rhd l[\![a\,!\,\langle V \rangle\, P]\!] \xrightarrow{a!V} \Omega, \langle V : \mathsf{T} \rangle \rhd l[\![P]\!]}$$

$$\text{(LTS-R)}\quad \frac{l \in \mathcal{R}_\Omega \qquad \Omega \vdash_l a : \mathrm{W}\langle \mathsf{T}' \rangle \qquad \Omega \vdash_l V : \mathsf{T}'}{\Omega \rhd l[\![a\,?\,(X : \mathsf{T})\, P]\!] \xrightarrow{a?V} \Omega \rhd l[\![P\{^V/_X\}]\!]}$$

$$\text{(LTS-COMM)}\quad \frac{\begin{array}{c}\Omega_M \rhd M \xrightarrow{(\Phi)a!V} \Omega'_M \rhd M' \\ \Omega_N \rhd N \xrightarrow{(\Phi)a?V} \Omega'_N \rhd N'\end{array}}{\begin{array}{l}\Omega \rhd M \mid N \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,\Phi)\, M' \mid N' \\ \Omega \rhd N \mid M \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,\Phi)\, N' \mid M'\end{array}}$$

$$\text{(LTS-OPEN)}\quad \frac{\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'}{\Omega \rhd (\mathsf{new}\, b : \mathsf{E})\, M \xrightarrow{(b:\mathsf{E};\Phi)a!V} \Omega' \rhd M'} \begin{array}{l} b \neq a \\ b \in \mathsf{fn}(V) \cup \mathsf{fn}(\Phi)\end{array}$$

$$\text{(LTS-WEAK)}\quad \frac{\Omega; \Omega_e \rhd M \xrightarrow{(\Phi)a?V} \Omega' \rhd M'}{\Omega \rhd M \xrightarrow{(\Omega_e;\Phi)a?V} \Omega' \rhd M'} \begin{array}{l} \mathsf{dom}(\Omega_e) \cap (\{a\} \cup \mathsf{fn}(M)) = \emptyset \\ \Omega_e \text{ is a loyal extension of } \Omega\end{array}$$

4 Loyal Bisimilarity

The definition given for the loyal barbed congruence is justified by intuitions but it is highly intractable: every proof of equivalence indeed requires a quantification over all contexts. So we also propose a complete proof technique for this equivalence: a bisimilarity. The idea of the bisimilarity is to provide an alternative but equivalent definition of the semantics using a *Labelled Transition System* (LTS) where the labels represent the possible interactions between the system and its environment. Then two systems can be distinguished if, after some preliminary interactions, one can perform a transition the other cannot.

The way the LTS is built is completely standard (see [1]): we associate the label τ to every internal reduction a system can perform, to indicate that the environment is not involved. The rule for migration (LTS-GOTO) is an example of this. Note that, since the interactions we are characterising are between some system M and an observer knowing Ω, we define transitions of *configurations* of the form $\Omega \rhd M$. Also note that the knowledge of the observer is left untouched in a τ transition since it is not interacting with the system. In this extended abstract, we present in Figure 4 only the most significant rules, namely the rules where a message is exchanged with the environment and the rule for communication. The omitted rules are:

- the two natural contextual rules (for contexts of the forms $(\mathsf{new}\, a : \mathsf{E})[\cdot]$ and $[\cdot] \mid M$);
- and some τ transitions that can be directly derived from the reduction semantics, the way (LTS-GOTO) is obtained from (R-GOTO).

Let us explain (LTS-W). The conditions of this rule are similar to the ones for barbs. Indeed an observer knowing Ω will be able to interact with a system outputting a message V on a channel a in a location l only when l is reachable ($l \in \mathcal{R}_\Omega$) and when the observer can input on that channel ($\Omega \vdash_l a : \mathrm{R}\langle \mathsf{T} \rangle$). The knowledge of the observer will consequently be enriched by the message: Ω becomes $\Omega, \langle V : \mathsf{T} \rangle$ along that transition. In this expression, the type T indicates all the rights the observer learns, calculated using the *meet* of the types associated with the channel. Suppose for instance that the meet of all the types associated with a in Ω is $\mathrm{RW}\langle \mathsf{T}_1, \mathsf{T}_2 \rangle @ l$; then T_1 sums up all the rights that can be obtained by inputting on a. We denote that type T_1 as $\Omega^r(a)$ in (LTS-W).

With those transitions, we would like to define an equivalence $\mathcal{R}$ as a standard bisimulation: when $\Omega \vDash M \mathcal{R} N$ and $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$ then there must exist some N' such that $\Omega \rhd N \xrightarrow{\tau}{}^* \xrightarrow{\hat{\mu}} \xrightarrow{\tau}{}^* \Omega' \rhd N'$ and $\Omega' \vDash M' \mathcal{R} N'$. But this definition cannot be used right away in our case, because of dependent types. Let us consider a case where the discrepancy appears. Suppose some channel c in l on which a passport can be transmitted (so c is of type $\mathrm{RW}\langle \sum x, y : \vec{\mathrm{LOC}}.\ x \mapsto y \rangle @ l$) and consider the following two systems:

$$(\mathsf{new}\, k' : \mathrm{LOC})\ (\mathsf{new}\, p : k, k' \mapsto l)\ l[\![c\,!\,\langle (k,l),(p) \rangle\, d\,!\,\langle k' \rangle]\!] \qquad (1)$$

$$(\mathsf{new}\, k' : \mathrm{LOC})\quad (\mathsf{new}\, p : k \mapsto l)\quad l[\![c\,!\,\langle (k,l),(p) \rangle\, d\,!\,\langle k' \rangle]\!] \qquad (2)$$

The only difference is the fact that the passport p can be used also from the new location k' in the first system. Since the observer receives p at the type $k \mapsto l$ in both cases, it should not be able to make the difference. But they can perform the following transitions with *distinct labels* (for simplicity, we ignore the type annotations in the labels):

$$\begin{array}{lrl}
\Omega \rhd (1) & \xrightarrow{(k',p)c!((k,l),(p))} & \Omega, p : k \mapsto l \rhd l[\![d\,!\,\langle k' \rangle]\!] \\
 & \xrightarrow{d!k'} & \Omega, p : k \mapsto l, k' : \mathrm{LOC} \rhd l[\![\mathsf{stop}]\!] \\
\Omega \rhd (2) & \xrightarrow{(p)c!((k,l),(p))} & \Omega, p : k \mapsto l \rhd (\mathsf{new}\, k' : \mathrm{LOC})\, l[\![d\,!\,\langle k' \rangle]\!] \\
 & \xrightarrow{(k')d!k'} & \Omega, p : k \mapsto l, k' : \mathrm{LOC} \rhd l[\![\mathsf{stop}]\!]
\end{array}$$

namely not opening the scope of k' in the same transition. To avoid this problem, we annotate configurations with a set of names whose scopes have been opened because of type dependencies, not because they were revealed. The labels are modified accordingly to mention only the names that are actually revealed.

Definition 5 (Actions). *The* annotated configuration $\Omega \rhd_{\tilde{a}} M$ *can perform the* action μ *and become* $\Omega' \rhd_{\tilde{a}'} M'$ *when:*

- *if* μ *is* τ *or* $(\Phi)a?V$*: the transition* $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$ *is provable in the LTS and* $\tilde{a} = \tilde{a}'$*;*
- *if* μ *is* $(\tilde{b})a!V$*: the transition* $\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'$ *is provable in the LTS,* $\tilde{b} = \mathsf{fn}(V) \cap (\mathsf{dom}(\Phi) \cup \tilde{a})$ *and* $\tilde{a}' = (\mathsf{dom}(\Phi) \cup \tilde{a}) \setminus \mathsf{fn}(V)$*.*

So, using an empty set as annotation, the first transition of the first system becomes:

$$\Omega \rhd_\emptyset (\mathsf{new}\, k' : \mathrm{LOC})\, (\mathsf{new}\, p : k, k' \mapsto l)\, l[\![c\,!\,\langle (k,l),(p) \rangle\, d\,!\,\langle k' \rangle]\!] \xrightarrow{(p)c!((k,l),(p))} \Omega, p : k \mapsto l \rhd_{k'} l[\![d\,!\,\langle k' \rangle]\!]$$

Definition 6 (Loyal bisimilarity). *The* loyal bisimilarity*, written* $\approx^{al}$*, is the largest bisimulation defined in the standard way over* actions *of annotated configurations.*

The rest of this section is devoted to the proof that the two equivalences coincide under some conditions to justify that the bisimilarity has been introduced as a proof technique. The proof of that property is significantly more complex than its equivalent in the literature: the control of migrations hinders tracking the knowledge of the observer (apart from the passports, note that we must keep track of annotations because they hide some names from the observer). We will describe here only the most interesting aspects of the proof; more details are provided in [13], the proof is fully developed in [14].

The first step to bridge the gap between the loyal barbed congruence and the loyal bisimilarity is to account for annotations in configurations. This can be done by simply defining *annotated typed relations*, written

$$\Omega \vDash M \ {}_{\tilde{a}_M}\mathcal{S}_{\tilde{a}_N} \ N$$

and adapting the notion of contextuality to those relations. It is quite easy to see that the *annotated loyal barbed congruence* which ensues coincides with $\cong^l$ when its annotations are the empty sets of names.

The expected result then amounts to proving that the loyal bisimilarity and the annotated loyal barbed congruence coincide. The proof that the bisimilarity is included in the barbed congruence is mainly the proof of the fact that the bisimilarity is contextual. This is naturally done by checking all three items defining contextuality, the major property to check being:

Theorem 3 (Bisimilarity is closed on parallel contexts). $\Omega \vDash M \ {}_{\tilde{a}_M}\approx^{al}_{\tilde{a}_N} N$, $l \in \mathcal{R}_\Omega$, $\Omega \vdash l[\![O]\!]$ *and* $\mathsf{fn}(O) \cap (\tilde{a}_M \cup \tilde{a}_N) = \emptyset$ *imply* $\Omega \vDash M \mid l[\![O]\!] \ {}_{\tilde{a}_M}\approx^{al}_{\tilde{a}_N} N \mid l[\![O]\!]$.

Idea of proof. To get this result we simply build a relation and prove that it is a bisimulation which induces the fact that it is included in the biggest bisimulation, $\approx^{al}$. Because that relation must be closed on reductions, we will consider a relation $\mathcal{S}$ in which systems have a very general form:

$$\Omega \vDash (\mathsf{new}\, \Phi_M)(M \mid \prod_i l_i[\![O_i]\!]) \ {}_{\tilde{a}_M}\mathcal{S}_{\tilde{a}_N} \ (\mathsf{new}\, \Phi_N)(N \mid \prod_i l_i[\![O_i]\!])$$

The main difficulty to tackle is the fact that, along reductions, the knowledge of the observer, initially completely located in Ω (because $l[\![O]\!]$ is well-typed in Ω), is split between Ω and $\prod_i l_i[\![O_i]\!]$. In particular, a part of the environments Φ and annotations $\tilde{a}$ should be included in the general knowledge of the observer since they might have been communicated to the processes O_i. A precise account of this knowledge must be kept to preserve the full-strength of the initial hypothesis of bisimilarity between M and N. In particular, M and N must be bisimilar for an observer having access to all the locations l_i since it has some processes O_i running there. □

Let us now consider the converse, namely the fact that the congruence is included in the bisimilarity. The guiding idea of the definition of the actions was to identify all the possible interactions between a system and its observer. So the proof of that inclusion can be based on the definition of contexts that characterise a given action of the system. Those contexts use the fact that we can put any environment Γ in a normal form looking like:

$$\begin{aligned} w_1 : \text{LOC}, \ldots, w_m : \text{LOC}, \\ u_1 : \tilde{w}_{i_1} \mapsto w_{i_1}, \ldots, u_n : \tilde{w}_{i_n} \mapsto w_{i_n}, \\ v_1 : \mathsf{C}_{1@}w_{j_1}, \ldots, v_o : \mathsf{C}_{o@}w_{j_o} \end{aligned}$$

where

- the w_k are all distinct;
- $u_k = u_{k'}$ only if $k = k'$ or if $w_{i_k} \neq w_{i_{k'}}$;
- $v_k = v_{k'}$ only if $k = k'$ or if $w_{j_k} \neq w_{j_{k'}}$.

So this normal form has the following structure: all the locations are defined first because types can depend only on location identifiers so that all the locations can be listed first; and every identifier is attributed exactly one type per location identifier to which it is attached[2]. The existence of such a normal form follows from the property of partial meets (Theorem 1) which ensures that all the types associated with a given identifier sum up to their meet.

This normal form of environments is relevant for the contexts that characterise the actions of a system because they provide a way to encode every environment into a value of the calculus.

Definition 7 (Reification of environments). *To an environment Γ of the following (normal) form*

$$\begin{aligned} w_1 : \text{LOC}, \ldots, w_m : \text{LOC}, \\ u_1 : \tilde{w}_{i_1} \mapsto w_{i_1}, \ldots, u_n : \tilde{w}_{i_n} \mapsto w_{i_n}, \\ v_1 : \mathsf{C}_{1@}w_{j_1}, \ldots, v_o : \mathsf{C}_{o@}w_{j_o} \end{aligned}$$

we associate the value V_Γ and the type T_Γ such that:

$$\begin{aligned} V_\Gamma &= ((w_1, \ldots, w_m), (u_1, \ldots, u_n, v_1, \ldots, v_o)) \\ \mathsf{T}_\Gamma &= \sum x_1, \ldots, x_m : \tilde{\text{LOC}}.\ \tilde{x}^\star_{i_1} \mapsto x_{i_1}, \ldots, \tilde{x}^\star_{i_n} \mapsto x_{i_n}, \mathsf{C}_{1@}x_{j_1}, \ldots, \mathsf{C}_{o@}x_{j_o} \end{aligned}$$

Proposition 1 (Soundness of the reification). *For any well-formed environment Γ and any location w defined in Γ, $\Gamma \vdash_w V_\Gamma : \mathsf{T}_\Gamma$.*

Thanks to this reification of environments, we can proceed as usual, namely we can define some system $\mathfrak{C}^\Omega_\mathcal{N}((\tilde{b})c!U)$ so that $M \mid \mathfrak{C}^\Omega_\mathcal{N}((\tilde{b})c!U)$ will be sending the value $V_{\Omega,\langle U:\Omega^r(c)\rangle}$ on some specific channel ω if and only if the system M has actually sent the message U over the channel c to $\mathfrak{C}^\Omega_\mathcal{N}((\tilde{b})c!U)$. So such a context

[2] When typechecking processes, a given channel or passport can be attached to more than one location variable.

would be of the form $l[\![O]\!]$ where l is the location of the channel c in which the action takes place. Note that the observer can launch some process in l since the action $(\tilde{b})c!U$ is visible to the observer Ω: by rule (LTS-W) this implies that l is in $\mathcal{R}_\Omega$. Then O performs the following steps.

1. It waits for a message on the channel c and, in parallel, exhibit a barb on some special channel δ.
2. It checks that the received value matches the expected U: this relies on the possibility to test the equality and inequality of names; in particular, to check that the names in $\tilde{b}$ are indeed fresh, the context is parameterised with a finite set of existing names $\mathcal{N}$ which contains all the names that are known to the observer. This test matches exactly the definition of the set $\tilde{b}$ in output actions: this set contains only the names which were hidden within the system or the annotation and which are revealed to the observer.
3. It finally cancels the barb on δ and outputs the value $V_{\Omega,\langle U:\Omega^r(c)\rangle}$ on the channel ω.

The channel δ used in the context serves only one purpose: to check that the step 2 has actually been performed: since the detected barbs always allow some preliminary τ transitions, the barb on ω is visible since the very beginning as soon as the system *can* perform the action.

By a very similar technique, it is possible to form contexts that characterise an input action, so that the following theorem can be proved:

Theorem 4. *The loyal annotated barbed congruence is included in the loyal bisimilarity.*

Idea of proof. We simply prove that $\cong^p$, the biggest annotated relation verifying the conditions of the loyal annotated barbed congruence apart from closure over $(\mathsf{new}\, a : \mathsf{E})[\cdot]$ contexts, is a bisimulation. For this consider $\Omega \vDash M \ {}_{\tilde{a}_M}\!\cong^p_{\tilde{a}_N} N$.

When the configuration $\Omega \rhd_{\tilde{a}_M} M$ performs a τ action to $\Omega \rhd_{\tilde{a}_M} M'$, the closure of $\cong^p$ on reductions gives a N' such that $\Omega \vDash M' \ {}_{\tilde{a}_M}\!\cong^p_{\tilde{a}_N} N'$.

For the action $\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\alpha} \Omega' \rhd_{\tilde{a}'_M} M'$, we know that $M \mid \mathfrak{C}^\Omega_\mathcal{N}(\alpha)$ can reduce into some system $(\mathsf{new}\, \Phi_M)\, M' \mid \lambda[\![\omega\,!\,\langle V_{\Omega'}\rangle]\!]$. By contextuality and closure on reductions, $N \mid \mathfrak{C}^\Omega_\mathcal{N}(\alpha)$ should reach an equivalent state, with a barb on ω and no barb on δ. By definition of the context $\mathfrak{C}^\Omega_\mathcal{N}(\alpha)$, that equivalent state must be of the form $(\mathsf{new}\, \Phi_N)\, N' \mid \lambda[\![\omega\,!\,\langle V_{\Omega'}\rangle]\!]$ with $\Omega \rhd_{\tilde{a}_N} N \stackrel{\alpha}{\Longrightarrow} \Omega' \rhd_{\tilde{a}'_N} N'$.

A fairly standard *scope extrusion lemma* (see for instance [1]) bridges the last gap by concluding $\Omega' \vDash M' \ {}_{\tilde{a}'_M}\!\cong^p_{\tilde{a}'_N} N'$ from

$$\lambda, \omega, \pi \vDash (\mathsf{new}\, \Phi_M)\, M' \mid \lambda[\![\omega\,!\,\langle V_{\Omega'}\rangle]\!] \ {}_{\tilde{a}'_M}\!\cong^p_{\tilde{a}'_N} (\mathsf{new}\, \Phi_N)\, N' \mid \lambda[\![\omega\,!\,\langle V_{\Omega'}\rangle]\!]$$

where: π is a universal passport to λ, $\tilde{a}'_M$ is $(\tilde{a}_M \cup \mathsf{dom}(\Phi_M)) \setminus \mathsf{dom}(\Omega)$ and a similar formula for $\tilde{a}'_N$. □

The results stated above directly entails the expected result:

Theorem 5 (Full abstraction of $\approx^{al}$ for $\cong^l$). *$\Omega \vDash M \cong^l N$ if and only if $\Omega \vDash M \ {}_\emptyset\!\approx^{al}_\emptyset N$*

5 Conclusion & Perspectives

This work presents a new approach to control the migrations of agents in the context of distributed computation, using simple passports that should correspond to the origin location of the migrating agent. We have developed the full theory of this idea, with a loyal barbed congruence that takes those passports into account to distinguish between systems. We have also provided a complete proof technique for this equivalence as a bisimilarity.

This work provides a solid ground on which to investigate subtler notions of security like the ones presented in [9] and [18]. We already started to study more complex passports in which resources that can be accessed after the migration depend on the passport actually used: when a new passport is generated, its type also embed all the rights to be granted to incoming processes.

It would also be interesting to refine passports to stricter notions of trust, where other locations are prevented from relaying processes for instance.

Acknowledgement The author would like to thank Matthew Hennessy for numerous helpful discussions and comments.

References

1. Hennessy, M.: A Distributed Pi-calculus. Cambridge University Press (2007)
2. Levi, F., Sangiorgi, D.: Controlling interference in ambients. In: 27th Annual Symposium on Principles of Programming Languages (POPL) (Boston, MA), ACM (January 2000) 352–364
3. Merro, M., Hennessy, M.: A bisimulation-based semantic theory of Safe Ambients. ACM Transactions on Programming Languages and Systems **28**(2) (March 2006) 290–330
4. Bugliesi, M., Crafa, S., Merro, M., Sassone, V.: Communication interference in mobile boxed ambients. In Agrawal, M., Seth, A., eds.: FSTTCS. Volume 2556 of Lecture Notes in Computer Science., Springer (2002) 71–84
5. Castagna, G., Nardelli, F.Z.: The Seal calculus revisited: Contextual equivalence and bisimilarity. In Agrawal, M., Seth, A., eds.: FSTTCS. Volume 2556 of Lecture Notes in Computer Science., Springer (2002) 85–96
6. Schmitt, A., Stefani, J.B.: The Kell calculus: A family of higher-order distributed process calculi. In Priami, C., Quaglia, P., eds.: Global Computing. Volume 3267 of Lecture Notes in Computer Science., Springer (2004) 146–178
7. Hennessy, M., Merro, M., Rathke, J.: Towards a behavioural theory of access and mobility control in distributed systems. Theoretical Computer Science **322** (2003) 615–669
8. Cardelli, L., Ghelli, G., Gordon, A.D.: Ambient groups and mobility types. In van Leeuwen, J., Watanabe, O., Hagiya, M., Mosses, P.D., Ito, T., eds.: IFIP TCS. Volume 1872 of Lecture Notes in Computer Science., Springer (2000) 333–347
9. Hennessy, M., Rathke, J., Yoshida, N.: SafeDpi: a language for controlling mobile code. Acta Informatica **42**(4-5) (2005) 227–290
10. Yoshida, N.: Channel dependent types for higher-order mobile processes (September 2004)

11. Martins, F., Vasconcelos, V.T.: History-based access control for distributed processes. In: TGC. (2005) 98–115
12. Boreale, M., Sangiorgi, D.: Bisimulation in name-passing calculi without matching. In: Thirteenth Annual Symposium on Logic in Computer Science (LICS) (Indiana), IEEE, Computer Society Press (July 1998)
13. Hym, S.: Mobility control via passports (2007) Preprint. Available on `https://hal.archives-ouvertes.fr/hal-00140527` and on `http://www.pps.jussieu.fr/~hym/r/`.
14. Hym, S.: Typage et contrôle de la mobilité. PhD thesis, Université Paris Diderot – Paris 7 (December 2006)
15. Hennessy, M., Riely, J.: Resource access control in systems of mobile agents. Information and Computation **173** (2002) 82–120
16. Hym, S., Hennessy, M.: Adding recursion to Dpi. Theoretical Computer Science **373**(3) (April 2007) 182–212
17. Milner, R., Sangiorgi, D.: Barbed bisimulation. In Kuich, W., ed.: 19th ICALP. Volume 623 of Lecture Notes in Computer Science., Springer-Verlag (1992) 685–695
18. Crary, K., Harper, R., Pfenning, F., Pierce, B.C., Weirich, S., Zdancewic, S.: Manifest security for distributed information. White paper (March 2006)

A Reduction Semantics

Fig. 5 Reduction semantics

$$\text{(R-GOTO)}\quad l[\![\mathsf{goto}_p\, k.\ P]\!] \longrightarrow k[\![P]\!]$$

$$\text{(R-NEWLOC)}\quad l[\![\mathsf{newloc}\, k, (\vec{c}), (\vec{p}), (\vec{q}) : \textstyle\sum x : \text{LOC}.\ \mathsf{T}\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!] \longrightarrow (\mathsf{new}\langle k, ((\vec{c}), (\vec{p}), (\vec{q})) : \mathsf{T}\{{}^{l}/_{x}\}\rangle)(k[\![P_k]\!] \mid l[\![P]\!])$$

$$\text{(R-NEWPASS)}\quad l[\![\mathsf{newpass}\, p\, \mathsf{from}\, \tilde{k}^{\star}\ \mathsf{in}\ P]\!] \longrightarrow (\mathsf{new}\, p : \tilde{k}^{\star} \mapsto l)\, l[\![P]\!]$$

$$\text{(R-COMM)}\quad l[\![a\,!\,\langle V\rangle\ P_1]\!] \mid l[\![a\,?\,(X : \mathsf{T})\ P_2]\!] \longrightarrow l[\![P_1]\!] \mid l[\![P_2\{{}^{V}/_{X}\}]\!]$$

$$\text{(R-IF-V)}\quad l[\![\mathsf{if}\ a = a\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!] \longrightarrow l[\![P_1]\!]$$

$$\text{(R-IF-F)}\quad l[\![\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!] \longrightarrow l[\![P_2]\!] \quad \text{when } a_1 \neq a_2$$

$$\text{(R-NEWCHAN)}\quad l[\![\mathsf{newchan}\, c : \mathsf{C}\ \mathsf{in}\ P]\!] \longrightarrow (\mathsf{new}\, c : \mathsf{C}@l)\, l[\![P]\!]$$

$$\text{(R-SPLIT)}\quad l[\![P_1 \mid P_2]\!] \longrightarrow l[\![P_1]\!] \mid l[\![P_2]\!]$$

$$\text{(R-REP)}\quad l[\![*P]\!] \longrightarrow l[\![P]\!] \mid l[\![*P]\!]$$

$$\text{(R-C-PAR)}\quad \frac{M_1 \longrightarrow M_1'}{M_1 \mid M_2 \longrightarrow M_1' \mid M_2} \qquad \text{(R-C-NEW)}\quad \frac{M_1 \longrightarrow M_1'}{(\mathsf{new}\, a : \mathsf{E})\, M_1 \longrightarrow (\mathsf{new}\, a : \mathsf{E})\, M_1'}$$

$$\text{(R-STRUCT)}\quad \frac{M_1 \equiv M_2 \longrightarrow M_2' \equiv M_1'}{M_1 \longrightarrow M_1'}$$