Towards a Behavioural Theory of Access and Mobility Control in Distributed Systems (Extended Abstract)

Matthew Hennessy[1], Massimo Merro[2], and Julian Rathke[1]

[1] School of Cognitive and Computing Sciences
University of Sussex
matthewh,julianr@cogs.sussex.ac.uk
[2] Dipartimento di Informatica
Universitá di Verona
merro@sci.univr.it

Abstract. We define a typed bisimulation equivalence for the language Dpi, a distributed version of the π-calculus in which processes may migrate between dynamically created locations. It takes into account resource access policies, which can be implemented in Dpi using a novel form of dynamic capability types. The equivalence, based on typed actions between configurations, is justified by showing that it is *fully-abstract* with respect to a natural distributed version of a contextual equivalence.
In the second part of the paper we study the effect of controlling the migration of processes. This affects the ability to perform observations at specific locations, as the observer may be denied access. We show how the typed actions can be modified to take this into account, and generalise the *full-abstraction* result to this more delicate scenario.

1 Introduction

The behaviour of processes in a distributed system depends on the resources they have been allocated. Moreover these resources, or a process's knowledge of these resources, may vary over time. Therefore an adequate behavioural theory of distributed systems must be based not only on the inherent abilities of processes to interact with other processes, but must also take into account the (dynamic) resource environment in which they are operating. In our approach judgements will take the form

$$\Gamma \models M \approx N,$$

where N and M are systems and Γ represents their computing environment. Intuitively this means that M and N offer the same behaviour, relative to the environment Γ. The challenge addressed by this paper is to give an adequate formalisation of this idea, where

– the systems M and N are collections of *location aware* processes, which may be allocated varying *access rights* to resources at different locations and may migrate between these locations to exercise their rights

© Springer-Verlag Berlin Heidelberg 2003

– the computing environment Γ may vary dynamically, reflecting both the overall resources available to M and N and the evolving knowledge that users may accumulate of these resources.

This is developed in terms of the language Dpi, [9], a version of the π-calculus, [13], in which processes may migrate between locations, which in turn can be dynamically created. As explained in [9] resource access policies in Dpi may be implemented using a *capability* based type system; thus in this setting it is sufficient to develop typed behavioural equivalences in order to capture the effect of resource access policies on process behaviour. But in this paper we extend the typing system of [9] by allowing types to be created *dynamically* and to depend on received data.

The language Dpi and its reduction semantics is summarised in Section 2, which relies heavily on the corresponding section of [9]. The typing system, discussed briefly in Section 3, contains two major extensions to the original typing system of [9]. The first introduces a new kind of type, $\mathsf{rc}\langle \mathrm{A}\rangle$ for *registered channel names*, which allows channels names to be used consistently at multiple locations. The second allows types to be constructed *dynamically* and be dependent on received data. Subject reduction still holds for this extended type system.

The second novelty of the paper, in Section 4.1, is a definition of *typed action*

$$\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright N \qquad (1)$$

indicating that in an environment constrained by the *type environment* Γ the system M may perform the action μ and be transformed into N; the environment Γ may also be changed by this interaction, to Γ', for example by the extrusion of new resources, or new capabilities on already known resources. Here the actions μ are either internal moves, τ, or *located* input or output actions, of the form $k.a?v$ or $(\tilde{b})k.a!v$. Informally, the action in (1) is possible if M is capable of performing the action μ in the standard manner *and* the environment Γ allows it to happen.

With these typed actions we can define a standard notion of (weak) bisimulations between *configurations*, consistent pairs of the form $\Gamma \triangleright M$; the formal definition is given in Section 4 and we use $\Gamma \triangleright M \approx^{bis} N$ to denote that there is a bisimulation containing the two configurations $\Gamma \triangleright M$ and $\Gamma \triangleright N$.

The first main result of the paper is that this notion of *typed bisimulation* captures precisely an independently defined *contextual equivalence*, which we denote by $\eqsim^{rbc}$.

The final topic of the paper is the effect of *migration* on the behaviour of systems. In Dpi the migration of processes is unconstrained. The relevant reduction rule is

$$k[\![\mathsf{goto}\, l.P]\!] \rightarrow l[\![P]\!].$$

Any agent is allowed to migrate from a site k to the site l. Indeed this rule is essential in establishing the above theorem. For example consider the systems M_1, M_2 given by

$$(\mathsf{new}\, k : \mathrm{K})\ l[\![c!\langle k\rangle]\!] \mid k[\![a!\langle\rangle\, \mathsf{stop}]\!] \qquad \text{and} \qquad (\mathsf{new}\, k : \mathrm{K})\ l[\![c!\langle k\rangle]\!] \mid k[\![\mathsf{stop}]\!] \quad (2)$$

where K is the declaration type $\mathsf{loc}[a : \mathsf{rw}\langle\rangle]$, and Γ an environment which has read capability at type K on c at l. These are not bisimilar in the environment Γ, as after exporting the new name k on c at l, that is performing the *bound* output action $(k)l.c!k$, only the former may have the typed action

$$(\Gamma' \triangleright k[\![a!\langle\rangle\,\mathsf{stop}]\!]) \xrightarrow{k.a!\langle\rangle} (\Gamma' \triangleright k[\![\mathsf{stop}]\!])$$

where Γ' represents the environment Γ updated with the new knowledge of a at k. Moreover they can be distinguished contextually because of the Γ-context

$$l[\![c?(x)\,.\mathsf{goto}\,x.a?(y)\,\mathsf{goto}\,l.\omega!\langle\rangle]\!] \mid -$$

An environment which has read or write capability on a channel at k can automatically send an agent there to perform a test and report back to base. Note that this test works only because systems allowed by Γ have the automatic ability to migrate to the site k. If on the other hand migration were constrained, as one would expect in more realistic scenarios, then these tests would no longer be necessarily valid and these terms may become contextually equivalent.

There are many mechanisms by which migration could be controlled in languages such as Dpi. In this paper we introduce one such mechanism, based on a simple extension of the typing system, which allows us to examine the effect of such control on behavioural equivalences. We introduce a new location capability **move**. Then migration from any location to k is only allowed with respect to an environment Γ, if in Γ the location k is known with a capability **move**; we say Γ has *migration* rights to k. This idea is easily implemented by using a slight extension to our typing system, and is sufficient to demonstrate the subtleties involved when migration is controlled.

The remainder of the paper is devoted to extending the characterisation result given above, characterising a contextual equivalence using bisimulations, to this language. The typed actions (1) above are readily adapted to this scenario. Here we allow, for example, the action

$$\Gamma \triangleright M \xrightarrow{k.a!v}_m \Gamma' \triangleright N \tag{3}$$

if, in addition to the requirements for (1), the environment has the capability **move** for k; intuitively for the environment to see the action (3) it must be able to move to the site k.

These actions lead to a new bisimulation equivalence, denoted $\approx^m_{bis}$, and we can prove

$\Gamma \models M \approx^m_{bis} N$ if and only if $\Gamma \models M \cong^m_{cxt} N$

where $\cong^m_{cxt}$ is a suitable modification of the contextual equivalence $\cong^{rbc}$; in the definition of $\cong^m_{cxt}$ we only require the equivalence to be preserved in the context of processes at locations to which the environment has migration rights. Thus referring to (2) above we will have

$$\Gamma \models M_1 \cong^m_{cxt} M_2$$

if Γ does not have migration rights to k. Note that neither of these systems can give rise to the modified typed actions, as given in (3) above.

However it is easy to envisage a natural version of contextual equivalence which does distinguish between M_1 and M_2 of (2) above. Although the environment may not have migration rights to k, it may, apriori, have a process running there. If this were allowed, and the environment had the appropriate capability on the channel a at k then the systems M_1 and M_2 could be distinguished. The question then arises of finding a bisimulation based characterisation for this modified contextual equivalence.

We address a parameterised version of this problem. Let $\mathcal{T}$ be the set of locations at which apriori the environment can place testing processes and let $\eqsim^{\mathcal{T}}_{cxt}$ be the resulting contextual equivalence. Unfortunately this is not characterised by the natural modification to the equivalence $\approx^{m}_{bis}$, which we denote by $\approx^{\mathcal{T}}_{bis}$. A counterexample is given in Section 5.2.

It turns out that we must be careful about the location at which information is learnt. Information about k learnt at l can not be used without the capability to move to k. However this information must be retained because that move capability may subsequently be obtained. This leads to a more complicated form of environment $\overline{\Gamma}$, which records

– locations at which testing processes may be placed, $\mathcal{T}$
– *globally* available information on capabilities at locations
– similar *locally* available information.

The details are given in Section 5.2, which also contains a generalisation of the typed actions of (1) above to these more complicated environments. The final result of the paper is that the new bisimulation equivalence based on these actions again captures the contextual equivalence.

In this extended abstract all proofs are omitted, as indeed is most of the technical exposition of the typing system. The reader is referred to the full version of the paper, [7], for all the details.

2 The Language Dpi

Syntax: The syntax, given in Figure 1, is a slight extension of that of Dpi from [9]. This presupposes a general set of names *Names* ranged over by n, m, and a set of variables *Vars* ranged over by x, y; informally we will often use $a, b, c, \ldots$ for names of channels and $l, k, \ldots$ for locations or sites. *Identifiers*, ranged over by u, v, w, may either be variables or names. The syntax also uses a set of types; discussion of these is postponed until the next section.

The syntax is built in a two-level structure, the lower level being processes, agents or threads. The syntax here is an extension of the π-calculus, [9], with primitives for migration between locations. At this level there are three forms of name creation:

Fig. 1. Syntax of Dpi

$M, N ::=$	*Systems*
$l[\![P]\!]$	Located Process
$M \mid N$	Composition
$(\mathsf{new}\, n : \mathrm{T})\ M$	Name Scoping
$\mathbf{0}$	Termination
$P, Q ::=$	*Processes*
$u!\langle V\rangle\, P$	Output
$u?(X : \mathrm{T})P$	Input
$\mathsf{goto}\, v.W$	Migration
$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$	Matching
$(\mathsf{newc}\, n : \mathrm{A})\ P$	Channel Name creation
$(\mathsf{newreg}\, n : \mathrm{G})\ P$	Registered Name creation
$(\mathsf{newloc}\, k : \mathrm{K})\ \mathsf{with}\ C\ \mathsf{in}\ P$	Location Name creation
$P \mid Q$	Composition
$* P$	Replication
stop	Termination
$U, V, W ::=$	*Values*
$(\alpha_1, \ldots, \alpha_n),\ n > 0$	tuples
$\alpha, \alpha' ::=$	*Generalised Identifiers*
u	Identifiers
$(u_1, \ldots, u_n)@u,\ n \geq 0$	Located Identifiers

– $(\mathsf{newc}\, a : \mathrm{A})\ P$, the creation of a new *local channel name* of type A called a.
– $(\mathsf{newreg}\, n : \mathsf{rc}\langle \mathrm{A}\rangle)\ P$, the creation of a new *registered name* for located channels of type A. These may be used in the declaration types of locations and treated uniformly across them.
– $(\mathsf{newloc}\, k : \mathrm{K})\ \mathsf{with}\ C\ \mathsf{in}\ P$, the creation of a new *location name* k of type K, with the code C running there, and P as a local continuation. Our typing system will ensure that K is a well-formed location type; for example this means that it may only use the registered channel names.

Systems are constructed from *located threads*, of the form $l[\![P]\!]$, representing the thread P running at location l. These may be combined with the parallel operator $\mid$ and names may be shared between threads using the construct $(\mathsf{new}\, e : \mathrm{T})$.

Reduction Semantics: This is given in terms of a binary relation between *closed* systems, system terms which have no free occurrences of variables:

$$M \rightarrow N$$

Fig. 2. Reduction semantics for DPI

(R-COMM)
$$\frac{}{k[\![c!\langle V\rangle\, Q]\!] \mid k[\![c?(X:\mathrm{T})\, P]\!] \rightarrow k[\![Q]\!] \mid k[\![P\{\!|V/x|\!\}]\!]}$$

(R-SPLIT)
$$\frac{}{k[\![P \mid Q]\!] \rightarrow k[\![P]\!] \mid k[\![Q]\!]}$$

(R-C−CREATE)
$$\frac{}{k[\![(\mathsf{newc}\, n:\mathrm{A})\ P]\!] \rightarrow (\mathsf{new}\, n:\mathrm{A}@k)\ k[\![P]\!]}$$

(R-MOVE)
$$\frac{}{k[\![\mathsf{goto}\, l.P]\!] \rightarrow l[\![P]\!]}$$

(R-R−CREATE)
$$\frac{}{k[\![(\mathsf{newreg}\, n:\mathrm{G})\ P]\!] \rightarrow (\mathsf{new}\, n:\mathrm{G})\ k[\![P]\!]}$$

(R-UNWIND)
$$\frac{}{k[\![* P]\!] \rightarrow k[\![P \mid * P]\!]}$$

(R-L−CREATE)
$$\frac{}{k[\![(\mathsf{newloc}\, l:\mathrm{L})\ \mathsf{with}\ C\ \mathsf{in}\ P]\!] \rightarrow (\mathsf{new}\, l:\mathrm{L})\ (l[\![C]\!] \mid k[\![P]\!])}$$

(R-STR)
$$\frac{M \equiv N,\ M \rightarrow M',\ M' \equiv N'}{N \rightarrow N'}$$

(R-EQ)
$$\frac{}{k[\![\mathsf{if}\ u = u\ \mathsf{then}\ P\ \mathsf{else}\ Q]\!] \rightarrow k[\![P]\!]}$$

(R-NEQ)
$$\frac{u \neq v}{k[\![\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q]\!] \rightarrow k[\![Q]\!]}$$

Fig. 3. Structural equivalence for DPI

(S-EXTR)	$(\mathsf{new}\, n:\mathrm{T})\ (M \mid N) = M \mid (\mathsf{new}\, n:\mathrm{T})\ N$ if $\mathsf{n}(n) \notin \mathsf{fn}(M)$
(S-COM)	$M \mid N = N \mid M$
(S-ASSOC)	$(M \mid N) \mid O = M \mid (N \mid O)$
(S-ZERO)	$M \mid \mathbf{0} = M$
(S-FLIP)	$(\mathsf{new}\, n:\mathrm{T})\ (\mathsf{new}\, n':\mathrm{T}')\ N = (\mathsf{new}\, n':\mathrm{T}')\ (\mathsf{new}\, n:\mathrm{T})\ N$ if $\mathsf{n}(n) \notin \mathrm{T}', \mathsf{n}(n') \notin \mathrm{T}$

and is a mild generalisation of that given in [9] for DPI. It is a *contextual relation* between systems; that is, it is preserved by the static operators $\mid$ and $(\mathsf{new}\, e:\mathrm{E})$. The defining rules are given in Figure 2, one of which uses a structural equivalence, whose axioms are given in Figure 3.

3 Typing

The collection of types, given in Figure 4, is an extension of those used in [9], to which the reader is referred for more background and motivation. Note that the formation of local channel type $\mathsf{rw}\langle\mathrm{T},\mathrm{U}\rangle$ requires a subtyping relation <:; however in the examples of this extended abstract we only use types of the form

Fig. 4. Types

Base Types:	$\mathrm{B} ::= \mathbf{int} \mid \mathbf{bool} \mid \mathbf{unit} \mid \top \mid \ldots$
Local Channel types:	$\mathrm{A} ::= \mathsf{r}\langle \mathrm{T}\rangle \mid \mathsf{w}\langle \mathrm{T}\rangle \mid \mathsf{rw}\langle \mathrm{T}, \mathrm{U}\rangle$ provided $\mathrm{U} <: \mathrm{T}$
Capability Types:	$\mathrm{R} ::= u : \mathrm{A}$
Location Types:	$\mathrm{K} ::= \mathsf{loc}[\mathrm{R}_1, \ldots, \mathrm{R}_n], n \geq 0$
Registered Name Types:	$\mathrm{G} ::= \mathsf{rc}\langle \mathrm{A}\rangle$
Value Types:	$\mathrm{C} ::= \mathrm{B} \mid \mathrm{A} \mid G \mid (\tilde{\mathrm{A}})@u \mid (\tilde{\mathrm{A}})@\mathrm{K}$
Transmission Types:	$\mathrm{T} ::= (\mathrm{C}_1, \ldots, \mathrm{C}_n), n \geq 0$

$\mathsf{rw}\langle \mathrm{T}, \mathrm{T}\rangle$, which we abbreviate to $\mathsf{rw}\langle \mathrm{T}\rangle$. But there are four major differences from [9]:

- We use a *top* type $\top$ for names with no capabilities.
- We use a new category of types, *registered name types*, to explicitly manage the resource names which can be shared between different locations.
- The types expressions are allowed to contain variables, thereby giving rise to what we call *dynamic* types; the constraints they place on agent behaviour is determined dynamically by instantiation of these variables.
- The notion of type environment is changed; they do not explicitly contain associations between names and location types.

A type judgement will take the form $\Gamma \vdash M$ where Γ is a *type environment*, a list of assumptions about the types to be associated with the identifiers in the system M. Typical examples used in environments include $w : \mathsf{rw}\langle \mathrm{T}\rangle@k$, meaning w is a read/write channel at location k, $w : \mathsf{rc}\langle \mathrm{A}\rangle$ meaning w is a registered channel name, and $w : \mathsf{loc}$ meaning that w is a location name. Because of lack of space in this extended abstract we omit the rules for the judgements; they may be found in the full version, [7], which also contains a proof of Subject Reduction.

4 Contextual Equivalence in Dpi

We now turn to the issue of defining a notion of equivalence for our language. In general the ability to distinguish between systems depends on our knowledge of the capabilities at the various sites. For example a client who is not aware of the resource a at the location k will be unable to perceive any difference between the two systems $k[\![a?(x)\, P]\!]$ and $k[\![\mathsf{stop}]\!]$. Thus, as explained in the Introduction, we develop notions of equivalences of the form

$$\Gamma \models M \approx N \tag{4}$$

where Γ is a well-defined type environment representing the user's knowledge of the capabilities of the systems M and N. It is important to realise that Γ may not contain enough information to type M, N; it represents that part of the type environment of these processes which has been released to the user.

A *knowledge-indexed relation over systems* is a family of binary relations between closed systems indexed by closed type environments. We write $\Gamma \models M \mathcal{R} N$ to mean that systems M and N are related by $\mathcal{R}$ at index Γ; moreover we assume that both M and N are typeable relative to some *extension* of Γ; that is some environment Δ with the same domain as Γ such that $\Delta <: \Gamma$. The desirable properties of knowledge-indexed relations which we consider are as follows:

Reduction closure: We say that a knowledge-indexed relation over systems is *reduction closed* if whenever $\Gamma \models M \mathcal{R} N$ and $M \to M'$ there exists some N' such that $N \to^* N'$ and $\Gamma \models M \mathcal{R} N'$.

Context closure: We say that a knowledge-indexed relation over systems is *contextual* if

(i) $\Gamma \models M \mathcal{R} N$ implies $\Gamma, \Gamma' \models M \mathcal{R} N$
(ii) $\Gamma \models M \mathcal{R} N$ and $\Gamma \vdash O$ implies $\Gamma \models (M \mid O) \mathcal{R} (N \mid O)$
(iii) $\Gamma, n : \mathrm{T} \models M \mathcal{R} N$ implies $\Gamma \models (\mathsf{new}\, n : \mathrm{T})\ M\ \ \mathcal{R}\ \ (\mathsf{new}\, n : \mathrm{T})\ N$

Note that in this last clause we have used an abbreviation to cover the three different forms of names which can be declared, local channels, registered names and locations, each differentiated by the form which T can take. Moreover we assume that n is new to Γ.

Barb preservation: For any given location k and any given channel a such that $\Gamma \vdash k : \mathsf{loc}$ and $\Gamma \vdash a : \mathsf{rw}\langle\rangle@k$ we write $\Gamma \vdash M \Downarrow^{\mathsf{barb}} a@k$ if there exists some M' such that $M \to^* (\mathsf{new}\, \tilde{n})\ (M' \mid k[\![a!\langle\rangle\, P]\!])$, where k, a do not appear in $\tilde{n}$. We say that a knowledge-indexed relation over systems is *barb preserving* if

$$\Gamma \models M \mathcal{R} N \quad \text{and} \quad \Gamma \vdash M \Downarrow^{\mathsf{barb}} a@k \quad \text{implies} \quad \Gamma \vdash N \Downarrow^{\mathsf{barb}} a@k$$

These three properties determine our *touchstone* equivalence:

Definition 1 (Reduction barbed congruence). *We let $\cong^{rbc}$ be the largest knowledge-indexed relation over systems which is*

– *pointwise symmetric, that is $\Gamma \models M \cong^{rbc} N$ implies $\Gamma \models N \cong^{rbc} M$*
– *contextual*
– *reduction closed*
– *barb preserving* □

We will now characterise $\cong^{rbc}$ using a labelled transition system and bisimulation equivalence, thereby justifying our particular notion of bisimulations.

Fig. 5. Typed actions

(LTS-RED)

$$\frac{M \longrightarrow M'}{(\Gamma \triangleright M) \xrightarrow{\tau} (\Gamma \triangleright M')}$$

(LTS-OUT)

$$\frac{\begin{array}{l}\Gamma \vdash k : \mathsf{loc}\\ a : \mathsf{r}\langle \mathrm{T}\rangle@k \in \Gamma\\ \Gamma \sqcap \langle V : \mathrm{T}\rangle@k \text{ exists}\end{array}}{(\Gamma \triangleright k[\![a!\langle V\rangle\, P]\!]) \xrightarrow{k.a!V} (\Gamma \sqcap \langle V : \mathrm{T}\rangle@k \triangleright k[\![P]\!])}$$

(LTS-IN)

$$\frac{\begin{array}{l}\Gamma \vdash k : \mathsf{loc}\\ a : \mathsf{w}\langle \mathrm{U}\rangle@k \in \Gamma\\ \Gamma \vdash V : \mathrm{U}@k\end{array}}{(\Gamma \triangleright k[\![a?(X : \mathrm{T})\, P]\!]) \xrightarrow{k.a?V} (\Gamma \triangleright k[\![P\{\!|V/X|\!\}]\!])}$$

(LTS-OPEN)

$$\frac{(\Gamma, n : \top \triangleright M) \xrightarrow{(\tilde{n})k.a!V} (\Gamma' \triangleright M')}{(\Gamma \triangleright (\mathsf{new}\, n : \mathrm{T})\ M) \xrightarrow{(n\tilde{n})k.a!V} (\Gamma' \triangleright M')} \quad \begin{array}{l} n \neq a, k\\ n \in \mathsf{fn}(V)\end{array}$$

(LTS-WEAK)

$$\frac{(\Gamma, n : T \triangleright M) \xrightarrow{(\tilde{n}:\tilde{\mathrm{T}})k.a?V} (\Gamma' \triangleright M')}{(\Gamma \triangleright M) \xrightarrow{(n:\mathrm{T}\tilde{n}:\tilde{\mathrm{T}})k.a?V} (\Gamma' \triangleright M')} \quad n \neq a, k$$

(LTS-PAR)

$$\frac{(\Gamma \triangleright M) \xrightarrow{\alpha} (\Gamma' \triangleright M')}{\begin{array}{l}(\Gamma \triangleright M \mid N) \xrightarrow{\alpha} (\Gamma' \triangleright M' \mid N)\\ (\Gamma \triangleright N \mid M) \xrightarrow{\alpha} (\Gamma' \triangleright N\ \mid M')\end{array}} \quad \mathsf{bn}(\alpha) \notin \mathsf{fn}(N)$$

(LTS-NEW)

$$\frac{(\Gamma, n : \top \triangleright M) \xrightarrow{\alpha} (\Gamma', n : \top \triangleright M')}{(\Gamma \triangleright (\mathsf{new}\, n : \mathrm{T})\ M) \xrightarrow{\alpha} (\Gamma' \triangleright (\mathsf{new}\, n : \mathrm{T})\ M')} \quad n \notin \mathsf{n}(\alpha)$$

4.1 A Labelled Transition Characterisation of Contextual Equivalence

A standard labelled transition system for DPI would describe the actions, inputs/outputs on located channels, which a system could in principle perform. However because of possible limited knowledge an external user may not be able to provoke these actions. Our labelled transition system uses *typed actions* of the form

$$\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright M'$$

and the defining rules are given in Figure 5. Again the intuition here is that M is not necessarily typeable by Γ; instead it represents that part of the environment which does type M which is known by the user.

As an example the rule (LTS-OUT) says that $k[\![a!\langle V\rangle\, P]\!]$ can only perform the obvious output action if

- k is known by Γ to be a location
- the user has the capability to accept a value from a at k, that is $\Gamma \vdash a : \mathsf{r}\langle \mathrm{T}\rangle_{@}k$ for some transmission type T
- the information which is being sent to the user does not contradict its current knowledge. This is formalised using a partial meet operation $\sqcap$ which is defined directly on types and extended to type environments; it is a method for *consistently* combining type information. In the the rule we also use the notation $\langle V : \mathrm{T}\rangle_{@}k$ to represent a simple environment obtained by adding the knowledge that V has type T at location k. The formal definition may be found in [7].

The rule for input, (LTS-IN), has a similar flavour.

Definition 2 (Bisimulations). *A binary relation $\mathcal{R}$ is said to be a* bisimulation *if $\Gamma \triangleright M \;\mathcal{R}\; \Gamma \triangleright N$ implies*

- *$\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright M'$ implies $\Gamma \triangleright N \overset{\hat{\mu}}{\Longrightarrow} \Gamma' \triangleright N'$ for some N' such that $\Gamma' \triangleright M' \;\mathcal{R}\; \Gamma' \triangleright N'$*
- *Symmetrically, $\Gamma \triangleright N \xrightarrow{\mu} \Gamma' \triangleright N'$ implies $\Gamma \triangleright M \overset{\hat{\mu}}{\Longrightarrow} \Gamma' \triangleright M'$ for some N' such that $\Gamma' \triangleright M' \;\mathcal{R}\; \Gamma' \triangleright N'$*

Here we are using the standard notation from [12]; $\overset{\mu}{\Longrightarrow}$ means $\xrightarrow{\tau}^{*} \circ \xrightarrow{\mu} \circ \xrightarrow{\tau}^{*}$ while $\overset{\hat{\mu}}{\Longrightarrow}$ is $\xrightarrow{\tau}^{*}$ if μ is τ and $\overset{\mu}{\Longrightarrow}$ otherwise; this allows a single internal move to be matched by zero or move internal moves.

We write $\Gamma \models M \approx^{bis} N$ if $(\Gamma \triangleright M)\;\mathcal{R}\;(\Gamma \triangleright N)$ for some bisimulation $\mathcal{R}$, and say that M and N are bisimilar in the environment Γ.

Theorem 1 (Full abstraction of $\eqsim^{rbc}$ for $\approx^{bis}$). *$\Gamma \models M \eqsim^{rbc} N$ iff $\Gamma \models M \approx^{bis} N$* □

5 Controlling Mobility

We now consider a richer calculus in which movement of processes may be controlled. As explained in the Introduction we extend DPI with a very simple means of mobility control and investigate the resulting contextual equivalence.

5.1 Migration Rights

Hennessy and Riely have already proposed a simple access control mechanism for Dpi in the form of the *move* capability [9], and here we investigate the effect this has on behavioural equivalence.

The location types in Dpi are of the form $\mathsf{loc}[u_1 : \mathrm{A}_1, \ldots, u_n : \mathrm{A}_n]$ where the $u_i : \mathrm{A}_i$ can be seen as capabilities at that location. We now introduce an extra type of capability by allowing location types to be also of the form

$$\mathsf{loc}[\mathbf{move}, a_1 : \mathrm{A}_1, \ldots, a_n : \mathrm{A}_n]. \tag{5}$$

If a location k is known at this type then agents resident at any location have migration rights to k; we realise that this is a somewhat simplistic approach to mobility control but it enables us to demonstrate the subtleties involved in developing behavioural theories in the presence of any such capabilities.

The type system, and the type-checking rules, can easily modified to cater for this new capability while retaining Subject Reduction; for details see the full version of the paper, [7].

Instead let us examine the effect migration rights have on behavioural equivalences. Suppose N_1, N_2 are given by

$$k[\![a!\langle\rangle\,\mathsf{stop}]\!] \qquad \text{and} \qquad k[\![\mathsf{stop}]\!] \tag{6}$$

The question of whether or not N_1 and N_2 are contextually equivalent relative to an environment Γ, now written $\Gamma \models N_1 \approx^m_{cxt} N_2$, depends on whether Γ has migration rights to k. If so, say at a location l, agents may be sent from l to k in order to observe the difference in behaviour between N_1 and N_2 at k. But will these agents be able to report back to the environment? This in turn depends on whether the environment allows migration from k to l. In general, the situation can get more complicated. Observing different behaviour at a site k may require a range of capabilities, and knowledge of these may be distributed throughout the environment at sites with limited migration rights between themselves.

For this extended language we give, in the following two subsections, two different generalisations to the full-abstraction result, Theorem 1.

5.2 Mobility Bisimulation Equivalence

It is straightforward to adapt the typed actions in Figure 5 to take into account these simple migration rights. Essentially for an action to be allowed at a site k the constraints discussed in Section 4.1 must be satisfied but in addition the environment must have migration rights to k. Formally we define actions

$$\Gamma \triangleright M \xrightarrow{\mu}_m \Gamma' \triangleright M'$$

by replacing the rules (LTS-OUT) and (LTS-IN) in Figure 5 with

$$\frac{\begin{array}{l}\text{(LTS-OUT}_{\text{M}}\text{)}\\ \Gamma \vdash k : \mathsf{loc}[\mathbf{move}] \\ a : \mathsf{r}\langle \mathrm{T} \rangle_{@} k \in \Gamma \\ \Gamma \sqcap \langle V : \mathrm{T} \rangle_{@} k \text{ exists}\end{array}}{(\Gamma \triangleright k[\![a!\langle V \rangle\, P]\!]) \xrightarrow{k.a!V}_m (\Gamma \sqcap \langle V : \mathrm{T} \rangle_{@} k \triangleright k[\![P]\!])}$$

$$\frac{\begin{array}{l}\text{(LTS-IN}_{\text{M}}\text{)}\\ \Gamma \vdash k : \mathsf{loc}[\mathbf{move}] \\ a : \mathsf{w}\langle \mathrm{U} \rangle_{@} k \in \Gamma \\ \Gamma \vdash V : \mathrm{U}_{@} k\end{array}}{(\Gamma \triangleright k[\![a?(X : \mathrm{T})\, P]\!]) \xrightarrow{k.a?V}_m (\Gamma \triangleright k[\![P\{\!|^{V}\!/_{X}|\!\}]\!])}$$

and leaving the other rules unchanged.

Definition 3 (Typed m-Bisimulations). *Let $\Gamma \models M \approx^m_{bis} N$ denote the resulting version of bisimulation equivalence, obtained by replacing the use of $\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright M'$ in Definition 2 with $\Gamma \triangleright M \xrightarrow{\mu}_m \Gamma' \triangleright M'$.*

Example 1. As in (6) above let N_1, N_2 denote $k[\![a!\langle\rangle\, \mathsf{stop}]\!]$ and $k[\![\mathsf{stop}]\!]$ respectively, and suppose Γ is such that $\Gamma \not\vdash k : \mathsf{loc}[\mathbf{move}]$. Then $\Gamma \models N_1 \approx^m_{bis} N_2$ because no m-typed actions are possible from these systems.

Example 2. Let N_3, N_4 represent $(\mathsf{new}\, k : \mathsf{loc}[\mathbf{move}, b : \mathsf{rw}\langle\rangle])\;\; l[\![a!\langle k \rangle]\!] \mid k[\![b!\langle\rangle]\!]$ and $(\mathsf{new}\, k : \mathsf{loc}[\mathbf{move}, b : \mathsf{rw}\langle\rangle])\;\; l[\![a!\langle k \rangle]\!] \mid k[\![\mathbf{0}]\!]$ respectively, and let Γ_1 denote the environment

$$l : \mathsf{loc},\; l : \mathbf{move},\; b : \mathsf{rc}\langle \mathsf{rw}\langle\rangle \rangle,\; a : \mathsf{rw}\langle \mathsf{loc}[b : \mathsf{rw}\langle\rangle] \rangle_{@} l$$

Here the environment can interact at the site l because it has migration rights there; and via the channel a located at l it can gain knowledge of k. But because of the type at which it knows a it can never gain migration rights to k. Consequently we have $\Gamma_1 \models N_3 \approx^m_{bis} N_4$.

However let Γ_2 denote

$$l : \mathsf{loc},\; l : \mathbf{move},\; b : \mathsf{rc}\langle \mathsf{rw}\langle\rangle \rangle,\; a : \mathsf{rw}\langle \mathsf{loc}[\mathbf{move}, b : \mathsf{rw}\langle\rangle] \rangle_{@} l$$

Here any location name received on the channel a at l comes with migration rights. So we have $\Gamma_2 \models N_3 \not\approx^m_{bis} N_4$.

The essential property of this new form of equivalence is a restricted form of contextuality:

Proposition 1. *Suppose $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$. Then $\Gamma \models M \approx^m_{bis} N$ and $\Gamma \vdash k[\![P]\!]$ implies $\Gamma \models M \mid k[\![P]\!] \approx^m_{bis} N \mid k[\![P]\!]$.*

This property allows us to give a contextual characterisation of $\approx^m_{bis}$. We need to slightly adapt the concepts defined in Section 4.

m-Context closure: Here the change is in the second clause of the definition of *Context closure*; it is changed to

(ii) $\Gamma \models M \mathcal{R} N$, $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$ and $\Gamma \vdash k[\![P]\!]$ implies $\Gamma \models (M \mid k[\![P]\!]) \mathcal{R} (N \mid k[\![P]\!])$

m-Barb preservation: Here we only allow barbs at locations to which migration rights exist. We write $\Gamma \vdash M \Downarrow^{\mathsf{mbarb}} a@k$ if in addition to the requirement that $\Gamma \vdash M \Downarrow^{\mathsf{barb}} a@k$ we have $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$ and $\Gamma \vdash a : \mathsf{rw}\langle\rangle@k$.

We now say that a typed relation over systems is *m-barb preserving* if $\Gamma \models M \mathcal{R} N$ and $\Gamma \vdash M \Downarrow^{\mathsf{mbarb}} a@k$ implies $\Gamma \vdash N \Downarrow^{\mathsf{mbarb}} a@k$.

Definition 4 (m-Reduction barbed congruence). *Let $\cong^m_{cxt}$ be the largest typed relation over systems which is reduction-closed, m-contextual and m-barb preserving.*

Our first generalisation may now be stated:

Theorem 2 (Full abstraction of $\cong^m_{cxt}$ for $\approx^m_{bis}$). *$\Gamma \models M \cong^m_{cxt} N$ iff $\Gamma \models M \approx^m_{bis} N$.* □

5.3 Re-examining Contextuality

The two examples given in the previous subsection deserve re-examination, particularly in view of the definition of m-contextuality. In Example 1 above it turns out that N_1 and N_2 are not equivalent with respect to any Γ which does not contain migration rights to k. But an alternative definition of *contextual* would require the behavioural equivalence to be preserved by *all* contexts typeable by Γ. Suppose Γ is the environment

$$h : \mathsf{loc},\ h : \mathbf{move},\ \mathit{eureka} : \mathsf{rw}\langle\rangle@h,\ k : \mathsf{loc},\ a : \mathsf{rw}\langle\rangle@k$$

Then one can check that $\Gamma \vdash k[\![a?()\,\mathsf{goto}\,h.\mathit{eureka}!\langle\rangle]\!]$ and running N_i in parallel with this well-typed context would enable us to distinguish between them.

This new, but still informal, notion of contextuality presupposes that the context can have already in place some testing agents running at certain sites to which it does not have migration rights. An obvious choice of sites would be all those which are known about, that is all k such that $\Gamma \vdash k : \mathsf{loc}$. However our results can be parameterised on this choice.

$\mathcal{T}$-Context closure: Let $\mathcal{T}$ be a collection of location names. The concept of $\mathcal{T}$-Context closure is obtained by changing the second clause in the definition of *Context closure* to:

(ii) $\Gamma \models M \mathcal{R} N$, $\Gamma \vdash k[\![P]\!]$, where either $k \in \mathcal{T}$ or $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}]$, implies $\Gamma \models (M \mid k[\![P]\!]) \mathcal{R} (N \mid k[\![P]\!])$

Definition 5 ($\mathcal{T}$-Reduction barbed congruence). *Let $\cong^{\mathcal{T}}_{cxt}$ be the largest typed relation over systems which is reduction-closed, $\mathcal{T}$-contextual and m-barb preserving.*

The question now is whether we can devise a bisimulation based characterisation of $\cong^{\mathcal{T}}_{cxt}$.

The obvious approach is to modify the definitions of the typed actions $\xrightarrow{\mu}_m$, to obtain actions $\xrightarrow{\mu}_{\mathcal{T}}$ which allow observations at a site k, if either the environment has migration rights to k as before, **or** $k \in \mathcal{T}$. With these actions we can modify Definition 2 to obtain a new behavioural equivalence, which we denote by $\approx^{\mathcal{T}}_{bis}$. Unfortunately this does not coincide with the contextual equivalence $\cong^{\mathcal{T}}_{cxt}$.

Example 3. Let N_5, N_6 be the systems defined by

$$h[\![a!\langle b@k\rangle]\!] \mid k[\![b!\langle\rangle]\!] \quad \text{and} \quad h[\![a!\langle b@k\rangle]\!] \mid k[\![\mathsf{stop}]\!]$$

and Γ the environment

$$h : \mathsf{loc},\ h : \mathbf{move},\ k : \mathsf{loc},\ a : \mathsf{rw}\langle \mathsf{r}\langle\rangle@k\rangle@h,\ b : \mathsf{w}\langle\top\rangle@k$$

Then if k is in $\mathcal{T}$ one can check that $N_5 \not\approx^{\mathcal{T}}_{bis} N_6$. This is because $\Gamma \triangleright N_5$ can perform the action $h.a!b@k$ followed by $k.b!\langle\rangle$, which can not be matched by $\Gamma \triangleright N_6$.

However $\Gamma \models N_5 \cong^{\mathcal{T}}_{cxt} N_6$ because it is not possible to find a context to distinguish between them. A context can be found to augment the knowledge of the environment at h with the read capability for b at k. But, in a well-typed context, it is not possible to transfer this information from h to where it can be put to use, namely k.

This example demonstrates that even with our very restricted move capability there are problems with the flow of information. Knowledge about the system learnt at l can not necessarily be passed to k if the environment does not have move capability at k. This motivates the new form of configurations we introduce for the labelled transition system necessary in order to characterise $\cong^{\mathcal{T}}_{cxt}$.

We replace a simple Γ with a structure $\overline{\Gamma} = (\Gamma, \Gamma_{k_1}, \ldots, \Gamma_{k_n})$ where the k_i make up $\mathcal{T}$. Each Γ_{k_i} represents localised knowledge at k_i whereas Γ represents the centralised knowledge, available at any location for which we have move capability. Given that we can store the centralised knowledge at a location k_0, provided by the environment (with move capability), we can always pass local knowledge on to k_0 (but not vice versa). Thus centralised knowledge is always greater than any of the local knowledge environments. This leads us to the following definition:

Definition 6 (Configurations). *A* configuration $(\overline{\Gamma} \triangleright M)$ *over $\mathcal{T}$ consists of a family of type environments $\overline{\Gamma} = \Gamma, \Gamma_{k_1}, \ldots, \Gamma_{k_n}$ such that*

Fig. 6. Labelled transition rules accounting for the move capability

(LTS-MOVE−OUT)

$$\frac{\begin{array}{l}\Gamma \vdash k : \mathsf{loc}[\mathbf{move}] \\ a : \mathsf{r}\langle \mathrm{T}\rangle@k \in \Gamma \\ \Gamma \sqcap \langle v : \mathrm{T}\rangle@k \text{ exists}\end{array}}{(\overline{\Gamma} \triangleright k[\![a!\langle v\rangle\, P]\!]) \xrightarrow{k.a!v} (\overline{\Gamma} \sqcap_0 \langle v : \mathrm{T}\rangle@k \triangleright k[\![P]\!])}$$

(LTS-T−OUT)

$$\frac{\begin{array}{l}\Gamma \not\vdash k : \mathsf{loc}[\mathbf{move}] \\ k \in \mathcal{T} \\ a : \mathsf{r}\langle \mathrm{T}\rangle@k \in \Gamma_k \\ \overline{\Gamma} \sqcap_0 \langle v : \mathrm{T}\rangle@k \sqcap_k \langle v : \mathrm{T}\rangle@k \text{ exists}\end{array}}{(\overline{\Gamma} \triangleright k[\![a!\langle v\rangle\, P]\!]) \xrightarrow{k.a!v} (\overline{\Gamma} \sqcap_0 \langle v : \mathrm{T}\rangle@k \sqcap_k \langle v : \mathrm{T}\rangle@k \triangleright k[\![P]\!])}$$

(LTS-MOVE−IN)

$$\frac{\begin{array}{l}\Gamma \vdash k : \mathsf{loc}[\mathbf{move}] \\ \Gamma \vdash a : \mathsf{w}\langle \mathrm{T}\rangle@k \\ \Gamma \vdash v : \mathrm{T}@k\end{array}}{(\overline{\Gamma} \triangleright k[\![a?(X : \mathrm{A})\, P]\!]) \xrightarrow{k.a?v} (\overline{\Gamma} \triangleright k[\![P\{\!|{}^{v}\!/\!{}_{x}|\!\}]\!])}$$

(LTS-T−IN)

$$\frac{\begin{array}{l}\Gamma \not\vdash k : \mathsf{loc}[\mathbf{move}] \\ k \in \mathcal{T} \\ \Gamma_k \vdash a : \mathsf{w}\langle \mathrm{T}\rangle@k \\ \Gamma_k \vdash v : \mathrm{T}@k\end{array}}{(\overline{\Gamma} \triangleright k[\![a?(X : \mathrm{A})\, P]\!]) \xrightarrow{k.a?v} (\overline{\Gamma} \triangleright k[\![P\{\!|{}^{v}\!/\!{}_{x}|\!\}]\!])}$$

(LTS-T−WEAK)

$$\frac{(\overline{\Gamma}, \overline{(n : T)}_{\triangledown} \triangleright M) \xrightarrow{(\tilde{n}:\tilde{\mathrm{T}})k.a?V} (\overline{\Gamma}' \triangleright M')}{(\overline{\Gamma} \triangleright M) \xrightarrow{(n:\mathrm{T}\tilde{n}:\tilde{\mathrm{T}})k.a?V} (\overline{\Gamma}' \triangleright M')} \quad n \neq a, k$$

– $\mathcal{T} = \{k_1, \ldots, k_n\}$
– $\Gamma <: \Gamma_{k_i}$ *for each* $1 \leq i \leq n$

and a system M which can be typed in some extension *of Γ.*

We will write $\overline{\Gamma}^{\mathcal{T}}_{\triangledown}$ to mean the family of environments $\Gamma, \Gamma_{k_1}, \ldots, \Gamma_{k_n}$ such that each component Γ_{k_i} is equal to the environment Γ; we will typically omit the parameter $\mathcal{T}$ here as it can usually be recovered from context. We understand $\overline{\Gamma}, \overline{\Gamma}'$ and $\overline{\Gamma} \sqcap \overline{\Gamma}'$ to be pointwise operations. Finally we need a notation for increasing knowledge in the individual components of a configuration, for which we use the notation $\sqcap_k$. For instance we write $\overline{\Gamma} \sqcap_0 \Gamma'$ to mean the family such that the global component becomes $\Gamma \sqcap_0 \Gamma'$ and all other components are unchanged. Similarly $\overline{\Gamma} \sqcap_k \Gamma'$ adds, if possible, Γ' to the k^{th} component.

We define our new labelled transition system, parameterised on $\mathcal{T}$, as binary relations between these new configurations. We replace the rules (LTS-OUT) and (LTS-IN) in Figure 5 with those in Figure 6 and modifying the remaining rules in Figure 5 in the obvious manner. The standard definition of (weak) bisimilarity may be applied to this new labelled transition system. To emphasise the role of the parameter $\mathcal{T}$ we will write the resulting equivalence as $\approx^{\mathcal{T}}_{bis}$.

We can now state the final result of the paper that, by considering a configuration in which the knowledge at every locality is initially Γ, then bisimilarity coincides with reduction barbed congruence with respect to Γ:

Theorem 3 (Full abstraction). *In* DPI *with restricted mobility*

$$\Gamma \models M \cong^{\mathcal{T}}_{cxt} N \qquad \textit{iff} \qquad \overline{\Gamma}_{\triangledown} \models M \approx^{\mathcal{T}}_{bis} N.$$

6 Conclusions and Related Work

We have presented two labelled transition systems for which bisimilarity coincides with a natural notion of contextual equivalence for distributed systems. The labelled transitions rely upon a type discipline for the language which can control resource access and mobility. As in [8,2], the use of a type environment representing the tester's knowledge of the system plays an important role in characterising the contextual equivalences. In particular it aided us in defining a labelled transition system which accounts for information flow in a distributed setting with restricted mobility.

There has been a great deal of interest in modelling distributed systems using calculi in recent years, [14,6,1,4,16,9,3]. The emphasis so far has largely been on design of the languages to give succinct descriptions of mobile processes with type systems given to constrain behaviour in a safe manner. Where equivalence has been used it has typically been introduced as some sort of contextual equivalence very similar to the one found in the present paper [6,1,11]. Proofs of correctness of protocols or language translations have been carried out with respect to these contextual equivalences. Recently in [5] a form of bisimulation has been suggested as a proof method for establishing contextual equivalence in the Seal calculus; but, as far as we know, the only existing example of an operational characterisation of behavioural equivalence in the distributed setting is found in [10].

The work in [15] takes a different, more intensional approach to equivalence in the distributed setting in that, in order to establish correctness of a particular protocol, a novel notion of equivalence based on coupled simulation tailored to accommodate migration is identified. Although having many interesting properties such as congruence, this equivalence is not shown to coincide with any independent contextually defined notion of equivalence.

References

1. Roberto M. Amadio and Sanjiva Prasad. Modelling IP mobility. In Davide Sangiorgi and Robert de Simone, editors, *CONCUR '98: Concurrency Theory (9th International Conference, Nice, France)*, volume 1466 of *LNCS*, pages 301–316. Springer, September 1998.
2. M. Boreale and D. Sangiorgi. Bisimulation in name-passing calculi without matching. In *13th LICS Conf.* IEEE Computer Society Press, 1998.
3. Luca Cardelli. A language with distributed scope. *Computing Systems*, 8(1):27–59, 1995. Short version in *Proceedings of POPL '95.* A preliminary version appeared as Report 122, Digital Systems Research, June 1994.
4. Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, June 2000.
5. G. Castagna and F. Zappa. The seal calculus revisited. In *22th Conference on the Foundations of Software Technology and Theoretical Computer Science.* Springer-Verlag, 2002.
6. Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile agents. In *7th International Conference on Concurrency Theory (CONCUR'96)*, pages 406–421, Pisa, Italy, August 26-29 1996. Springer-Verlag. LNCS 1119.
7. M. Hennessy, M. Merro, and J. Rathke. Towards a behavioural theory of access and mobility control in distributed systems. Computer Science Report 2002:01, University of Sussex, 2002.
8. M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. In *Proc. CATS2002, Computing: Australasian Theory Symposium, Melbourne 2002*, 2002. Also available as a University of Sussex technical report.
9. M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173:82–120, 2002.
10. M. Merro and M. Hennessy. Bisimulation congruences in safe ambients. *ACM SIGPLAN Notices*, 31(1):71–80, January 2002.
11. M. Merro, J. Kleist, and U. Nestmann. Mobile Objects as Mobile Processes. *To appear in Journal of Information and Computation*, 2002.
12. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
13. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.
14. Peter Sewell. Global/local subtyping and capability inference for a distributed pi-calculus. In *ICALP 98*, volume 1443 of *LNCS*. Springer, 1998.
15. Asis Unyapoth and Peter Sewell. Nomadic pict: Correct communication infrastructure for mobile computation. In *Conference Record of POPL'01: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 236–247, London, United Kingdom, January 17–19, 2001.
16. J. Vitek and G. Castagna. A calculus of secure mobile computations. In *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*. Springer, 1999.