Towards a Behavioural Theory of Access and Mobility Control in Distributed Systems[1]

Matthew Hennessy[a], Massimo Merro[b], Julian Rathke[a]

[a] *University of Sussex, UK*

[b] *University of Verona*

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.

Key words: distributed picalculus, access control, capability types, mobility control, contextual equivalence, bisimulations

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 should be based not only on the inherent abilities of processes to interact with other processes, but must also take into account the

[1] The authors acknowledge the financial support of EPSRC grant GR/M71169, the Royal Society and the Mikado and Myths projects, and the helpful criticisms of the anonymous referees.

(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
- 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, [11], an extension of the π-calculus, [17], in which processes may migrate between locations, which in turn can be dynamically created. As explained in [11] 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. However in this paper we extend the typing system of [11] by allowing types to be created *dynamically* and to depend on received data.

In DPI a typical system can take the form

$$l[\![P]\!] \mid (\mathsf{new}\, e : \mathrm{E})\ (k[\![Q]\!] \mid l[\![R]\!])$$

Here there are two threads P and R running at l and one, Q, running at k. The threads Q and R share the private name e at type E. The threads $P,\ Q,\ R$ are similar to processes in the π-calculus in that they can receive and send values on local channels; the types of these channels indicate the kind of values which may be transmitted. For example in

$$l[\![(\mathsf{newloc}\, k : \mathrm{K})\ \ \mathsf{with}\ P\ \mathsf{in}\ (\mathsf{xpt}_1!\langle k\rangle \mid \mathsf{xpt}_2!\langle k\rangle \quad \mid \quad R)]\!],$$

in parallel with the execution of R at l, a new location k is created at type K, the code P is installed there, and the name of the new location is exported via the channels xpt_i; as we will see this system evolves to

$$(\mathsf{new}\, k : \mathrm{K})\ (l[\![\mathsf{xpt}_1!\langle k\rangle \mid \mathsf{xpt}_2!\langle k\rangle \ \mid\ R]\!] \quad \mid \quad k[\![P]\!])$$

Location types are similar to record types, their form being

$$\mathsf{loc}[a_1 : \mathrm{A}_n, \ldots a_n : \mathrm{A}_n]$$

This indicates that the channels or resources a_i at types A_i are available at the location; each pair $a_i : \mathrm{A}_i$ can be viewed as a capability at that location. So for example K above could be

$$\mathsf{loc}[\mathsf{ping} : \mathsf{rw}\langle \mathrm{A}\rangle, \mathsf{finger} : \mathsf{rw}\langle \mathrm{B}\rangle]$$

indicating that the services ping and finger are supported (via both read and write communication) at k. However the types at which k becomes known depend on the types of the exporting channels. Suppose for example these had the types

$$\begin{aligned}&\mathsf{xpt}_1 : \mathsf{w}\langle \mathsf{loc}[\mathsf{ping} : \mathsf{w}\langle \mathrm{A}\rangle]\rangle\\ &\mathsf{xpt}_2 : \mathsf{w}\langle \mathsf{loc}[\mathsf{finger} : \mathsf{w}\langle \mathrm{B}\rangle]\rangle\end{aligned}$$

This means, for example, that xpt_1 has the capability of writing values of type $\mathsf{loc}[\mathsf{ping} : \mathsf{w}\langle \mathrm{A}\rangle]$, that is locations which have a resource called ping, which may be used there at type $\mathsf{w}\langle \mathrm{A}\rangle$. Then processes receiving the name k from the source xpt_1 would only be able to *write* to the ping service at k, while the source xpt_2 only allows similar access to the finger service. In effect different capabilities at k are obtained via different sources. It is in this way, by selectively distributing names at particular super-types, that resource access policies are implemented in DPI.

The language DPI and its reduction semantics is summarised in Section 2, which relies heavily on the corresponding section of [11]. The typing system is discussed in Section 3. This contains two major extensions to the original typing system of [11]. 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. The first main result of the paper is a Subject Reduction theorem for this new typing system.

Section 4.1 contains a definition of *typed action*

$$\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright N \tag{1}$$

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 $k.a!v$, meaning the input or output of the value v along the channel a located at k. 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) bisimulation between *configurations*, consistent pairs of the form $\Gamma \triangleright M$; the formal definition is given in Section 4 and we use $\Gamma \models M \approx_{bis} N$ to denote that there is a bisimulation containing the two configurations $\Gamma \triangleright M$ and $\Gamma \triangleright N$.

The second main result of the paper is that this notion of *typed bisimulation* captures precisely an independently defined *contextual equivalence.* In Section 4 we define $\Gamma \models M \underset{\sim}{\approx}_{rbc} N$ to be the largest parameterised equivalence which is

- closed with respect to reductions, that is preserves in some sense the reduction semantics
- preserved, in a suitable sense, with respect to Γ-system contexts
- preserves simple observations, which we call *distributed barbs.*

We prove the theorem:

$$\text{In } \textsc{Dpi}, \quad \Gamma \models M \approx_{bis} N \text{ if and only if } \Gamma \models M \underset{\sim}{\approx}_{rbc} N \tag{2}$$

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

$$(\mathsf{new}\, k : \mathrm{K})\ l[\![c!\langle k\rangle]\!] \mid k[\![a!\langle\rangle\, \mathsf{stop}]\!] \qquad \text{and} \qquad (\mathsf{new}\, k : \mathrm{K})\ l[\![c!\langle k\rangle\, \mathsf{stop}]\!] \mid k[\![\mathsf{stop}]\!] \tag{3}$$

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

$$\mathbf{move}_*$$

Then, relative to an environment Γ, migration is only allowed to a location k if k is known in Γ with the capability $\mathbf{move}_*$. 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 details are given in Section 5, and more realistic generalisations are discussed in the Conclusion.

The remainder of the paper is devoted to extending the result (2) above to this language. The typed actions (1) are readily adapted to this scenario. Here we allow, for example, the action

$$\Gamma \triangleright M \xrightarrow{k.a!v}_m \Gamma' \triangleright N \tag{4}$$

if, in addition to the requirements for (1) we require that the environment has the capability $\mathbf{move}_*$ for k; intuitively for the environment to see the action (4) 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

$$\Gamma \models M \approx^m_{bis} N \text{ if and only if } \Gamma \models M \eqsim^m_{rbc} N$$

where $\eqsim^m_{rbc}$ is a suitable modification of the contextual equivalence $\eqsim_{rbc}$. For this modification we only require the equivalence to be preserved by processes at locations to which the environment has migration rights. Thus referring to (3) above we will have

$$\Gamma \models M_1 \eqsim^m_{rbc} M_2$$

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 (4) above.

However it is easy to envisage a natural version of contextual equivalence which does distinguish between M_1 and M_2 of (3) 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 $\cong^{\mathcal{T}}_{rbc}$ be the resulting contextual equivalence. Unfortunately this is not characterised by the natural modification to the equivalence $\approx^{m}_{bis}$, which we denote by $\approx^{\mathcal{T}}_{bis}$. This is defined using actions such as

$$\Gamma \triangleright M \xrightarrow{k.a!v}_{\mathcal{T}} \Gamma' \triangleright N \tag{5}$$

which are only allowed if either the environment has migration rights to k, as before, or k is in $\mathcal{T}$. 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 Dpi with controlled migration, $\overline{\Gamma} \models M \approx^{\mathcal{T}}_{bis} N$ if and only if $\overline{\Gamma} \models M \cong^{\mathcal{T}}_{rbc} N$.

2 The language Dpi

Syntax: The syntax, given in Figure 1, is a slight extension of that of Dpi from [11]. 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, which are defined in Figure 4; discussion of these is postponed until the next section.

From Figure 1 we can see that values take the form of tuples $(\alpha_1, \ldots, \alpha_n)$, for $n > 0$. Each α_i can be a simple identifier, referring to a local channel or a location, or it may take the form $(u_1, \ldots, u_n)@u$, representing a sequence of channels $(u_1, \ldots, u_n)$ each located at u.

Compound values are deconstructed using *patterns*, ranged over by the meta-

$M, N \ ::=$	*Systems*
$l[\![P]\!]$	Located Process
$M \mid N$	Composition
$(\mathsf{new}\, n : \mathrm{E})\ M$	Name Scoping
$\mathbf{0}$	Termination

$P, Q \ ::=$	*Processes*
$u!\langle V\rangle\, P$	Output
$u?(X : \mathrm{T})P$	Input
$\mathsf{goto}\, v.P$	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
$(\mathsf{newloc}\, k : \mathrm{K})\ \mathsf{with}\ P\ \mathsf{in}\ Q$	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

Fig. 1. Syntax of Dpi

variables $X, Y, \ldots$.; these are values comprised entirely of distinct variables. For example the pattern $(x, (y_1, y_2)_{@}z)$ will deconstruct a value into two components, requiring the second one to have the form $(n_1, n_2)_{@}k$.

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, [17], with primitives for migration between locations. As in the π-calculus, we have input and output of values on channels, parallelism and the terminated process stop. We also allow matching and mismatching, with the construct $\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$, a form of recursion, $*P$, and three forms of name creation:

- $(\mathsf{newc}\, a : \mathrm{A})\ P$, the creation of a new *local channel* 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.
- $(\mathsf{newloc}\, k : \mathrm{K})\ \mathsf{with}\ P\ \mathsf{in}\ Q$, the creation of a new *location* k of type K, with the code P running there, in parallel with the code Q running at the original location. 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 $|$ and names may be shared between threads using the construct $(\mathsf{new}\, n : \mathrm{E})$; the form of type E will depend on whether n is a location, a local channel, or a registered name.

Processes, systems and indeed types may contain occurrences of variables, and these may be bound in the construct $u?(X : \mathrm{T})\ P$; if x appears in the pattern X then all occurrences of x in T and P are bound. This leads to the notions of free and bound variables, capture-avoiding substitution of identifiers for variables, $P\{\!|{}^{v}\!/\!_{x}|\!\}$, and α-equivalence. These are all standard apart from substitutions into (location) types, which is not quite syntactic; the details of substitution into types may be found in Definition 2 . We say that a system or process term is *closed* if it contains no free occurrences of variables.

The language also contains binding constructs for names, $(\mathsf{newc}\, n : \mathrm{A})\ P$, $(\mathsf{newreg}\, n : \mathrm{G})\ P$ and $(\mathsf{newloc}\, k : \mathrm{K})\ \mathsf{with}\ C\ \mathsf{in}\ P$ in processes. So we also have the notions of free and bound names in terms, and as usual the definition of α-equivalence identifies terms which only differ by their use of bound names.

Reduction Semantics: This is given in terms of a binary relation between *closed* systems:

$$M \to N$$

and is a mild generalisation of that given in [11] for DPI. It is a *contextual relation* between systems; that is, it is preserved by the static operators $|$ and $(\mathsf{new}\, e : \mathrm{E})$. It is defined to be the least such relation which satisfies the axioms and rules in Figure 2. The rule (R-STR) merely says that the we are working up to a structural equivalence, $\equiv$, which abstracts from inessential details in the terms representing systems. Formally structural equivalence is

(R-COMM)

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

(R-C—CREATE)

$$\frac{}{k[\![(\mathsf{newc}\, n:\mathrm{A})\ \ P]\!] \to (\mathsf{new}\, n:\mathrm{A}@k)\ \ k[\![P]\!]}$$

(R-MOVE)

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

(R-R—CREATE)

$$\frac{}{k[\![(\mathsf{newreg}\, n:\mathrm{G})\ \ P]\!] \to (\mathsf{new}\, n:\mathrm{G})\ \ k[\![P]\!]}$$

(R-UNWIND)

$$\frac{}{k[\![*\, P]\!] \to k[\![P \mid *\, P]\!]}$$

(R-L—CREATE)

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

(R-SPLIT)

$$\frac{}{k[\![P \mid Q]\!] \to k[\![P]\!] \mid k[\![Q]\!]}$$

(R-EQ)

$$\frac{}{k[\![\mathsf{if}\ u=u\ \mathsf{then}\ P\ \mathsf{else}\ Q]\!] \to k[\![P]\!]}$$

(R-NEQ)

$$\frac{}{k[\![\mathsf{if}\ u=v\ \mathsf{then}\ P\ \mathsf{else}\ Q]\!] \to k[\![Q]\!]}\ u \neq v$$

(R-STR)

$$\frac{M \equiv N,\ M \to M',\ M' \equiv N'}{N \to N'}$$

Fig. 2. Reduction semantics for DPI

(S-EXTR) $(\mathsf{new}\, n:\mathrm{E})\ (M \mid N) = M \ \mid\ (\mathsf{new}\, n:\mathrm{E})\ \ N$ if $n \notin \mathsf{fn}(M)$

(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) $(\mathsf{new}\, n:\mathrm{E})\ \ (\mathsf{new}\, n':\mathrm{E}')\ \ N = (\mathsf{new}\, n':\mathrm{E}')\ \ (\mathsf{new}\, n:\mathrm{E})\ \ N$ if $n \notin \mathrm{E}', n' \notin \mathrm{E}$

Fig. 3. Structural equivalence for DPI

defined to be the least contextual relation between (*closed*) systems which satisfies the axioms in Figure 3. One of the main forms of reduction involves *local communication* and is governed by the axiom (R-COM):

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

This uses an obvious generalisation of substitution of values into patterns $P\{\!|{}^{V}\!/x|\!\}$; of course this may not be well-defined if the structure of the pattern X does not match that of the value V. The other main form of reduction is *migration*, governed by the rule (R-MOVE):

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

In addition to these we have the unwinding of recursive definitions (R-UNWIND) and the testing of identifiers for identity, (R-EQ) and (R-NEQ).

The remaining rules are housekeeping in nature. The rule (R-SPLIT) allows the structural reorganisation of threads so that the main reduction rules can be applied, while the three rules associated with name binding (R-C − CREATE), (R-R − L − CREATE) and (R-R − CREATE) allow names declared locally in threads to appear globally at the system level, with their types appropriately modified.

3 Typing

In this section we outline the use of types to control resources and the accompanying typing system. The starting point is similar to the typing system of [11], but there are three major differences:

- We use a new category of types, *registered name types*, to explicitly manage the resource names which can be shared between different locations.
- The type 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; here they do not explicitly contain associations between names and location types.

3.1 The Types

The collection of types is an extension of those used in [11], to which the reader is referred for more background and motivation. In particular we will inherit a subtyping relation $\mathrm{T} <: \mathrm{U}$ with the property of *partial meets*; that is if two types $\mathrm{T}_1, \mathrm{T}_2$ have a lower bound, which we denote by $\mathrm{T}_1 \downarrow \mathrm{T}_2$, then they have a greatest lower bound $\mathrm{T}_1 \sqcap \mathrm{T}_2$. Intuitively the existence of $\mathrm{T}_1 \sqcap \mathrm{T}_2$ means that T_1 and T_2 are *consistent*, in that they allow compatible capabilities on values at these types.

The basic set of types may be classified as follows:

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 $\mathrm{U} <: \mathrm{T}$, U, T closed
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:	$\mathrm{C} ::= \mathrm{B} \mid \mathrm{A} \mid G \mid (\tilde{\mathrm{A}})@\mathrm{K}$
Transmission Types:	$\mathrm{T} ::= (\mathrm{C}_1, \ldots, \mathrm{C}_n), n \geq 0$

Fig. 4. Types

Base types: includes predefined types such as **int**, **bool**,.... Note that it also includes the type $\top$, at which names can only be used for comparison with other names.

Local Channel types: ranged over by A, take the form $\mathsf{rw}\langle \mathrm{T}, \mathrm{U} \rangle$, indicating that values may be read at type T and written at type U. These may be restricted to read-only capability $\mathsf{r}\langle \mathrm{T} \rangle$ or write-only capability $\mathsf{w}\langle \mathrm{U} \rangle$. We usually abbreviate the type $\mathsf{rw}\langle \mathrm{T}, \mathrm{T} \rangle$ to $\mathsf{rw}\langle \mathrm{T} \rangle$.

Location types: ranged over by L, K and may take the form $\mathsf{loc}[u_1 : \mathrm{A}_1, \ldots, u_n : \mathrm{A}_n]$. A process which obtains a location name at this type may use the resources u_i there, with the capabilities ordained by the local channel type A_i. As in [11] we require u_i to be distinct, although this side condition is omitted from Figure 4. We identify these types up to re-ordering of the resources u_i. We also abbreviate $\mathsf{loc}[]$ to loc.

Registered name types: ranged over by G, and may take the form $\mathsf{rc}\langle \mathrm{A} \rangle$, where A is a local channel type. One may think of these as types of names which have been registered as available for use in the declaration of new locations. The intention is that distinct locations can maintain a uniform naming scheme for common services. For example print could be declared as a registered name, and then used as part of the declared resources at a range of different sites or locations. Then code using the name print can be executed at any of these sites.

The formation rules for all types are given in Figure 4. The *transmission types*, ranged over by T, are the types at which values may be sent or received over channels. They consist of tuples the components of which may be base values, local channels, registered names, or *structured values* of the composite type $(\tilde{\mathrm{A}})@\mathrm{K}$. The usefulness of the composite types $(\tilde{\mathrm{A}})@\mathrm{K}$ has been explained at length in [11] to which the reader is referred to for more details; briefly these

(SUB-CTOP) $$\frac{}{\mathrm{T} <: \top}$$

(SUB-TUPLE) $$\frac{}{\textbf{base} <: \textbf{base}}$$

(SUB-CHAN) $$\frac{\mathrm{T}_1 <: T_2, \mathrm{U}_1 <: U_2 \quad \mathrm{T}_2 <: \mathrm{U}_1}{\mathsf{w}\langle T_2\rangle <: \mathsf{w}\langle T_1\rangle \quad \mathsf{r}\langle U_1\rangle <: \mathsf{r}\langle U_2\rangle \quad \mathsf{rw}\langle U_1, T_2\rangle <: \mathsf{rw}\langle U_2, T_1\rangle}$$

$$\frac{\mathrm{T} <: U}{\mathsf{rw}\langle U, T\rangle <: \mathsf{w}\langle T\rangle \quad \mathsf{rw}\langle U, T\rangle <: \mathsf{r}\langle U\rangle}\ \mathrm{T}, \mathrm{U}\ \text{closed}$$

(SUB-LOC) $$\frac{\mathrm{L}(u_i) <: \mathrm{A}_i,\ \ 1 \leq i \leq k}{\mathrm{L} <: \mathsf{loc}[u_1 : \mathrm{A}_1, \ldots, u_k : \mathrm{A}_k]}$$

(SUB-CAP) $$\frac{\mathrm{A} <: \mathrm{A}'}{u : \mathrm{A} <: u : \mathrm{A}'}$$

(SUB-HOM) $$\frac{A_1 <: A_2,\ K_1 <: K_2}{A_1 @ K_1 <: A_2 @ K_2 \quad \mathsf{rc}\langle \mathrm{A}_1\rangle <: \mathsf{rc}\langle \mathrm{A}_2\rangle}$$

(SUB-TUPLE) $$\frac{C_i <: \mathrm{C}'_i}{(\tilde{\mathrm{C}}) <: (\widetilde{\mathrm{C}'})}$$

Fig. 5. Subtyping

may be viewed as dependent types, with values of the form $(\tilde{c})@u$; here c_i is the name of a channel of type A_i located at k, a location name of type K. Note that a location type K can also be viewed as a transmission type, by identifying it with $()@\mathrm{K}$. It is also worth pointing out that variables are only allowed to appear in location types, as in a capability $u : \mathrm{A}$, the identifier u may be either a name or a variable.

Formally the types must be defined simultaneously with the subtype relation $<:$ because of the side-condition in the rules for local channels. The rules defining subtype relation are given in Figure 5, and again these are a minor modification from the subtyping rules in [11], where motivation may be found, which in turn are a generalisation of the subtyping rules originally introduced for the π-calculus in [18]. In the rule (SUB-LOC) we use the obvious notation $\mathrm{L}(u)$ to denote the channel type associated in the location type L with the identifier u. It is easy to check that the defined relation $<:$ is a pre-order (even a partial order) but it also has another important property.

Definition 1 (Partial meets) A preorder $\langle A, <\rangle$ is said to have *partial meets* if every pair of elements a_1, a_2 in A which has a lower bound also has a greatest lower bound. Formally if there is an element b such that $b < a_1,\ b < a_2$ then there exists an element $a_1 \sqcap a_2$ satisfying

- $a_1 \sqcap a_2 < a_1$ and $a_1 \sqcap a_2 < a_2$
- for every b such that $b < a_1$ and $b < a_2$, we have $b < a_1 \sqcap a_2$. □

Proposition 1 *The set of types* **Types***, ordered by* $<:$ *has partial meets.*

Proof: As in [11]. □

Intuitively $T_1 \sqcap T_2$ exists if the capabilities described by the individual types T_i are consistent, and it is obtained by "unioning" their capabilities. This operation will be used extensively in our type inference system. It is also used in the definition of syntactic substitution of identifiers for variables into types, referred to in the previous section.

Definition 2 (Substitution into types) The partial operation of substitution into types,$T\{\!|{}^{v}\!/\!{}_{x}|\!\}$, is defined by induction. For location types we have

$$\mathsf{loc}[u_1 : \mathrm{A}_1, \ldots, u_n : \mathrm{A}_n]\{\!|{}^{v}\!/\!{}_{x}|\!\} = \\ \mathsf{loc}[u_1\{\!|{}^{v}\!/\!{}_{x}|\!\} : \mathrm{A}_1] \;\sqcap \ldots \sqcap\; \mathsf{loc}[u_n\{\!|{}^{v}\!/\!{}_{x}|\!\} : \mathrm{A}_n]$$

For channel types the definition is trivial, as they are required to be closed. For the remaining types the definition is extended homomorphically. So for example $(\tilde{\mathrm{T}})\{\!|{}^{v}\!/\!{}_{x}|\!\} = (\mathrm{T}_1\{\!|{}^{v}\!/\!{}_{x}|\!\}, \ldots \mathrm{T}_n\{\!|{}^{v}\!/\!{}_{x}|\!\})$ □

We end this subsection with some examples; we will often omit individual type annotations, particularly when they play no role in the discussion, and will use standard abbreviations, such as omitting trailing occurrences of stop.

Example 1 (Remote channel types) Consider the location type

$$\mathrm{L}_s = \mathsf{loc}[\mathsf{quest} : \mathrm{T}_q,\ \mathsf{ping} : \mathrm{T}_p,\ \mathsf{kill} : \mathrm{T}_k]$$

at which a typical service site s might be declared. Such a service would respond to calls on the three ports, quest, ping, and kill. The first might be a method which provides a specific function, such as testing integers for primality, the second might allow the state of the service to be tested, while the third would give a client the ability to disable the site. The agent responsible for creating s has the possibility of publicising its existence either at the declaration type L_s or at one of its subtypes, such as:

$$\mathsf{loc}[\mathsf{quest} : \mathrm{T}_q,\ \mathsf{ping} : \mathrm{T}_p] \\ \mathsf{loc}[\mathsf{quest} : \mathrm{T}_q] \\ \mathsf{loc}$$

This allows the agent to provide selective access to the services available at the server.

A typical server would take the form

$$\begin{aligned} s[\![\ \ \textit{internals} \mid {*}\mathsf{quest}?(X : \mathrm{U}_q) \ldots \\ {*}\, \mathsf{ping}?(X : \mathrm{U}_p) \ldots \\ {*}\, \mathsf{kill}?(X : \mathrm{U}_k) \ldots \ \ ]\!] \end{aligned}$$

where *internals* represents some internal code necessary to set-up and control the services. Let us look at one example of servicing requests. Suppose the service checks whether or not a supplied integer is a prime number. So at the channel quest the service receives an integer, and a return address; it checks if the integer is a prime and returns the answer at the proffered address:

$$\begin{aligned} & s[\![\ldots \mid {*}\mathsf{quest}?(x, y@z)\ \mathsf{goto}\, z.y!\langle \textit{isprime}(x) \rangle \\ & \quad {*}\, \mathsf{ping}?(X : \mathrm{U}_p) \ldots \\ & \quad {*}\, \mathsf{kill}?(X : \mathrm{U}_k) \ldots \ \ ]\!] \end{aligned}$$

Here the integer is bound to x, while the address consists of two parts, a channel, bound to y, at some *unknown* site, bound to z. We use $\textit{isprime}(x)$ as shorthand for some computation at the site s which returns the value true or false, depending on whether the input x is prime.

A typical client, residing at c, takes the form:

$$c[\![(\mathsf{newc}\, r : \mathsf{rw}\langle \mathbf{bool} \rangle)\ \ \mathsf{goto}\, s.\mathsf{quest}!\langle v, r@c \rangle \ \mid\ r?(z) \ldots]\!]$$

Here a new return channel r is generated and a process is sent to the service s with the integer to be tested v, and the return address $r@c$. Meanwhile back at the client the result is awaited on the local channel r.

The type of the service at the port quest, denoted T_q above, takes the form $\mathsf{r}\langle \mathrm{U}_q \rangle$, where U_q is a tuple type. The first component is **int** while the second is a type for a remote channel at some *unknown* location; the fact that the location (of the client) is unknown, or arbitrary, allows the service to be used by any client. Since the capability to write a boolean is required of the remote channel the type U_q is given by

$$\langle\ \mathbf{int},\ \mathsf{w}\langle \mathbf{bool} \rangle @\mathsf{loc}\ \rangle$$

□

Example 2 (Personalised service) Here we consider a variation of the servers in Example 1 which can be personalised so as to respond only to a specific site. Consider the following system, which receives requests for new services:

$$\begin{aligned} \mathsf{center}[\![\ \ \mathsf{setup}?((x_1, x_2)x@z)\ \ (\mathsf{newloc}\, s : \mathrm{L}_s)\ \ \mathsf{with}\ P(x_2, z)\ \mathsf{in} \\ \mathsf{goto}\, z.x_1!\langle s \rangle]\!] \end{aligned}$$

Here a request is received at setup for a new service, which is established at a new site s, whose name is returned to the address bound to $x_1@z$. The second address, $x_1@z$ is used as part of the code $P(x_2, z)$ which is installed in the newly generated site:

$$*\mathsf{quest}?(x)\ \ \mathsf{goto}\, z.x_2!\langle \mathit{isprime}(x)\rangle\ \ldots$$

So the type of the server site would take the form

$$\mathrm{L}_s = \mathsf{loc}[\mathsf{quest} : \mathsf{rw}\langle \mathbf{int}\rangle,\ \mathsf{ping} :\ \ldots]$$

Then an example client such as

$$\mathsf{me}[\![(\mathsf{newc}\, r_1 : \mathsf{rw}\langle \mathsf{loc}\rangle)\ \ (\mathsf{newc}\, r_2 : \mathsf{rw}\langle \mathbf{bool}\rangle)\ \ \mathsf{goto}\, \mathsf{center.setup}!\langle (r_1, r_2)@\mathsf{me}\rangle \ldots\ ]\!]$$

receives personalised treatment; the new site will always reply to a channel at the site me. □

Example 3 (Shared interfaces) Here we demonstrate the usefulness of the new type category of *registered names* in setting up shared interfaces among different sites. Consider a system of the form

$$(\mathsf{newreg}\ \ \mathsf{put} : \mathsf{rc}\langle \mathrm{T}_p\rangle,\ \mathsf{get} : \mathsf{rc}\langle \mathrm{T}_g\rangle)\ (\mathsf{Bserver} \mid \mathsf{Client}_1 \mid \mathsf{Client}_2 \mid \ldots\quad)$$

consisting of a bank account server $\mathsf{Bserver}$ and a number of clients. The system is within the scope of two registered names, put and get, registered at specific types T_p and T_g on which we will not elaborate. This pair of typed names may serve, informally, as the interface for bank accounts created by the server for the various clients. An example server would take the form:

$$\begin{aligned}\mathsf{Bserver} \Leftarrow s[\![*\mathsf{request}?(x : \mathbf{int}, y@z)\\ &(\mathsf{newloc}\, b : \mathrm{L}_b)\ \ \mathsf{with} \ldots \mathsf{put},\ \mathsf{get} \ldots\ \mathsf{in}\\ &\quad \mathsf{goto}\, z.y!\langle b\rangle\quad]\!]\end{aligned}$$

Here a request is received, consisting of an initial amount x and a return address $y@z$. A new bank account is established at some new site b, whose name is forwarded to the return address. For simplicity we ignore the actual code for running the bank account but it uses put and get as access ports. The declaration type of the new account uses the registered names:

$$\mathrm{L}_b = \mathsf{loc}[\mathsf{put} : \mathrm{T}_p,\ \mathsf{get} : \mathrm{T}_g]$$

A typical client will look like:

$$\mathsf{Client} \Leftarrow \mathsf{me}[\![(\mathsf{newc}\, r : \mathsf{rw}\langle \mathrm{L}_b\rangle)\ \ \mathsf{goto}\, s.\mathsf{request}!\langle 100, r@\mathsf{me}\rangle\ \mid r?(x) \ldots]\!]$$

It generates an appropriate reply channel, sends it to the server and then awaits the name of the new account.

All new accounts received by clients will now have the same interface, consisting of the two methods put, get at the types T_p and T_g. More importantly the code developed by each client is independent of the actual account at which it will be run, so long as it respects the published interface; that is so long as it uses put, get in accordance with the types T_p and T_g. □

Example 4 (Dynamic interfaces) In the previous example the server generates the new bank accounts and informs the client. An alternative scheme would be for the clients to be responsible for setting up the accounts and the server would merely administer the shared interface:

$$\begin{aligned}\mathsf{Server} \Leftarrow\ & (\mathsf{newreg}\ \ \mathsf{put} : \mathsf{rc}\langle \mathrm{T}_p\rangle,\ \mathsf{get} : \mathsf{rc}\langle \mathrm{T}_g\rangle)\\ & s[\![{*}\mathsf{request}?(y@z)\\ & \qquad\qquad \mathsf{goto}\, z.y!\langle \mathsf{put}, \mathsf{get}\rangle]\!]\end{aligned}$$

Here, on receipt of a request the server simply forwards the two registered names put and get. A typical client would look like:

$$\begin{aligned}\mathsf{Client} \Leftarrow \mathsf{me}[\![& (\mathsf{newc}\, r : \mathrm{T}_r)\ \ \mathsf{goto}\, s.\mathsf{request}!\langle r@\mathsf{me}\rangle\ \mid\\ & \qquad r?(y,z)\, (\mathsf{newloc}\, b : \mathrm{L}^{y,z})\ \ \mathsf{with}\ \ldots\, \mathit{code} \ldots\ \mathsf{in}\ \ \ldots]\!]\end{aligned}$$

Here the client, in response to a request, receives two registered names which are bound to y and z and then a new bank account is set up with a declaration type

$$\mathrm{L}^{y,z} = \mathsf{loc}[y : \mathrm{T}_g, z : \mathrm{T}_p]$$

Note that this again is a dynamic type, which will be instantiated at run-time. Also the type of the reply channel used by clients, T_r is for *registered names*, rather than *channels*. Here it may be $\langle \mathsf{put} : \mathsf{rc}\langle \mathrm{T}_p\rangle, \mathsf{get} : \mathsf{rc}\langle \mathrm{T}_g\rangle\rangle$.

The net effect is that all bank accounts established by clients who use the server will share the same interface. □

3.2 Type environments

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.

These can take the form

- $u : \mathsf{loc}$, meaning that u is a location.

(E-EMPTY)
$$\frac{}{\vdash \mathbf{env}}$$

(E-BASE)
$$\frac{\Gamma \vdash \mathbf{env}}{\Gamma,\ u : \top \vdash \mathbf{env}}\ u \notin \Gamma$$

(E-NEW−LCHAN)
$$\frac{\Gamma \vdash \mathbf{env} \quad \Gamma \vdash w : \mathsf{loc}}{\Gamma,\ u : \mathrm{A}_{@}w \vdash \mathbf{env}}\ \begin{array}{l} u \notin \Gamma \\ u \notin \mathrm{A}\end{array}$$

(E-REF−LCHAN)
$$\frac{\Gamma \vdash \mathbf{env} \quad \Gamma \vdash w : \mathsf{loc} \quad \Gamma \vdash u : \mathsf{rc}\langle \mathrm{B}\rangle,\ \mathrm{B} <: \mathrm{A}}{\Gamma,\ u : \mathrm{A}_{@}w \vdash \mathbf{env}}\ \begin{array}{l} u_{@}w \notin \mathsf{dom}(\Gamma) \\ u \notin \mathrm{A}\end{array}$$

(E-RCHAN)
$$\frac{\Gamma \vdash \mathbf{env}}{\Gamma,\ u : \mathsf{rc}\langle \mathrm{A}\rangle \vdash \mathbf{env}}\ u \notin \Gamma$$

(E-LOC)
$$\frac{\Gamma \vdash \mathbf{env}}{\Gamma,\ u : \mathsf{loc} \vdash \mathbf{env}}\ u \notin \Gamma$$

Fig. 6. Well-formed Environments

(T-CHAN)
$$\frac{\Gamma, u : \mathrm{A}_{@}w, \Gamma' \vdash \mathbf{env}}{\Gamma, u : \mathrm{A}_{@}w, \Gamma' \vdash_w u : \mathrm{A}'}\ \mathrm{A} <: \mathrm{A}'$$

(T-RCHAN)
$$\frac{\Gamma, u : \mathsf{rc}\langle \mathrm{A}\rangle, \Gamma' \vdash \mathbf{env}}{\Gamma, u : \mathsf{rc}\langle \mathrm{A}\rangle, \Gamma' \vdash_w u : \mathsf{rc}\langle \mathrm{A}'\rangle}\ \mathrm{A} <: \mathrm{A}'$$

(T-LOC)
$$\frac{\Gamma, u : \mathsf{loc}, \Gamma' \vdash \mathbf{env}}{\Gamma, u : \mathsf{loc}, \Gamma' \vdash_w u : \mathsf{loc}}$$

(T-BASE)
$$\frac{\Gamma \vdash \mathbf{env}}{\Gamma \vdash_w v : \mathbf{base}}\ v \in \mathbf{base}$$

(T-LOC)
$$\frac{\Gamma \vdash_w v : \mathsf{loc} \quad \Gamma \vdash_w u_i : \mathrm{A}_{i@}v \quad \Gamma \vdash_w u_i : \mathsf{rc}\langle \mathrm{D}_i\rangle,\ \mathrm{D}_i <: \mathrm{A}_i}{\Gamma \vdash_w v : \mathsf{loc}[u_1 : \mathrm{A}_1, \ldots, u_n : \mathrm{A}_n]}$$

(T-LOCATED−CHANNEL)
$$\frac{\Gamma \vdash_w u_i : \mathrm{A}_{i@}v \quad \Gamma \vdash_w v : \mathrm{K}}{\Gamma \vdash_w (\widetilde{u})_{@}v : (\widetilde{A})_{@}\mathrm{K}}$$

(T-TUPLE)
$$\frac{\Gamma \vdash_w u_i : \mathrm{T}_i}{\Gamma \vdash_w (\widetilde{u}) : (\widetilde{\mathrm{T}})}$$

Fig. 7. Type rules for values

- $u : \mathsf{rc}\langle \mathrm{A}\rangle$, meaning u represents a registered name of type A
- $u : \mathrm{A}_{@}w$, meaning the channel u located at w has type A

In general, an arbitrary list of such assumptions may not be consistent. For example we should not be able to introduce an assumption $u : \mathsf{loc}$ if u is already designated as a channel, or introduce $u : \mathrm{A}_{@}w$ unless w is known to be a location. In order to describe the set of valid, or well-formed, environments we introduce judgements of the form

$$\Gamma \vdash \mathbf{env}$$

An environment may contain several entries for a name u but the judgements ensure that each instance is either as a registered name or located at a unique location. The inference rules are given in Figure 6 and are straightforward. A valid environment Γ can always be extended by an entry $u : \mathbf{base}$, $u : \mathsf{loc}$ or $u : \mathsf{rc}\langle \mathrm{A}\rangle$ provided the identifier u is new to Γ. Also, using (E-NEW − LCHAN), it can be extended by a located channel association $u : \mathrm{A}@w$ provided u is new and w is known to be a location; this corresponds to adding dynamically a completely new channel name at the location w. On the other hand the rule (E-REF − LCHAN) allows locations to share channel names. Here the side-condition (see the definition of the domain of an environment below) ensures that u can not already exist at the location w, but it may exist elsewhere; that is Γ may contain an association $u : \mathrm{A}'@w'$ for some w' different than w. But to introduce such a name, to be shared among various locations, it must already be declared as a registered name, and it can only be introduced at w with a subtype of its declared type. This is the import of the premise $u : \mathsf{rc}\langle \mathrm{B}\rangle$ and the condition $\mathrm{B} <: \mathrm{A}$. So in general local channel names may exist at different locations but all their local types are consistent, in that they have the declared type B as a lower bound.

Definition 3 (Environment domains) For any environment Γ we define its domain $\mathsf{dom}(\Gamma)$ to be $\{\, u \mid \Gamma \vdash u : \mathrm{T}$ for some global type $\mathrm{T}\,\} \cup \{\, u@w \mid \Gamma \vdash u : \mathrm{A}@w$ for some located type $\mathrm{A}\,\}$ □

The association between identifiers and types may be generalised in a natural manner to values. This is achieved by judgements of the form $\Gamma \vdash_w V : \mathrm{T}$ and the rules are given in Figure 7. The basic axioms are (T-CHAN), (T-RCHAN) and (T-LOC); the other rules merely extend the resulting associations structurally to other values and other types. Note that for many judgements, such as $\Gamma \vdash_w v : \mathsf{rc}\langle \mathrm{A}\rangle$, the inference is independent of the location w; in such cases the subscript will normally be omitted. Note also that to infer a location type, $\Gamma \vdash v : \mathsf{loc}[u_1 : \mathrm{A}, \ldots u_n : \mathrm{A}_n]$, it is necessary for each u_i to be a registered name in Γ.

Proposition 2 *Suppose* Γ *is a valid environment, that is* $\Gamma \vdash \mathbf{env}$. *Then*

(i) $\Gamma \vdash_w V : \mathrm{T}_1$ *and* $\Gamma \vdash_w V : \mathrm{T}_2$ *implies* $\Gamma \vdash_w V : \mathrm{T}_1 \sqcap \mathrm{T}_2$
(ii) $\Gamma \vdash_w u : \mathrm{A}$ *and* $\Gamma \vdash_w u : \mathsf{rc}\langle \mathrm{B}\rangle$ *implies* $\Gamma \vdash_w u : \mathsf{rc}\langle \mathrm{A} \sqcap \mathrm{B}\rangle$
(iii) $\Gamma \vdash_w u : \mathsf{r}\langle \mathrm{T}\rangle$ *and* $\Gamma \vdash_w u : \mathsf{w}\langle \mathrm{U}\rangle$ *implies* $\mathrm{U} <: \mathrm{T}$
(iv) $\Gamma \vdash_w u : \mathrm{U}$ *and* $\mathrm{U} <: \mathrm{T}$ *implies* $\Gamma \vdash_w u : \mathrm{T}$.

Proof: Straightforward inductions on the inferences of the judgements. □

Valid type environments may also be compared by their ability to associate types to identifiers:

Definition 4 (Environment extensions) For valid type environments let $\Gamma_1 \leq \Gamma_2$ if for all identifiers w, u, $\Gamma_2 \vdash_w u : \mathrm{T}_2$ implies $\Gamma_1 \vdash_w u : \mathrm{T}_1$ for some $\mathrm{T}_1 <: \mathrm{T}_2$ □

Proposition 3 *Let* **Envs** *be the set of all valid environments. Then the preorder* $\langle \mathbf{Envs}, \leq \rangle$ *has partial meets.*

Proof: First note that **Envs** ordered by $\leq$ is indeed a preorder but not a partial order. For example if Γ_1, Γ_2 denote the environments

$$k : \mathsf{loc},\ l : \mathsf{loc} \quad \text{and} \quad l : \mathsf{loc},\ k : \mathsf{loc}$$

respectively, then $\Gamma_1 \leq \Gamma_2$ and $\Gamma_2 \leq \Gamma_1$ but they are different environments.

Suppose there is a valid environment Δ such that $\Delta \leq \Gamma_i$ for $i = 1, 2$ we show how to construct a valid environment $\Gamma_1 \sqcap \Gamma_2$. The construction is by induction on the size of Γ_2. If it is empty then the result is obviously Γ_1 itself. Otherwise it is of the form $\Gamma_2', u : \mathrm{T}$ and we may assume $\Gamma_1 \sqcap \Gamma_2'$ exists. Then $\Gamma_1 \sqcap \Gamma_2$ is constructed by extending $\Gamma_1 \sqcap \Gamma_2'$; the precise extension depends on u and T. If $u \notin \mathsf{dom}(\Gamma_1 \sqcap \Gamma_2')$ then the construction gives $\Gamma_1 \sqcap \Gamma_2', u : \mathrm{T}$. So let us assume that $u \in \mathsf{dom}(\Gamma_1 \sqcap \Gamma_2')$.

- T is loc: The construction gives $\Gamma_1 \sqcap \Gamma_2'$ itself.
- T is **base**: Similar.
- T is $\mathsf{rc}\langle \mathrm{A} \rangle$: Here there are two cases:
  - · If $u : \mathsf{rc}\langle \mathrm{B} \rangle$ appears in $\Gamma_1 \sqcap \Gamma_2'$ then the result is obtained by replacing that entry with $u : \mathsf{rc}\langle \mathrm{B} \sqcap \mathrm{A} \rangle$.
  - · Otherwise we can assume that $u : \mathsf{rc}\langle \mathrm{B} \rangle$ does not appear in $\Gamma_1 \sqcap \Gamma_2'$ for any B but we do have an entry $u : \mathrm{B}@w$. Let Δ be obtained by removing this entry. Then the construction gives $\Delta, u : \mathsf{rc}\langle \mathrm{B} \sqcap \mathrm{A} \rangle, u : \mathrm{B}@w$.
- T has the form $\mathrm{A}@w$: Here again there are a number of cases:
  - · Suppose $u : \mathsf{rc}\langle \mathrm{B} \rangle$ and $u : \mathrm{A}'@w$ appear in $\Gamma_1 \sqcap \Gamma_2'$. Then the construction gives the result of replacing these with $u : \mathsf{rc}\langle \mathrm{B} \sqcap \mathrm{A} \rangle$, $u : (\mathrm{A} \sqcap \mathrm{A}')@w$ respectively.
  - · Suppose $u : \mathsf{rc}\langle \mathrm{B} \rangle$ appears in $\Gamma_1 \sqcap \Gamma_2'$ but $u : \mathrm{A}'@w$ does not, for any A'. Here the construction gives $\Delta, u : \mathrm{A}'@w$ where Δ is the result of replacing the entry $u : \mathsf{rc}\langle \mathrm{B} \rangle$ in $\Gamma_1 \sqcap \Gamma_2'$ with $u : \mathsf{rc}\langle \mathrm{B} \sqcap \mathrm{A} \rangle$.
  - · Suppose there is no entry of the form $u : \mathsf{rc}\langle \mathrm{B} \rangle$ but there is $u : \mathrm{A}'@w$. Then the construction replaces that entry with $u : \mathrm{A} \sqcap \mathrm{A}'$.
  - · Finally suppose there is no entry $u : \mathsf{rc}\langle \mathrm{B} \rangle$ but there is one of the form $u : \mathrm{A}'@w'$ for some w' different from w. Let Δ be the result of removing that entry. Then the construction gives $\Delta, u : \mathsf{rc}\langle \mathrm{A} \sqcap \mathrm{A}' \rangle, u : \mathrm{A}@w, u : \mathrm{A}@w'$.

We leave the reader to check that this construction is correct; that is

- $\Gamma_1 \sqcap \Gamma_2 \vdash \mathbf{env}$
- $\Gamma_1 \sqcap \Gamma_2 \leq \Gamma_i$ for $i = 1, 2$
- If $\Delta \leq \Gamma_i$ for $i = 1, 2$ then $\Delta \leq \Gamma_1 \sqcap \Gamma_2$. □

Our first use of this partial meet operation is to construct a type environment

$$\frac{\Gamma, n : \mathsf{rc}\langle \mathrm{A} \rangle \vdash M}{\Gamma \vdash (\mathsf{new}\, n : \mathsf{rc}\langle \mathrm{A}\rangle)\ M}\ \text{(T-RNEW)} \qquad \frac{\Gamma, n : \mathrm{A}@k \vdash M}{\Gamma \vdash (\mathsf{new}\, n : \mathrm{A}@k)\ M}\ \text{(T-CNEW)} \qquad \frac{\begin{array}{l}\Gamma \sqcap \langle k : \mathrm{K}\rangle \vdash M \\ \Gamma \sqcap \langle k : \mathrm{K}\rangle \vdash_w k : \mathrm{K}\end{array}}{\Gamma \vdash (\mathsf{new}\, k : \mathrm{K})\ M}\ \text{(T-LNEW)}$$

$$\frac{\begin{array}{l}\Gamma \vdash M \\ \Gamma \vdash N\end{array}}{\Gamma \vdash M \mid N}\ \text{(T-PAR)} \qquad \frac{\begin{array}{l}\Gamma \vdash_k P : \mathbf{proc} \\ \Gamma \vdash k : \mathsf{loc}\end{array}}{\Gamma \vdash k[\![P]\!]}\ \text{(T-THREAD)}$$

Fig. 8. Typing Systems

from a value V and a type T, relative to a location identifier w; this will be denoted by $\langle V : \mathrm{T}\rangle@w$.

Definition 5 (Constructing Environments) We define the list of type associations $\langle V : \mathrm{T}\rangle@w$ by induction on the structure of V; in certain cases it will also be a valid type environment.

- V is an identifier u and T is a local channel type A. Then $\langle V : \mathrm{T}\rangle@w$ is the list of size two, $w : \mathsf{loc}, u : \mathrm{T}@w$. If T is a located channel type $\mathrm{A}@k$ then $\langle V : \mathrm{T}\rangle@w$ is is $k : \mathsf{loc}, u : \mathrm{A}@k$. Note that in the latter case w plays no role in the construction of the list.
- V is an identifier u and T is a location type $\mathsf{loc}[v_1 : \mathrm{B}_1, \ldots, v_k : \mathrm{B}_k]$. Here $\langle V : \mathrm{T}\rangle@w$ is the list $u : \mathsf{loc}, v_1 : \mathrm{B}_1@u, \ldots, v_k : \mathrm{B}_k@u$. Note again that w plays no role.
- V is the structured name $(u_1, \ldots u_n)@v$.
  Here T must have the form $(\mathrm{A}_1, \ldots, \mathrm{A}_n)@K$ and again, the resulting list $\langle V : \mathrm{T}\rangle@w$ will be independent of w. It is constructed in the natural manner; first we construct the list associated with K, and then add on the associations for u_i. This gives $\langle v : \mathrm{K}\rangle@w,\ u_1 : \mathrm{A}_1@v, \ldots, u_n : \mathrm{A}_n@v$.
- V is the tuple $(\alpha_1, \ldots, \alpha_n)$. In this case we need T to be of the form $(\mathrm{C}_1, \ldots, \mathrm{C}_n)$, in which case the resulting list $\langle V : \mathrm{T}\rangle@w$ is constructed by induction: $\langle \alpha_1 : \mathrm{T}_1\rangle@w \sqcap \ldots \sqcap \langle \alpha_n : \mathrm{T}_n\rangle@w$. □

We have seen that the construction of $\langle V : \mathrm{T}\rangle@w$ is often independent of the location w, for example in the case when T is a location type. In such cases we will render this simply as $\langle k : \mathrm{K}\rangle$.

We are now ready to describe the type inference system for ensuring that systems are well-typed. There are two form of judgements, for systems and threads. The type inference rules for the first,

$$\Gamma \vdash M,$$

meaning that M is a well-typed system relative to Γ, are given in Figure 8. The intention is that whenever such a judgement can be inferred it will follow that Γ is a well-formed environment.

The main inference rule is (T-THREAD). In order to ensure that $k[\![P]\!]$ is a well-typed system we must show that the thread is well-typed to run at k. The typing of threads must be relative to a location because it may use *local* channels; these channels must exist at k. There is also a subtlety in the typing of name creation. First note that in these, and all subsequent rules, we assume that all bound names in a conclusion do not appear free in any assumptions. Thus in (T-LNEW) when constructing $\Gamma \sqcap \langle k : \mathrm{K}\rangle$ we know that k is actually new to Γ; so effectively the type associations in $\langle k : \mathrm{K}\rangle$ are simply appended to those in Γ. However we also have to check that K is a properly formed location type, that is that all the names it uses are registered resources; this is achieved by the judgement in the second premise.

Finally the typing rules for the judgements on threads

$$\Gamma \vdash_w P : \mathbf{proc}$$

are given in Figure 9, many of which should be familiar from typing systems for the π-calculus. For example (T-IN) says that to ensure the process $u?(X : \mathrm{T})\, P$ is well-typed relative to Γ to run at location w we must ensure that

- u is a channel with read capability of the appropriate type at w, that is $\Gamma \vdash_w u : \mathsf{r}\langle \mathrm{T}\rangle$
- the residual is well-typed in the environment Γ augmented by assuming the variables in the pattern X have the types assigned to them by the incoming type T, that is $\Gamma \sqcap \langle X : \mathrm{T}\rangle @w \vdash_w P : \mathbf{proc}$.

The rules (T-OUTPUT), (T-STOP), (T-PAR) and (T-REP) are informed in the same manner from similar rules for the π-calculus. The rule (T-GO) is a natural one for typing the process $\mathsf{goto}\, u.P$ and note that the requirements are actually independent of the current location w. The three rules governing the generation of new names at the three kinds of types A, K and G should be self-explanatory. Finally the rule (T-MATCH) is motivated at length in [11] where it is argued to be essential in capability based type systems. Briefly

(T-OUTPUT)
$$\frac{\begin{array}{l}\Gamma \vdash_w P : \mathbf{proc}\\ \Gamma \vdash_w V : \mathrm{T}\\ \Gamma \vdash_w u : \mathsf{w}\langle \mathrm{T}\rangle\end{array}}{\Gamma \vdash_w u!\langle V\rangle\, P : \mathbf{proc}}$$

(T-IN)
$$\frac{\begin{array}{l}\Gamma \sqcap \langle X : \mathrm{T}\rangle@w \vdash_w P : \mathbf{proc}\\ \Gamma \vdash_w u : \mathsf{r}\langle \mathrm{T}\rangle\end{array}}{\Gamma \vdash_w u?(X : \mathrm{T})\, P : \mathbf{proc}}$$

(T-GO)
$$\frac{\begin{array}{l}\Gamma \vdash_w u : \mathsf{loc}\\ \Gamma \vdash_u P : \mathbf{proc}\end{array}}{\Gamma \vdash_w \mathsf{goto}\, u.P : \mathbf{proc}}$$

(T-STOP)
$$\frac{\Gamma \vdash \mathbf{env}}{\Gamma \vdash_w \mathsf{stop} : \mathbf{proc}}$$

(T-L—NEW)
$$\frac{\begin{array}{l}\Gamma \sqcap \langle k : \mathrm{K}\rangle \vdash_w P : \mathbf{proc}\\ \Gamma \sqcap \langle k : \mathrm{K}\rangle \vdash_k Q : \mathbf{proc}\\ \Gamma \sqcap \langle k : \mathrm{K}\rangle \vdash_w k : \mathrm{K}\end{array}}{\Gamma \vdash_w (\mathsf{newloc}\, k : \mathrm{K})\ \mathsf{with}\ Q\ \mathsf{in}\ P : \mathbf{proc}}$$

(T-MATCH)
$$\frac{\begin{array}{l}\Gamma \vdash_w u : \mathrm{T}, v : \mathrm{U}\\ \Gamma \vdash_w Q : \mathbf{proc}\\ \Gamma \sqcap \langle u : \mathrm{U}\rangle@w \sqcap \langle v : \mathrm{T}\rangle@w \vdash_w P : \mathbf{proc},\\ \text{(whenever } (\Gamma\sqcap\langle u{:}\mathrm{U}\rangle\sqcap\langle v{:}\mathrm{T}\rangle) \text{ exists)}\end{array}}{\Gamma \vdash_w \mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q : \mathbf{proc}}$$

(T-R—NEW)
$$\frac{\Gamma, n : \mathrm{G} \vdash_w P : \mathbf{proc}}{\Gamma \vdash_w (\mathsf{newreg}\, n : \mathrm{G})\ P : \mathbf{proc}}$$

(T-C—NEW)
$$\frac{\Gamma, n : \mathrm{A}@w \vdash_w P : \mathbf{proc}}{\Gamma \vdash_w (\mathsf{newc}\, n : \mathrm{A})\ P : \mathbf{proc}}$$

(T-REP)
$$\frac{\Gamma \vdash_w P : \mathbf{proc}}{\Gamma \vdash_w *\, P : \mathbf{proc}}$$

(T-PAR)
$$\frac{\begin{array}{l}\Gamma \vdash_w P : \mathbf{proc}\\ \Gamma \vdash_w Q : \mathbf{proc}\end{array}}{\Gamma \vdash_w P \mid Q : \mathbf{proc}}$$

Fig. 9. Typing Threads

when establishing that $\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$ is well-typed with respect to Γ we need to ensure that both P and Q are well-typed. However in the case of P is executed we can take advantage of the fact that the identifiers u and v are in fact the same. Consequently any typing information associated with them can be amalgamated. So we need only establish that P is well-typed with respect to the augmented environment $\Gamma \sqcap \langle u : \mathrm{U}\rangle@w \sqcap \langle v : \mathrm{T}\rangle@w$; here the type of u is augmented by that of v, namely U, while that of v is augmented with T, the type of u. Note however that this is only necessary if the augmented environment actually exists; when it does not exist we know that u can never be the same as v and therefore P will never be executed.

3.4 Properties of the typing system

We are mainly interested in establishing Subject Reduction but this requires a series of preliminary results which we first outline. We often abbreviate abbreviate the judgement $\Gamma \vdash_w P : \mathbf{proc}$ to $\Gamma \vdash_w P$. First two standard properties one would expect:

Proposition 4 • *(Weakening) Suppose* Γ, Γ' *are two well-defined environments such that* $\Gamma' <: \Gamma$. *Then* $\Gamma \vdash M$ *implies* $\Gamma' \vdash M$.
• *(Strengthening) Suppose* $\Gamma, u : \mathrm{E} \vdash M$ *and* u *does not occur in the free identifiers of* M. *Then* $\Gamma \vdash M$.

Proof: Standard. Note however that corresponding results must be first established for the typing systems for values and processes. □

One standard property which does **not** hold is Interchange:

$$\Gamma_1, u_1 : \mathrm{T}_1, u_2 : \mathrm{T}_2, \vdash M \text{ implies } \Gamma_1, u_2 : \mathrm{T}_2, u_1 : \mathrm{T}_1, \vdash M$$

because one can not arbitrarily switch the entries in a well-typed environment. This property usually plays a central role in proofs of Subject Reduction and here we have to find a replacement. In a preorder $\langle A, < \rangle$ with partial meets (as opposed to a partial order) the meet $a \sqcap b$ of two elements is not uniquely determined; there may be more than one element which satisfies the definition. But all are related with respect to the equivalence relation $\equiv$ defined by

$$a \equiv b \ \text{ if } \ a < b \ \text{ and } \ b < a$$

Moreover in any preorder with partial meet we have the identities

$$a \sqcap b \equiv b \sqcap a \tag{6}$$
$$a \sqcap (b \sqcap c) \equiv (a \sqcap b) \sqcap c \tag{7}$$

Recall that **Envs** ordered by $<$ is a preorder is such a structure. Moreover from Weakening we know that whenever $\Gamma_1 \equiv \Gamma_2$

$$\Gamma_1 \vdash M \ \text{ if and only if } \ \Gamma_2 \vdash M$$

and similarly for processes and values. Thus we can rearrange valid environments using the identities (6), (7) above without changing their use in the inference of typing judgements. These judgements will be used in place of Interchange.

The main difficulty in establishing the Subject Reduction resides in showing the the reduction rule (R-COM) preserves well-typing. This amounts to showing that $\Gamma \vdash_k c!\langle V \rangle\, Q \mid c?(X)\, R$ implies $\Gamma \vdash_k Q \mid R\{\!|V/x|\!\}$ and proving

$$\Gamma \vdash_k R\{\!|V/x|\!\} \tag{8}$$

is the non-trivial part. After some analysis, the premise yields the statements

$$\Gamma \sqcap \langle X : \mathrm{T} \rangle @k \vdash_k R \ \text{ and } \ \Gamma \vdash V : \mathrm{T}@w \tag{9}$$

Our aim is to establish a *Substitution* result, which will be sufficient to infer (8) from (9).

However here the notation for the constructed environment $\langle X : \mathrm{T}\rangle_{@}k$ hides considerable complexity; the type T may be any of the allowed transmission types, for local or non-local channels, for locations, or for structured values. Accordingly we will isolate the particular cases, and treat some of them individually. Combining them will give Our general Substitution result, Theorem 1 on 27.

The most difficult individual case is when the value being substituted is a location; this is tackled in Proposition 6. But we start with considering the case when the value substituted is a local channel.

Proposition 5 (Local channel substitutions) *Suppose* $\Gamma \vdash_w v : \mathrm{A}$ *and* $\Gamma \vdash_w w_1 : \mathsf{loc}$. *Then, if* x *does not appear in* Γ

Values: $\Gamma, x : \mathrm{A}_{@}w \vdash_{w_1} U : \mathrm{T}$ *implies* $\Gamma \vdash_{w_1} U\{\!|{}^{v}\!/\!{}_{x}|\!\} : \mathrm{T}$
Processes: $\Gamma, x : \mathrm{A}_{@}w \vdash_{w_1} R$ *implies* $\Gamma \vdash_{w_1} : R\{\!|{}^{v}\!/\!{}_{x}|\!\}$

Proof: Throughout the proof we let α' denote $\alpha\{\!|{}^{v}\!/\!{}_{x}|\!\}$ for any appropriate syntactic object α.

The result for values is easily established by induction on the inference of the judgement $\Gamma, x : \mathrm{A}_{@}w \vdash_{w_1} U : \mathrm{T}$. The base cases, when one of the axioms (T-NAME),(T-RCHAN) or (T-LOC) is used, requires an argument which depends on whether U is the variable x or not. All other cases follow straightforwardly by induction.

Similarly the result for processes is proved by induction on the inference of $\Gamma, x : \mathrm{A}_{@}w \vdash_{w_1} R$ and an analysis of the last rule used. We examine two typical cases.

- Suppose $\Gamma, x : \mathrm{A}_{@}w \vdash_{w_1} u?(X : \mathrm{T})\, R$ because

(i) $\Gamma, x : \mathrm{A}_{@}w \vdash_{w_1} u : \mathsf{r}\langle \mathrm{T}\rangle$ and
(ii) $(\Gamma, x : \mathrm{A}_{@}w) \sqcap \langle X : \mathrm{T}\rangle_{@}w \vdash_{w_1} R$
Applying the first result to (i) we obtain
(i') $\Gamma \vdash_{w_1} u' : \mathsf{r}\langle \mathrm{T}\rangle$.
In (ii), since $\Gamma \vdash_{w_1} w : \mathsf{loc}$, the environment may be written as $(\Gamma \sqcap \langle x : \mathrm{A}\rangle_{@}w) \sqcap \langle X : \mathrm{T}\rangle_{@}w_1$ which is equivalent to $(\Gamma \sqcap \langle X : \mathrm{T}\rangle_{@}w_1) \sqcap \langle x : \mathrm{A}\rangle_{@}w$. Thus (ii) may be rewritten as
(ii') $(\Gamma \sqcap \langle X : \mathrm{T}\rangle_{@}w_1) \sqcap \langle x : \mathrm{A}\rangle_{@}w \vdash_{w_1} R$
Here we can apply induction to obtain
(ii") $(\Gamma \sqcap \langle X : \mathrm{T}\rangle_{@}w_1) \vdash_{w_1} R'$
Now the input rule (T-IN) can be applied to (i') and (ii") to obtain the required $\Gamma \vdash_{w_1} u'?(X : \mathrm{T})\, R'$. Note that our conventions about bound vari-

ables ensures that $u'?(X : \mathrm{T})\, R'$ is the same as $(u?(X : \mathrm{T})\, R)'$.

- Suppose $\Gamma, x : \mathrm{A}@w \vdash_{w_1} \mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P\ \mathsf{else}\ Q$. Then we know
  (i) $\Gamma, x : \mathrm{A}@w \vdash_{w_1} u_1 : \mathrm{T}, u_2 : \mathrm{U}$
  (ii) $\Gamma, x : \mathrm{A}@w \vdash_{w_1} Q$
  (iii) $(\Gamma, x : \mathrm{A}@w) \sqcap \langle u_1 : \mathrm{U}\rangle@w_1 \sqcap \langle u_2 : \mathrm{T}\rangle@w_1 \vdash_{w_1} P$, provided the environment $(\Gamma, x : \mathrm{A}@w) \sqcap \langle u_1 : \mathrm{U}\rangle@w_1 \sqcap \langle u_2 : \mathrm{T}\rangle@w_1$ exists.

  Applying the first result to (i) and induction to (ii) we obtain
  (i') $\Gamma \vdash_{w_1} u_1' : \mathrm{T}, u_2' : \mathrm{U}$
  (ii') $\Gamma \vdash_{w_1} Q'$

  Now suppose the environment $\Gamma \sqcap \langle u_1' : \mathrm{U}\rangle@w_1 \sqcap \langle u_2' : \mathrm{T}\rangle@w_1$ exists. In this case we need to show
$$\Gamma \sqcap \langle u_1' : \mathrm{U}\rangle@w_1 \sqcap \langle u_2' : \mathrm{T}\rangle@w_1 \vdash_{w_1} P' \tag{10}$$
  from which we can obtain the required conclusion
$$\Gamma \vdash_{w_1} \mathsf{if}\ u_1' = u_2'\ \mathsf{then}\ P'\ \mathsf{else}\ Q'.$$

  Establishing (10) requires an analysis of whether u_1, or u_2, or both coincide with x. As an example suppose $u_1 = x$ but $u_2 \neq x$, in which case w and w_1 must be the same location. So we know that $\Gamma \sqcap \langle v : \mathrm{U}\rangle@w_1 \sqcap \langle u_2 : \mathrm{T}\rangle@w_1 \vdash \mathbf{env}$, and since $\Gamma \vdash_w v : \mathrm{A}$ this means that
$$\Gamma, x : \mathrm{A}@w \sqcap \langle x : \mathrm{U}\rangle@w_1 \sqcap \langle u_2 : \mathrm{T}\rangle@w_1 \vdash \mathbf{env}$$
  So from (iii) we know
$$\Gamma, x : \mathrm{A}@w \sqcap \langle x : \mathrm{U}\rangle@w_1 \sqcap \langle u_2 : \mathrm{T}\rangle@w_1 \vdash_{w_1} P$$
  However this environment may be written as $\Gamma \sqcap \langle u_2 : \mathrm{T}\rangle@w_1,\ x : (\mathrm{A} \sqcap \mathrm{U})@w$ and therefore by Weakening we have
$$\Gamma \sqcap \langle v : \mathrm{U}\rangle@w_1 \sqcap \langle u_2 : \mathrm{T}\rangle@w_1,\ x : (\mathrm{A} \sqcap \mathrm{U})@w \vdash_{w_1} P$$
  Since $\Gamma \sqcap \langle v : \mathrm{U}\rangle@w_1 \sqcap \langle u_2 : \mathrm{T}\rangle@w_1 \vdash v : (\mathrm{A} \sqcap \mathrm{U})@w_1$ we may apply induction to obtain the required (10).

□

Unfortunately the substitution of locations requires a more complicated formulation. In this case our premise is that $\Gamma \vdash v : \mathrm{K}$, for some location type K, the inductive hypothesis is
$$\Gamma \sqcap \langle x : \mathrm{K}\rangle \vdash_w R \tag{11}$$
and we need to prove $\Gamma \vdash_w R\{\!|^v\!/\!_x|\!\}$. As an example suppose R has the form $\mathsf{goto}\, x.c?(y : \mathrm{A})\, P$. Then from the second premise we will be able to deduce

that

$$\Gamma \vdash v : \mathrm{K},\ y : \mathrm{A}@x \vdash_x P$$

However at this point we will be unable to perform induction because this is not an instance of the inductive hypothesis (11). Instead we will need to generalise (11) and unfortunately this will mean substituting v for x not only in process terms but also in environments.

Definition 6 (Substituting into environments) Suppose Γ is a valid environment. We define $\Gamma[\,^v\!/\!_x\,]$, the *substitution* of v for x in Γ, by induction on the size of Γ. If it is empty then so is $\Gamma[\,^v\!/\!_x\,]$. So we may assume Γ has the form $\Gamma', u : \mathrm{T}$ and that $\Gamma'[\,^v\!/\!_x\,]$ has been defined.

- If T has the form $\top$, $\mathsf{rc}\langle \mathrm{A}\rangle$ or $\mathsf{loc}[]$ then $\Gamma[\,^v\!/\!_x\,]$ is given by $\Gamma'[\,^v\!/\!_x\,] \sqcap u\{\!|^v\!/\!_x|\!\} : \mathrm{T}$.
- Otherwise it must be of the form $\mathrm{A}@w$ and $\Gamma[\,^v\!/\!_x\,]$ is defined to be $\Gamma'[\,^v\!/\!_x\,] \sqcap \langle u\{\!|^v\!/\!_x|\!\} : \mathrm{A}\rangle @ w\{\!|^v\!/\!_x|\!\}$.

$\square$

Lemma 1 *Suppose* $\Gamma \vdash \mathbf{env}$. *Then*

- $\Gamma \sqcap \langle x : \mathsf{loc}\rangle \sqcap \langle v : \mathsf{loc}\rangle \vdash \mathbf{env}$ *implies* $\Gamma[\,^v\!/\!_x\,] \vdash \mathbf{env}$.
- $\Gamma \sqcap \langle x : \mathsf{rc}\langle \mathrm{A}\rangle\rangle \sqcap \langle v : \mathsf{rc}\langle \mathrm{A}\rangle\rangle \vdash \mathbf{env}$ *implies* $\Gamma[\,^v\!/\!_x\,] \vdash \mathbf{env}$.

Proof: By induction on the size of Γ. $\square$

With this new notation we are now able to formulate an appropriate substitution result for locations.

Proposition 6 (Location substitutions) *Suppose* $\Gamma_1 \vdash v : \mathrm{K}$ *and* x *does not appear in* Γ_1. *Then*

Environments: $\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2 \vdash \mathbf{env}$ *implies* $\Gamma_1 \sqcap \Gamma_2[\,^v\!/\!_x\,] \vdash \mathbf{env}$
Values: $\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2 \vdash_w U : \mathrm{T}$ *implies* $\Gamma_1 \sqcap \Gamma_2[\,^v\!/\!_x\,] \vdash_{w\{^v\!/\!_x\}} U\{\!|^v\!/\!_x|\!\} : (\mathrm{T}\{\!|^v\!/\!_x|\!\})$
Processes: $\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2 \vdash_w R$ *implies* $\Gamma_1 \sqcap \Gamma_2[\,^v\!/\!_x\,] \vdash_{w\{^v\!/\!_x\}} R\{\!|^v\!/\!_x|\!\}$

Proof: Note that the previous Lemma ensures that $\Gamma_2[\,^v\!/\!_x\,]$ is a well-defined environment. The first result is proved by induction on Γ while the second is by induction on the inference of the judgement $\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2 \vdash_w U : \mathrm{T}$; we leave the details to the reader.

The result for processes is by induction on the inference of $\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2 \vdash_w R$ and an analysis of the last rule used. We give one representative example.

Suppose $\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2 \vdash_w (\mathsf{newloc}\, l : \mathrm{L})$ with Q in P because

(i) $(\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2) \sqcap \langle l : \mathrm{L}\rangle \vdash_l Q$
(ii) $(\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2) \sqcap \langle l : \mathrm{L}\rangle \vdash_w P$
(iii) $\Gamma_1 \sqcap \langle x : \mathrm{K}\rangle \sqcap \Gamma_2 \sqcap \langle l : \mathrm{L}\rangle \vdash_w l : \mathrm{L}$

Using the associativity of $\sqcap$ we can rearrange (i) to the form

$$\Gamma_1 \sqcap \langle x : \mathrm{K} \rangle \sqcap (\Gamma_2 \sqcap \langle l : \mathrm{L} \rangle) \vdash_l Q$$

to which induction can be applied to give

$$\Gamma_1 \sqcap (\Gamma_2 \sqcap \langle l : \mathrm{L} \rangle)[\,{}^{v}\!/\!_{x}\,] \vdash_l Q\{\!|{}^{v}\!/\!_{x}|\!\}$$

However once more the environment can be rearranged to give

(i') $\Gamma_1 \sqcap \Gamma_2[\,{}^{v}\!/\!_{x}\,] \sqcap \langle l : \mathrm{L} \rangle) \vdash_l Q\{\!|{}^{v}\!/\!_{x}|\!\}$

A similar argument can be used to obtain

(ii') $\Gamma_1 \sqcap \Gamma_2[\,{}^{v}\!/\!_{x}\,] \sqcap \langle l : \mathrm{L} \rangle) \vdash_{w\{{}^{v}\!/\!_{x}\}} P\{\!|{}^{v}\!/\!_{x}|\!\}$

while the result for values gives

(iii') $\Gamma_1 \sqcap \Gamma_2[\,{}^{v}\!/\!_{x}\,] \sqcap \langle l : \mathrm{L} \rangle) \vdash_{w\{{}^{v}\!/\!_{x}\}} l : \mathrm{L}$

Now (T-NEWL) can be applied to (i'),(ii') and (iii') to obtain the required $\Gamma_1 \sqcap \Gamma_2[\,{}^{v}\!/\!_{x}\,] \vdash_{w\{{}^{v}\!/\!_{x}\}} (\mathsf{newloc}\, l : \mathrm{L})$ with $(Q\{\!|{}^{v}\!/\!_{x}|\!\})$ in $(P\{\!|{}^{v}\!/\!_{x}|\!\})$ □

The substitution of registered names needs a formulation similar to that of locations. For example consider an attempt to prove

$$\Gamma,\ x : \mathsf{rc}\langle \mathrm{A} \rangle \vdash_w (\mathsf{newloc}\, k : \mathsf{loc}[x : \mathrm{B}])\ \mathsf{with}\ Q\ \mathsf{in}\ P \qquad (12)$$

This will be reduced to an attempt to prove

$$\Gamma,\ x : \mathsf{rc}\langle \mathrm{A} \rangle,\ k : \mathsf{loc},\ x : \mathrm{B}@k \vdash_w P$$

which is not of the form (12).

Proposition 7 (Registered name substitutions) *Suppose* $\Gamma_1 \vdash v : \mathsf{rc}\langle \mathrm{A} \rangle$ *and x does not appear in* Γ_1*. Then*

Environments: $\Gamma_1 \sqcap \langle x : \mathsf{rc}\langle \mathrm{A} \rangle \rangle \sqcap \Gamma_2 \vdash \mathbf{env}$ *implies* $\Gamma_1 \sqcap \Gamma_2[\,{}^{v}\!/\!_{x}\,] \vdash \mathbf{env}$
Values: $\Gamma_1 \sqcap \langle x : \mathsf{rc}\langle \mathrm{A} \rangle \rangle \sqcap \Gamma_2 \vdash_w U : \mathrm{T}@$ *implies* $\Gamma_1 \sqcap \Gamma_2[\,{}^{v}\!/\!_{x}\,] \vdash_w U\{\!|{}^{v}\!/\!_{x}|\!\} : (\mathrm{T})\{\!|{}^{v}\!/\!_{x}|\!\}$
Processes: $\Gamma_1 \sqcap \langle x : \mathsf{rc}\langle \mathrm{A} \rangle \rangle \sqcap \Gamma_2 \vdash_w R$ *implies* $\Gamma_1 \sqcap \Gamma_2[\,{}^{v}\!/\!_{x}\,] \vdash_w R\{\!|{}^{v}\!/\!_{x}|\!\}$

Proof: Left to the reader. □

We can now state the Substitution result in the required form:

Theorem 1 (Substitutions) *Suppose* $\Gamma \vdash_w V : \mathrm{T}$*,* $\Gamma \vdash w : \mathsf{loc}$ *and the variables in X do not appear in* Γ*. Then* $\Gamma \sqcap \langle X : \mathrm{T} \rangle @w \vdash_w R$ *implies* $\Gamma \vdash_{w\{{}^{V}\!/\!_{X}\}} R\{\!|{}^{V}\!/\!_{X}|\!\}$.

Proof: The proof is by induction on the structure of the type T. When it has the form A the result follows from Proposition 5 and the cases $A@k$ and **base** are similar. When T is a location type it follows from Proposition 6 and when it is of the form $\mathsf{rc}\langle B\rangle$ it follows from Proposition 7. The remaining cases can be proved by induction. □

We are now ready to outline the main result of this section.

Theorem 2 (Subject Reduction) *If* $\Gamma \vdash M$ *and* $M \rightarrow N$ *then* $\Gamma \vdash N$.

Proof: It is a question of examining in turn each of the rules in Figure 2. The rule (R-STR) requires the result

$M \equiv N$ and $\Gamma \vdash M$ implies $\Gamma \vdash N$,

the details of which may be found in [11]. We examine two typical cases from the remaining in Figure 2.

(R-COMM): Suppose $\Gamma \vdash k[\![c!\langle V\rangle\, Q]\!] \mid k[\![c?(X : \mathrm{T})\, P]\!]$. We have to show that $\Gamma \vdash k[\![Q]\!] \mid k[\![P\{\!|V/x|\!\}]\!]$, which will follow if we can prove
(i) $\Gamma \vdash_k Q$ and
(ii) $\Gamma \vdash_k P\{\!|V/x|\!\}$
The first is an easy consequence of the hypothesis while the second will follow from Theorem 1 if we can establish
(a) $\Gamma \vdash_k V : \mathrm{T}$ and
(b) $\Gamma \sqcap \langle X : \mathrm{T}\rangle@w \vdash_k P$
The hypothesis implies implies $\Gamma \vdash_k c?(X : \mathrm{T})\, P$ which means (b) is satisfied but also that $\Gamma \vdash_k c : \mathsf{r}\langle \mathrm{T}\rangle$. On the other hand the hypothesis also implies that $\Gamma \vdash_k c!\langle V\rangle\, Q$ which means that $\Gamma \vdash_k V : \mathrm{U}$ for some type U such that $\Gamma \vdash_k c : \mathsf{w}\langle \mathrm{U}\rangle$. However Proposition 2 (iii) implies that $\mathrm{U} <: \mathrm{T}$ and part (iv) of the same proposition gives (a) and we are finished.
(R-C − CREATE): Suppose $\Gamma \vdash k[\![(\mathsf{newc}\, n : \mathrm{A})\;\; P]\!]$. To establish the judgement $\Gamma \vdash (\mathsf{new}\, n : \mathrm{A}@k)\;\; k[\![P]\!]$ it is sufficient, by (T-C − NEW), to prove

$$\Gamma, n : \mathrm{A}@k \vdash_k P \tag{13}$$

But the only way to establish the hypothesis is by the rule (T-CNEW) in Figure 8, for which we need $\Gamma, n : \mathrm{A}@k \vdash k[\![P]\!]$, which can only be established from (T-THREAD), for which (13) is necessary. □

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

$$k[\![a?(x)\, P]\!] \qquad k[\![\mathsf{stop}]\!]$$

Thus, as explained in the Introduction, we develop equivalences of the form

$$\Gamma \models M \approx N \tag{14}$$

where Γ is a well-defined type environment representing the user's knowledge of the capabilities of the systems M and N. Since we are only interested in *closed* systems, that is containing no occurrences of free variables, we confine our attention to *closed* environments, those with no variables in their domains.

It may seem reasonable to assume that the user knows everything about the systems under scrutiny, but DPI is specifically designed to model scenarios in which clients are given selective knowledge of dynamically created resources.

Example 5 (Extruding names) Let K be the type $\mathsf{loc}[a : \mathrm{A}, b : \mathrm{B}]$ Consider the system M defined by

$$l[\![(\mathsf{newloc}\, k : \mathrm{K})\ \ \mathsf{with}\ a?(x)\, S\ \mathsf{in}\ \ c!\langle k\rangle]\!]$$

This generates a new location k, exports its name along c and installs a service S at k via the resource a.

The ability of a user to use the service depends on its capability on the channel c at k. Suppose this can only send values at the type K_b where K_b is $\mathsf{loc}[b : \mathrm{B}]$, a super-type of K. When the new name is exported along c there will now be a divergence between the knowledge of the user and that of the system M. The latter knows that k supports two resources a and b, while the user is under the illusion that it only support one, b. □

So we will have to consider triples in (14) above where Γ will not in general have sufficient information to type M and N. We will be able to maintain some constraints about the typability of M and N by insisting that Γ, the knowledge of the user, represents a subset of the knowledge of the system. This motivates the following definition:

Definition 7 (Simple configurations) A simple configuration is written as $\Gamma \triangleright M$, where

- M is a closed system
- there exists some Δ such that $\Delta <: \Gamma$, $\mathsf{dom}(\Delta) = \mathsf{dom}(\Gamma)$ and
- $\Delta \vdash M$. □

Rather than simply defining an ad-hoc bisimulation based equivalence over simple configurations we first introduce a *touchstone* equivalence by which considering natural desirable properties that one would expect of behavioural equivalences. We choose to base this on a generalisation of the reduction barbed congruence of [12].

A *knowledge-indexed relation over systems* is a family of binary relations between 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 and moreover, $\Gamma \triangleright M$ and $\Gamma \triangleright N$ form simple configurations. 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 $M \rightarrow M'$ there exists some N' such that $N \rightarrow^* N'$ and $\Gamma \models M' \mathcal{R} N'$.

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

(i) $\Gamma \models M \mathcal{R} N$ and $\Gamma, \Gamma' \vdash_{\mathbf{env}}$ 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) $\Gamma \sqcap \langle n : \mathrm{T} \rangle \models M \mathcal{R} N$ 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. The first clause also contains a subtlety; this implies that the equivalence should be preserved even if the user invents some new names. It would be unreasonable to rewrite this as

(i') $\Gamma \models M \mathcal{R} N$ and $\Gamma' <: \Gamma$, where $\Gamma' \vdash_{\mathbf{env}}$, implies $\Gamma' \models M \mathcal{R} N$.

This would allow the user to invent *new* capabilities on resources it has received from the systems under investigation.

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 $M \rightarrow^* M'$ for some M' structurally equivalent to a system of the form $(M'' \mid k[\![a!\langle\rangle\, P]\!])$. Here $\mathsf{rw}\langle\rangle$ indicates the type of a channel for transmitting the trivial value consisting of the empty tuple; more correctly it should be rendered as $\mathsf{rw}\langle()\rangle$. 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 8 (Reduction barbed congruence) We let $\approx_{rbc}$ be the largest knowledge-indexed relation over systems which is

- point-wise symmetric, that is $\Gamma \models M \approx_{rbc} N$ implies $\Gamma \models N \approx_{rbc} M$
- contextual
- reduction closed

- barb preserving. $\square$

We will now characterise $\approx_{rbc}$ using a labelled transition system and bisimulation equivalence, thereby justifying our particular notion of bisimulations. Note that knowledge-indexed relations generalise the more usual notion of type-indexed relations in which one would also demand that $\Gamma \vdash M$ and $\Gamma \vdash N$ whenever $\Gamma \models M \mathcal{R} N$. Our characterisations can be instantiated to account for these situations.

4.1 A labelled transition characterisation of contextual equivalence

The labelled transition system we present in this section is informed by recent work by two of the authors on characterising contextual equivalences for π-calculus with input/output subtyping [10]. This in turn was influenced by work by Boreale and Sangiorgi for a similar language in the absence of the name equality test [2].

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

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

where $\Gamma \triangleright M$ is a simple configuration and μ takes one of the forms:

output: of the value V along the channel c located at k, and exporting the new names $\tilde{n}$, $\quad (\tilde{n})k.c!V$
input: of the value V along the channel c located at k, using new names $\tilde{n}$ of type $(\tilde{\mathrm{E}})$, $\quad (\tilde{n} : \tilde{\mathrm{E}})c?V$
internal: unobservable activity, $\quad \tau$

The rules determining these relations are given in Figure 10, and they deserve some comment. First, for simplicity, internal action is equated with reduction, in the rule (LTS-RED). 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 $\Gamma \vdash_k a : \mathsf{r}\langle \mathrm{T}\rangle$ for some transmission type T
- the information which is being sent to the user does not contradict its current knowledge, that is $\Gamma \sqcap \langle V : \mathrm{T}\rangle @k$ exists

In fact in the rule the second requirement is slightly more stringent; we only use one particular transmission type for V, namely that which appears in Γ

(LTS-RED)

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

(LTS-OUT)

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

(LTS-IN)

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

(LTS-OPEN)

$$\frac{(\Gamma, n : \top \triangleright M) \xrightarrow{(\tilde{n})k.a!V} (\Gamma' \triangleright M')}{(\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\;\; M) \xrightarrow{(n\tilde{n})k.a!V} (\Gamma' \triangleright M')} \quad \begin{array}{l} n \neq a, k \\ n \in \mathsf{fn}(V)\end{array}$$

(LTS-WEAK)

$$\frac{(\Gamma, \langle n : \mathrm{E}\rangle \triangleright M) \xrightarrow{(\tilde{n}:\tilde{\mathrm{E}})k.a?V} (\Gamma' \triangleright M')}{(\Gamma \triangleright M) \xrightarrow{(n:\mathrm{E}\tilde{n}:\tilde{\mathrm{E}})k.a?V} (\Gamma' \triangleright M')} \quad n \neq a, k$$

(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)

$$\frac{(\Gamma, n : \top \triangleright M) \xrightarrow{\alpha} (\Gamma', n : \top \triangleright M')}{(\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\;\; M) \xrightarrow{\alpha} (\Gamma' \triangleright (\mathsf{new}\, n : \mathrm{E})\;\; M')} \quad n \notin \mathsf{n}(\alpha)$$

Fig. 10. Typed actions

for a at k; this simply cuts down on the number of possible moves.

The rule for input, (LTS-IN), has a similar flavour. The located process $k[\![a?(X : \mathrm{T})\, P]\!]$ can only read a value V at the channel a located at k if

(i) the user knows k is a location

(ii) the user can write on a located at k at the required type, that is $\Gamma \vdash_k \mathsf{w}\langle \mathrm{U} \rangle$, for some type U

(iii) the user can type the value V at this the required type, $\Gamma \vdash_k V : \mathrm{U}$.

As in the output case the second requirement is written slightly differently; the actual type U used will be that which appears for a in Γ. Note also that apriori there is no relationship required between the type at which the value is sent, U, and the type at which it will be used, T. But it turns out that in the context in which these rules will be applied (see Definition 7 below) the latter will be a super-type of the former.

The remaining rules are familiar from standard treatments of the pi-calculus, but to handle the environments we have introduced some extra notation. First $\langle n : \mathrm{E} \rangle$ represents the obvious environment, if E is a location or registered name type; otherwise E has the form $\mathrm{A}@k$, when it represents the simple environment $\langle n : \mathrm{A} \rangle @k$. This is used in the rule (LTS-WEAK), where we write $\Gamma, \langle n : \mathrm{E} \rangle$ rather than $\Gamma \sqcap \langle n : \mathrm{E} \rangle$ to emphasise that n does not appear in Γ.

We show that the transition rules are in fact well-defined, in the sense that they form a binary relation between simple configurations.

Proposition 8 *Suppose* $\Gamma \triangleright M$ *is a simple configuration. If* $\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright N$ *is a typed action then* $\Gamma' \triangleright N$ *is also a simple configuration. Moreover,*

- *if* μ *is* $(\tilde{n})k.c!V$ *then* $c : \mathsf{r}\langle \mathrm{T} \rangle @k \in \Gamma$ *for some* T *and* Γ' *is* $\Gamma \sqcap \langle V : \mathrm{T} \rangle @k$
- *if* μ *is* $(\tilde{n} : \tilde{\mathrm{E}})k.c?V$ *then* Γ' *is* $\Gamma, \langle \tilde{n} : \tilde{\mathrm{E}} \rangle$.

Proof: By induction on the inference of the typed action $\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright N$ and an analysis of the last rule used. As an example consider an application of the rule (LTS-IN). Here we have

$$(\Gamma \triangleright k[\![c?(X : \mathrm{T})\, P]\!]) \xrightarrow{k.c?V} (\Gamma \triangleright k[\![P\{\!|V/X|\!\}]\!])$$

because

(i) $\Gamma \vdash k : \mathsf{loc}$

(ii) $c : \mathsf{w}\langle \mathrm{U} \rangle @k \in \Gamma$ for some type U

(iii) $\Gamma \vdash_k V : \mathrm{U}$.

We know that $\Delta \vdash k[\![c?(X : \mathrm{T})\, P]\!]$ for some Δ such that $\Delta <: \Gamma$ and $\mathsf{dom}(\Delta) = \mathsf{dom}(\Gamma)$. We will have the required result if we can show that $\Delta \vdash k[\![P\{\!|V/X|\!\}]\!]$, that is

$$\Delta \vdash_k P\{\!|V/X|\!\} \tag{15}$$

From the hypothesis we know that $\Delta \vdash k[\![c?(X : \mathrm{T})\, P]\!]$, that is $\Delta \vdash_k c?(X : \mathrm{T})\, P$. This can only be derived by the rule (T-IN) which means we must have

(i') $\Delta \vdash_k c : \mathsf{r}\langle \mathrm{T} \rangle$
(ii') $\Delta \sqcap \langle X : \mathrm{T} \rangle @k \vdash_k P : \mathbf{proc}$

But (ii) above, and the fact that $\Delta <: \Gamma$, implies that $\Delta \vdash_k c : \mathsf{w}\langle \mathrm{U} \rangle$, and therefore by the third part of Proposition 2 we have $\mathrm{U} <: \mathrm{T}$; the fourth part of the same Proposition, together with (iii) above, then gives $\Gamma \vdash_k V : \mathrm{T}$. This, and the above (ii') are the required hypotheses in the Substitution Theorem, Theorem 1, to obtain the required (15). □

The net effect of this proposition is that in typed actions $\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright N$ the resulting environment Γ' is completely determined by Γ and μ.

It is very easy to view these typed actions as simple restrictions on a natural operational semantics for Dpi. Let us write

$$M \xrightarrow{\mu} N$$

if $\Delta \triangleright M \xrightarrow{\mu} \Delta' \triangleright N$ for some Δ, Δ' using a variation on the rules from Figure 10 in which the typing constraints on Δ are not enforced (note that side-conditions to maintain freshness of new names are still in place). Then we will have the typed action

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

if and only if

- M can in principle perform the action μ, that is $M \xrightarrow{\mu} N$
- and the environment Γ *allows the action.*

We make the latter statement more precise in the next proposition.

Proposition 9 *Suppose* $\Gamma \triangleright M$ *is a simple configuration.*

- $(\Gamma \triangleright M) \xrightarrow{\tau} (\Gamma' \triangleright N)$ *if and only if* $M \rightarrow N$ *and* Γ' *is* Γ
- $(\Gamma \triangleright M) \xrightarrow{(\tilde{n})k.a!V} (\Gamma' \triangleright N)$ *if and only if* $M \xrightarrow{(\tilde{n})k.a!V} N$ *and*
  - · $\Gamma \vdash k : \mathsf{loc}$
  - · $a : \mathsf{r}\langle \mathrm{T} \rangle @k$ *occurs in* Γ, *for some type* T
- $(\Gamma \triangleright M) \xrightarrow{(\tilde{n}:\tilde{\mathrm{E}})k.a?V} (\Gamma' \triangleright N)$ *if and only if* $M \xrightarrow{k.a?V} N$ *and*
  - · $\Gamma \vdash k : \mathsf{loc}$
  - · $\Gamma \vdash_k a : \mathsf{w}\langle \mathrm{T} \rangle$, *for some type* T
  - · $\Gamma, \langle \tilde{n} : \tilde{\mathrm{E}} \rangle \vdash_k V : \mathrm{T}$.

Proof: Each statement only requires a simple proof by rule induction. □

Definition 9 (Bisimulations) A binary relation $\mathcal{R}$ over simple configurations is said to be a *bisimulation* if $C\ \mathcal{R}\ D$ implies

- $C \xrightarrow{\mu} C'$ implies $D \overset{\hat{\mu}}{\Longrightarrow} D'$ for some D' such that $C' \mathcal{R} D'$
- Symmetrically, $D \xrightarrow{\mu} D'$ implies $C \overset{\hat{\mu}}{\Longrightarrow} C'$ for some C' such that $C' \mathcal{R} D'$

Here we are using the standard notation from [16]; $\overset{\mu}{\Longrightarrow}$ 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. $\square$

Note that the relation $\approx_{bis}$ forms a knowledge-indexed relation over systems by considering Γ as a parameter to the relation. Moreover it satisfies all for the properties in Definition 8. As an example we will prove that $\approx_{bis}$ is contextual. The following three lemmas will be helpful in establishing this.

Lemma 2 *If* $\Gamma \models M \approx_{bis} N$ *and* $\Gamma <: \Gamma'$, *where* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Gamma')$, *then* $\Gamma' \models M \approx_{bis} N$.

Proof: Straightforward co-induction. $\square$

The next lemma ensures that when new values are extruded to the environment the types at which they become known there are super-types of the type at which they were declared by the system. Here it will be convenient to use the operator $\langle - \rangle_{@}k$, previously only used for environments, to types. $\langle \mathrm{A} \rangle_{@}k$ gives the system level type expression $\mathrm{A}_{@}k$; for any other T, that it location and registered name types, $\langle \mathrm{T} \rangle_{@}k$ returns T itself; it is generalised to lists in the standard manner.

Lemma 3 *If* $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma' \triangleright M'$ *then* $M \equiv (\mathsf{new}\, \tilde{n} : \langle \tilde{\mathrm{T}} \rangle_{@}k)\ M''$ *such that if* $\Gamma' \vdash_k \tilde{n} : \tilde{\mathrm{U}}$ *then* $\tilde{\mathrm{T}} <: \tilde{\mathrm{U}}$.

Proof: We show the case where V is is a simple identifier; but we can use induction to extend this to the more general case.

Suppose we have $\Gamma \triangleright M \xrightarrow{(n)k.c!n} \Gamma' \triangleright M'$. It is straightforward to check that Γ' is $\Gamma \sqcap \langle n : \mathrm{T}_0 \rangle_{@}k$ for some T_0 such that Γ contains $c : \mathsf{r}\langle \mathrm{T}_0 \rangle_{@}k$ and that $M \equiv (\mathsf{new}\, n : \langle \mathrm{T} \rangle_{@}k)\ M''$ for some T and M''. Suppose that $\Gamma' \vdash_k n : U$. We know from the typing rules that it must be the case that $\mathrm{T}_0 <: U$; We show that $\mathrm{T} <: \mathrm{T}_0$. We know that $\Gamma \triangleright M$ is a configuration, so there must exist a Δ such that $\Delta \vdash M$, $\mathsf{dom}(\Delta) = \mathsf{dom}(\Gamma)$ and $\Delta <: \Gamma$. We know from this and the typing rule for outputs that

(i) $\Delta \vdash_k c : \mathsf{w}\langle \mathrm{T}_1 \rangle$
(ii) $\Delta \sqcap \langle n : \mathrm{T} \rangle \vdash_k n : \mathrm{T}_1$

We use (ii) to see that $\mathrm{T} <: \mathrm{T}_1$. We also note that $\Delta(c) <: \Gamma(c)$ implies that

Δ contains $c : \mathsf{r}\langle \mathrm{T}_2\rangle @ k$ for some $\mathrm{T}_2 <: \mathrm{T}_0$. This fact, along with (i), tells us

$$\mathrm{T}_1 <: \mathrm{T}_2 <: \mathrm{T}_0$$

because of the variance condition on read/write channels. Collecting these together we obtain $\mathrm{T} <: \mathrm{T}_0$ as required. □

We now show that actions performed jointly by a system and an observer in its environment can be decomposed into individual components; moreover these components can be recomposed to form again the joint action. The results depend on the fact that the system is part of a simple configuration.

Lemma 4 (Composition/Decomposition)

(i)

(a) If $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma' \triangleright M'$ *and* $O \xrightarrow{k.c?V} O'$ *then* $\Gamma \triangleright M \mid O \overset{\tau}{\Longrightarrow} \Gamma \triangleright (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ M' \mid O'$ *for some* $\tilde{\mathrm{E}}$

(b) If $\Gamma \triangleright M \xrightarrow{(\tilde{n}:\tilde{\mathrm{E}})k.c?V} \Gamma' \triangleright M'$ *and* $O \xrightarrow{(\tilde{n})k.c!V} O'$ *then* $\Gamma \triangleright M \mid O \overset{\tau}{\Longrightarrow} \Gamma \triangleright (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ M' \mid O'$

(ii) If $\Gamma \triangleright M \mid O \xrightarrow{\tau} \Gamma \triangleright M'$ *and* $\Gamma \vdash O$ *then one of the following hold*

(a) $\Gamma \triangleright M \xrightarrow{\tau} \Gamma \triangleright M''$ *such that* $M' \equiv M'' \mid O$

(b) $O \xrightarrow{\tau} O'$ *such that* $M' \equiv M \mid O'$

(c) $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma' \triangleright M''$ *and* $O \xrightarrow{k.c?V} O'$ *such that* $M' \equiv (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ M'' \mid O'$ *for some* $\tilde{\mathrm{E}}$

(d) $\Gamma \triangleright M \xrightarrow{(\tilde{n}:\tilde{\mathrm{E}})k.c?V} \Gamma' \triangleright M''$ *and* $O \xrightarrow{(\tilde{n})k.c!V} O'$ *such that* $M' \equiv (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ M'' \mid O'$

Proof: Part (i) is relatively straightforward. We only show the first case as the other is similar. We can proceed by induction on the number of τ actions in the derivation from the system. For the inductive case this follows easily by the inductive hypothesis and the fact that $\mid$ and $(\mathsf{new}\,)$ are evaluation contexts. We consider the base case in which $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma' \triangleright M'$.

By Proposition 9 we see that $M \xrightarrow{(\tilde{n})k.c!V} M'$. By inspecting the transition rules we note that the following structural forms must hold

- $M \equiv (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ (\mathsf{new}\,\tilde{m}' : \tilde{\mathrm{E}}')\ (k[\![c!\langle V\rangle\, P]\!] \mid M'')$
- $M' \equiv (\mathsf{new}\,\tilde{m}' : \tilde{\mathrm{E}}')\ (k[\![P]\!] \mid M'')$
- $O \equiv (\mathsf{new}\,\tilde{n}' : \tilde{\mathrm{U}}')\ (k[\![c?(X : \mathrm{U})\, Q]\!] \mid O'')$
- $O' \equiv (\mathsf{new}\,\tilde{n}' : \tilde{\mathrm{U}}')\ (k[\![Q\{\!|{}^{V}\!/x|\!\}]\!] \mid O'')$

for k, c not in $\tilde{n}, \tilde{n}', \tilde{m}'$. It is clear that, by alpha-converting where necessary,

$$M|O \equiv (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ ((\mathsf{new}\,\tilde{m}' : \tilde{\mathrm{E}}')\ (k[\![c!\langle V\rangle\, P]\!]|M'')|O) \longrightarrow (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ (M'|O')$$

so we then conclude, by (LTS-RED), that $\Gamma \triangleright M \mid O \xrightarrow{\tau} \Gamma \triangleright (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ (M' \mid O')$ as required.

For Part(ii) we suppose $\Gamma \vdash O$ and consider how the judgement $\Gamma \triangleright M \mid O \xrightarrow{\tau} \Gamma \triangleright M'$ is derived. This transition must be derived using an instance of one of the base axioms for reduction in Figure 2. Call this axiom instance A. The cases which arise are

- A is a subterm of M. In which case O does not contribute to the transition and (a) holds.
- A is a subterm of O. In which case M does not contribute to the transition and (b) holds.
- A is not a subterm of M or O. In which case, by inspecting the rules, we see that the only possibility is that A must be an instance of rule (R-COMM). Let us suppose that A is of the form

$$k[\![c!\langle V\rangle\, P]\!] \mid k[\![c?(X : \mathrm{U})\, Q]\!] \to k[\![P]\!] \mid k[\![Q\{\!|{}^{V}\!/x|\!\}]\!].$$

There are two ways in which this could occur: either M provides the output action, say $\xrightarrow{(\tilde{n})k.c!V}$, and N the corresponding input (in which case (c) will hold), or vice-versa (and (d) will hold). We concentrate on the former as the latter can be dealt with in a similar way. We know that it must be the case that (up to structural equivalence)

$$M \equiv (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ (\mathsf{new}\,\tilde{m}' : \tilde{\mathrm{E}}')\ (k[\![c!\langle V\rangle\, P]\!] \mid M''')$$

$$O \equiv (\mathsf{new}\,\tilde{m} : \tilde{\mathrm{U}})\ (k[\![c?(X : \mathrm{U})\, Q]\!] \mid O'')$$

such that k and c are not in $\tilde{n}, \tilde{m}', \tilde{m}$. Let M'' be the term

$$(\mathsf{new}\,\tilde{m}' : \tilde{\mathrm{E}}')\ (k[\![P]\!] \mid M''')$$

and O' be

$$(\mathsf{new}\,\tilde{m} : \tilde{\mathrm{U}})\ (k[\![Q\{\!|{}^{V}\!/x|\!\}]\!] \mid O'').$$

It is clear that $M' \equiv (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{E}})\ (M'' \mid O')$ so it suffices to demonstrate that $O \xrightarrow{k.c?V} O'$ and $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma' \triangleright M''$ for some Γ' such that $\tilde{\mathrm{E}} <: \Gamma'(\tilde{n})$. The former is immediate from the transition rules for input. We also know that $M \xrightarrow{(\tilde{n})k.c!V} M''$ so we must show that Γ permits the typed action. We know by hypothesis that $\Gamma \vdash O$. This means, in particular, that $\Gamma \sqcap \langle \tilde{m} : \tilde{\mathrm{U}}\rangle@k \vdash k[\![c?(X : \mathrm{U})\, Q]\!]$. This immediately tells us that $\Gamma \vdash k : \mathsf{loc}$ (as k is not in $\tilde{m}$) and $\Gamma \sqcap \langle \tilde{m} : \tilde{\mathrm{U}}\rangle@k \vdash c : \mathsf{r}\langle U\rangle$. Again, c is not in $\tilde{m}$, so it must be that

$$c : \mathsf{r}\langle \mathrm{T}\rangle@k \text{ appears in } \Gamma \text{ for some } T.$$

These two facts, and the fact that $\Gamma \triangleright M$ is a simple configuration, allow us to conclude using Proposition 9 that $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma \sqcap \langle V : \mathrm{T}\rangle@k \triangleright M''$ as required. □

The next series of propositions examine the individual requirements for $\approx_{bis}$ to be contextual.

Proposition 10

$$\Gamma \models M \approx_{bis} N \text{ and } \Gamma, \Gamma' \vdash_{\mathbf{env}} \text{ implies } \Gamma, \Gamma' \models M \approx_{bis} N.$$

Proof: Although this should be straightforward the proof is complicated by the fact that we do not have an Interchange rule for environments.

For the purposes of the proof, let us fix a type environment Γ_0 and an association list Γ'_0 such that $\Gamma_0, \Gamma'_0 \vdash_{\mathbf{env}}$. Using these we define two grammars for extensions of Γ_0, Γ'_0 and Γ_0 respectively, corresponding to the ways in which typed actions can increase the knowledge of an environment.

Let Γ^+ denote environments which can be described by the grammar

$$\begin{aligned}\Gamma^+ ::= {} & \Gamma_0, \Gamma'_0 \\ & \mid \Gamma^+ \sqcap \Gamma \text{ provided } \Gamma \vdash_{\mathbf{env}} \text{ and } \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Gamma'_0) = \emptyset \\ & \mid \Gamma^+, \Gamma \text{ provided } \Gamma^+, \Gamma \vdash_{\mathbf{env}} \text{ and } \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Gamma'_0) = \emptyset\end{aligned}$$

Let the set of environments Γ^- be defined in a similar manner, but starting from Γ_0 rather then Γ_0, Γ'_0. By construction we therefore have that both $\Gamma^+ \vdash_{\mathbf{env}}$ and $\Gamma^- \vdash_{\mathbf{env}}$ and that $\mathsf{dom}(\Gamma'_0)$ is disjoint from $\mathsf{dom}(\Gamma^-)$.

We now define $\mathcal{R}$ such that $(\Gamma^+ \triangleright M) \; \mathcal{R} \; (\Gamma^+ \triangleright N)$ if and only if $\Gamma^- \models M \approx_{bis} N$ and show that $\mathcal{R}$ forms a bisimulation. The result follows easily from this, using the initial case when Γ^+, Γ^- are Γ_0 and Γ'_0 respectively.

The relation is clearly symmetric, so we simply need to show that

$$\Gamma^+ \triangleright N \overset{\hat{\mu}}{\Longrightarrow} \Gamma^+_1 \triangleright N'$$

with

$$(\Gamma^+_1 \triangleright M') \; \mathcal{R} \; (\Gamma^+_1 \triangleright N')$$

whenever $\Gamma^+ \triangleright M \xrightarrow{\mu} \Gamma^+_1 \triangleright M'$. So, suppose that $\Gamma^+ \triangleright M \xrightarrow{\mu} \Gamma^+_1 \triangleright M'$. If μ is a τ action then the definition of $\mathcal{R}$ gives an immediate match from N as τ reductions are independent of the environment. The interesting cases are those actions which are constrained.

Consider the case in which μ is $(\tilde{n})k.c!V$. We know that Γ^+_1 is of the form $\Gamma^+ \sqcap \langle V : \mathrm{T} \rangle_{@k}$. Note that the fact that $\Gamma^- \triangleright M$ is a simple configuration assures us that the domain of $\langle V : \mathrm{T} \rangle_{@k}$ does not intersect that of Γ'_0, so that Γ^+_1 is an environment allowed by our first grammar.

We also know by Proposition 9 that

- $M \xrightarrow{\mu} M'$

- $\Gamma^+ \vdash k : \mathsf{loc}$
- Γ^+ contains $c : \mathsf{r}\langle \mathrm{T}\rangle_{@}k$ for some T.

We know that $\mathsf{dom}(\Gamma_0')$ is disjoint from $\mathsf{dom}(\Gamma^-)$ and that $\Gamma^- \triangleright M$ is a simple configuration, so it must be the case that $\Gamma^- \vdash k : \mathsf{loc}$ and Γ^- contains $c : \mathsf{r}\langle \mathrm{T}\rangle_{@}k$ also. By Proposition 9 again, we see that $\Gamma^- \triangleright M \xrightarrow{\mu} \Gamma^- \sqcap \langle V : \mathrm{T}\rangle_{@}k \triangleright M'$. By definition of $\mathcal{R}$ we know that there must exist some

$$\Gamma^- \triangleright N \overset{\mu}{\Longrightarrow} \Gamma^- \sqcap \langle V : \mathrm{T}\rangle_{@}k \triangleright N'$$

such that

$$\Gamma^- \sqcap \langle V : \mathrm{T}\rangle_{@}k \models M' \approx_{bis} N'.$$

This, and Proposition 9, tells us that $\Gamma^+ \triangleright N \overset{\mu}{\Longrightarrow} \Gamma_1^+ \triangleright N'$ with $(\Gamma_1^+ \triangleright M')\ \mathcal{R}\ (\Gamma_1^+ \triangleright N')$ as required.

The case in which μ is an input transition can be treated similarly, but using the third rule in the grammar for extending environments. □

Proposition 11

$$\Gamma \sqcap \langle n : \mathrm{E}\rangle \models M \approx_{bis} N \textit{ implies } \Gamma \models (\mathsf{new}\, n : \mathrm{E})\ M \approx_{bis} (\mathsf{new}\, n : \mathrm{E})\ N.$$

Proof: In fact, due to Lemma 2, it suffices to show

$$\Gamma \sqcap \langle n : \top\rangle \models M \approx_{bis} N \text{ implies } \Gamma \models (\mathsf{new}\, n : \mathrm{E})\ M \approx_{bis} (\mathsf{new}\, n : \mathrm{E})\ N.$$

We proceed by defining a relation $\mathcal{R}$ which contains $\approx_{bis}$ and relates $(\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\ M)$ and $(\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\ N)$ whenever $\Gamma, n : \top \models M \approx_{bis} N$. We show that $\mathcal{R}$ forms a bisimulation.

Take any two configurations related by $\mathcal{R}$: if these are bisimilar then we can be sure that $\mathcal{R}$ satisfies the necessary closure properties. Thus we can assume that we have chosen configurations of the form $\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\ M$ and $\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\ N$. Suppose $\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\ M \xrightarrow{\mu} \Gamma' \triangleright M'$. There are two possibilities regarding which was the last rule involving (new) used to infer this transition. If rule (LTS-OPEN) was used then μ is of the form $(n)\mu'$ and

$$\Gamma, n : \top \triangleright M \xrightarrow{\mu'} \Gamma' \triangleright M'.$$

We know by the definition of $\mathcal{R}$ that $\Gamma, n : \top \models M \approx_{bis} N$, so we have

$$\Gamma, n : \top \triangleright N \overset{\mu'}{\Longrightarrow} \Gamma' \triangleright N'$$

such that $\Gamma' \triangleright M' \approx_{bis} N'$. In turn we see that $\Gamma \triangleright (\mathsf{new}\, n : \mathrm{E})\ N \overset{\mu}{\Longrightarrow} \Gamma' \triangleright N'$ so that $(\Gamma' \triangleright M')\ \mathcal{R}\ (\Gamma' \triangleright N')$ as required. Otherwise, it must be the case that (LTS-NEW) was used. The matching transition from N can be found using a similar argument. □

Proposition 12

$$\Gamma \models M \approx_{bis} N \text{ and } \Gamma \vdash O \text{ implies } \Gamma \models M \mid O \approx_{bis} N \mid O.$$

Proof: We do this by defining a relation $\mathcal{R}$ such that

$$(\Gamma \triangleright (\mathsf{new}\,\tilde{n}_0 : \tilde{\mathrm{U}}_1)\ M \mid O)\ \mathcal{R}\ (\Gamma \triangleright (\mathsf{new}\,\tilde{n}_0 : \tilde{\mathrm{U}}_2)\ N \mid O)$$

if and only if there exists some $\tilde{\mathrm{E}}$ such that $\tilde{\mathrm{U}}_1 <: \tilde{\mathrm{E}}$ and $\tilde{\mathrm{U}}_2 <: \tilde{\mathrm{E}}$ and $\Gamma \sqcap \langle \tilde{n}_0 : \tilde{\mathrm{E}} \rangle \models M \approx_{bis} N$ and $\Gamma \sqcap \langle \tilde{n}_0 : \tilde{\mathrm{E}} \rangle \vdash O$. We must show that $\mathcal{R}$ forms a bisimulation. For the purposes of exposition we will assume that $\tilde{n}_0$ is empty. The more general case follows in a similar manner.

Take $(\Gamma \triangleright M \mid O)\ \mathcal{R}\ (\Gamma \triangleright N \mid O)$ and suppose that $\Gamma \triangleright M \mid O \xrightarrow{\mu} \Gamma' \triangleright M'$. If μ is not a τ action then it clearly derives entirely from M or O. In either case, a matching μ transition can be found from N because $\Gamma \models M \approx_{bis} N$. Suppose then that μ is a τ action. We use Lemma 4, Part (ii), to observe that one of four cases hold.

(a) $\Gamma \triangleright M \xrightarrow{\tau} \Gamma \triangleright M''$. Again, matching transitions are easily found because $\Gamma \models M \approx_{bis} N$.

(b) $O \xrightarrow{\tau} O'$. But then $\Gamma \triangleright N \mid O \xrightarrow{\tau} \Gamma \triangleright N \mid O'$ and, by Subject Reduction, Theorem 2, we know that $\Gamma \vdash O'$ also so $(\Gamma \triangleright M \mid O')\ \mathcal{R}\ (\Gamma \triangleright N \mid O')$ as required.

(c) $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma' \triangleright M''$ and $O \xrightarrow{k.c?V} O'$ such that $M' \equiv (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{U}}_1)\ (M'' \mid O')$ for some $\tilde{\mathrm{U}}_1$. We note that there must exist some $\Gamma \triangleright N \overset{(\tilde{n})k.c!V}{\Longrightarrow} \Gamma' \triangleright N'$ such that $\Gamma' \models M'' \approx_{bis} N'$ and moreover, by Lemma 4, Part (i), we see that $\Gamma \triangleright N \mid O \overset{\tau}{\Longrightarrow} \Gamma \triangleright (\mathsf{new}\,\tilde{n} : \tilde{\mathrm{U}}_2)\ N' \mid O'$ for some $\tilde{\mathrm{U}}_2$. But we know that Γ' is $\Gamma \sqcap \langle V : \mathrm{E} \rangle_{@} k$ and that $\tilde{n}$ are all contained in V, so Γ' is necessarily of the form $\Gamma'' \sqcap \langle \tilde{n} : \tilde{\mathrm{E}} \rangle$. We know by Lemma 3 that $\tilde{\mathrm{U}}_1 <: \tilde{\mathrm{E}}$ and $\tilde{\mathrm{U}}_2 <: \tilde{\mathrm{E}}$. In particular, we have
$$\Gamma'' \sqcap \langle \tilde{n} : \tilde{\mathrm{E}} \rangle \models M' \approx_{bis} N'.$$
Now, we know that $\Gamma \triangleright M \xrightarrow{(\tilde{n})k.c!V} \Gamma' \triangleright M''$ so this means that Γ contains $c : \mathsf{r}\langle \mathrm{T} \rangle_{@} k$ for some T. We also know that $\Gamma \vdash O$ and $O \xrightarrow{k.c?V} O'$. Up to structural equivalence then, this means that

$$O \equiv (\mathsf{new}\,\tilde{m} : \tilde{\mathrm{U}})\ (k[\![c?(X : \mathrm{U})\, Q]\!] \mid O'')$$

and
$$O' \equiv (\mathsf{new}\,\tilde{m} : \tilde{\mathrm{U}})\ (k[\![Q\{\!|V/x|\!\}]\!] \mid O'')$$
with k, c not in $\tilde{m}$. By inspecting the typing rules we see that

$$\Gamma \vdash c : \mathsf{r}\langle \mathrm{U} \rangle_{@} k$$

and
$$\Gamma \sqcap \langle V : \mathrm{T} \rangle_{@} k \sqcap \langle \tilde{m} : \tilde{\mathrm{U}} \rangle \sqcap \langle X : \mathrm{U} \rangle_{@} k \vdash_{k} Q.$$

The former tells us that T <: U because we know Γ contains $c : \mathsf{r}\langle \mathrm{T}\rangle@k$, and the latter, along with the fact that

$$\Gamma \sqcap \langle V : \mathrm{T}\rangle@k \sqcap \langle \tilde{m} : \tilde{\mathrm{U}}\rangle \vdash_k V : \mathrm{T}$$

and Theorem 1, tells us that $\Gamma \sqcap \langle V : \mathrm{T}\rangle@k = \Gamma'' \sqcap \langle \tilde{n} : \tilde{\mathrm{E}}\rangle \vdash O'$ so we can conclude

$$(\Gamma' \triangleright (\mathsf{new}\, \tilde{n} : \tilde{\mathrm{U}}_1)\ M' \mid O')\ \mathcal{R}\ (\Gamma' \triangleright (\mathsf{new}\, \tilde{n} : \tilde{\mathrm{U}}_2)\ N' \mid O')$$

as required.

(d) $\Gamma \triangleright M \xrightarrow{(\tilde{n}:\tilde{\mathrm{E}})k.c?V} \Gamma' \triangleright M''$ and $O \xrightarrow{(\tilde{n})k.c!V} O'$. Similar to previous case. □

Corollary 1 *The knowledge-indexed relation $\approx_{bis}$ over systems is contextual.*

Proof: We need to show the three relevant properties of contextuality; namely, preservation under weakening, $(\mathsf{new}\, n : \mathrm{T})\ [\ ]$ contexts and parallel composition. Note that these are exactly the content of the previous three propositions. □

Moreover it is the largest knowledge-indexed relation which satisfies all the properties of Definition 8:

Theorem 3 (Full abstraction of $\cong_{rbc}$ for $\approx_{bis}$)

$$\Gamma \models M \cong_{rbc} N \qquad \textit{iff} \qquad \Gamma \models M \approx_{bis} N$$

Proof: One direction is straightforward. We have just shown that $\approx_{bis}$ is contextual. By definition it is point-wise symmetric and reduction closed, and it is easy to prove that it is barb preserving. It follows that $\approx_{bis}$ is contained in $\cong_{rbc}$.

The converse is more difficult. It involves constructing a context $C[-]_{\alpha,\Gamma}$ which in some sense characterises the ability of a system to perform the external action α in the environment Γ. Approximately this context should have the property that

$$\Gamma \triangleright M \stackrel{\alpha}{\Longrightarrow} \Gamma' \triangleright N$$

if and only if $C[M]_{\alpha,\Gamma} \Rightarrow D[N]$, where $D[-]$ is a canonical context from which both N and Γ' are in some sense *recoverable.*

The formal proof can be recovered as an instance of the more complicated Theorem 5 and is therefore omitted. □

5 Controlling mobility

We now consider a richer calculus in which movement of processes may be controlled. As explained in the Introduction, in DPI any process which is in possession of the name of a location may travel to that place and begin executing arbitrary code there. 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 [11]. To explain intuitively how such capabilities can be used let us consider an example.

Example 6 (The taxman) Let us re-examine the bank account server in Example 3. The server $\mathsf{Bserver}$ automatically gains knowledge of all generated bank accounts, and therefore apriori has migration rights to them; it can run whatever code it wishes at these sites. A simple variation, which takes advantage of this fact could be defined as:

$$\begin{aligned}\mathsf{Bserver} \Leftarrow s[\![&\mathsf{request}?(x:\mathbf{int}, y@z)\\ &\quad(\mathsf{newloc}\, b:\mathrm{L}_b)\ \mathsf{with}\ \ldots\mathsf{put},\ \mathsf{get}\ldots\ \mathsf{in}\\ &\qquad \mathsf{goto}\, z.y!\langle b\rangle\ \mid \mathsf{Taxman}\ ]\!]\end{aligned}$$

where the agent Taxman is defined by:

$$\begin{aligned}&\mathsf{deduct}?(x:\mathbf{int}, y:\mathrm{L}_b, z)\\ &\quad \mathsf{goto}\, y.\ldots \textit{collect tax with}\ \mathsf{get},\mathsf{put}\\ &\quad \ldots \textit{return to}\ \ z\end{aligned}$$

This agent can be sent by the server to any client bank account, bound to y, to collect an amount of tax, bound to x.

With move capabilities an alternative server could be defined which generates bank accounts with nominated migration rights; indeed the form of move capabilities could be such that the client can determine those sites which may use the accounts:

$$\begin{aligned}&(\mathsf{newreg}\ \ \mathsf{put}:\mathsf{rc}\langle \mathrm{T}_p\rangle,\ \mathsf{get}:\mathsf{rc}\langle \mathrm{T}_g\rangle)\\ \mathsf{BserverCon} \Leftarrow\ & s[\![\mathsf{request}?(x:\mathbf{int}, y@z, W) \qquad - W\ \textit{allowed sites}\\ &\qquad (\mathsf{newloc}\, b:\mathrm{L}_b^W)\ \mathsf{with}\ \ldots\mathsf{put}\ldots\mathsf{get}\ldots]\!]\end{aligned}$$

$$\frac{\Gamma \vdash u : \mathsf{loc}}{\Gamma, u : \mathbf{move}_* \vdash \mathbf{env}} \qquad \frac{}{\Gamma, u : \mathbf{move}_*, \Gamma' \vdash_w u : \mathsf{loc}[\mathbf{move}_*]}$$

Fig. 11. Extra rules for move capability

Here we use L_b^W to denote (informally) the type of a location with the resources required of a bank account, but in addition only locations mentioned in W have immigration rights there. □

As we shall see the introduction of move capabilities, and associated immigration rights, complicates considerably the nature of contextual equivalences. Here we study a particularly simple form, a *universal* move capability $\mathbf{move}_*$. This allows us to have location types in two different forms:

$$\begin{array}{ll} \mathsf{loc}[u_1 : \mathrm{A}_1, \ldots] & \text{as before} \\ \mathsf{loc}[\mathbf{move}_*,\ \ u_1 : \mathrm{A}_1, \ldots] & \end{array}$$

Now having a location name l at the second type endows a process with emigration rights to l. In contrast having l at the first type means that the process has knowledge of the resources there, but has not yet gained immigration rights. Of course this is not a very realistic, or useful, capability, but it does serve to demonstrate the complexity inherent in the treatment of move capabilities in general. In the conclusion we discuss generalisations.

Formally to incorporate this new capability into DPI we need to modify the type system. The details are straightforward:

- We redefine the capabilities in Figure 4 to read

$$\text{Capabilities: R} ::= u : \mathrm{A} \ \mid \ \mathbf{move}_*$$

- Type environments can now also include entries of the form $u : \mathbf{move}_*$. We add rules to the type judgements for environments and values accordingly; see Figure 11.
- Finally, we change the type inference of the migration primitive by replacing the rule (T-GO) from Figure 9 with

$$\text{(T-MOVE−GO)} \quad \frac{\begin{array}{l}\Gamma \vdash u : \mathsf{loc}[\mathbf{move}_*] \\ \Gamma \vdash_u P : \mathbf{proc}\end{array}}{\Gamma \vdash_w \mathsf{goto}\, u.P}$$

We make no change to the reduction semantics, nor the definition of contextual equivalence for the language. It is straightforward to check that Theorem 2 (Subject Reduction) also holds for this extended calculus.

Let us now examine the effect migration rights, even in this simple form, have on behavioural equivalences. Suppose N_1, N_2 are given by

$$k[\![a!\langle\rangle\,\mathsf{stop}]\!] \qquad \text{and} \qquad k[\![\mathsf{stop}]\!] \tag{16}$$

The question of whether or not N_1 and N_2 are contextually equivalent relative to an environment Γ, now written

$$\Gamma \models N_1 \approxeq^{m}_{rbc} N_2$$

depends on whether there are locations known to the environment Γ which have migration rights to k. If so, say at a location l_1, agents may be sent from l_1 to k in order to observe the difference in behaviour between M_1 and M_2 at k. Moreover if the environment maintains a central location, with universal immigration rights, then these agents may report back the result of their observations.

In the following two subsections, two different generalisations to the full-abstraction result, Theorem 3 for DPI augmented with this simple form of move capability.

5.2 Mobility Bisimulation equivalence

It is straightforward to adapt the typed actions in Figure 10 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 10 with

(LTS-OUT$_\mathrm{M}$)

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

(LTS-IN$_\mathrm{M}$)

$$\frac{\begin{array}{l}\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*] \\ a : \mathsf{w}\langle \mathrm{U}\rangle @k \in \Gamma \\ \Gamma, \Gamma' \vdash_k V : \langle \mathrm{U}\rangle @\end{array}}{(\Gamma \triangleright k[\![a?(X : \mathrm{T})\, P]\!]) \xrightarrow{k.a?V}_m (\Gamma, \Gamma' \triangleright k[\![P\{\!|V/X|\!\}]\!])}$$

and leaving the other rules unchanged.

Definition 10 (Typed m-Bisimulations) A typed relation $\mathcal{R}$ over systems is said to be a *typed m-bisimulation* if it satisfies the requirements of Definition 9, with the relation $\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright M'$ replaced by $\Gamma \triangleright M \xrightarrow{\mu}_m \Gamma' \triangleright M'$.

We use $\Gamma \models M \approx^m_{bis} N$ to denote the resulting version of bisimulation equivalence. □

Example 7 () As in (16) 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 8 () Here let $N_3,\ N_4$ represent the systems

$$\begin{array}{ll}(\mathsf{new}\, k : \mathsf{loc}[b : \mathsf{rw}\langle\rangle])\ \ l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!] & \quad\text{and}\\ (\mathsf{new}\, k : \mathsf{loc}[b : \mathsf{rw}\langle\rangle])\ \ l[\![a!\langle k\rangle]\!] \mid k[\![\mathbf{0}]\!] & \end{array}$$

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 equivalence is a restricted form of contextuality:

Proposition 13 *Suppose* $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*]$*. Then* $\Gamma \models M \approx^m_{bis} N$ *and* $\Gamma \vdash k[\![P]\!]$ *implies* $\Gamma \models M \mid k[\![P]\!] \ \approx^m_{bis}\ N \mid k[\![P]\!]$.

Proof: The proof is similar to that of Proposition 12, where now the hypothesis $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*]$ is necessary to allow interaction between the two systems in parallel. □

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. A typed relation over systems is *m-contextual* if

(i) $\Gamma \models M \mathcal{R} N$ and $\Gamma, \Gamma' \vdash_{\mathbf{env}}$ implies $\Gamma, \Gamma' \models M \mathcal{R} N$

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

(iii) $\Gamma \sqcap \langle n : \mathrm{T} \rangle \models M \mathcal{R} N$ implies $\Gamma \models (\mathsf{new}\, n : \mathrm{T})\ M\ \mathcal{R}\ (\mathsf{new}\, n : \mathrm{T})\ N$

m-Barb preservation: Here we only allow barbs at locations to which migration rights exist. We write $\Gamma \vdash M \Downarrow^{\mathsf{m\text{-}barb}} a@k$ if

- $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*]$ and $\Gamma \vdash a : \mathsf{rw}\langle\rangle@k$
- there exists some M' such that $M \rightarrow^* M'$ where M' is structurally equivalent to a term of the form $(M'' \mid k[\![a!\langle\rangle\, P \mid Q]\!])$

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

Definition 11 (m-Reduction barbed congruence) Let $\approxeq^{m}_{rbc}$ be the largest typed relation over systems which is reduction-closed, m-contextual and m-barb preserving. □

Theorem 4 (Full abstraction of $\approxeq^{m}_{rbc}$ for $\approx^{m}_{bis}$)

$$\Gamma \models M \approxeq^{m}_{rbc} N \qquad \textit{iff} \qquad \Gamma \models M \approx^{m}_{bis} N.$$

Proof: The formal proof can be recovered as an instance of the more complicated Theorem 5 and is therefore omitted. □

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 7 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. A typed relation $\mathcal{R}$ is said to be $\mathcal{T}$*-contextual* if

(i) $\Gamma \models M \mathcal{R} N$ and $\Gamma, \Gamma' \vdash \mathbf{env}$ implies $\Gamma, \Gamma' \models M \mathcal{R} N$
(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 $\Gamma \models (M \mid k[\![P]\!]) \mathcal{R} (N \mid k[\![P]\!])$
(iii) $\Gamma \sqcap \langle n : \mathrm{T}\rangle \models M \mathcal{R} N$ implies $\Gamma \models (\mathsf{new}\, n : \mathrm{T})\ M\ \ \mathcal{R}\ \ (\mathsf{new}\, n : \mathrm{T})\ N$

Definition 12 ($\mathcal{T}$-Reduction barbed congruence) Let $\cong^{\mathcal{T}}_{rbc}$ be the largest typed relation over systems which is reduction-closed, $\mathcal{T}$-contextual and m-barb preserving. $\square$

Note that here we still only allow barbs at locations to which we have migration rights. This could be generalised to also allow barbs at locations in $\mathcal{T}$. But it would not change the equivalence as these local barbs can always be replaced by barbs at predefined locations which the environment declares with migration rights.

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

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 9 to obtain a new behavioural equivalence, which we denote by $\approx^{\mathcal{T}}_{bis}$. Unfortunately this does not coincide with the contextual equivalence $\cong^{\mathcal{T}}_{rbc}$.

Example 9 () 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

$$h : \mathsf{loc},\ h : \mathbf{move}_*,\ k : \mathsf{loc},\ a : \mathsf{rw}\langle \mathrm{T}\rangle@h$$

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

However $\Gamma \models N_5 \cong^{\mathcal{T}}_{rbc} N_6$ 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 fact that b exists at k. But it is not possible to transfer this information from h to where it can be put to use, namely k. $\square$

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. Thus, any direct interactions performed at a location k in $\mathcal{T}$, without using migration, need to be made with a localised knowledge at k. This motivates the new form of configurations we introduce for the labelled transition system necessary in order to characterise $\approx_{rbc}^{\mathcal{T}}$.

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 13 (Configurations)

- An *environment structure*, or simply a structure, over $\mathcal{T}$, consists of a family of type environments $\overline{\Gamma} = \Gamma, \Gamma_{k_1}, \ldots, \Gamma_{k_n}$ such that
  - $\cdot$ $\mathcal{T} = \{k_1, \ldots, k_n\}$
  - $\cdot$ $\Gamma <: \Gamma_{k_i}$ for each $1 \leq i \leq n$

  We sometimes use Γ_{k_0} to denote the first component of the structure $\overline{\Gamma}$.
- A *configuration* $\overline{\Gamma} \triangleright M$ (over $\mathcal{T}$) consists of an environment structure $\overline{\Gamma}$ and a system M such that there exists some environment Δ with
  - $\cdot$ $\Delta \vdash M$
  - $\cdot$ $\Delta <: \Gamma$
  - $\cdot$ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Gamma)$ □

We will write $\overline{\Gamma}_{\triangledown}^{\mathcal{T}}$ 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 the context. We understand $\overline{\Gamma}, \overline{\Gamma}'$ and $\overline{\Gamma} \sqcap \overline{\Gamma}'$ to be point-wise 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 10 with those in Figure 12 and modify the remaining rules in Figure 10 in the obvious manner; an example of the required modification is given in the rule (LTS-T $-$ WEAK), where the new knowledge, a name n of type T, is extended throughout all components of the environment. A similar modification is required for the rules involving (new) .

(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)

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

(LTS-T−IN)

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

(LTS-T−WEAK)

$$\frac{(\overline{\Gamma}, \overline{(n : \mathrm{E})}_{\triangledown} \triangleright M) \xrightarrow{(\tilde{n}:\tilde{\mathrm{E}})k.a?V} (\overline{\Gamma'} \triangleright M')}{(\overline{\Gamma} \triangleright M) \xrightarrow{(n:\mathrm{E}\tilde{n}:\tilde{\mathrm{E}})k.a?V} (\Gamma' \triangleright M')} \quad n \neq a, k$$

Fig. 12. Labelled transition rules accounting for the move capability

Note that each of the new rules, involving input and output, have two cases, depending on whether the $\mathbf{move}_*$ capability of the location under scrutiny is present, or if it is in the parameter $\mathcal{T}$. The difference between the cases lies in whether the ability to perform the corresponding action is checked locally or globally. For outputs, if the move capability is present then only the global environment is increased with the new knowledge; in its absence the corresponding local environment also has to be updated. Note also that the effect of the rule (LTS-T − WEAK) is that all new names generated by the environment are made known to every component.

It is straightforward to check that the resulting labelled transition system is well-defined, and has many of the desirable properties of the more straightforward labelled transition system of Section 4.

Proposition 14

- *If* $\overline{\Gamma} \triangleright M$ *is a configuration and* $(\overline{\Gamma} \triangleright M) \xrightarrow{\alpha} (\overline{\Gamma}' \triangleright M')$ *then* $\overline{\Gamma}' \triangleright M'$ *is also a configuration.*
- *For every* $\overline{\Gamma}$ *and every action* α *there exists a unique structure* $(\overline{\Gamma}\ \mathsf{after}\ \alpha)$ *with the property that* $(\overline{\Gamma} \triangleright M) \xrightarrow{\alpha} (\overline{\Gamma}' \triangleright M')$ *implies* $\overline{\Gamma}'$ *is* $(\overline{\Gamma}\ \mathsf{after}\ \alpha)$*.*

Proof: Similar to that of Proposition 8. □

The evolution from $\overline{\Gamma}$ to $(\overline{\Gamma}\ \mathsf{after}\ \alpha)$ involves two distinct kinds of increase in knowledge. The first is when the types associated with names already known to Γ are changed (to a subtype) and the second is when new names are created. The latter, for example happens when the action is an input rule and the environment creates new names; it can also happen in the output case, when the system extrudes new names. But in all cases the new knowledge is distributed to each component of the new environment structure. We call this new information the *extension of* $\overline{\Gamma}$ *by* α.

The standard definition of (weak) bisimilarity may now 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 show that this co-inductive equivalence characterises the contextual equivalence $\cong^{\mathcal{T}}_{rbc}$.

However to carry out the proof we must first generalise the latter, from simple environments Γ to the structures $\overline{\Gamma}$. This involves generalising the notion of *context-closure* to families of relations indexed by environment structures.

$\mathcal{T}$-Context closure revisited: A family of relations $\mathcal{R}$, parameterised over environment structures, is said to be $\mathcal{T}$*-contextual* if

(i) $\overline{\Gamma} \models M\ \mathcal{R}\ N$ and $\Gamma, \Gamma' \vdash_{\mathbf{env}}$ implies $\overline{\Gamma}, \overline{\Gamma'}_{\triangledown} \models M\ \mathcal{R}\ N$
(ii) $\overline{\Gamma} \models M\ \mathcal{R}\ N$, $\Gamma \vdash k[\![P]\!]$, where $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*]$, implies $\overline{\Gamma} \models (M \mid k[\![P]\!])\ \mathcal{R}\ (N \mid k[\![P]\!])$
(iii) $\overline{\Gamma} \models M\ \mathcal{R}\ N$, $\Gamma_i \vdash k_i[\![P]\!]$, where $k_i \in \mathcal{T}$, implies $\overline{\Gamma} \models (M \mid k_i[\![P]\!])\ \mathcal{R}\ (N \mid k_i[\![P]\!])$
(iv) $\overline{\Gamma} \sqcap \overline{\langle n : \mathrm{T}\rangle}_{\triangledown} \models M\ \mathcal{R}\ N$ implies $\overline{\Gamma} \models (\mathsf{new}\, n : \mathrm{T})\ M\ \ \mathcal{R}\ \ (\mathsf{new}\, n : \mathrm{T})\ N$

Definition 14 ($\mathcal{T}$-Reduction barbed congruence revisited) Let $\cong^{\mathcal{T}}_{rbc}$ be the largest family of relations indexed by environment structures which is reduction-closed, $\mathcal{T}$-contextual and m-barb preserving. □

These definitions are designed with the following property in mind:

Proposition 15 $\Gamma \models M \underset{\sim}{\approx}^{\mathcal{T}}_{rbc} N$ *(according to Definition 12) if and only if* $\overline{\Gamma}_{\triangledown} \models M \underset{\sim}{\approx}^{\mathcal{T}}_{rbc} N$ *(according to Definition 14) .*

Proof: Straightforward unravelling of the definitions. □

So now we can concentrate on relating the relation $\overline{\Gamma} \models M \underset{\sim}{\approx}^{\mathcal{T}}_{rbc} N$ with $\overline{\Gamma} \models M {\approx}^{\mathcal{T}}_{bis} N$, and thereby obtain a co-inductive characterisation of the real relation of interest, $\Gamma \models M \underset{\sim}{\approx}^{\mathcal{T}}_{rbc} N$.

Proposition 16 (Soundness of $\approx_{bis}$ for $\underset{\sim}{\approx}_{rbc}$) *For any* $\mathcal{T}$, $\overline{\Gamma} \models M \approx^{\mathcal{T}}_{bis} N$ *implies* $\overline{\Gamma} \models M \underset{\sim}{\approx}^{\mathcal{T}}_{rbc} N$.

Proof: This involves showing that the co-inductive relation satisfies the defining properties of $\underset{\sim}{\approx}^{\mathcal{T}}_{rbc}$, the most difficult one being the preservation of relevant contexts. However the proof is a mild generalisation of that of Corollary 1, and its preceding propositions. □

The remainder of the section is devoted to the proof of the converse of this proposition, namely completeness. The main challenge is to design contexts which *characterise*, in some sense the typed actions of Figure 12. The intuitive idea is to maintain some *home-base*, which we will denote with the new location name k_0, which maintains all global information, available at another new resource named r_0. Each site k_i in $\mathcal{T}$ will maintain a record of *local information*, available at a local resource which we call r_i. An invariant of the testing context is that the information available at each r_i is also available globally, at r_0.

The context for an action at a site l depends on whether the environment has migration rights to l. If it does then an agent is launched from the *home-base* to l and reports back to home-base, updating the information at r_0. If it does not then the test is purely local; it is launched from the relevant location k_i in $\mathcal{T}$, although to maintain the invariant the global knowledge is also updated at the home base k_0.

The definition of the defining contexts, $C^{\overline{\Gamma}}_{\alpha}$, for the action α relative to the structure $\overline{\Gamma}$ is given in Figure 13. However to keep the description manageable we only consider the case where α involves the transmission of a single name; this contains all the essential details of the more general case. The description also uses a considerable number of notational conventions we now outline.

Notation 1

- *For the remainder of this section we assume that the names* k_0, $r_0, r_1, \ldots, r_n$ *are always chosen to be fresh wherever they are used; their use has already been informally explained. We also require the fresh names* δ, δ_{fail} *and* δ_{succ}

For notational convenience below we use $\overline{\Gamma}'$ as an abbreviation for $(\overline{\Gamma}\,\mathsf{after}\,\alpha)$.

- If α is $(\tilde{m})k.a!v$ and $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*]$ then $C_\alpha^{\overline{\Gamma}} =$

$$k_0[\![\mathsf{goto}\,k.a?X.\mathsf{if}\ (X =_\Gamma (\mathsf{new}\,\tilde{m})\ v)\ \mathsf{then}\ \mathsf{goto}\,k_0.(r_0!\left\langle v_{\overline{\Gamma}'_{k_0}}\right\rangle \mid \delta!\langle\rangle)\ \mathsf{else}\ \mathbf{0}]\!] \\ \mid \mathsf{LReport}$$

- If α is $(\tilde{m})k_i.a!v$, where $\Gamma \not\vdash k_i : \mathsf{loc}[\mathbf{move}_*]$ but $k_i \in \mathcal{T}$ then $C_\alpha^{\overline{\Gamma}} =$

$$k_i[\![a?X.\mathsf{if}\ (X =_\Gamma (\mathsf{new}\,\tilde{m})\ v)\ \mathsf{then}\ r_i!\left\langle v_{\overline{\Gamma}'_{k_i}}\right\rangle \mid \mathsf{goto}\,k_0.(r_0!\left\langle v_{\overline{\Gamma}'_{k_i}}\right\rangle \mid \delta!\langle\rangle)\ \mathsf{else}\ \mathbf{0}]\!] \\ \mid \prod_{k_j\in\mathcal{T},i\neq j} k_j[\![r_j!\left\langle v_{\overline{\Gamma}'_{k_j}}\right\rangle]\!]$$

- If α is $(\tilde{m}:\tilde{T})k.a?v$ and $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*]$ then $C_\alpha^{\overline{\Gamma}} =$

$$(\mathsf{new}\,\tilde{m}:\tilde{\mathrm{E}})\ \ k_0[\![\mathsf{goto}\,k.a!\langle v\rangle\,.\mathsf{goto}\,k_0.(r_0!\left\langle v_{\overline{\Gamma}'_{k_0}}\right\rangle \mid \delta!\langle\rangle)]\!] \mid \mathsf{LReport}$$

- If α is $(\tilde{m}:\tilde{T})k_i.a?v$, where $\Gamma \not\vdash k_i : \mathsf{loc}[\mathbf{move}_*]$ but $k_i \in \mathcal{T}$ then $C_\alpha^{\overline{\Gamma}} =$

$$(\mathsf{new}\,\tilde{m}:\tilde{\mathrm{E}})\ \ k_i[\![a!\langle v\rangle\,.\mathsf{goto}\,k_0.(r_0!\left\langle v_{\overline{\Gamma}'_{k_0}}\right\rangle \delta!\langle\rangle)]\!] \mid \mid \mathsf{LReport}$$

Fig. 13. Contexts for actions

to be used periodically as barbs.

- *For a type environment Γ we use v_Γ to represent a tuple consisting of all the identifiers (and compound identifiers) in the domain of Γ. This is our way of representing the knowledge of an environment, typeable by Γ. Thus in Figure 13 when a test has been completed the new knowledge in the form of v_Δ for some Δ, is made available at the global resource name r_0. In all but one case this is $(\overline{\Gamma}\,\mathsf{after}\,\alpha)_{k_0}$. The exception is in the second case, where for reasons of typability this has to be restricted to the local knowledge at the location of the action. But in all cases local knowledge in a similar form is also made available at the local resource names r_i; this uses the process term*

$$\prod_{k_j\in\mathcal{T}} k_j[\![r_j!\left\langle v_{(\overline{\Gamma}\mathsf{after}\alpha)_{k_j}}\right\rangle]\!]$$

which we abbreviate to $\mathsf{LReport}$.
- *For a type environment Γ we use (Γ) to represent the tuple of types listed in Γ in such a way that $\Gamma \vdash v_\Gamma : (\Gamma)$. These will be used to ensure that the action contexts are properly typed.*
- *The action contexts for outputs receive a value v and test its identity against*

all known identifiers. In Figure 13 this testing is expressed using the notation $(X =_\Gamma (\mathsf{new}\,\tilde{m})\ v)$*, which is defined by*

$$\begin{array}{ll} X = n & \textit{if}\ \ v = n,\ \textit{and}\, n \notin (\tilde{m}) \\ X \notin \Gamma & \textit{if}\ \ v \in (\tilde{m}) \\ (X_1 =_\Gamma (\mathsf{new}\,\tilde{m})\ V_1) \wedge (X_2 =_\Gamma (\mathsf{new}\,\tilde{m})\ V_2) & \textit{if}\ \ X = X_1 @ X_2 \\ & \qquad \textit{and}\ V = V_1 @ V_2 \end{array}$$

Here $X \notin \Gamma$ *is the obvious encoding of nested tests for* X *against each identifier in the domain of* Γ*, and we use* $\wedge$ *as a shorthand for a programmed conjunction of tests.*

We need to ensure that these action contexts can be properly typed, and that the definition of $\mathcal{T}$-context closure actually allows them to be used as contexts; this is far from apparent from the definition of $\mathcal{T}$-context closure. A useful description of allowed contexts is given in the following definition.

Definition 15 ($\mathcal{T}$-contexts) Let $\mathsf{obs}(\overline{\Gamma}, N)$ be the least relation which satisfies the following conditions:

- $\mathsf{obs}(\overline{\Gamma}, \mathbf{0})$
- $\mathsf{obs}(\overline{\Gamma}, N)$, $\Gamma \vdash k : \mathsf{loc}[\mathbf{move}_*]$ and $\Gamma \vdash k[\![P]\!]$ implies $\mathsf{obs}(\overline{\Gamma}, N \mid k[\![P]\!])$
- $\mathsf{obs}(\overline{\Gamma}, N)$, $k \in \mathcal{T}$ and $\Gamma_{k_i} \vdash k_i[\![P]\!]$ implies $\mathsf{obs}(\overline{\Gamma}, N \mid k_i[\![P]\!])$
- $\mathsf{obs}(\overline{\Gamma, n : \mathrm{T}_\triangledown}, N)$ implies $\mathsf{obs}(\overline{\Gamma}, (\mathsf{new}\, n : \mathrm{T})\ N)$ □

Proposition 17 *Let* $\mathcal{R}$ *be any family of relations which is* $\mathcal{T}$*-contextual. If* $\mathsf{obs}(\overline{\Gamma}, O)$ *then* $\Gamma \models M\ \mathcal{R}\ N$ *implies* $\Gamma \models (M \mid O)\ \mathcal{R}\ (N \mid O)$

Proof: By induction on the definition of $\mathsf{obs}(\overline{\Gamma}, O)$. □

We should not expect $C^{\overline{\Gamma}}_\alpha$ to be an allowed context for $\overline{\Gamma}$, that is $\mathsf{obs}(\overline{\Gamma}, C^{\overline{\Gamma}}_\alpha)$; we need to add types for the new housekeeping names k_0, r_i etc. used. Unfortunately the precise typings for these new names depends to some extent on the action α. The basic reason is that the test positioned at $k_i \in \mathcal{T}$ has to be typeable by the local knowledge Γ_{k_i}. This test includes an agent which reports to the home base; however to ensure typeability its behaviour there can only depend on the knowledge Γ_{k_i}; see the second case in Figure 13, where at r_0 only the knowledge known at k_i can be reported globally. With this in mind let us define $\Gamma_{H,\alpha}$, a list of type associations as follows:

$$\begin{array}{l} k_0{:}\mathsf{loc},\ \ \mathbf{move}@k_0,\ \ \delta{:}\mathsf{rw}\langle\rangle@k_0,\ \ \delta_{\mathrm{fail}}{:}\mathsf{rw}\langle\rangle@k_0,\ \ \delta_{\mathrm{succ}}{:}\mathsf{rw}\langle\rangle@k_0, \\ r_0{:}\mathsf{rw}\langle(\overline{\Gamma}\ \mathsf{after}\ \alpha)_n\rangle@k_0,\ \ r_1{:}\mathsf{rw}\langle(\overline{\Gamma}\ \mathsf{after}\ \alpha)_{k_1}\rangle@k_1, \ldots, r_n{:}\mathsf{rw}\langle(\overline{\Gamma}\ \mathsf{after}\ \alpha)_{k_n}\rangle@k_n \end{array}$$

where

- if α is an output move, $\Gamma \not\vdash k_i : \mathsf{loc}[\mathbf{move}_*]$ but $k_i \in \mathcal{T}$ then the index n is k_i
- otherwise n is k_0.

We can now state the main result which formalises the correspondence between typed actions and typed contexts. It uses a term $\mathsf{GReport}_\alpha$ to give the state of knowledge, after the successful completion of the action; its composition depends slightly on the action in question.

Proposition 18 *Suppose* $(\overline{\Gamma} \triangleright M)$ *is a configuration. Then* $(\overline{\Gamma} \triangleright M) \overset{\alpha}{\Longrightarrow} (\overline{\Gamma}' \triangleright M')$ *if and only if*

(1) $\mathsf{obs}(\overline{\Gamma}, \overline{\Gamma_{H,\alpha_\nabla}}, C_\alpha^{\overline{\Gamma}})$, *that is* $C_\alpha^{\overline{\Gamma}}$ *is an allowed context for the extended environment structure*
(2) $M \mid C_\alpha^{\overline{\Gamma}} \rightarrow^* (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ (M' \mid k_0[\![\delta!\langle\rangle]\!] \mid \mathsf{LReport} \mid \mathsf{GReport}_\alpha)$

If α *is an output action at* k_i *where* $\Gamma \not\vdash k_i : \mathsf{loc}[\mathbf{move}_*]$ *but* $k_i \in \mathcal{T}$ *then* $\mathsf{GReport}_\alpha$ *is the term*

$$k_0[\![r_0!\big\langle v_{(\overline{\Gamma}\mathsf{after}\alpha)_{k_i}}\big\rangle]\!]$$

Otherwise it is

$$k_0[\![r_0!\big\langle v_{(\overline{\Gamma}\mathsf{after}\alpha)_{k_0}}\big\rangle]\!]$$

Proof: (Outline) In one direction the result is straightforward; it is sufficient to prove, by induction on the derivation, that if $(\overline{\Gamma} \triangleright M) \overset{\alpha}{\Longrightarrow} (\overline{\Gamma}' \triangleright M')$ then the action context is allowed and that when run parallel with M the overall system can reach the desired state.

The converse is more difficult and depends on the precise definition of the context. But the crucial point is that in the reduction from $M \mid C_\alpha^{\overline{\Gamma}}$ the subsystem $k_0[\![\delta!\langle\rangle]\!]$ can only be reached if M performs the action α, possibly preceded or followed by some internal actions. □

The usefulness of the components of these contexts which retain the local and global knowledge is apparent from the following result:

Lemma 5 (Extrusion) *Suppose* $\tilde{m}{:}\tilde{\mathrm{E}}$ *is the extension of* $\overline{\Gamma}$ *by the action* α. *Then*

$$\overline{\Gamma}, \overline{(\Gamma_{H,\alpha})_\nabla} \models (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ (M \mid \mathsf{GReport}_\alpha \mid \mathsf{LReport}) \\ \approx_{\mathcal{T}}^{rbc} (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ (N \mid \mathsf{GReport}_\alpha \mid \mathsf{LReport})$$

implies

$$(\overline{\Gamma}\ \mathsf{after}\ \alpha) \models M \approx_{\mathcal{T}}^{rbc} N.$$

Proof: Unfortunately the proof of this result is notationally quite complex and is relegated to the appendix. □

We next give an outline of the completeness proof, which relies heavily on this Extrusion Lemma.

Proposition 19 (Completeness of $\approx_{bis}$ for $\eqsim_{rbc}$) *For any $\mathcal{T}$, $\overline{\Gamma} \models M \eqsim^{rbc}_{\mathcal{T}} N$ implies $\overline{\Gamma} \models M \approx^{\mathcal{T}}_{bis} N$.*

Proof: We define a relation $\mathcal{R}$ such that $\overline{\Gamma} \models M \;\mathcal{R}\; N$ if $\overline{\Gamma} \models M \eqsim^{rbc}_{\mathcal{T}} N$ and show that $\mathcal{R}$ forms a bisimulation. The proposition will then follow by co-induction.

Suppose $\overline{\Gamma} \models M \;\mathcal{R}\; N$ and let $\overline{\Gamma} \rhd M \xrightarrow{\alpha} (\overline{\Gamma}\ \mathsf{after}\ \alpha) \rhd M'$. We must find a matching transition from $(\overline{\Gamma} \rhd N)$. We only outline the *monadic* case, where only single values are transmitted.

The idea of the proof is to use a particular context which mimics the effect of the action α, and also allows us to subsequently compare the residuals of the two systems. This context has the form

$$D^{\overline{\Gamma}}_{\alpha}[-] = (\mathsf{new}\, \delta)\ (\ -\ \mid C^{\overline{\Gamma}}_{\alpha} \mid \mathsf{Flip})$$

where $C^{\overline{\Gamma}}_{\alpha}$ is given in Figure 13 and Flip is the system

$$k_0[\![\delta_{\text{fail}}!\langle\rangle \mid \delta?()\,.\delta_{\text{fail}}?()\,.\delta_{\text{succ}}!\langle\rangle]\!].$$

Intuitively the existence of the barb δ_{fail} at the home-base k_0 indicates that the action has not yet happened, whereas that of δ_{succ} ensures that is has occurred, and has been reported via δ. In the context above this reporting channel δ has been restricted and we have omitted its obvious type.

Using Proposition 18 (the only if implication), we can deduce that $C^{\overline{\Gamma}}_{\alpha} \mid \mathsf{Flip}$ is an allowed context for the extended environment $\overline{\Gamma}, \overline{\Gamma_{H,\alpha_\nabla}}$ and because $\eqsim^{rbc}_{\mathcal{T}}$ is $\mathcal{T}$-contextual we therefore have

$$\overline{\Gamma}, \overline{\Gamma_{H,\alpha_\nabla}} \models D^{\overline{\Gamma}}_{\alpha}[M] \eqsim^{rbc}_{\mathcal{T}} D^{\overline{\Gamma}}_{\alpha}[N]$$

By inspecting the reduction rules of $D^{\overline{\Gamma}}_{\alpha}[M]$ we observe that,

$$D^{\overline{\Gamma}}_{\alpha}[M] \qquad \rightarrow^* (\mathsf{new}\, \tilde{m} : \tilde{\mathrm{E}})\ (M' \mid k_0[\![\delta_{\text{succ}}!\langle\rangle]\!] \mid \mid \mathsf{GReport}_{\alpha} \mid \mathsf{LReport}).$$

where $(\tilde{m} : \tilde{T})$ is the extension of $\overline{\Gamma}$ by α. Let us call this latter system M_0.

This reduction must be matched by a corresponding reduction

$$D^{\overline{\Gamma}}_{\alpha}[N] \rightarrow^* N_0$$

where

$$\overline{\Gamma}, \overline{\Gamma_{H,\alpha_\nabla}} \models M_0 \eqsim^{rbc}_{\mathcal{T}} N_0. \tag{17}$$

However the possible matching reductions are constrained by the barbs of M_0 in the extended environment; it has the barb $\delta_{\text{succ}}@k_0$ but it does not have $\delta_{\text{fail}}@k_0$. Effectively the reduction must have the form

$$D_{\alpha}^{\overline{\Gamma}}[N] \qquad \to^{*} (\mathsf{new}\,\tilde{m} : \tilde{\mathrm{E}})\ (N' \mid k_0[\![\delta_{\text{succ}}!\langle\rangle]\!] \mid \mathsf{GReport}_{\alpha} \mid \mathsf{LReport})$$

for some N'. This in turn implies that there must be a reduction

$$N \mid C_{\alpha}^{\overline{\Gamma}} \to^{*} (\mathsf{new}\,\tilde{m} : \tilde{\mathrm{E}})\ (N' \mid k_0[\![\delta!\langle\rangle]\!] \mid \mathsf{GReport}_{\alpha} \mid \mathsf{LReport}\,.$$

At this stage we can apply Proposition 18 (in the opposite direction) to obtain a required weak typed action

$$\overline{\Gamma} \triangleright N \stackrel{\alpha}{\Longrightarrow} \overline{\Gamma}' \triangleright N'.$$

However we must establish that

$$(\overline{\Gamma}\ \mathsf{after}\ \alpha) \models M' \mathrel{\approxeq^{rbc}_{\mathcal{T}}} N'. \tag{18}$$

It is easy to remove the success barbs from (17) above to obtain

$$\begin{aligned}\overline{\Gamma}, \overline{\Gamma_{H,\alpha_{\nabla}}} \models &(\mathsf{new}\,\tilde{m}{:}\tilde{\mathrm{E}})\ (M' \mid \mathsf{GReport}_{\alpha} \mid \mathsf{LReport}) \\ &\approxeq^{rbc}_{\mathcal{T}} (\mathsf{new}\,\tilde{m}{:}\tilde{\mathrm{E}})\ (N' \mid \mathsf{GReport}_{\alpha} \mid \mathsf{LReport}).\end{aligned}$$

However this is precisely the premise in the Extrusion Lemma above, Lemma 5, which gives the required (18). □

Finally, we can state the final result of the paper which follows from Propositions 16 and 19.

Theorem 5 (Full abstraction) *In* Dpi *with restricted mobility*

$$\overline{\Gamma} \models M \approxeq^{\mathcal{T}}_{rbc} N \qquad \textit{if and only if} \qquad \overline{\Gamma} \models M \approx^{\mathcal{T}}_{bis} N.$$

Note that Theorem 3 and Theorem 4 can be recovered as an instance of this result by considering every location type to contain the move capability. In this case the extra labelled transitions and extra structure we require in configurations becomes redundant.

6 Conclusion

We have presented two labelled transition systems for which bisimilarity coincides with a natural notion of contextual equivalence for distributed systems in which types are used to control access rights and a primitive form of mobility. As in [10,2], 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.

The set of basic types used for DPI, those in Figure 4, is a slight generalisation of those in [11]; the novelty is the introduction of the new category of *registered names*, which we feel clarifies the sharing of resource names among different locations. To discuss this point consider the following system, written in the syntax of the current paper:

$$l[\![(\mathsf{newc}\, c : \mathrm{C})\ \ (\mathsf{newloc}\, k : \mathsf{loc}[c : \mathrm{D}])\ \ \mathsf{with}\ P\ \mathsf{in}\ Q]\!]$$

Here a new channel c is declared local to l, yet when a new location k is generated it is also allowed to install at k another local channel named c. The type inference of [11] allows such systems to be well-typed, even if there is no relationship between the types C and D; one could allow read at type **bool** and the other write at type **int**. Here this system is not well-typed as l and k can only have a common resource c if it has already been declared as a *registered name*. Moreover in such circumstances the types C and D would have to be consistent, as they both would be supertypes of the type of the registered name.

The mobility control presented here is simply a first step towards identifying the nature of contextual equivalence in this setting. An obvious generalisation would be to introduce more specific move capabilities, of the form $\mathbf{move}_{\mathrm{S}}$, where S is a set of locations. Then having a location l at with this capability would endow all locations mentioned in S with immigration rights to l; this is essentially what was used informally in Example 6. In another vein, we can investigate how the parameter $\mathcal{T}$ affects equivalence. The use we make of it here is to allow testing at any (initially) known location. At the other extreme we could fix $\mathcal{T}$ to be empty. This would only allow tests to be placed at *fresh* locations — thereby changing the nature of observability and simplifying the semantics considerably. This may be the appropriate choice for testing equivalences [8].

A related, but different, approach to mobility control, would be to base migration rights not on the source location, as we do here, but on the potential behaviour of the incoming agent. This would involve developing a fine-grained type system for agent behaviour, and then restricting access to agents which satisfy some published access types. Type systems along these lines have already been developed for the π-calculus, [22], and have recently been extended to DPI, [9].

There has been a great deal of interest in modelling distributed systems using calculi in recent years, [19,7,1,5,21,11,4]. 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 [7,1,13]. Proofs of correctness of protocols or language translations have been carried out with respect to these contextual equivalences. But, as far as we know, the only existing examples of bisimulation-based characterisations of contextual equivalences in a distributed setting are found in [14,15,3], for versions of ambients. Recently bisimulations have been proposed for the Seal Calculus, [6]. The resulting equivalence is shown to be sound with respect to contextual equivalence, but is not complete; in general it distinguishes systems which are identified by the contextual equivalence. The work in [20] 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.

A The Extrusion Lemma

To prove the Extrusion lemma we need to formulate a more general statement, involving environment structures which are consistent with $\overline{\Gamma}$, the current knowledge about the system. It uses a general environment structure Δ and presupposes that the channels r_i used in the main body of the text are typed to support the reporting of the components of Δ. So we use Γ_D to denote the following list of environments:

$$k_0{:}\mathsf{loc},\quad \mathbf{move}@k_0,\quad r_0{:}\mathsf{rw}\langle \Delta_{k_0}\rangle@k_0$$
$$r_1{:}\mathsf{rw}\langle \Delta_{k_1}\rangle@k_1, \ldots, r_n{:}\mathsf{rw}\langle \Delta_{k_n}\rangle@k_n$$

Lemma 6 (General Extrusion) *Let $\overline{\Delta}$ be an environment structure such that $\overline{\Gamma} \sqcap \overline{\Delta} \triangleright M$ and $\overline{\Gamma} \sqcap \overline{\Delta} \triangleright N$ are configurations. Then*

$$\overline{\Gamma}, \overline{(\Gamma_D)}_{\triangledown} \models (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ \ (M \mid k_0[\![r_0!\langle v_\Delta\rangle]\!] \mid \textstyle\prod_{k_j \in \mathcal{T}} k_j[\![r_j!\left\langle v_{\Delta_{k_j}}\right\rangle]\!])$$
$$\approx^{rbc}_{\mathcal{T}}$$
$$(\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ \ (N \mid k_0[\![r_0!\langle v_\Delta\rangle]\!] \mid \textstyle\prod_{k_j \in \mathcal{T}} k_j[\![r_j!\left\langle v_{\Delta_{k_j}}\right\rangle]\!])$$

implies

$$\overline{\Gamma} \sqcap \overline{\Delta} \models M \approx^{rbc}_{\mathcal{T}} N.$$

Proof: We define a family of relations as follows: Let $\mathcal{R}_{\overline{\Gamma}\sqcap\overline{\Delta}}$ be the set of all pairs of systems (M, N) such that

$$\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown} \models (\mathsf{new}\,\tilde{m}{:}\tilde{\mathrm{E}})\ \ (M \mid k_0[\![r_0!\langle v_\Delta\rangle]\!] \mid \prod_{k_j \in \mathcal{T}} k_j[\![r_j!\left\langle v_{\Delta_{k_j}}\right\rangle]\!]) \approx^{rbc}_{\mathcal{T}}$$
$$(\mathsf{new}\,\tilde{m}{:}\tilde{\mathrm{E}})\ \ (N \mid k_0[\![r_0!\langle v_\Delta\rangle]\!] \mid \prod_{k_j \in \mathcal{T}} k_j[\![r_j!\left\langle v_{\Delta_{k_j}}\right\rangle]\!])$$

When it is apparent from the context we will refer to these systems simply as A, B respectively.

The result will follow if we prove that $\mathcal{R}_{\overline{\Gamma}\sqcap\overline{\Delta}}$ is

- reduction closed
- $\mathcal{T}$-contextual
- m-barb preserving

We outline some of the required proofs.

First let us consider the third requirement, *m-barb preserving*; From $\overline{\Gamma} \sqcap \overline{\Delta} \vdash M \Downarrow^{\text{m-barb}} a@k$ we need to show that $\overline{\Gamma} \sqcap \overline{\Delta} \vdash N \Downarrow^{\text{m-barb}} a@k$. We know A and B have the same barbs but the problem is that $a@k$ may not be a barb of A because a (or even k) may be restricted via the occurrence of $(\mathsf{new}\,\tilde{m} : \tilde{M})$ in A. We overcome this problem by placing A and B in an appropriate context.

Let K_{barb}be the system

$$k_0[\![r_0?(X : (\Delta)).\mathsf{goto}\,k.a?().\mathsf{goto}\,k_0.w!\langle\rangle\ \{\!|^X\!/\!_{v_\Delta}|\!\}]\!]$$

where w is a fresh channel. It is easy to check that

$$\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown}, w : \mathsf{rw}\langle\rangle@k_0 \models A \mid K_{\text{barb}} \approx^{rbc}_{\mathcal{T}} B \mid K_{\text{barb}}$$

and that

$$\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown}, w : \mathsf{rw}\langle\rangle@k_0 \vdash (A \mid K_{\text{barb}}) \Downarrow^{\text{m-barb}} w@k_0$$

Effectively we have turned the barb $a@k$ of M into the barb $w@k_0$ of A in the context of K_{barb}.

By the m-barb preservation property of $\approx^{rbc}_{\mathcal{T}}$ (and Weakening), we have

$$\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown}, w : \mathsf{rw}\langle\rangle@k_0 \vdash (B \mid K_{\text{barb}}) \Downarrow^{\text{m-barb}} w@k_0$$

As w is fresh this could only have arisen by interaction with N along channel a at k, that is $N \rightarrow^* (N' \mid k[\![a!\langle\rangle \ldots. \mid \ldots]\!])$. This suffices to conclude that $\overline{\Gamma} \sqcap \overline{\Delta} \vdash N \Downarrow^{\text{m-barb}} a@k$, as required.

The other requirements are proved in a similar manner. Let us look briefly at perhaps the most difficult one, namely that $\mathcal{R}$ is preserved by suitable parallel compositions. So suppose $\overline{\Gamma} \sqcap \overline{\Delta} \models M \mathcal{R} N$ and $\overline{\Gamma} \sqcap \overline{\Delta} \vdash k[\![P]\!]$. We must show that $\overline{\Gamma} \sqcap \overline{\Delta} \models M \mid k[\![P]\!] \mathcal{R} N \mid k[\![P]\!]$ whenever either $\overline{\Gamma} \sqcap \overline{\Delta} \vdash k : \mathsf{loc}[\mathbf{move}_*]$ or $k \in \mathcal{T}$. The details only vary slightly between the cases.

First suppose $\overline{\Gamma} \sqcap \overline{\Delta} \vdash k : \mathsf{loc}[\mathbf{move}_*]$. Here we use the context

$$K_{\text{move}} = k_0[\![r_0?(X : (\Delta)).(\mathsf{goto}\, k.P)\{\!\!\{^{X}\!/\!_{v_\Delta}\}\!\!\} \mid r_0'!\langle X \rangle]\!]$$

where r_0' is fresh. Since $\overline{\Gamma} \sqcap \overline{\Delta} \vdash k[\![P]\!]$ and we abstract over the values v_Δ, it is easy to check that

$$\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown}, r_0' : \mathsf{rw}\langle\Delta\rangle@k_0 \vdash K_{\text{move}}$$

and therefore by contextuality we have

$$\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown}, r_0' : \mathsf{rw}\langle\Delta\rangle@k_0 \models (\mathsf{new}\, r_0)\ (A \mid K_{\text{move}}) \approx^{rbc}_{\mathcal{T}} (\mathsf{new}\, r_0)\ (B \mid K_{\text{move}})$$

where we have omitted the obvious declaration type on the restricted channel r_0, namely $\mathsf{rw}\langle\Delta\rangle@k_0$.

A simple argument gives the identity

$$\begin{aligned}&\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown}, r_0' : \mathsf{rw}\langle(\Delta)\rangle@k_0 \models (\mathsf{new}\, r_0)\ (A \mid K_{\text{move}}) \approx^{rbc}_{\mathcal{T}} \\ &\qquad (\mathsf{new}\, \tilde{m})\ ((M \mid k[\![P]\!]) \mid k_0[\![r_0'!\langle v_\Delta \rangle]\!] \mid \prod_{k_j \in \mathcal{T}} k_j[\![r_j!\left\langle v_{\Delta_{k_j}} \right\rangle]\!])\end{aligned}$$

and similar one relating B with N.

As r_0 and r_0' are both fresh and have the same type we can conclude that

$$\begin{aligned}\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown} \models (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ &((M \mid k[\![P]\!]) \mid k_0[\![r_0!\langle v_\Delta \rangle]\!] \mid \textstyle\prod_{k_j \in \mathcal{S}} k_j[\![r_j!\left\langle v_{\Delta_{k_j}} \right\rangle]\!]) \\ &\approx^{rbc}_{\mathcal{T}} \\ (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ &((N \mid k[\![P]\!]) \mid k_0[\![r_0!\langle v_\Delta \rangle]\!] \mid \textstyle\prod_{k_j \in \mathcal{S}} k_j[\![r_j!\left\langle v_{\Delta_{k_j}} \right\rangle]\!]).\end{aligned}$$

This suffices to witness

$$\overline{\Gamma} \sqcap \overline{\Delta} \models (M \mid k[\![P]\!]) \mathcal{R} (N \mid k[\![P]\!])$$

as required.

The structure of the proof in the second case, when $k \in \mathcal{T}$, say $k = k_i$ is similar but we use the context $K_{\text{local}} = k_i[\![r_i?(X : \Delta_{k_i})]\!].(P\{\!\!\{^{X}\!/\!_{v_{\Delta_{k_i}}}\}\!\!\} \mid r_i'!\langle X \rangle)$.
□

Let us now see how the version of Extrusion required for the Completeness proof can be obtained from this general result.

Corollary 2 *Suppose $\tilde{m}$:$\tilde{\mathrm{E}}$ is the extension of $\overline{\Gamma}$ by the action α. Then*

$$\begin{aligned}\overline{\Gamma}, \overline{(\Gamma_D)_\triangledown} \models (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ (M' \mid \mathsf{GReport}_\alpha \mid \mathsf{LReport} \\ \approx^{rbc}_{\mathcal{T}} (\mathsf{new}\, \tilde{m}{:}\tilde{\mathrm{E}})\ (N' \mid \mathsf{GReport}_\alpha \mid \mathsf{LReport}\end{aligned}$$

implies

$$(\overline{\Gamma}\ \mathsf{after}\ \alpha) \models M' \approx^{rbc}_{\mathcal{T}} N'$$

Proof: The proof depends on instantiating the environment structure $\overline{\Delta}$ in the more general result. It suffices to let this be $\overline{(\Gamma\ \mathsf{after}\ \alpha)}$. Note that in all cases except one this instantiation gives $\Gamma_{k_i} <: \Delta_{k_i}$ for each i. The exception is the troublesome case when α is an output at some location k_i to which we do not have migration rights but which is in $\mathcal{T}$; In this case we do not have $\Gamma_{k_0} <: \Delta_{k_0}$ but only the weaker statement $\Gamma_{k_0} \sqcap \Delta_{k_0}$. Hence the requirement for the more general extrusion lemma. □

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] M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication and mobility control in boxed ambients. Accepted for publication in the Journal of Information & Computation. 2003.

[4] 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.

[5] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, June 2000.

[6] G. Castagna and F. Zappa Nardelli. The Seal calculus revisited: Contextual equivalences and bisimilarity. In *Proceedings of FSTTCS*, Lecture Notes in Computer Science, 2002.

[7] 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.

[8] M. Hennessy. *Algebraic Theory of Processes.* The MIT Press, Cambridge, Mass., 1988.

[9] M. Hennessy, J. Rathke, and N. Yoshida. Safedpi:a language for controlling mobile code. Computer Science Report 02:2003, University of Sussex, 2003.

[10] Matthew Hennessy and Julian Rathke. Typed behavioural equivalences for processes in the presence of subtyping. In James Harland, editor, *Electronic Notes in Theoretical Computer Science*, volume 61. Elsevier Science Publishers, 2002. To appear in *Mathematical Structures in Computer Science.*

[11] Matthew Hennessy and James Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173:82–120, 2002.

[12] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

[13] M. Merro, J. Kleist, and U. Nestmann. Mobile Objects as Mobile Processes. *Journal of Information and Computation*, 177(2):195–241, 2002.

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

[15] Massimo Merro and Francesco Zappa Nardelli. Bisimulation proof techniques for mobile ambients. In *Proc. 30th International Colloquium on Automata, Languages, and Programming (ICALP 2003), Eindhoven*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[16] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[17] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[18] Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–454, 1996. Extended abstract in LICS '93.

[19] Peter Sewell. Global/local subtyping and capability inference for a distributed pi-calculus. In *ICALP 98*, volume 1443 of *LNCS.* Springer, 1998.

[20] Asis Unypoth 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.

[21] 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.

[22] Nobuko Yoshida and Matthew Hennessy. Assigning types to processes. *Information and Computation*, 172:82–120, 2002.