Computer Science Report

Proof methodologies for behavioural equivalence in Distributed picalculus

Alberto Ciaffaglione
Matthew Hennessy
Julian Rathke

Report 2005:03

April 2005

Department of Informatics
University of Sussex
Brighton BN1 9QH

ISSN 1350–3170

Proof methodologies for behavioural equivalence in Distributed picalculus

Alberto Ciaffaglione, Matthew Hennessy and Julian Rathke

Abstract. We focus on techniques for proving behavioural equivalence between systems in D

categories, for systems, and agents. A typical system takes the form

$$(\mathsf{new}\, e : \mathsf{E})(l\llbracket P\rrbracket \;\mathrm{j}\; k\llbracket Q\rrbracket) \;\mathrm{j}\; l\llbracket R\rrbracket$$

This represents a system with two sites, *l* and *k*, with the agents *P* and *R* running at the former and *Q* at the latter; moreover *P* and *Q*, although executing at different sites, share some private information, *e*, of type E. The syntax for agents, or processes, is an extension of that of the picalculus [SW01]. There are input and output on local channels, parallelism, matching of values, iteration, and a migration construct. For example, in the system

$$l\llbracket P \;\mathrm{j}\; \mathsf{goto}\, k.Q\rrbracket \;\;\mathrm{j}\;\; k\llbracket R\rrbracket$$

the process *Q* can migrate from *l* to *k*, leading to the resulting system

$$l\llbracket P\rrbracket \;\;\mathrm{j}\;\; k\llbracket Q \;\mathrm{j}\; R\rrbracket$$

Finally, processes have the ability to create new instances of names (channels, newc, and sites, newloc); their declaration types dictate the use to which these will be put. Finall7. crer(l)]TJ/F14 9.96 Tf1s These,5.7.TJ/f 7.1 These,

The values, *V*, communicated along channels consist of tuples of *simple values*, *v*[SW8 goto[(S V)]8 9:onTJ/F1 9occurr(pic Tf 6.85 0 TD[(,)42 0 TTD[(l)]TJ/F14 9.96 Tf 2.

Q of tuple76.64 0 TD[(l)]TJ/F14 9.96 Tf4ormer and

ThesW8 TD[(l)]TJ/F14 9.96 Tf(e)TJ/F14 9.96TD[.949)(w)]TJ/F8 9.96 Tf 18.82 0 TD[(e]T

Base Types:	$\textbf{base} ::= \textbf{int} \;j\; \textbf{bool} \;j\; \textbf{unit} \;j\; > \;j\; :::$
Value Types:	$\mathsf{A} ::= \textbf{base} \;j\; \mathsf{C} \;j\; \mathsf{C}@\mathsf{loc} \;j\; \mathsf{K}$
Local Channel types:	$\mathsf{C} ::= \mathsf{rhTi} \;j\; \mathsf{whTi} \;j\; \mathsf{rwhTi}$
Location Types:	$\mathsf{K} ::= \mathsf{loc}[c_1 : \mathsf{C}_1; :::; c_n : \mathsf{C}_n];\; n \quad 0$
	(provided $c_i = c_j$ implies $i = j$)
Transmission Types:	$\mathsf{T} ::= (\mathsf{A}_1; :::; \mathsf{A}_n);\, n \quad 0$

Figure 2. Types for Dpi - informal

This generates a new reply channel, r, at the declaration type R, and awaits input on this channel to be printed. Concurrently, it sends to the server site an agent, which sends to the request channel the tuple consisting of some value, v_c, hopefully an integer, and the reply address, $r@c$. Then, running the combined system

$$S \,j\, C \tag{1}$$

should result in a boolean being printed at the client's site, the value of which is determined by the primality of v_c.

2.2 Typing

Dpi is a capability based language, in the sense that the behaviour of processes depends on the capabilities the various entities have received in their environment. Formally, these capabilities are represented as types, and the various categories of types we use are given in Figure 2. Apart from the standard base types, and the special *top* type >, the main ones are

local channel types: these are ranged over by C and can take the form rwhTi, giving the ability to both read and write values of type T, or the restricted supertypes rhTi and whTi;

non-local channel types: these take the form C@loc, and a value of this type is a structured value, $c@l$;

location types: these take the form $\mathsf{loc}[c_1 : \mathsf{C}_1; :::; c_n : \mathsf{C}_n]$; receiving a value l of this type gives access to the channels, or resources, c_i at type C_i, for $1 \quad i \quad n$.

In this overview we omit one further category of types, that of *registered names*, as they play no part in the current paper; as usual, the reader is referred to [HMR04] for an explanation of their role in ensuring consistency between the

The rules for typing agents are more or less borrowed from the picalculus [PS00], with the addition of a rule for migration. For example, (local) input and output are handled by the rules

(ty-out)

$$\frac{\vdash_w V : \mathsf{T} \qquad \vdash_w P \qquad \vdash_w u : \mathsf{w}\langle \mathsf{T} \rangle}{\vdash_w u!\langle V \rangle P}$$

(ty-in)

that is required of in order to type both the server and the client is to let S, the type associated with the request channel, to be rwh **int**; wh**bool**i@loc i.

There is an interesting point to be made here. The client generates the reply channel r with both read and write capabilities; only the latter is sent to the server, via req, and the former is retained for internal use. This use of restricted capabilities provides a certain level of protection to the client, as it knows that the reply from the server can not be usurped by any other client.

2.3 Behaviour

The behaviour of a system, that is the ability of its agents to interact with other agents, depends on the knowledge these agents have of each others capabilities. In the example just discussed we have seen the client generating a reply channel with two capabilities, but only making one of these externally available; indeed, the proper functioning of the client/server interaction depends on such decisions.

Definition 2.1 (Configurations). A *configuration* consists of a pair I BM, where

- I is a type environment which associates some type to every free name in M
- there is a type environment such that ` M and <: I

This latter requirement means that if I can assign a type T_{I} to a name n, then can assign a type T such that T <: T_{I}. Again, viewing types as sets of capabilities, this means that T_{I}, representing the knowledge of the external user, is a subset of T , the actual set of capabilities used to type the system M.

So we define the behaviour in terms of actions over configurations; these are of the form

$$\mathsf{I} \; \mathsf{B} \, M \quad ! \quad \mathsf{I}^0 \, \mathsf{B} \, M^0 \qquad (2)$$

where the label can take any of the following forms

- : an internal action, requiring no participation by the user;
- $(\tilde{e} : \tilde{\mathsf{E}})k{:}a?V$: the input of value V along the channel a, located at the site k. The bound names in $(\tilde{e})$ are freshly generated by the user;
- $(\tilde{e} : \tilde{\mathsf{E}})k{:}a!V$: the output of value V along the channel a, located at the site k. The bound names in $(\tilde{e})$ are freshly generated by the environment.

The rules for defining these actions are given in Figure 3 and Figure 4, a slightly different but equivalent formulation to that given in [HMR04]. The guiding principle for (2) to happen, is that M must be able to perform the action , and the user must have, in I, the capability to participate in the action. The rules use some new notation for looking up the types associated with channels in environments: the partial functions $\mathsf{I}^r(k; a)$ and $\mathsf{I}^w(k; a)$ return the read, respectively write, type associated with the channel a at the location k in I (of course these

I B k[[?(X bd]ampetion[with fr ou 5.531.6 TD[(

(m-in)

$$\frac{\mathrm{I}^{w}(k;a)\ \#\qquad \mathrm{I}\ `_{k}\ V : \mathrm{I}^{w}(k;a)}{\mathrm{I}\ \mathrm{B}\ k[\![a?(X)\,R]\!]\ \overset{k:a?V}{!}\ \mathrm{I}\ \mathrm{B}\ k[\![R\,\mathrm{f}^{V}\!=\!_{X}\mathrm{g}]\!]}$$

(m-weak)

$$\frac{\mathrm{I};\mathrm{h}e:\mathrm{E}\mathrm{i}\ \mathrm{B}\ M\ \overset{(\tilde{d}:\tilde{\mathrm{D}})k:a?V}{!}\ \mathrm{I}^{0}\ \mathrm{B}\ M^{0}}{\mathrm{I}\ \mathrm{B}\ M\ \overset{(e:\mathrm{E}\ \tilde{d}:\tilde{\mathrm{D}})k:a?V}{!}\ \mathrm{I}^{0}\ \mathrm{B}\ M^{0}}\quad \mathrm{bn}(e)\not\in \mathrm{I}$$

(m-out)

$$\frac{\mathrm{I}^{r}(k;a)\ \#}{\mathrm{I}\ \mathrm{B}\ k[\![a!\mathrm{h}V\mathrm{i}\ P]\!]\ \overset{k:a!V}{!}\ \mathrm{I};\ \mathrm{h}V:\mathrm{I}^{r}(k;a)\mathrm{i}_{@}k\ \mathrm{B}\ k[\![P]\!]}$$

(m-open)

$$\mathrm{I};\mathrm{h}e:>\mathrm{i}\ \mathrm{B}\ M\ \overset{(\tilde{d}:\tilde{\mathrm{D}})k:a!V}{!}\ \mathrm{I}^{0}\ \mathrm{B}\ ($$

[EF8.6]TJ/F19.96 Tf 7.193 0 TD[(()T.00

2 [4] is extended actions /F1 9.96 Tf 5.215

2 -188.29 -29.38 158 TD[(D)]TJ 21Tf 4.5n4

(m-weak), allows us to derive the following action from the server S

$$\Gamma \triangleright S \xrightarrow{(r@c:\mathsf{R})\alpha} \Gamma, r@c : \mathsf{R} \triangleright s[\![\mathsf{goto}\ c.r!\langle isprime(v_c)\rangle\ \mathsf{stop}]\!] \tag{3}$$

where α is the input action $s.\mathsf{req}?(v_c, r@c)$, because

$$\Gamma, r@c : \mathsf{R} \triangleright S \xrightarrow{\alpha} \Gamma, r@c : \mathsf{R} \triangleright s[\![\mathsf{goto}\ c.r!\langle isprime(v_c)\rangle\ \mathsf{stop}]\!]$$

Similarly, (m-out) requires Γ to have a *read* capability on a at k, in order for $k[\![a!\langle V\rangle\, P]\!]$ to be able to perform the obvious output; note that here the current knowledge of the user, Γ, is augmented by whatever new knowledge which can be gleaned from the received value V. Intuitively, $\langle V : \mathsf{T}\rangle@k$ decomposes the value V, relative to the type T, from the standpoint of k; this last only comes into play when V contains instances of local channels, which are then interpreted as channels at k. But the important point in (m-out) is that the type at which V is added to Γ is $\Gamma^r(k, a)$, the reception type that the user currently has on a at k. Thus (m-open) allows us to deduce

$$\Gamma \triangleright (\mathsf{new}\, r@c : \mathsf{R})\, s[\![\mathsf{req}!\langle v_c, r@c\rangle\ \mathsf{stop}]\!] \xrightarrow{(r@c:\mathsf{R})\alpha} \Gamma, r@c : \mathsf{R}_w \triangleright s[\![\mathsf{stop}]\!] \tag{4}$$

where α is the output action $s.\mathsf{req}!\langle v_c, r@c\rangle$, because with (m-out) we can derive

$$\Gamma, r@c : \top \triangleright s[\![\mathsf{req}!\langle v_c, r@c\rangle\ \mathsf{stop}]\!] \xrightarrow{\alpha} \Gamma, r@c : \mathsf{R}_w \triangleright s[\![\mathsf{stop}]\!]$$

The use of $\top$ is simply to ensure that we have a valid configuration; but note that the user has gained only the restricted capability R_w on the new channel r, rather than the more liberal declaration capability R, because the former is the type at which the user can receive values along req.

The rules for the internal actions are given in Figure 4, and most are straightforward. We have labelled some as β-actions, which will be useful in the next section; but for the moment these labels can be ignored. The only interesting rule is (m-comm), which formalises *communication*. Note that, in the hypotheses of both variations, arbitrary user environments, Γ_1 and Γ_2, are allowed. This may be surprising at first, but intuitively τ-actions should be independent of all external knowledge. For example, we can use (3) and (4) above to derive

$$\begin{aligned}&\Gamma \triangleright S \mid (\mathsf{new}\, r@c : \mathsf{R})\, s[\![\mathsf{req}!\langle v_c, r@c\rangle\ \mathsf{stop}]\!] \xrightarrow{\tau}\\ &\Gamma \triangleright (\mathsf{new}\, r@c : \mathsf{R})\, s[\![\mathsf{goto}\ c.r!\langle isprime(v_c)\rangle\ \mathsf{stop}]\!] \mid s[\![\mathsf{stop}]\!]\end{aligned}$$

for an arbitrary Γ.

We now have a labelled transition system in which the states are configurations, and we can apply the standard definition of (weak) bisimulation.

(m-comm)

$$\frac{I_1\ \mathrm{B}\ M \overset{(\tilde{e}:\tilde{E})k{:}a?V}{!} I_1^0\ \mathrm{B}\ M^0 \qquad I_2\ \mathrm{B}\ N \overset{(\tilde{e}:\tilde{E})k{:}a!V}{!} I_2^0\ \mathrm{B}\ N^0}{I\ \mathrm{B}\ M\ \mathrm{j}\ N \quad ! \quad I\ \mathrm{B}\ (\mathrm{new}\,\mathbf{e} : \hat{\mathrm{E}})(M^0\ \mathrm{j}\ N^0)}$$

(m-comm)

$$\frac{I_1\ \mathrm{B}\ M \overset{(\tilde{e}:\tilde{E})k{:}a!V}{!} I_1^0\ \mathrm{B}\ M^0 \qquad I_2\ \mathrm{B}\ N \overset{(\tilde{e}:\tilde{E})k{:}a?V}{!} I_2^0\ \mathrm{B}\ N^0}{I\ \mathrm{B}\ M\ \mathrm{j}\ N \quad ! \quad I\ \mathrm{B}\ (\mathrm{new}\,\mathbf{e} : \hat{\mathrm{E}})(M^0\ \mathrm{j}\ N^0)}$$

(m-move)

$$I\ \mathrm{B}\ k\llbracket \mathrm{goto}\ l{:}P \rrbracket \quad ! \quad I\ \mathrm{B}\ l\llbracket P \rrbracket$$

(m-c:create)

$$I\ \mathrm{B}\ k\llbracket (\mathrm{newc}\ c : \mathrm{C})\ \ P \rrbracket \quad ! \quad I\ \mathrm{B}\ (\mathrm{new}\ c@k : \mathrm{C})\ k\llbracket P \rrbracket$$

(m-l:create)

$$I\ \mathrm{B}\ k\llbracket (\mathrm{newloc}\ l : \mathrm{L})\ \ P \rrbracket \quad ! \quad I\ \mathrm{B}\ (\mathrm{new}\ l : \mathrm{L})\ k\llbracket P \rrbracket$$

(m-eq)

$$I\ \mathrm{B}\ k\llbracket \mathrm{if}\ v = v\ \mathrm{then}\ P\ \mathrm{else}\ Q \rrbracket \quad ! \quad I\ \mathrm{B}\ k\llbracket P \rrbracket$$

(m-neq)

$$I\ \mathrm{B}\ k\llbracket \mathrm{if}\ v_1 = v_2\ \mathrm{then}\ P\ \mathrm{else}\ Q \rrbracket \quad ! \quad I\ \mathrm{B}\ k\llbracket Q \rrbracket \qquad (v_1 \neq v_2)$$

(m-split)

$$I\ \mathrm{B}\ k\llbracket P\ \mathrm{j}\ Q \rrbracket \quad ! \quad I\ \mathrm{B}\ k\llbracket P \rrbracket\ \mathrm{j}\ k\llbracket Q \rrbracket$$

(m-unwind)

$$I\ \mathrm{B}\ k\llbracket\ \ P \rrbracket \quad ! \quad I\ \mathrm{B}\ k\llbracket\ \ P\ \mathrm{j}\ P \rrbracket$$

Figure 4. Internal actions-in-context for Dpi

Definition 2.2 (Bisimulations). We say a binary relation over configurations is a *bisimulation* if both it, and its inverse, satisfy the following transfer property

$$\begin{tikzcd}
(\mathsf{I}_M \, \mathsf{B} \, M) \;\; \mathsf{R} \;\; (\mathsf{I}_N \, \mathsf{B} \, N) \arrow[d, dash, "?"'] & \text{implies} & (\mathsf{I}_M \, \mathsf{B} \, M) \;\; \mathsf{R} \;\; (\mathsf{I}_N \, \mathsf{B} \, N) \arrow[d, Rightarrow, "\hat{}"'] \\
(\mathsf{I}_{M^0} \, \mathsf{B} \, M^0) & & (\mathsf{I}_{M^0} \, \mathsf{B} \, M^0) \;\; \mathsf{R} \;\; (\mathsf{I}_{N^0} \, \mathsf{B} \, N^0)
\end{tikzcd}$$

Here we use standard notation, see [MPW92], with =) representing ! ! ! , and $\hat{=})$ meaning ! , if is , and =) otherwise. This allows a single internal move to be matched by zero or more internal moves.

We let $_{bis}$ denote the largest bisimulation between configurations.

Rather than writing (I B M) $_{bis}$ (I B N); we use the more suggestive notation

$$\mathsf{I} \models M \quad _{bis} N$$

This can be viewed as a relation between systems, parameterised over type environments which represent user's knowledge of the systems' capabilities.

It is this bisimilarity $_{bis}$ which is the object of our study: we aim to show that, despite the complexity of its definition, tractable proof techniques can be developed for it.

Finally, we should remark this is not an arbitrarily chosen version of bisimulation equivalence. In [HMR04] its definition is justified in detail: it is shown to be, in some sense, the largest reasonable typed equivalence between Dpi systems.

3 Proof techniques

The basic method for showing that two systems M and N are equivalent, relative to an environment I, is to exhibit a parameterised relation R such that I $\models$ MRN, and demonstrate that it satisfies the requirements of being a bisimulation. In this section we give a number of auxiliary methods, which can considerably relieve the burden of exhibiting such relations.

The following Theorem is proved in [HMR04], and justifies a form of contextual reasoning.

Theorem 3.1 (Contextuality).

- I $\models$ M $_{bis}$ N *and* I ` O *imply* I $\models$ M j O $_{bis}$ N j O
- I ; he : E i $\models$ M $_{bis}$ N *implies* I $\models$ (new e : E) M $_{bis}$ (new e : E) N

We can also manipulate system descriptions. Let be the least equivalence relation which satisfies the rules in Figure 5, and is preserved by the constructs j and (new e : E)( ); this is referred to as *structural equivalence*.

(s-extr) $(\mathsf{new}\, e : \mathsf{E})(M \mathsf{j} N)$ M j $(\mathsf{new}\, e : \mathsf{E})\, N$ if $\mathsf{bn}(e) \notin \mathsf{fn}(M)$

(s-com) $M \mathsf{j} N$

[illegible]

-actions, include

$$\begin{aligned} k[\![P \ \mathrm{j} \ Q]\!] & \quad {}_{bis} \ k[\![P]\!] \ \mathrm{j} \ k[\![Q]\!] \\ k[\![\mathsf{goto} \ l{:}P]\!] & \quad {}_{bis} \ l[\![P]\!] \\ k[\![(\mathsf{newc} \ c : \mathsf{C}) \ \ P]\!] & \quad {}_{bis} \ (\mathsf{new} \ c_{@}k : \mathsf{C}) \ k[\![P]\!] \end{aligned}$$

But these -labelled internal actions also provide us with a very powerful method for approximating bisimulations, in the spirit of [JR04].

Definition 3.5 (Bisimulations up-to-). A binary relation between configurations is said to be a *bisimulation up-to-* if it satisfies the following transfer properties

$$\begin{tikzcd}
(\mathsf{I}_M \ \mathsf{B} \ M) \ \ \mathsf{R} \ \ (\mathsf{I}_N \ \mathsf{B} \ N) \arrow[d, "?"'] & & (\mathsf{I}_M \ \mathsf{B} \ M) & \mathsf{R} & (\mathsf{I}_N \ \mathsf{B} \ N) \\
(\mathsf{I}_{M^0} \ \mathsf{B} \ M^0) & \text{implies} & (\mathsf{I}_{M^0} \ \mathsf{B} \ M^0) \ \ \mathsf{A}_l & &
\end{tikzcd}$$

Proof: We leave to the reader to check that the relation ($_{bis}$ R $_{bis}$) is a bisimulation over configurations. The key properties for establishing this are the two inclusions ! $_{bis}$ (Proposition 3.4) and $_{bis}$ (Proposition 3.2), Lemma 3.3 and transitivity, in Definition 3.5, of both A_l (due to Lemma 3.6) and A_r. The result then follows, since ($_{bis}$ R $_{bis}$) trivially contains R.

4 Crossing a firewall

Let us consider the *firewall* example, first proposed in [CG98] and studied at length in [GC99, LS00, MN03] within versions of Mobile Ambients. Intuitively, a firewall is a domain to which access is restricted: only agents which are permitted, in some sense, by the firewall are allowed in. A simple example takes the form

$$F \;(\;\; (\mathsf{new}\; f : \mathsf{F})\; f[\![P \;\mathrm{j}\;\; \mathsf{goto}\, a{:}\mathsf{tell}!\mathrm{h}f\mathrm{i}]\!]$$

Here f is the name of the firewall, which is created with the capabilities described in the location type F, and P is some code which maintains the internal business of the firewall. A typical example of the capabilities could be given by

$$\mathsf{F} = \mathsf{loc}[\mathsf{info} : \mathsf{rwhIi};\;\; \mathsf{req} : \mathsf{rwhRi}]$$

which allow reading to and writing from two resources info and req in f. Then P could, for example, maintain appropriate services at the resources; of course, it would also be able to use non-local resources it knows about in its current environment.

The existence of the firewall is made known only to another domain, a, via the information channel tell located there. An example is the following

$$A \;(\;\; a[\![R \,\mathrm{j}\, \mathsf{tell}?(x)\, \mathsf{goto}\; x{:}Q]\!]$$

where a is informed of f by inputing on the local channel tell. If we consider an arbitrary type environment , we have the execution

$$\mathsf{B}\; F \,\mathrm{j}\, A \;\;!\;\; (\mathsf{new}\; f : \mathsf{F})(f[\![P \;\mathrm{j}\;\; \mathsf{goto}\, a{:}\mathsf{tell}!\mathrm{h}f\mathrm{i} \;\mathrm{j}\; Q]\!]) \;\;\mathrm{j}\;\; a[\![R]\!] \qquad (5)$$

so the code Q is allowed to execute locally within the firewall.

Not]TJ -42.3 -11anthinit (resoofn9800llo)24Tf 12.37 0 TD[(a40]TJ/F1 9.96 Tf 9.69

4vironme-395(first)0(codeph)-3ed [info : rwR req 1-390(The) 946 d65 rd upervi 65 T 9 F 60021 e 24 6 d l declh Tf 8.55 0 TD[(=)D[(9]TJ/F1 9.96 Tf 3.54 0 TD[(

Then one might expect to be able to derive

$$\models\ F\ \mathrm{j}\,A\ \ _{bis}\ (\mathsf{new}\ f : \mathsf{F})(f[\![P\ \mathrm{j}\ \ \mathsf{goto}\,a.\mathsf{tell}!\mathrm{h}f\mathrm{i}\ \mathrm{j}\ Q]\!])\ \ \mathrm{j}\ \ a[\![R]\!] \qquad (7)$$

But this happens not to be true, because of the implicit assumption that the information channel tell in a can only be accessed by partners in the entry protocol, f and a. But, in order for (6) to be true, we must have $\ \grave{}_a$ tell : rwhF$_r$i; and this allows other agents in the environment access to tell. For example, consider

$$\mathsf{Rogue}\ \ (\ \ b[\![\mathsf{goto}\,a.\mathsf{tell}!\mathrm{h}b\mathrm{i}]\!]$$

and suppose that the only type inference from involving b is ` $b$ : loc; so  is not aware of any resources at $b$. Nevertheless  ` Rogue, and therefore *Contextuality* (Theorem 3.1) applied to (7) would give

$$\begin{aligned}&\models\ F\ \mathrm{j}\,A\ \mathrm{j}\ \mathsf{Rogue}\ \ _{bis}\\ &\quad(\mathsf{new}\ f : \mathsf{F})(f[\![P\ \mathrm{j}\ \ \mathsf{goto}\,a.\mathsf{tell}!\mathrm{h}f\mathrm{i}\ \mathrm{j}\ Q]\!])\ \ \mathrm{j}\ \ a[\![R]\!]\ \ \mathrm{j}\ \ \mathsf{Rogue}\end{aligned}$$

But this is obviously not the case, as the left-hand system can reduce via a series of -steps (representing the interaction between A and Rogue) to the state

$$\mathsf{B}\ F\ \mathrm{j}\ a[\![R]\!]\ \mathrm{j}\ b[\![Q]\!]$$

Under reasonable assumptions about the code Q, the right-hand system has no corresponding reduction to a similar state. On the left-hand side the code Q, now located at b, can not run, while on the right-hand side, no matter what -steps are made, Q will be able to execute at f.
Thus (7) can not be true.

However, our framework allows us to amend the correctness statement (7) above, taking into account the implicit assumption about the information channel tell. The essential point is that the protocol works provided that *only the firewall can write on* tell. This can be formalised by proving the equivalence between the two systems relative to a restricted environment, one which does not allow write access to tell.

First some notation. Let us write $\ \grave{}_k^{max}\ V : \mathsf{T}$ to mean

$\grave{}_k\ V : \mathsf{T}$

$\grave{}_k\ V : \mathsf{T}^0$ implies $\mathsf{T} <: \mathsf{T}^0$

In other words, T is the *largest* type which can be assigned to V. Now suppose I is a type environment which satisfies

(i) I $\grave{}_a^{max}$ tell : rhFi

(ii) I ` $a[\![R]\!]$

(iii) I ` (new f : F) $f[\![P]\!]$

The import of the first requirement, which is the most important, is that systems in the computational context can not write on tell. The other requirements, which are mainly for convenience, ensure that the residual behaviour at a and f is well-behaved, although a side-effect is that they also can not write on tell. Under these assumptions, we prove

$$I \models F \mid A \approx_{bis} (\mathsf{new}\, f : \mathsf{F})(f[\![P \mid \mathsf{goto}\, a.\mathsf{tell}!\langle f\rangle \mid Q]\!]) \mid a[\![R]\!] \qquad (8)$$

First note that (up-to structural equivalence)

$$I \triangleright F \mid A \xrightarrow{\tau} F \mid A_t \mid a[\![R]\!]$$

via (m-split) and (m-ctxt), where A_t is a shorthand for $a[\![\mathsf{tell}?(x)\, \mathsf{goto}\, x.Q]\!]$. So, by Propositions 3.2 and 3.4, it is su

M has the form F_g j A_t j $_n$ (a⟦tell!hfi⟧)^n

N has the form F_g j f⟦Q⟧ j $_n$ (a⟦tell!hfi⟧)^n

where $_n$ (a⟦tell!hfi⟧)^n, for some n 0, means n copies of a⟦tell!hfi⟧ running in parallel.

Proposition 4.1. *The parameterised relation* R *defined above is a bisimulation up-to- .*

Proof: Suppose J $\models$ M R N. Let us consider all possible actions from J B M. In fact, it is sufficient to consider the case (b) above, when J and M and N are of the prescribed form. The actions fall into one of three categories (for convenience we shorten $_n$ (a⟦tell!hfi⟧)^n with $_n$).

Note that the firewall F allows, in principle, multiple entries of agents from a. So, for example, if R

The first requirement establishes that *the computational context can not read on* req, while the following points ensure that the residual behaviour at the server and the clients is well-behaved, with the side-effect that neither S^0 nor C_i^0 can read on req.

First, let us show that one client interacts correctly with the server

$$\mathrm{I}\ \mathrm{j}\!\!=\ S\ \mathrm{j}\ C_1 \quad {}_{bis}\ S\ \mathrm{j}\ c_1[\![(\mathrm{newc}\, r : \mathrm{R})\ \ r!\mathrm{h}isprime(v_1)\mathrm{i}\ \mathrm{j}\ C_1^0]\!] \tag{10}$$

Note that (up-to-structural equivalence)

$$\mathrm{I}\ \mathrm{B}\, S\ \mathrm{j}\ C_1 \quad ! \quad (\mathrm{new}\, r@c_1 : \mathrm{R})\, S_r\ \mathrm{j}\ s[\![S^0]\!]\ \mathrm{j}\ s[\![\mathrm{req}!\mathrm{h}v_1; r@c_1\,\mathrm{i}]\!]\ \mathrm{j}\ c_1[\![C_1^0]\!]$$

where we use S_r as a shorthand for $s[\![\ \ \mathrm{req}?(x; y@z)\mathrm{goto}\, z{:}y!\mathrm{h}isprime(x)\mathrm{i}\,]\!]$, and

$$\begin{aligned}&\mathrm{I}\ \ \mathrm{B}\ S\ \mathrm{j}\ c_1[\![(\mathrm{newc}\, r : \mathrm{R})\ \ r!\mathrm{h}isprime(v_1)\mathrm{i}\ \mathrm{j}\ C_1^0]\!] \quad !\\ &\qquad (\mathrm{new}\, r@c_1 : \mathrm{R})\, S_r\ \mathrm{j}\ s[\![S^0]\!]\ \mathrm{j}\ c_1[\![r!\mathrm{h}isprime(v_1)\mathrm{i}]\!]\ \mathrm{j}\ c_1[\![C_1^0]\!]\end{aligned}$$

By Propositions 3.2, 3.4,

Proposition 5.1. *The parameterised relation* R *defined above is a bisimulation up-to- .*

Proof: Suppose J $\models$ M R N. The actions from J B M in the case (b) above fall into one of three categories.

First S_r is responsible

$$\mathsf{I}_r\,\mathsf{B}\,M \quad ! \quad s[\![\ \mathsf{req}?(x;y_@z)\mathsf{goto}\,z{:}y!\mathsf{h}isprime(x)\mathsf{i}\ \mathsf{j}\,R^0]\!]\ \mathsf{j}\ s[\![\mathsf{req}!\mathsf{h}v_1;\,r_@c_1\,\mathsf{i}]\!]\ \mathsf{j} \quad _n$$

where R^0 is a shorthand for $\mathsf{req}?(x;y_@z)\mathsf{goto}\,z{:}y!\mathsf{h}isprime(x)\mathsf{i}$. But

$$\mathsf{I}_r\ \mathsf{B}\ s[\![\ \mathsf{req}?(x;y_@z)\mathsf{goto}\,z{:}y!\mathsf{h}isprime(x)\mathsf{i}\ \mathsf{j}\,R^0]\!]\ \mathsf{j}\ s[\![\mathsf{req}!\mathsf{h}v_1;\,r_@c_1\,\mathsf{i}]\!]\ \mathsf{j} \quad _n \quad ! \\ S_r\ \mathsf{j} \quad _1\ \mathsf{j}\ s[\![\mathsf{req}!\mathsf{h}v_1;\,r_@c_1\,\mathsf{i}]\!]\ \mathsf{j} \quad _n$$

and this can be matched by

$$\mathsf{I}_r\,\mathsf{B}\,N \quad ! \quad S_r\ \mathsf{j} \quad _1\ \mathsf{j}\ c_1[\![r!\mathsf{h}isprime(v_1)\mathsf{i}]\!]\ \mathsf{j} \quad _n$$

because both configurations belong to R, clause (b), up-to structural equivalence.

The third component, $\ _n\ (s[\![\mathsf{req}?(x;y_@z)\mathsf{goto}\,z{:}y!\mathsf{h}isprime(x)\mathsf{i}]\!])^n$, is responsible for the action, which is either $s{:}\mathsf{req}?\ \ i_j;d_j@k_j$ or $(e{:}E)s{:}\mathsf{req}?\ \ i_j;d_j@k_j$. These actions correspond to the delivery of (new) data by the environment (from which the system is allowed to learn infinitely new names), and are followed by the action (m-move). However, it is easy to see that $\mathsf{I}_r\,\mathsf{B}\,N$ can

This completes our proof of (10), that one client can interact correctly with the server. Contextual reasoning can now be employed to generalise this result to an arbitrary number of clients. For example, let us show

$$\mathsf{I} \models S \,\mathsf{j}\, C_1 \,\mathsf{j}\, C_2 \quad {}_{bis} \; S \,\mathsf{j} \quad {}_{i\mathsf{2f1;2g}}\, c_i[\![(\mathsf{newc}\, r : \mathsf{R})\; r!\mathsf{h}isprime(v_i)\mathsf{i} \,\mathsf{j}\, C_i^0]\!] \qquad (11)$$

Because of $\mathsf{I} \;\grave{}\; C_2$ (requirement (iii) above), *Contextuality* applied to (10) gives

$$\mathsf{I} \models S \,\mathsf{j}\, C_1 \,\mathsf{j}\, C_2 \quad {}_{bis} \; S \,\mathsf{j}\, c_1[\![(\mathsf{newc}\, r : \mathsf{R})\; r!\mathsf{h}isprime(v_1)\mathsf{i} \,\mathsf{j}\, C_1^0]\!] \,\mathsf{j}\, C_2 \qquad (12)$$

On the other hand, repeating the analysis of C_1 on C_2, we obtain

$$\mathsf{I} \models S \,\mathsf{j}\, C_2 \quad {}_{bis} \; S \,\mathsf{j}\, c_2[\![(\mathsf{newc}\, r : \mathsf{R})\; r!\mathsf{h}isprime(v_2)\mathsf{i} \,\mathsf{j}\, C_2^0]\!]$$

But $\mathsf{I} \;\grave{}\; C_1$ (again (iii)) also implies $\mathsf{I} \;\grave{}\; c_1[\![(\mathsf{newc}\, r : \mathsf{R})\; r!\mathsf{h}isprime(v_1)\mathsf{i} \,\mathsf{j}\, C_1^0]\!]$, and therefore, by *Contextuality*

$$\begin{aligned} \mathsf{I} \models S \,\mathsf{j}\, C_2 \,\mathsf{j}\, c_1[\![(\mathsf{newc}\, r : \mathsf{R})\; r!\mathsf{h}isprime(v_1)\mathsf{i} \,\mathsf{j}\, C_1^0]\!] \quad {}_{bis} \\ S \,\mathsf{j} \quad {}_{i\mathsf{2f1;2g}}\, c_i[\![(\mathsf{newc}\, r : \mathsf{R})\; r!\mathsf{h}isprime(v_i)\mathsf{i} \,\mathsf{j}\, C_i^0]\!] \end{aligned}$$

So we conclude (11) from (12), Proposition 3.2, and transitivity of ${}_{bis}$.

It is then a simple matter to extend this reasoning, using induction, to show that an arbitrary number of clients can be handled

$$\mathsf{I} \models S \,\mathsf{j} \quad {}_{i\mathsf{2f1;:::;}n\mathsf{g}}\, C_i \quad {}_{bis} \; S \,\mathsf{j} \quad {}_{i\mathsf{2f1;:::;}n\mathsf{g}}\, c_i[\![(\mathsf{newc}\, r : \mathsf{R})\; r!\mathsf{h}isprime(v_i)\mathsf{i} \,\mathsf{j}\, C_i^0]\!]$$

This we leave to the reader.

As a further example of the modularity of our proofs, let us consider a particular instantiation of the residual processes, S^0 and C_i^0: we set S^0 to stop and C_i^0 to $r?(x)\,\mathsf{print}_i!\mathsf{h}x\mathsf{i}$, where print_i are local channels. For convenience we restrict attention to two clients, and let us assume that they send the integer values $v_1 = 4$ and $v_2 = 3$, respectively, to the server. So we have

$$\begin{aligned} S^{00} \;&\mathsf{(}\;\; s[\![\;\; \mathsf{req}?(x;y_{@}z)\mathsf{goto}\, z{:}y!\mathsf{h}isprime(x)\mathsf{i}]\!] \qquad)\,\mathsf{i}\,\mathsf{j}\,C^0 \;\; \mathsf{- - 6\,8 \;\; /F1\,1 \;\; 9\,.\,96\;T\,(\,using} \\ C_1^{00} \;&\mathsf{(}\;\; c_1[\![(\mathsf{newc}\, r : \mathsf{R})\;\; \mathsf{goto}\; s{:}\mathsf{req}!\mathsf{h}4;r_{@}c_1\,\mathsf{i} \,\mathsf{j}\, r?(x)\,\mathsf{prC} \end{aligned}$$

ther, to the tasks

$$\mathcal{I} \models S'' \mid c_1[\![(\mathsf{newc}\, r : \mathsf{R})\ \ r!\langle isprime(4)\rangle \mid r?(x)\, \mathsf{print}_1!\langle x\rangle]\!] \approx_{bis} \\ S'' \mid c_1[\![\mathsf{print}_1!\langle false\rangle]\!]$$

$$\mathcal{I} \models S'' \mid c_2[\![(\mathsf{newc}\, r : \mathsf{R})\ \ r!\langle isprime(3)\rangle \mid r?(x)\, \mathsf{print}_2!\langle x\rangle]\!] \approx_{bis} \\ S'' \mid c_2[\![\mathsf{print}_2!\langle true\rangle]\!]$$

Note that *Contextuality* does not allow us to eliminate S'' from these judgements, since $\mathcal{I} \vdash S''$ is not true. Nevertheless, it is a simple matter to construct a witnessing bisimulation to demonstrate directly these two equivalences, as the reader can check.

6 Metaservers

In this section we describe a *memory service* by involving the newloc operator of Dpi, which allows the creation of new instances of sites. A (meta)server contains a resource setup, where requests are received, and installs the service at a new site, thus providing personalised treatment to its clients.

A first version of the server receives a return address, generates a new located memory cell, and installs some code there, meanwhile delivering the new

An alternative, slightly different version of the server leaves to the clients the responsibility to create the memory cells, just installing the servicing code at the proffered site

$$S' \Leftarrow s'[\![\ast\,\mathsf{setup}'?(x, y@z)\ \mathsf{goto}\ x.\mathsf{Mem} \mid \mathsf{goto}\ z.y!]\!]$$

Correspondingly, clients generate an acknowledgement channel and a new location, send a request to the server, and await the server to acknowledge the service has been installed

$$C_i' \Leftarrow c_i[\![(\mathsf{newc}\ t : \mathsf{T})\ (\mathsf{newloc}\ m_i : \mathsf{M})\ \mathsf{goto}\ s'.\mathsf{setup}'!\langle m_i, t@c_i\rangle \mid t?P_i(m_i)]\!]$$

where $\mathsf{T} = \mathsf{rw}\langle\mathbf{unit}\rangle$.

We want now to relate the two different approaches, therefore connecting the behaviour of the two following systems, relative to a typing environment $\mathcal{I}$

$$\mathcal{I} \models S \mid C_1 \mid C_2 \tag{13}$$

$$\mathcal{I} \models S' \mid C_1' \mid C_2' \tag{14}$$

Our goal is to establish that, from the point of view of the clients, under certain hypotheses the two kinds of servers S and S' lead to equivalent behaviour. This means finding a suitable type environment $\mathcal{I}$ such that

$$\mathcal{I} \models S \mid C_1 \mid C_2 \approx_{bis} S' \mid C_1' \mid C_2' \tag{15}$$

It is immediate to notice that the correctness of this protocol requires that *the computational context should have neither write nor read access to the* setup *and* setup' *channels*. Thus, the equivalence can be proved relative to a restricted environment $\mathcal{I}$, satisfying

$$\mathcal{I} \vdash_s^{max} \mathsf{setup} : \top \qquad\qquad \mathcal{I} \vdash_{s'}^{max} \mathsf{setup}' : \top$$

Now, the internal actions can be used to deduce a derivation from (13) and (14) to the systems

$$\mathcal{I} \models S \mid \prod_{i\in\{1,2\}} (\mathsf{new}\ m_i : \mathsf{M})(m_i[\![\mathsf{Mem}]\!] \mid c_i[\![$$

(b) or (new $r_{1@}c_1$: R; $r_{2@}c_2$: R; m_1 : M) S j $_n$ j $S_{!2}$ j $C_{?2}$ j M_1 j $C_{!1}$ j $C_{?1}$

(c) or (new $r_{1@}c_1$: R; $r_{2@}c_2$: R; m_2 : M) S j $_n$ j $S_{!1}$ j $C_{?1}$ j M_2 j $C_{!2}$ j $C_{?2}$

(d) or (new $r_{2@}c_2$: R; m_1 : M) S j $_n$ j $S_{!2}$ j $C_{?2}$ j M_1 j C_{P_1}

(e) or (new $r_{1@}c_1$: R; m_2 : M) S j $_n$ j S

with the Dpi calculus [HR02b]. In order to cope with bisimulation equivalence in Dpi [HMR04], it is natural to look for bisimulations up-to in the spirit of [SM92]. More precisely, we have introduced in our work *bisimulations up-to -reductions*, which have been inspired by a similar approach to concurrent ML [JR04]. This technique actually relieves the burden of exhibiting witness bisimulations, and its feasibility has been proved to be successful, combined mainly with *Contextuality*

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes (I and II). *Information and Computation*, 100(1,2), 1992.

[PS00] Benjamin C. Pierce and Davide Sangiorgi. Behavioral equivalence in the polymorphic picalculus. *Journal of ACM* 47(3), 2000.

[SM92] Davide Sangiorgi and Robin Milner. The problem of “weak bisimulation up to”. In Proc. of *CONCUR*, *Lecture Notes in Computer Science* 630, Springer, 1992.

[SW01] Davide Sangiorgi and David Walker. *The* picalculus*: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[US01] Asis Unyapoth and Peter Sewell. Nomadic pict: correct communication infrastructure for mobile computation. In Proc. of *POPL*, 2001.