Un

Location and Link Failure in a Distributed π-calculus

Adrian Francalanza and Matthew Hennessy

Abstract. We develop a behavioural theory of distributed systems in the presence of failures. The framework we use is that of Dπ, a language in which located processes, or agents, may migrate between dynamically created locations. These processes run on a distributed network, in which individual nodes may fail, or the links between them may be broken. The original language, Dπ, is extended by a new construct for detecting and reacting to these failures together with constructs that induce failure.

We define a bisimulation equivalence between these systems, based on labelled actions which record, in addition to the effect actions have on the processes, the actual state of the underlying network and the view of this state known to observers. We prove that the equivalence is *fully abstract*, in the sense that two systems will be differentiated if and only if, in some sense, there is a computational context, consisting of a network and an observer, which can see the difference.

Contents

1 Introduction

It is generally accepted that location transparency is not attainable over *wide-area networks*, [4], large computational infrastructures which may even span the globe. Because of this, various *location-aware* calculi and programming languages have arisen in the literature; not only do these emphasise the *distributed* nature of systems but they also assume that the various system components, processes or agents, are aware of their location in the network, and perhaps, also aware of some aspect of the underlying network topology. In these languages, computations take place at distinct locations, physical or virtual, and processes may migrate between the locations of which they are aware, to participate in such computations.

It is also argued in [4] that *failures*, and the ability to react to them, are also an inevitable facet of these infrastructures, which must be taken into account when designing languages for location-aware computation. The

| inv

state of the system, to N in the usual manner, but it may also a ect the nature of the underlying network. For example, an observer may extend the network by creating new locations; we also allow the observer to kill sites, or in the second framework, break links between sites, thereby capturing changes in the behaviour of N in response to dynamic failures.

In the framework with link failures, the definition of these actions turns out to be relatively sophisticated. Intuitively, the action (1) above is meant to simulate the interaction between an observer and the system. However, even though the system and the observer may initially share the same view of the underlying network, interactions quickly give rise to situations in which these views *diverge*. In general, observers may not be aware of the status of all the nodes and links in a network because they might be *unreachable*; the system, on the other hand may reach such nodes through the knowledge of scoped names. So in (1) above, the network representation needs to record the actual state of the underlying network, together with the observers *partial view* of it. This in turn will require developing variations on the actions (1) above, where the actual network representations , are replaced by more abstract representations.

In Section 2 we treat the case of *location failure*. We define the language DπLoc, an extension of Dπ [1] [11], with an additional operator ping $l.P\vdash Q$, for checking the accessibility of a location l

duction semantics is now given relative to network configuration, ranged over by ▲, which record both the status of nodes and their connectivity. Interestingly, ping $l.P \vdash Q$ executed on site k, now checks whether l is *accessible* from k; it may not be accessible either because l is dead or because the link between l and k is broken. Moreover, in DπF, *node creation* is more complicated, since one must also specify the connectivity of a new node; in DπF this is achieved using a simple type system.

In 3.3 we present a variation of the actions (1) above for DπF. It turns out that much of the information in the network representations ▲ is irrelevant; for example if a node

k

Table 1. *Syntax of typed DπLoc*

Types

$$\mathtt{T},\mathtt{U} ::= \mathtt{loc}[\mathtt{S}] \mid \mathtt{ch} \qquad \mathtt{S},\mathtt{R} ::= \mathtt{a} \mid \mathtt{d}$$

Processes

$$\begin{array}{l} P, Q ::= u!\rceil\tilde{V}.P \mid u?(X).P \mid !u?(X).P \mid \mathsf{if}\ v = u.P \vdash Q \mid P \mid Q \\ \qquad\quad \mid (\nu\, n:\mathtt{T})P \mid \mathsf{go}\ u.P \mid \mathsf{kill} \mid \mathsf{ping}\ u.P \vdash Q \mid \mathbf{0} \end{array}$$

Systems

$$M, N, O ::= l[\![P]\!] \mid N \mid M \mid (\nu\, n:\mathtt{T})N$$

but the presence of this construct will facilitate the definition of the reduction semantics.

There are three syntactic categories in DπLoc. The first, *local processes* ranged over by P, Q, includes the standard π-calculus constructs for communication, $a!\rceil\tilde{V}.P$ and $a?(X).P$, replicated input, $!a?(X).P$, name restriction $(\nu\, n:\mathtt{T})P$, where $\mathtt{T}$ types n as a channel or a location name, comparison $\mathsf{if}\ v = u.P \vdash Q$, inaction, $\mathbf{0}$, and parallel composition, $P \mid Q$. The values transmitted as part of a communication, ranged over by V, consist of tuples of identifiers. When input on a channel, they are deconstructed using patterns, ranged over by X; patterns are simply tuples of variables, each having a unique occurrence.

The major innovation is a programming construct which allows processes to react to *perceived faults* in the underlying communication network. In addition to the Dπ migration construct $\mathsf{go}\ l.P$, [11], we add a testing construct, $\mathsf{ping}\ l.P \vdash Q$, inspired from [2, 1, 16]. This constructs acts as a conditional, based on the *perceived state* of the location l; thus if l is reachable, it launched process P, otherwise it launches Q.

As explained in the introduction, we also wish to consider the behaviour of systems under dynamic network faults. To simulate these instances,

state of the scoped location (dead or alive).

In contrast to Dπ, DπLoc uses also an additional third level of *configurations*. At this level, we have a representation of the network on which the system is running. A typical configuration takes the form

$$\triangleright\, N$$

where

Table 2. *Local Reduction Rules for DπLoc*

Assuming ⊢ $l : \mathbf{alive}$

(r-comm)

$$\triangleright\ l[\![a!\langle V\rangle.P]\!] \mid l[\![a?(X)]\!]$$

Table 3. *Network Reduction Rules for DπLoc*

Assuming $\vdash l : \mathbf{alive}$

(r-go)
$$\frac{}{\triangleright\, l[\![\mathsf{go}\ k.P]\!] \quad \triangleright\, k[\![P]\!]} \ \Pi \quad k \quad l$$

(r-ngo)
$$\frac{}{\triangleright\, l[\![\mathsf{go}\ k.P]\!] \quad \triangleright\, k[\![\mathbf{0}]\!]} \ \Pi \not\sim k \quad l$$

(r-ping)
$$\frac{}{\triangleright\, l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \quad \triangleright\, l[\![P]\!]} \ \Pi \quad k \quad l$$

(r-nping)
$$\frac{}{\triangleright\, l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \quad \triangleright\, l[\![Q]\!]} \ \Pi \not\sim k \quad l$$

(r-new)
$$\frac{}{\triangleright\, l[\![(\nu\, n{:}\mathtt{T})P]\!] \quad \triangleright\, (\nu\, n{:}\mathtt{T})\, l[\![P]\!]}$$

(r-kill)
$$\frac{}{\triangleright\, l[\![\mathsf{kill}]\!] \quad (\quad l) \triangleright\, l[\![\mathbf{0}]\!]}$$

locations. Dynamic network faults are engendered in the obvious manner by (r-kill), and finally (r-new), allows us to export locally generated new names to the system level, as in Dπ.

The rules in Table 4 are adaptations of standard rules for the π-calculus. For instance, the first rule, (r-str), states that the reduction semantics is defined up to a *structural equivalence*, defined in the usual manner, as the least equivalence relation on systems which satisfies the set of rules and axioms in Table 5. The remaining reduction rules in Table 4 state that reduction is preserved by parallel composition and name scoping operations on configurations. But note that the rule for scoping, (r-ctxt-rest), uses an obvious notation $+\, n : \mathtt{T}$ for extending network representations with new names, which is formally defined in the Appendix. Note also that this rule needs to allow for the type of the scoped name to change; this is because types for locations actually carry *dynamic* state information, namely whether they are alive or dead, as explained in the following example.

Example 2.2.2. Consider the following system

$$\triangleright\, k[\![\mathsf{go}\ l.a?(x).P]\!] \quad l[\![(\nu\, k_0 : \mathtt{loc[a]})(a!\rceil k_0\langle .Q \quad \mathsf{go}\ k_0.\mathsf{kill})]\!] \qquad (3)$$

where is the network representation $\rceil\, l, k, a\neg\langle$, consisting of the two *live* locations l, k; the addition of the construct $\mathsf{go}\ k_0.\mathsf{kill}$ indicates that we wish to consider the newly created location k_0, as defective, and thus it may become faulty some time in the future.

Table 4. *Contextual Reduction Rules for DπLoc*

(r-str)

$$\frac{\triangleright N \;-\; \triangleright N \qquad \triangleright N \qquad \triangleright M \qquad \triangleright M \;-\; \triangleright M}{\triangleright N \qquad \triangleright M}$$

(r-ctxt-rest)

$$\frac{+\,n : \mathtt{T} \triangleright N}{\triangleright (\nu\, n : \mathtt{T})N} \qquad \frac{+\,n : \mathtt{U} \triangleright M}{\triangleright (\nu\, n : \mathtt{U})M}$$

(r-ctxt-par)

$$\frac{\triangleright N}{\triangleright N \setminus M} \qquad \frac{\triangleright N}{\triangleright N \setminus M} \;\Pi\; M$$

$$\triangleright M \setminus N \qquad \triangleright M \setminus N$$

Table 5. *Structural Rules for DπLoc*

(s-comm)	$N \setminus M \;-\; M \setminus N$	
(s-assoc)	$(N \setminus M) \setminus M \;-\; N \setminus (M \setminus M)$	
(s-unit)	$N \setminus l[\![\mathbf{0}]\!] \;-\; N$	
(s-extr)	$(\nu\, n{:}\mathtt{T})(N \setminus M) \;-\; N \setminus (\nu\, n{:}\mathtt{T})M$	$n \notin \mathbf{fn}(N)$
(s-flip)	$(\nu\, n{:}\mathtt{T})(\nu\, m{:}\mathtt{U})N \;-\; (\nu\, m{:}\mathtt{U})(\nu\, n{:}\mathtt{T})N$	
(s-inact)	$(\nu\, n{:}\mathtt{T})N \;-\; N$	$n \notin \mathbf{fn}(N)$

As in Dπ, an application of (r-go), based on the fact that both k and l are alive, and (r-par-ctxt) on (3), yields

$$\triangleright\, l[\![a?(x).P]\!] \setminus l[\![(\nu\, k_0 : \mathtt{loc[a]})a!\rceil k_0 \langle .Q \setminus \mathsf{go}\ k_0.\mathsf{kill}]\!]$$

which can be followed by an application of (r-fork), (r-new) (and (r-par-ctxt)) to launch a new location k_0 and get

$$\triangleright\, l[\![a?(x).P]\!] \setminus (\nu\, k_0 : \mathtt{loc[a]})(\, l[\![a!\rceil k_0 \langle .Q]\!] \setminus l[\![\mathsf{go}\ k_0.\mathsf{kill}]\!]\,)$$

At this point, we can perform a communication on channel a using (r-par-comm), thereby and

Finally, (r-kill), followed by (r-ctxt-par), can be used to kill k_0 and derive

$$(\quad + k_0 : \texttt{loc[a]}) \triangleright l[\![P\ ^{k_0}/_x]\!] \ \backslash\ l[\![Q]\!] \ \backslash\ k_0[\![\mathsf{kill}]\!] \\ (\quad + k_0 : \texttt{loc[d]}) \triangleright l[\![P\ ^{k_0}/_x]\!] \ \backslash\ l[\![Q]\!] \ \backslash\ k_0[\![\mathbf{0}]\!]$$

where the type of k_0 changes from `loc[a]` to `loc[d]`, and thus, an application of (r-ctxt-rest) reduces (5) to

$$\triangleright (\nu\, k_0 : \texttt{loc[d]})(\ l[\![P\ ^{k_0}/_x]\!] \ \backslash\ l[\![Q]\!] \ \backslash\ k_0[\![\mathbf{0}]\!]\ ) \tag{6}$$

■

This example also serves to illustrate another important point that we shall refer to repeatedly in this report. In general, in a configuration

$$\triangleright N$$

denotes the network representation on which the system N is running. But there may be subsystems of N which are running on extended (internal) networks. For example in (3) above, the subsystem $l[\![a!\rceil k\langle\ .Q]\!]$ is running with respect to the network represented by $(\quad + k_0 : \texttt{loc[a]})$, while in (6) the subsystem $l[\![Q]\!]$ is running with respect to $(\quad + k_0 : \texttt{loc[d]})$.

Reduction barbed congruence

In view of the reduction semantics, we can now adapt the standard approach [9, 8] to obtain a contextual equivalence for DπLoc; we use the variation first proposed in [12]. We wish to compare the behaviour of systems running on the same network; consequently we use the following framework, borrowed from [8]:

Definition 2.2.3 (Typed Relation). A *typed relation* over systems is a family of binary relations between systems, $\mathcal{R}$, indexed by network representations. We write $\quad = M\ \mathcal{R}\ N$ to mean that systems M and N are related by $\mathcal{R}$ at index $\quad$, that is $M\ \mathcal{R}_{\Pi}\ N$, and moreover $\quad \triangleright M$ and $\quad \triangleright N$ are valid configurations. ■

The definition of our equivalence hinges on what it means for a typed relation to be *contextual*, which must of course take into account the presence of the network. Our definition has two requirements:

$$\langle \quad \mathcal{R} \vdash \quad \bullet \quad \vdash \quad \langle \quad$$

Definition 2.2.4 (Observers). The intuition of *valid* observer system O in a distributed setting Γ, denoted as $\Gamma \vdash_{\mathsf{obs}} O$, is that O originates from some *live fresh* location k_0, migrates to any location in $\mathbf{loc}(\mathcal{N})$ to interact with (observe) processes there and then returns back to the originating fresh location k_0 to compare its observations with other observers. For convenience, we often omit the mentioning of the fresh locations k_0 and place observing code immediately at locations in $\mathbf{loc}(\mathcal{N})$. We note that, according to the definition of the reduction rule (r-ngo), observing code can never reach dead locations and we therefore have to encode this in our definition of $\Gamma \vdash_{\mathsf{obs}} O$. For convenience, we also disallow observer to be located at scoped *dead* loactions, denoted as $\Gamma \vdash_{\mathsf{obs}} \mathtt{T}$ and defined in the Appendix. $\Gamma \vdash_{\mathsf{obs}} O$ is recursively defined as:-

	$\Gamma \vdash_{\mathsf{obs}} l[\![P]\!]$	if $\mathbf{fn}(P) \subseteq \mathcal{N}$ and $\Gamma \vdash l : \mathbf{alive}$
|	$\Gamma \vdash_{\mathsf{obs}} (\nu\, n{:}\mathtt{T})N$	if $\Gamma \vdash_{\mathsf{obs}} \mathtt{T}$ and $(\Gamma, n{:}\mathtt{T}) \vdash_{\mathsf{obs}} N$
|	$\Gamma \vdash_{\mathsf{obs}} M \mid N$	if $\Gamma \vdash_{\mathsf{obs}} M$ and $\Gamma \vdash_{\mathsf{obs}} N$ (ContRtion

■

Definition 2.2.4, defining allowed observer systems, determines the definition of contextuality given below.

cont /R46 14.3462 Tf 6.3408 60 Td (n)TjO/R66 14.34

Definition 2.2.5 (Contextual typed relations). A typed relation $\mathcal{R}$ over configurations is *contextual* if:

(Parallel Systems)

|	$\Gamma \vdash M \mathcal{R} N$ and $\Gamma \vdash$	$= M_{\setminus}$

Then ≅, called *reduction barbed congruence*, is the largest symmetric typed relation over configurations which is:

| barb preserving

| reduction closed

| contextual ■

We leave the reader to check that pointwise ≅ is an equivalence relation.

Example 2.2.7. Consider the systems onePkt and twoPkt defined as:

$$\begin{array}{ll} \mathsf{onePkt} & l[\![\mathsf{go}\ k.(a!\langle\rangle \mid b!\langle\rangle)]\!] \\ \mathsf{twoPkt} & l[\![\mathsf{go}\ k.a!\langle\rangle]\!] \mid l[\![\mathsf{go}\ k.b!\langle\rangle]\!] \end{array}$$

They represent two di erent strategies for sending the messages $a!\langle\rangle \mid b!\langle\rangle$ from l to k. The first system, onePkt, transfers the two messages as one unit (one packet), whereas the second system, twoPkt, uses a distinct packet for every message. In a calculus with no network failure, it would be hard to distinguish between these two systems.

The two configurations are however

We also note that

$$_{k} \triangleright k[\![a!\langle\,\rangle]\!] \downarrow_{a@k}$$
$$_{k} \triangleright k[\![a!\langle\,\rangle]\!] \downarrow_{b@k}$$

However the left hand side, $_{lk} \triangleright \mathsf{onePkt} \setminus l[\![\mathsf{kill}]\!]$ can never reduce to a configuration with such barbs. Formally, there is no configuration $\triangleright N$ such that

$$_{lk} \triangleright l[\![\mathsf{go}\ k.(a!\langle\,\rangle \setminus b!\langle\,\rangle)]\!] \setminus l[\![\mathsf{kill}]\!] \qquad \triangleright N$$

where $\triangleright N \downarrow_{a@k}$ and $\triangleright N \downarrow_{b@k}$. ■

Example 2.2.8. Consider the two systems:

$$\mathsf{nonDet1} \quad (\nu\, k{:}\mathtt{loc[a]})\ k[\![\mathsf{kill}]\!] \setminus k[\![\mathsf{go}\ l.a!\langle\,\rangle]\!]$$
$$\mathsf{nonDet2} \quad (\nu\, b{:}$$

By contrast, the next two server implementations introduce a degree of *distribution*, by processing the request across a number of locations:

$$\mathsf{srvDis} \quad (\nu\, data : \mathsf{ch})\ l[\![req?(x,y).\mathsf{go}\ k_1.data!\langle x,y\rangle]\!] \mid k_1[\![data?(x,y).\mathsf{go}\ l.y!\langle f(x)\rangle]\!]$$

$$\mathsf{srv2Rt} \quad (\nu\, data : \mathsf{ch})\ l[\![req?(x,y).(\nu\, sync : \mathsf{ch})(\mathsf{go}\ k_1.data!\langle x, sync\rangle \mid \mathsf{go}\ k_2, k_1.data!\langle x, sync\rangle \mid sync?(x).y!\langle x\rangle)]\!] \mid k_1[\![data?(x,y).(\mathsf{go}\ l.y!\langle f(x)\rangle \mid \mathsf{go}\ k_2, l.y!\langle f(x)\rangle)]\!]$$

Both servers, srvDis and srv2Rt, distributed the internal database *remotely* at location k_1. Server srvDis thus receives a client request at l, migrates *directly* to k_1 and queries the database; the database then returns to l and reports back the processed value, $f(x)$, on the requested return channel y. The other server, srv2Rt, accepts a client request at l, but attempts to access the unique remote database located at k_1 through *two di erent routes*, one *directly* from l to k_1 and the other *indirectly* from l through the intermediate node k_2 and then finally to k_1 where the database resides; similarly, the internal database of srv2Rt returns the answer $f(x)$ on y along these two routes. In a scenario where no fault occurs to k_1 and k_2, srv2Rt will receive two answers back at l. To solve this, the original requests are sent with a scoped return channel *sync*; a process waiting for answers on this channel at location l chooses non-deterministically between any two answers received and relays the answer on the original channel y.

We leave the reader to check that the local server, server and remote implementations, srvDis and srv2Rt, are di erent, that is:

$$\not\cong \mathsf{server} \cong \mathsf{srvDis} \quad \text{and} \quad \not\cong \mathsf{server} \cong \mathsf{srv2Rt}$$

because of their behaviour in the context

$$C_2[\ ] = [\ ] \mid k_1[\![\mathsf{kill}]\!]$$

However, it turns out that the two remote server implementations are reduction barbed congruent in DπLoc:

$$\cong \mathsf{srvDis} \cong \mathsf{srv2Rt}$$

Unfortunately, Definition 2.2.6 makes it hard to prove this statement because it uses quantification over all possible contexts. ■

Due to the problems associated with Definition 2.2.6, we need an inductive definition of behavioural equivalence that is easier to prove but still consistent with reduction barbed congruence. In the remainder of this section we define a *bisimulation equivalence* which allows us to relate 76 Td (Due)Tj 28.13 0 Td (to)T

Table 6. *Operational Rules(1) for DπLoc*

Assuming	$\vdash l : \mathbf{alive}$

Table 7. *Operational Rules(2) for DπLoc*

(l-open)

$$\frac{+\,n:\mathtt{T} \triangleright N \xrightarrow{(\tilde{\nu} n\tilde{\mathtt{T}})l:a!\ V} \triangleright N}{\triangleright (\nu\, n:\mathtt{T})N \xrightarrow{(n:\mathbf{T},\tilde{\nu} n\tilde{\mathbf{T}})l:a!\ V} \triangleright N}\ l, a \neq n \quad V$$

(l-weak)

$$\frac{+\,n:\mathtt{T} \triangleright N \xrightarrow{(\tilde{\nu} n\tilde{\mathtt{T}})l:a?(V)} \triangleright N}{\triangleright N \xrightarrow{(n:\mathbf{T},\tilde{\nu} n\tilde{\mathbf{T}})l:a?(V)} \triangleright N}\ l, a \neq n \quad V$$

(l-rest)

$$\frac{+\,n:\mathtt{T} \triangleright N \xrightarrow{\mu} +\,n:\mathtt{U} \triangleright N}{\triangleright (\nu\, n:\mathtt{T})N \xrightarrow{\mu} \triangleright (\nu\, n:\mathtt{U})N}\ n \notin \mathbf{fn}(\mu)$$

(l-par-ctxt)

$$\frac{\triangleright N \xrightarrow{\mu} \triangleright N}{\begin{array}{l}\triangleright N \setminus M \xrightarrow{\mu} \triangleright N \setminus M \\ \triangleright M \setminus N \xrightarrow{\mu} \triangleright M \setminus N\end{array}}\ \Pi \quad M$$

(l-par-comm)

$$\frac{\triangleright N \xrightarrow{(\tilde{\nu} n\tilde{\mathtt{T}})l:a!\ V} \triangleright N \qquad \triangleright M \xrightarrow{(\tilde{\nu} n\tilde{\mathtt{T}})l:a?(V)} \triangleright M}{\begin{array}{ll}\triangleright N \setminus M & \xrightarrow{\tau} \triangleright (\nu\, \tilde{n}:\tilde{\mathtt{T}})(N \setminus M) \\ \triangleright M \setminus N & \xrightarrow{\tau} \triangleright (\nu\, \tilde{n}:\tilde{\mathtt{T}})(M \setminus N)\end{array}}$$

rules inherited from distributed π-calculi such as Dπ; note, however, that actions can only occur at live locations. The rules in Table 7 are also adaptations of the standard rules for actions-in-context from [9] together with the rule (l-par-comm), for local communication. Here, we highlight the rule (r-weak), dealing with the learning of the existence of new location names and their state as a result of an input from the context; this rule was adopted from a variant used already in [9, 8]. Note also the general form of (l-rest), where the type of n may change from T to U; this phenomena is inherited directly from (r-ctxt-rest) of Table 4 in the reduction semantics and explained in Example 2.2.9.

The rules dealing with the new constructs of DπLoc, are contained in Table 8, most of which are inherited from the reduction semantics. The only new one is (l-halt), where the action kill : l represents a failure induced by an observer. This is in contrast with the rule (l-kill), where l is killed by the system itself and the associated action is τ.

Table 8. *Operational Rules(3) for DπLoc*

Assuming $\vdash l : \mathbf{alive}$

(l-kill) $\dfrac{}{\triangleright\, l[\![\mathsf{kill}]\!] \xrightarrow{\tau} (\quad l) \triangleright\, l[\![\mathbf{0}]\!]}$ (l-halt) $\dfrac{}{\triangleright\, N \xrightarrow{\mathsf{kill}:l} (\quad l) \triangleright\, N}$

(l-new) $\dfrac{}{\triangleright\, l[\![(\nu\, n:\mathrm{T})P]\!] \xrightarrow{\tau} \triangleright\, (\nu\, n:\mathrm{T})l[\![P]\!]}$

(r-go) $\dfrac{}{\triangleright\, l[\![\mathsf{go}\ k.P]\!] \xrightarrow{\tau} \triangleright\, k[\![P]\!]}\ \Pi\ k\ l$ (r-ngo) $\dfrac{}{\triangleright\, l[\![\mathsf{go}\ k.P]\!] \xrightarrow{\tau} \triangleright\, k[\![\mathbf{0}]\!]}\ \Pi \not\vdash k\ l$

(r-ping) $\dfrac{}{\triangleright\, l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \xrightarrow{\tau} \triangleright\, l[\![P]\!]}\ \Pi\ k\ l$

(r-nping) $\dfrac{}{\triangleright\, l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \xrightarrow{\tau} \triangleright\, l[\![Q]\!]}\ \Pi \not\vdash k\ l$

The first sanity check we prove about our lts is the property that in an action

$$\triangleright\, N \xrightarrow{\mu} \triangleright\, N$$

were μ is an external action, the residual network is completely determined by the network and the external action μ.

Definition 2.3.2 (Action residuals). The partial function after from tuples of network representations and external actions μ to network representation is defined as:

- after $(\tilde{n} : \tilde{\mathrm{T}})l : a!\rceil V$ is defined as $+\, \tilde{n} : \tilde{\mathrm{T}}$
- after $(\tilde{n} : \tilde{\mathrm{T}})l : a?(V)$ is defined as $+\, \tilde{n} : \tilde{\mathrm{T}}$
- after $\mathsf{kill} : l$ is defined as l

The second sanity check is that the actions are indeed well-defined relations over configurations.

Proposition 2.3.4. The lts defined in Definition 2.3.1 forms a binary relation between well-defined configurations. That is, if $\triangleright N \xrightarrow{\mu} \triangleright N$ and $\vdash N$ then $\vdash N$.

Proof. By induction on the derivation of the action $\triangleright N \xrightarrow{\mu} \triangleright N$. Note that if μ is an external action then Proposition 2.3.3 gives the precise form of . Moreover if it is the internal action τ then it is possible to prove that either coincides with or takes the form l for some l live in . □

Using the lts of actions we can now define, in the standard manner, *weak bisimulation equivalence* over configurations. Our definition uses the standard notation for weak actions, namely $\Longleftarrow^{\mu}$ denotes $\Longleftarrow \xrightarrow{\mu} \Longleftarrow$, and $\Longleftarrow^{\widehat{\mu}}$ denotes

$\Longleftarrow \xrightarrow{\tau}$ if $\mu = \tau$

$\Longleftarrow^{\mu}$ otherwise.

Definition 2.3.5 (Weak bisimulation equivalence). This is denoted as $\cong$, and is defined to be the largest typed relation over configurations such that if

It is clear that $\mathcal{R}$ is a typed relation; we only have to show that $\mathcal{R}$ it is a bisimulation. The proof proceeds by induction on the structure of $\triangleright M$ and $\triangleright N$. □

Example 2.3.7. We recall that in Example 2.2.8, we claimed that $_l \triangleright \mathsf{nonDet1}$ was equivalent to $_l \triangleright \mathsf{nonDet2}$. We here show that they are *bisimilar*, by giving

Table 9. *Syntax of DπF*

Types

$\mathtt{T},\mathtt{U},\mathtt{W} ::= \mathtt{ch} \mid \mathtt{loc}[\mathtt{S},\mathtt{C}]$ $\qquad$ $\mathtt{S},\mathtt{R} ::= \mathtt{a} \mid \mathtt{d}$

$\mathtt{C},\mathtt{D} ::= u_1,\ldots,u_n$

Processes

$P, Q ::= \ldots \mid \mathsf{break}\ l$

Systems

$N, M ::= \ldots$

describe the live *connections* to other locations. Thus, in DπF, a location type is denoted as $\mathtt{loc}[\mathtt{S},\mathtt{C}]$, where the first element $\mathtt{S}$ is inherited from Section 2, and the second element $\mathtt{C}$ is a set of locations $l_1,\ldots,l_n$. If a new k location is declared at this type, then it is intended to be linked in the underlying network with each of the locations l_i, although there will be complications; see Example 3.2.1. The only other modificiation to the syntax is the addition of the process construct $\mathsf{break}\ l$, which breaks a live connection between the location hosting the process and location l. Contextual equivalences then take into account the effect of link faults on system behaviour, in the same manner as the presence of kill takes node faults into account.

The final major extension in the DπF syntax is in the network representation; in a setting where not every node is interconnected, the network representation needs also to represent which nodes are connected apart from their current alive/dead status.

Definition 3.1.1 (Network representation). First let us introduce some notation to represent the *links*the

| $\mathcal{D}$ **loc**($\mathcal{N}$) represents the set of dead locations, as before.

| $\mathcal{L}$ **loc**($\mathcal{N}$) **loc**($\mathcal{N}$) represents the set of connections between locations ■

As with DπLoc network representations, we use the notation $\blacktriangle_{\mathcal{N}}$, $\blacktriangle_{\mathcal{D}}$ and $\blacktriangle_{\mathcal{L}}$ to refer to the individual components of ▲. We will also have various notation for checking properties of DπF network representations, and updating them; these will be explained informally, with the formal definitions relegated to the Appendix.

3.2 Reduction Semantics of DπF

The definition of well-formed configurations, Definition 2.2.1 generalises in a straightforward manner: we say $\blacktriangle \triangleright M$ is a well-formed configuration if every free name occuring in M is also in $\blacktriangle_{\mathcal{N}}$. Then the judgements of the reduction semantics take the form

$$\blacktriangle \triangleright M \qquad \blacktriangle \triangleright M$$

where $\blacktriangle \triangleright M$ and $\blacktriangle \triangleright M$ are well-formed configurations. This is defined as the least relation which satisfies the rules in Table 2 and Table 4 (substituting ▲ for) , all inherited from the reduction rules for DπLoc, together with the new reduction rules of Table 10, which we now explain. We note that, as usual, all of these rules require that the location where the activity is to occur is alive, $\blacktriangle \vdash l$: **alive**.

The most subtle but important changes to the network reductions rules are those concerning the constructs go and ping. Even though the general intuition remains the same to that of 2.2, the former notion of k being accessible from l, used by rules such as (r-go) and (r-nping), and still denoted as $\blacktriangle \vdash k \quad l$, changes; for k to be *accessible* from l, two conditions must hold, namely that k is alive *and* that the link between l and k is alive as well. If any of these two conditions do not hold, then k is deemed to be *inaccessible* from the point of view

Table 10. *New Network Reduction Rules for DπF*

Assuming $\blacktriangle \vdash l : \mathbf{alive}$

(r-go)

$$\frac{}{\blacktriangle \triangleright l[\![\mathsf{go}\ k.P]\!] \quad \blacktriangle \triangleright k[\![P]\!]}\ \Delta \quad k \quad l$$

(r-ngo)

$$\frac{}{\blacktriangle \triangleright l[\![\mathsf{go}\ k.P]\!] \quad \blacktriangle \triangleright k[\![\mathbf{0}]\!]}\ \Delta \not\vdash k \quad l$$

(r-ping)

$$\frac{}{\blacktriangle \triangleright l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \quad \blacktriangle \triangleright l[\![P]\!]}\ \Delta \quad k \quad l$$

(r-nping)

$$\frac{}{\blacktriangle \triangleright l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \quad \blacktriangle \triangleright l[\![Q]\!]}\ \Delta \not\vdash k \quad l$$

(r-newc)

$$\frac{}{\blacktriangle \triangleright l[\![(\nu\, c : \mathsf{ch})\, P]\!] \quad \blacktriangle \triangleright (\nu\, c : \mathsf{ch})\, l[\![P]\!]}$$

(r-newl)

$$\frac{}{\blacktriangle \triangleright l[\![(\nu\, k : \mathtt{loc}[\mathtt{S}, \mathtt{C}])\, P]\!] \quad \blacktriangle \triangleright (\nu\, k : \mathtt{loc}[\mathtt{S}, \mathtt{D}])\, l[\![P]\!]}\ \mathbf{loc}[\mathbf{S}, \mathbf{D}] = \mathsf{inst}(\mathbf{loc}[\mathbf{S}, \mathbf{C}], l, \Delta)$$

(r-kill)

$$\frac{}{\blacktriangle \triangleright l[\![\mathsf{kill}]\!] \quad (\blacktriangle \quad l) \triangleright l[\![\mathbf{0}]\!]}$$

(r-brk)

$$\frac{}{\blacktriangle \triangleright l[\![\mathsf{break}\ k]\!] \quad (\blacktriangle \quad l \_ k) \triangleright l[\![\mathbf{0}]\!]}\ \Delta \quad l \quad k$$

Table 11. *New Structural Rules for DπF*

. . .
(s-flip-1) $(\nu\, n : \mathtt{T})(\nu\, m : \mathtt{U})N \equiv (\nu\, m : \mathtt{U})(\nu\, n : \mathtt{T})N \qquad n \notin \mathbf{fn}(\mathtt{U})$
(s-flip-2) $(\nu\, n : \mathtt{T})(\nu\, m : \mathtt{U})N \equiv (\nu\, m : \mathtt{U} \quad n)(\nu\, n : \mathtt{T} + m)N \quad n \quad \mathbf{fn}(\mathtt{U})$
. . .

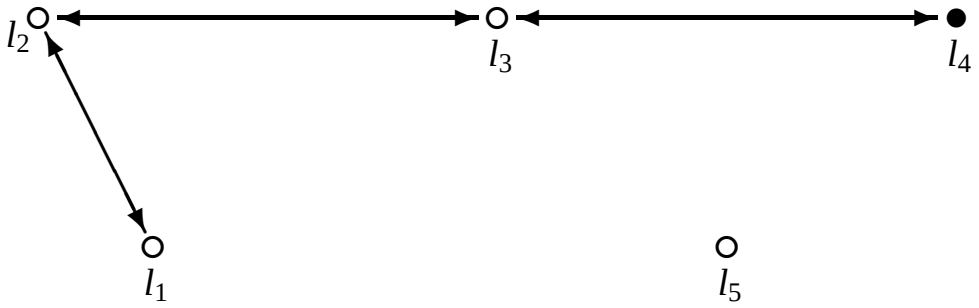
of which is relegated to the Appendix; intuitiv

rules in Table 5, defining the structural equivalence. The revision is detailed in Table 11; the rule (s-flip) is replaced by the two rules (s-flip-1) and (s-flip-2). This enables us to flip two successively scoped locations even if the first is used in the type of the second, that is there is a link between the two scoped locations.

Example 3.2.1. Consider the system:

$$\mathsf{launchNewLoc} \qquad l_3[\![a!\lceil l_1' \rceil \,\|\, l_3[\![a?(x).(\nu k : \mathtt{loc}[\mathtt{a},\ x, l_2, l_4, l_5\ ])P]\!]$$

running on a network ▲ consisting of four locations $l_1..l_5$, all of which are alive except l_4, with l_2 connected to l_1 and l_3, and l_3 connected to l_4. Diagrammatically this is easily represented as:



where, open nodes ( ) represent live locations and closed ones (|) dead locations; we systematically omit reflexive links in these network diagrams. Formally describing ▲ is more tedious:

- ▲$_{\mathcal{N}}$ is $a,\ l_1,\ l_2,\ l_3,\ l_4,\ l_5$
- ▲$_{\mathcal{D}}$ is l_4
- the link set ▲$_{\mathcal{L}}$ is given by

$$\begin{pmatrix} \lceil l_1, l_1' , \lceil l_2, l_2' , \lceil l_3, l_3' , \lceil l_4, l_4' , \lceil l_5, l_5' , \\ \lceil l_1, l_2' , \lceil l_2, l_3' , \lceil l_3, l_4' , \lceil l_2, l_1' , \lceil l_3, l_2' , \lceil l_4, l_3' \end{pmatrix}$$

Clearly there is considerable redundancy in this representation of link sets; ▲$_{\mathcal{L}}$ can be more reasonably represented as:

$$\blacktriangle_{\mathcal{L}} \ = \ l_1 \rightsquigarrow l_2,\ l_2 \rightsquigarrow l_3,\ l_3 \rightsquigarrow l_4,\ l_5 \rightsquigarrow l_5$$

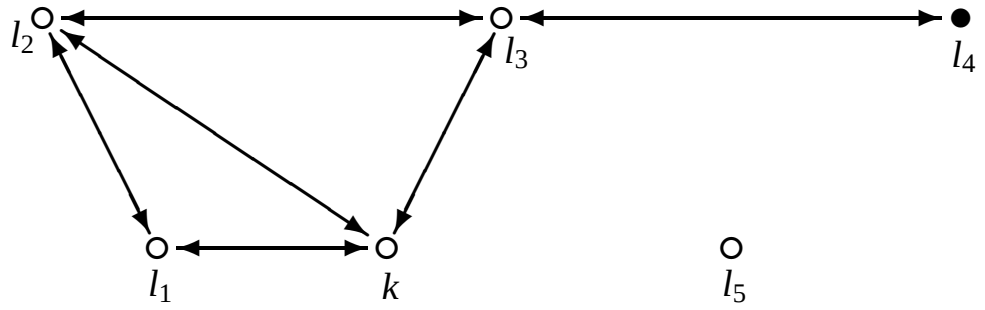
where $l \rightsquigarrow k$ denotes the pair of pairs $\lceil l, k' , \lceil k, l'$ together with the reflexive pairs $\lceil l, l' , \lceil k, k'$; in such cases, a reflexive bi-direcitional link $l \rightsquigarrow l$ would be used for completely disconnected nodes such as l_5. When we apply the reduction semantics to the configuration ▲ ▷ launchNewLoc, the rule (r-comm) is used first to allow the communication of the value l_1 along a, and then (r-newl) can be used to launch the declaration of k to the system level. However, the evaluation of inst($l_2, \mathtt{loc}[\mathtt{a},\ l_1, l_2, l_4, l_5\ ]$, ▲) at launching turns out to be $\mathtt{loc}[\mathtt{a},\ l_1, l_2, l_3\ ]$ because:

- the location from where the k is launched, that is l_3, is automatically connected to k.
- l_1 and l_2 are reachable from the location where the new location k is launched, that is $\blacktriangle \vdash l_1, l_2 \leftsquigarrow l_3$; l_2 is directly accessible from l_3 while l_1 is reachable indirectly through l_2
- l_4 and l_5 are not reachable from l_2; l_4 is dead and thus it is not accessible from any other node; l_5 on the other hand, is completely disconnected.

So the resulting configuration is:

$$\blacktriangle \triangleright (\nu\, k : \mathtt{loc}[\mathtt{a},\ l_1, l_{2,3}\ ])\ \ l_3[\![P\ ^{l_1}/_x\ ]\!]$$

The network $\blacktriangle$ of course does not change, but if we focus on the system $l_2[\![P\ ^{l_1}/_x\ ]\!]$, we see that it is running on the internal network represented by:



■

This distinction between the internal networks used by di erent subsystems has already occurred in the semantics of DπLoc; see the discussion of Example 2.2.2. Nevertheless, we warn to the reader that there will be more serious consequences for DπF, due to the complex nature of reachability that comes into play.

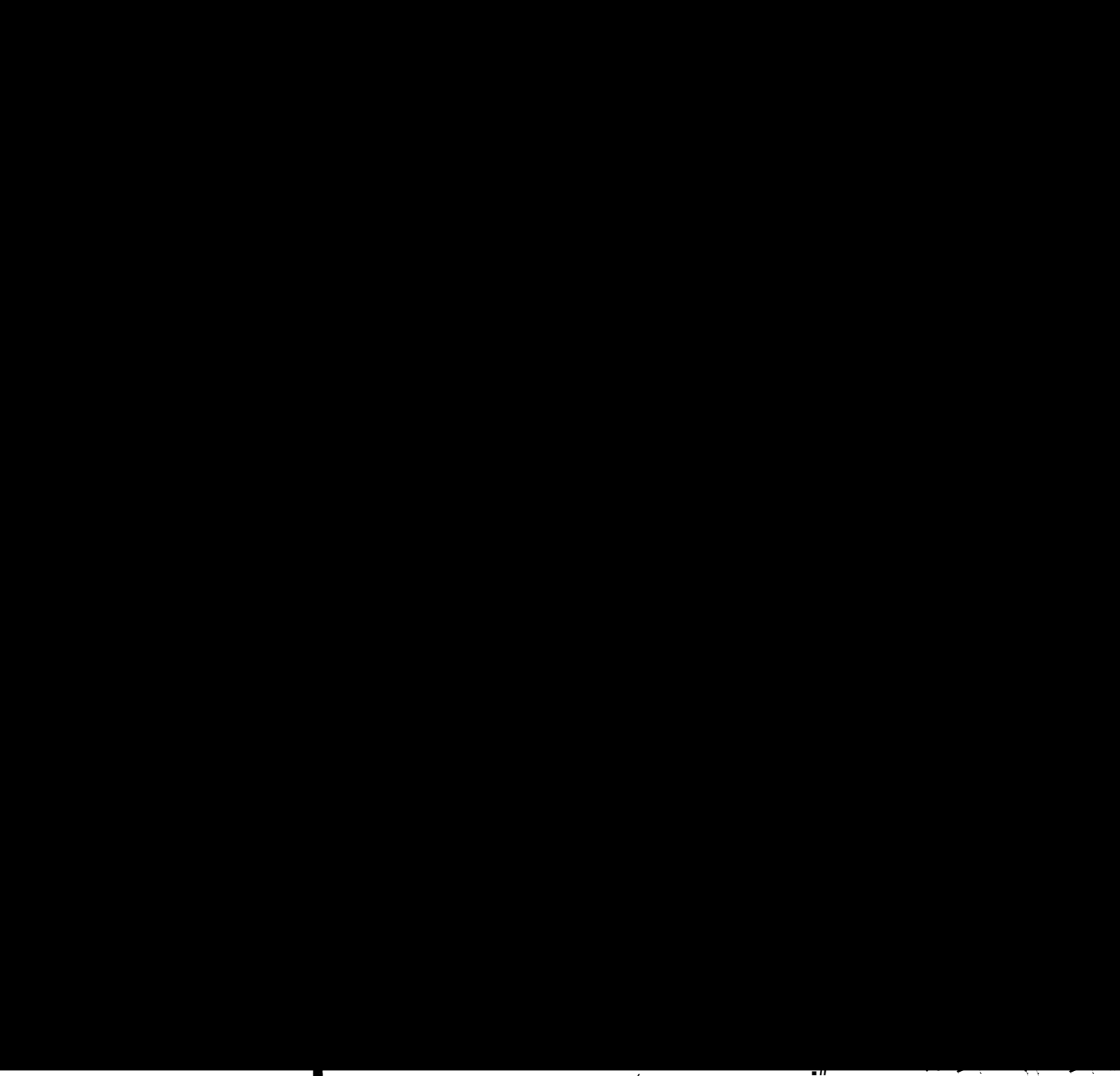
Reduction barbed congruence

The definition of Reduction barbed congruence, Definition 2.2.6, originally developed for DπLoc configurations, can be adapted to apply also to DπF. The formal definition is delayed to Section 4, but let us use the the same notation,

$$\blacktriangle \models M \cong N \tag{7}$$

to indicate that the systems M and N are equivalent relative to the network $\blacktriangle$; the discussion in the section only relies on an intuitive understanding of this concept.

Let us now reconsider the three implementations of a client server discussed in Example 2.2.9, but this time running on a network with explicit links. For con-


k_1 $data?(x,y).$
go $l.y!\lceil f(x)$
monitor $l\vdash$go $k_2, l.y!\lceil f(x)$.

Example 3.2.3. Consider the following three networks,

$\blacktriangle_1 = \blacktriangle_l + k{:}\mathtt{loc}[\mathtt{d}, l\,] =$

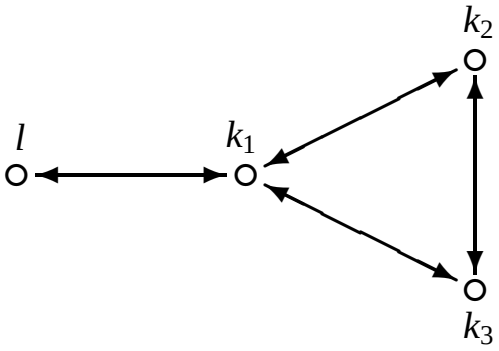
$\blacktriangle_2 = \blacktriangle_l + k{:}\mathtt{loc}[\mathtt{d}, \,] =$

$\blacktriangle_3 = \blacktriangle_l + k{:}\mathtt{loc}[\mathtt{a}, \,] =$

These are the e ective networks for the system $l[\![a!\langle k\rangle \,]\!]$ in the three configurations $\blacktriangle_l \triangleright N_i$, where N_i are defined by

$$N_1 \quad (\nu\, k : \mathtt{loc}[\mathtt{d}, l\,])\, l[\![a!\langle k\rangle \,]\!]$$
$$N_2 \quad (\nu\, k : \mathtt{loc}[\mathtt{d}, \,])N$$

$l[\![a!\langle k_2, k_3\rangle.P]\!]$ is e ectively running on the following *internal* network:



(8)

Let us now see to what knowledge of this internal network can be gained by an observer O at site l, such as $l[\![a?(x,y).O(x,y)]\!]$. Note, that prior to any interaction, O is running on the network $\blacktriangle_l$, and thus, is only aware of the unique location l. By inputting along a, it can gain knowledge of the two names k_1 and k_2, thereby evolving to $l[\![O(k_2,k_3)]\!]$. Yet, even though it is in possession of these two names, it cannot discover the link between them, due to the fact that it is not aware of the local name k_1; in other words, it cannot discover the full extent of the internal network (8) above.

This means that, there is now a di erence between the actual network being used by the system,

Location and Link

$\mathcal{L}$, denoted and defined as:

$$\mathcal{L} \leftrightsquigarrow l_1, \ldots, l_n \stackrel{\text{def}}{=} \bigcup_{i=1} \mathcal{L} \leftrightsquigarrow l_i$$

Finally, if $C = k_1, \ldots, k_n$ is a set of locations representing connections, and l is a location such that $l \notin C$, then $l \rightsquigarrow C$ denotes the component defined as:

$$l \rightsquigarrow C \stackrel{\text{def}}{=} \lceil k, \rceil \ , \lceil l, \rceil \ , \lceil k, \rceil \ , k \quad C \qquad \lceil l, \rceil$$

where locations in C are symmetrically related to l, while l is also related to itself. In the resultant component $l \rightsquigarrow C$, all the locations in C in the component $l \rightsquigarrow C$ are connected as a star formation to l and as a result, all locations are reachable from one another in *at most* two accesses by going through the central node l. Using previous shorthand notation, we could have alternatively defined $l \rightsquigarrow C$ as:

$$l \rightsquigarrow C \stackrel{\text{def}}{=} l \rightsquigarrow k, \ k \quad C$$

■

Lemma 3.3.4 (Subtracting a Component from a Linkset). For any linkset $\mathcal{L}$ and component $\mathcal{K}$ such that $\mathcal{K} \quad \mathcal{L}$, the set $\mathcal{L}/\mathcal{K}$ is also a linkset.

Proof. Immediate from the fact that $\mathcal{L}$ can be expressed as $\bigcup_{i=1}^{n} \mathcal{K}_i$ where $\mathcal{K}$ must be equal to one $\mathcal{K}_i$. Thus, if $\mathcal{K} = \mathcal{K}_j$, the set $\bigcup_{j \neq i=1}^{n} \mathcal{K}_i$, which translates to $\mathcal{L}/\mathcal{K}$, would still be a linkset. □

We now revert our discussion back to e ective network representation, and show how the definition of components facilitates the procedure for extending networks.

Definition 3.3.5 (Augmenting effective networks). Let n be fresh to the network and C be a set of locations such that C **dom**( ). Then we define the operation $+n{:}\mathtt{T}$ as:

$| \quad +n : \mathtt{ch} \stackrel{\text{def}}{=} \lceil \quad n \ , \quad , \quad \langle$

$| \quad +n : \mathtt{loc}[\mathtt{d},$

In the above definition, extending a network with a fresh channel is trivial; adding a fresh dead node is similarly simple, due to the fact that does not represent dead nodes or links to dead nodes explicitly. The only subcase that deserves some explanation is that of adding fresh *live* nodes. A fresh live location is added to either $_O$ or depending on its links. If it is not linked to any observable location, $\subseteq$ **dom**($_O$) $=\emptyset$, then the new fresh location is not reachable from the context and is therefore added to . If, on the other hand, it is linked to an observable location, $\subseteq$ **dom**($_O$) $\neq\emptyset$, then it becomes observable as well. There is also the case where the fresh location is linked to both observable and hidden locations, still represented above by the case where $\subseteq$ **dom**($_O$) $\neq\emptyset$; in such a case, the fresh location, together with any components in the hidden state linked to it, that is $\rightsquigarrow\subseteq$, become observable and thus transfered from to $_O$. The following example elucidates this operation for extending e ective networks.

Example 3.3.6. Consider the e ective network , representing six locations $l, k_1, \ldots, k_5$:

$$= \langle l, k_1, k_2, k_3, k_4, k_5 \ , \ l \rightsquigarrow l \ , \ k_1 \rightsquigarrow k_2, k_2 \rightsquigarrow k_3, k_4 \rightsquigarrow k_4 \rangle$$

Accroding to Definition 3.3.2, l is the only observable location by the context; locations $k_{1..4}$ are alive but not reachable from any observable location while the remaining location, k_5, is dead since it is not in **dom**($_O$ $\mathcal{H}$). Moreover, the linkset representing the hidden state, , can be partitioned into two components, $\mathcal{K}_1 = k_1 \rightsquigarrow k_2, k_2 \rightsquigarrow k_3$ and $\mathcal{K}_2 = k_4 \rightsquigarrow k_4$ whereas the linkset representing the observable state, $_O$, can only be partitioned into one component, itself.

The operation $+k_0 : \mathtt{loc}[\mathtt{a}, l\ ]$ would make the fresh location, k_0, observable in the resultant e ective network since it is linked to, thus reachable from, the observable location l. On the other hand, the operation $+k_0 : \mathtt{loc}[\mathtt{a}, \emptyset\ ]$ would make k_0 hidden since it is a completely disconnected node, just like k_4. The operation $+k_0 : \mathtt{loc}[\mathtt{a}, k_1\ ]$ would still make k_0 hidden in the resultant e ective network, since it is only link to the hidden node k_1. Finally, the operation $+k_0 : \mathtt{loc}[\mathtt{a}, l, k_1\ ]$ intersects with both $_O$ and . This means that k_0 itself becomes observable, but as a side e ect, the components reachable through it, that is $\rightsquigarrow l, k_1 = \mathcal{K}_1$, becomes observable as well. Thus, according to Definition 3.3.5, the updated network translates to:

$$\begin{array}{l} +k_0 : \mathtt{loc}[\mathtt{a}, l, k_1\ ] = \\ \langle \ \mathcal{N} \quad k_0 \ , \quad _O \quad (k_0 \rightsquigarrow l, k_1\ ) \quad (\quad \rightsquigarrow l, k_1\ ), \quad /(\quad \rightsquigarrow l, k_1\ )\rangle \\ \langle \ \mathcal{N} \quad k_0 \ , \quad _O \quad k_0 \rightsquigarrow l, k_0 \rightsquigarrow k_1 \quad \mathcal{K}_1, \quad /\mathcal{K}_1 \quad \rangle \\ \langle \ \mathcal{N} \quad k_0 \ , \quad l \rightsquigarrow k_0, k_0 \rightsquigarrow k_1, k_1 \rightsquigarrow k_2, k_2 \rightsquigarrow k_3 \ , \quad k_4 \rightsquigarrow k_4 \quad \rangle \end{array}$$

■

Table 12. *Network Operational Rules(2) for DπF*

Assuming $\vdash l : \mathbf{alive}$

(l-kill)
$$\frac{}{\triangleright\, l[\![\mathsf{kill}]\!] \xrightarrow{\tau} (\quad l) \triangleright l[\![\mathbf{0}]\!]}$$

(l-brk)
$$\frac{}{\triangleright\, l[\![\mathsf{break}\ k]\!] \xrightarrow{\tau} (l \frown k) \triangleright l[\![\mathbf{0}]\!]}\ \Sigma\ l\ k$$

(l-halt)
$$\frac{}{\triangleright\, N \xrightarrow{\mathsf{kill}:l} (\quad l) \triangleright N}\ \Sigma_{\mathrm{obs}}\ l : \mathbf{alive}$$

(l-disc)
$$\frac{}{\triangleright\, N \xrightarrow{k \nleftrightarrow k} (l \frown k) \triangleright N}\ \Sigma_{\mathrm{obs}}\ l\ k$$

(l-go)
$$\frac{}{\blacktriangle \triangleright\, l[\![\mathsf{go}\ k.P]\!] \xrightarrow{\tau} \blacktriangle \triangleright\, k[\![P]\!]}\ \Delta\ k\ l$$

(l-ngo)
$$\frac{}{\blacktriangle \triangleright\, l[\![\mathsf{go}\ k.P]\!] \xrightarrow{\tau} \blacktriangle \triangleright\, k[\![\mathbf{0}]\!]}\ \Delta \not\sim k\ l$$

(l-ping)
$$\frac{}{\blacktriangle \triangleright\, l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \xrightarrow{\tau} \blacktriangle \triangleright\, l[\![P]\!]}\ \Delta\ k\ l$$

(l-nping)
$$\frac{}{\blacktriangle \triangleright\, l[\![\mathsf{ping}\ k.P \vdash Q\ ]\!] \xrightarrow{\tau} \blacktriangle \triangleright\, l[\![Q]\!]}\ \Delta \not\sim k\ l$$

(l-newc)
$$\frac{}{\blacktriangle \triangleright\, l[\![(\nu\, c : \mathsf{ch})\, P]\!] \xrightarrow{\tau} \blacktriangle \triangleright\, (\nu\, c : \mathsf{ch})\, l[\![P]\!]}$$

(l-newl)
$$\frac{}{\blacktriangle \triangleright\, l[\![(\nu\, k : \mathtt{loc[S,C]})\, P]\!] \xrightarrow{\tau} \blacktriangle \triangleright\, (\nu\, k : \mathtt{loc[S,D]})\, l[\![P]\!]}\ \mathbf{loc}[\mathbf{S},\mathbf{D}] = \mathsf{inst}(\mathbf{loc}[\mathbf{S},\mathbf{C}], l, \Delta)$$

Let us now return to the defintion of our lts for DπF. The transitions between e ective configurations (9) are determined by the rules and axioms already given in Table 6 from Section 2.3, together with the new rules in Table 12 and Table 13. Most of the rules in Table 12 are inherited directly from their counterpart reduction rules in Table 10. The new rule is (l-disc) which introduces the new label $l \nleftrightarrow k$ and models the breaking of a link from the observing context, in the same fashion as (l-fail) in Table 8 models external location killing. Both of these rules are

Table 13. *Contextual Operational Rules(3) for DπF*

(l-open)

$$\frac{+n:\mathsf{T} \triangleright N \xrightarrow{(\tilde{\nu}n\tilde{\mathsf{T}})l:a!\ V} \triangleright N}{\triangleright (\nu\, n:\mathsf{T})N \xrightarrow{(n:\mathsf{U},\tilde{\nu}n\tilde{\mathsf{T}})l:a!\ V} \triangleright N} \; l, a \neq n \quad V,\ \mathsf{U} = \mathsf{T}/\Sigma_{\mathcal{D}}$$

(l-weak)

$$\frac{+n:\mathsf{T} \triangleright N \xrightarrow{(\tilde{\nu}n\tilde{\mathsf{T}})l:a?(V)} \triangleright N}{\triangleright N \xrightarrow{(n:\mathsf{T},\tilde{\nu}n\tilde{\mathsf{T}})l:a?(V)} \triangleright N} \; l, a \neq n \quad V,\ (\Sigma + \tilde{\nu}n\tilde{\mathsf{T}}) \vdash_{\mathsf{obs}} \mathsf{T}$$

(l-rest)

$$\frac{+n:\mathsf{T} \triangleright N \xrightarrow{\mu} +n:\mathsf{U} \triangleright N}{\triangleright (\nu\, n:\mathsf{T})N \xrightarrow{\mu} \triangleright (\nu\, n:\mathsf{U})N} \; n \notin \mathbf{fn}(\mu)$$

(l-rest-typ)

$$\frac{+k:\mathsf{T} \triangleright N \xrightarrow{(\tilde{\nu}n\tilde{\mathsf{T}})l:a!\ V} (\ +\tilde{n}:\tilde{\mathsf{U}})\ +k:\mathsf{U} \triangleright N}{\triangleright (\nu\, k:\mathsf{T})N \xrightarrow{(\tilde{\nu}n\tilde{\mathsf{U}})l:a!\ V} +\tilde{n}:\tilde{\mathsf{U}} \triangleright (\nu\, k:\mathsf{U})N} \; l, a \neq k \quad \mathbf{fn}(\tilde{\mathsf{T}})$$

(l-par-ctxt)

$$\frac{\triangleright N \xrightarrow{\mu} \triangleright N}{\begin{array}{l}\triangleright N \setminus M \xrightarrow{\mu} \triangleright N \setminus M \\ \triangleright M \setminus N \xrightarrow{\mu} \triangleright M \setminus N\end{array}} \; \Sigma \quad M$$

(l-par-comm)

$$\frac{(\ ) \triangleright N \xrightarrow{(\tilde{\nu}n\tilde{\mathsf{T}})l:a!\ V} \triangleright N \qquad (\ ) \triangleright M \xrightarrow{(\tilde{\nu}n\tilde{\mathsf{T}})l:a?(V)} \triangleright M}{\begin{array}{ll}\triangleright N \setminus M & \xrightarrow{\tau} \triangleright (\nu\, \tilde{n}:\tilde{\mathsf{T}})(N \setminus M) \\ \triangleright M \setminus N & \xrightarrow{\tau} \triangleright (\nu\, \tilde{n}:\tilde{\mathsf{T}})(M \setminus N)\end{array}}$$

tleties are required to deal with the interaction between scoped location names and their occurrence in location types. For instance, the rule (l-open) filters the type of scope extruded locations by removing links to locations that are already dead and that will not a ect the e ective network ; this is done through the operation $\mathsf{T}/\ _{\mathcal{D}}$ defined in the Appendix. A side condition is added to (l-weak), $(\ +\tilde{n}:\tilde{\mathsf{T}}) \vdash_{\mathsf{obs}} \mathsf{T}$, limiting the types of imported fresh locations to only contain

locations which are externally accessible, since intuitively, the context can only introduce fresh locations linked to locations it can access. The internal communication rule (l-par-comm) also changes slightly from the one given earlier for DπLoc; communication is defined in terms of the system view (( )) rather than the observer view dictated by . The intuition for this alteration is that internal communication can still occur, even at locations that the observer cannot access, thus we denote the ability to output and input of systems with respect to the maximal observer view ( ). Finally, a completely new rule is (l-rest-typ), which restricts the links exported in location types if one endpoint of the link is still scoped. The utility of this rule is illustrated further in the following example. The rules (l-rest) and (l-par-ctxt) remain unchanged for DπLoc.

Example 3.3.7. Let us revisit Example 3.3.1 to see how the e ect of the observer O on M_1, running on the e ective network $_l$ having only one location l which is alive, that is $(\blacktriangle_l)$. This e ectively means calculating the result of M_1 performing an output on a at l.

It is easy to see that an application of (l-out), followed by two applications of (l-open) gives

$$_l + k_1 : l \triangleright M_1 \xrightarrow{\alpha} {}_l + k_1 : l + k_2 : k_1 + k_3 : k_1, k_2 \triangleright l[\![P]\!] \qquad (10)$$

where M_1 is $(\nu\, k_2 : k_1)(\nu\, k_3 : k_1, k_2)l[\![a!\rceil k_2, k_3^{\prime}.P]\!]$ and α is the action $(k_2 : k_1, k_3 : k_1, k_2)l : a!\rceil k_2, k_3^{\prime}$. Note that (l-rest) can not be applied to this judgement, since k_1 occurs free in the action α. However (10) can be re-arranged to read

$$_l + k_1 : l \triangleright M_1 \xrightarrow{\alpha} {}_l + k_2 : \overline{\ } + k_3 : k_2 + k_1 : l, k_1, k_2 \triangleright l[\![P]\!]$$

moving the addition of location k_1 in the reduct to the outmost position. At this point, (l-rest-typ) can be applied, to give

$$_l \triangleright M_1 \xrightarrow{\beta} {}_l + k_2 : \overline{\ } + k_3 1$$

.3462 Tf 5.98652 0 Td (u7lR57 14.3/R71 10.04 Tf -1 Td 4462 Tf y4 (1)Tj /-2.42.16001 T8.4 z 0 Td (tT

and may be represented diagramatically b

where the links of hidden components are

With these actions we can now define
equivalence between configurations, whic
tual reasoning. Let us write

to mean that there is a (weak) bisimulation
$\triangleright N$ using the current actions. This ne
positive results. For example, for

by giving the relation

Example 3.3.8. Let us consider a slight variation on the system M_1 used in Example 3.3.1 and Example 3.3.7:

$$M_2 \quad (\nu\, k_1 : l\,)(\nu\, k_2 : k_1\,)(\nu\, k_3 : k_1\,)l[\![a!\langle k_2, k_3\rangle.P]\!]$$

again running on the simple (extended) network $_l$. Note that here

In order to obtain a bisimulation equivalence which coincides with reduction barbed congruence it is necessary to abstract away from

Table 14. *The derived lts for DπF*

(l-deriv-1)

$$\frac{\triangleright N \xrightarrow{\mu} \triangleright N}{\triangleright N \xrightarrow{\mu} \triangleright N}\ \mu \quad \tau, \mathsf{kill} : l, l \leftrightarrow k$$

(l-deriv-2)

$$\frac{\triangleright N \xrightarrow{(\tilde{\nu}n\tilde{T})l:a!\ V} \triangleright N}{\triangleright N \mid \triangleright N \xrightarrow{(\tilde{\nu}n\tilde{L})l:a!\ V} \triangleright N}\ \tilde{L} = \mathsf{lnk}(\tilde{\nu}n\tilde{T}, \Sigma)$$

(l-deriv-3)

$$\frac{\triangleright N \xrightarrow{(\tilde{\nu}n\tilde{T})l:a?(V)} \triangleright N}{\triangleright N \mid \triangleright N \xrightarrow{(\tilde{\nu}n\tilde{L})l:a?(V)} \triangleright N}\ \tilde{L} = \mathsf{lnk}(\tilde{\nu}n\tilde{T}, \Sigma)$$

then we do not add anything to either $\mathcal{O}$ or as is the case for $\mathtt{T} = \mathsf{ch}$. Based on this definition of $+n:\mathtt{T}$, we give the following definition for $\mathsf{lnk}(n:\mathtt{T},)$:

$$\mathsf{lnk}(n:\mathtt{T},) \stackrel{\text{def}}{=} \begin{cases} (n \rightsquigarrow \mathsf{C}) \quad (\quad \leftsquigarrow \mathsf{C}) & \text{if } \mathtt{T} = \mathtt{loc}[\mathtt{a}, \mathsf{C}] \text{ and } \mathsf{C} \not\downarrow \mathbf{loc}(\ \mathcal{O}) \\ \emptyset & \text{otherwise} \end{cases}$$

This function is extended to sequences of typed names in the obvious manner:

$$\mathsf{lnk}(n, \tilde{n}:\mathtt{T}, \tilde{\mathtt{T}}, \ ) \ = \ \mathsf{lnk}(n:\mathtt{T}, \ ), \mathsf{lnk}(\tilde{n}:\tilde{\mathtt{T}}, \ )$$

where denotes $+\,n:\mathtt{T}$. ■

These revised actions give rise to a new (weak) bisimulation equivalence over configurations, $\cong$, defined in the usual way, but based on *derived actions*. We use

$$= M \cong N$$

to mean that the configurations $\triangleright M$ and $\triangleright N$ are bisimilar.

Example 3.4.3. Here we re-examine the systems in Example 3.3.8 and Example 3.3.9. We recall that in Example 3.3.8 we had the following actions with respect to the original lts: -

$$l \triangleright M_1 \xrightarrow{\mu_1} \quad + k_2 : \quad + k_3 : \ k_2 \ \triangleright (\nu\, k_1 : \ l, k_2, k_3 \ )\, l[\![P]\!]$$

$$l \triangleright M_2 \xrightarrow{\mu_2} \quad + k_2 : \quad + k_3 : \ \triangleright (\nu\, k_1 : \ l, k_2, k_3 \ )\, l[\![P]\!]$$

But l contains only one accessible node l; extending it with the new node k_2, linked to nothing does not increase the set of accessible nodes. Further increasing it with a new node k_3, linked to the inaccessible k_2 (in the case of $\triangleright M_1$) also leads to no increase in the accessible nodes. Correspondingly, the calculations of $\mathsf{lnk}(k_2 : \ , \ )$ and $\mathsf{lnk}(k_3 : \ k_2 \ , (\ + k_2 : \ ))$ both lead to the empty link set.

Formally, we get the derived action

$$\triangleright M_1 \xrightarrow{\alpha} + k_2\overline{:} + k_3 : k_2 \triangleright (\nu k_1 : l, k_2, k_3)\, l[\![P]\!]$$

where α is $(k_2\overline{:}, k_3\overline{:})l : a!\rceil k_2, k_3\langle$. Similar calculations gives exactly the same derived action from M_2:

$$\triangleright M_2 \xrightarrow{\alpha}$$

4.1 Reduction barbed congruence for DπF

The key to the definition is the isolation of the externally observable information in an extended environment. We use $\mathcal{I}$ to range over *knowledge representations*, pairs $\rceil\mathcal{N},\mathcal{O}\acute{}$ where

- $\mathcal{N}$ is a set of names, as usual divided into **loc**($\mathcal{N}$) and **chan**($\mathcal{N}$),
- $\mathcal{O}$ is a linkset over $\mathcal{N}$.

These can be obtained from e ective networks in the obvious manner:

$$\mathcal{I}(\ ) \stackrel{\text{def}}{=} \rceil_{\mathcal{N}},\ \acute{\mathcal{O}}$$

The key property of this subset of the information in a network representation is that it is preserved by derived actions:

Definition 4.1.1 (Action residuals). We overload the partial function after from Definition 2.3.2 so that now it ranges over knowledge representations $\mathcal{I}$ and external derived actions μ. The newly defined partial function returns knowledge representation, defined as:

- $\mathcal{I}$ after $(\tilde{n} : \tilde{L})l : a!\rceil\acute{V}$ is defined as $\mathcal{I} + \tilde{n} : \tilde{L}$
- $\mathcal{I}$ after $(\tilde{n} : \tilde{L})l : a?(V)$ is defined as $\mathcal{I} + \tilde{n} : \tilde{L}$
- $\mathcal{I}$ after kill : l is defined as $\mathcal{I}\quad l$
- $\mathcal{I}$ after $l \leftrightarrow k$ is defined as $\mathcal{I}\quad l \backsim k$ ■

Proposition 4.1.2. If $\triangleright N \xrightarrow{\ \mu\ } \triangleright N$ where μ is a derived external action, then $\mathcal{I}(\ )$ coincides with $\mathcal{I}(\ )$ after μ

Proof. A straightforward induction on the inference of $\triangleright N \xrightarrow{\ \mu\ } \triangleright N$. □

We find appropriate to use $\mathcal{I}$ as a means to define *the right circumstances*, necessary to allow an action μ to happen. Thus, we define the following predicate. **4.1 The predicate**

Definition 4.1.7 (Typed Relations for DπF). A *typed relation* over extended configurations is a binary relation between such configurations with the property that

$$\triangleright M \ \mathcal{R} \quad \triangleright N \ \text{ implies } \ \mathcal{I}(\ ) = \mathcal{I}(\ )$$

We can mimic the notation in Definition 2.2.3 by writing

$$\mathcal{I} \ \overset{\cdot}{=} \quad \triangleright M \ \mathcal{R} \quad \triangleright N$$

to mean that systems $\triangleright M$ and $\triangleright N$ are related by $\mathcal{R}$ and that both $\mathcal{I}(\ )$ and $\mathcal{I}(\ )$ coincide with $\mathcal{I}$. ■

The definition of contextuality depends on what a given $\mathcal{I}$ allows to be observable; for this we adapt Definition 2.2.4.

Definition 4.1.8 (Observables). For any $\mathcal{I}$ let:

- $\mathcal{I} \vdash l : \mathbf{alive}$, if l is in $\mathbf{dom}(\mathcal{I}_O)$; this implies that l is not only alive, but in the accessible part of any such that $\mathcal{I}(\ )$ coincides with $\mathcal{I}$.
- $\mathcal{I} \vdash l \rightsquigarrow k$, if $l \rightsquigarrow k \in \mathcal{I}_O$; this implies that the link $l \rightsquigarrow k$ is not only alive, but in the accessible part of any such that $\mathcal{I}(\ )$ coincides with $\mathcal{I}$.
- $\mathcal{I} \vdash \mathtt{T}$ if $\mathtt{T}$ is either $\mathtt{ch}$ or $\mathtt{loc}[\mathtt{a}, \mathtt{C}]$ such that $\mathtt{C} \in \mathbf{dom}(\mathcal{I}_O)$.

We can now define the relation $\mathcal{I} \vdash O$ as:

- $\mathcal{I} \vdash l[\![P]\!]$ if $\mathbf{fn}(P) \subseteq \mathcal{I}_N$ and $\mathcal{I} \vdash l : \mathbf{alive}$
- $\mathcal{I} \vdash (\nu\, n{:}\mathtt{T})N$ if $\mathcal{I} \vdash \mathtt{T}$ and $(\mathcal{I} + n{:}\mathtt{T}) \vdash_{\mathsf{obs}} N$
- $\mathcal{I} \vdash M \mid N$ if $\mathcal{I} \vdash M$ and $\mathcal{I} \vdash N$

We can now adapt the notation of Definition 2.2.4 as:

$$\blacktriangle \vdash_{\mathsf{obs}} l{:}\mathbf{alive},\ l \rightsquigarrow k,\ \mathtt{T},\ O \ \overset{\text{def}}{=} \ \mathcal{I}(\ (\blacktriangle)) \vdash l{:}\mathbf{alive},\ l \rightsquigarrow k,\ \mathtt{T},\ O$$

$$\vdash_{\mathsf{obs}} l{:}\mathbf{alive},\ l \rightsquigarrow k,\ \mathtt{T},\ O \ \overset{\text{def}}{=} \ \mathcal{I}(\ ) \vdash l{:}\mathbf{alive},\ l \rightsquigarrow k,\ \mathtt{T},\ O$$

The intuition of $\blacktriangle \vdash_{\mathsf{obs}} O$ and $\vdash_{\mathsf{obs}} O$ are still the same as that of Definition 2.2.4: an observer O is restricted to the observable network. However, the updated definition reflects the fact that the observable network is now not only defined in terms of live nodes but live, *reachable* nodes. ■

As a result of this adaptation, we can carry forward to the section the definition of contextual typed relations, defined earlier in 2.2.5. However, before we go on an define reduction barbed congruence for DπF terms, we need also to update the notion of a barb; a barb is observable by the context in DπF, if the location at which the barb occurs is alive *and observable*.

Definition 4.1.9. $\triangleright N \downarrow_{a@l}$ denotes an *observable barb* exhibited by the configuration $\triangleright N$, on channel a at location l. Formally, it means that $\blacktriangle(\ )\triangleright N \equiv \blacktriangle(\ )\triangleright N$ for some $\triangleright N$ such that $N \equiv M \setminus l[\![a!\tilde{V}.Q]\!]$ and $\mathcal{I} \vdash_{\mathsf{obs}} l$: **alive**. ■

With these modifications, Definition 2.2.6 can be applied to obtain a definition of *reduction barbed congruence* for DπF, which we denote by

$$\mathcal{I} \models\ _1\triangleright M_1 \cong\ _2\triangleright M_2 \text{ whenever } \mathcal{I}(\ _1) = \mathcal{I}(\ _2)$$

Note that this enables us to compare aribtrary configurations, $_1\triangleright M_1$ and $_2\triangleright M_2$, but it can be specialised to simply comparing systems running on the same network. Let us write

$$\models M \cong N$$

to mean that $\mathcal{I}(\ ) \models\ \triangleright M \cong\ \triangleright N$. Then, for example, the informal notation (7) used in Section 3.2 can be taken to mean

$$(\blacktriangle) \vdash M \cong N$$

The second main result of the paper can now be stated:

Theorem 4.1.10. Suppose $\mathcal{I}(\ _1) = \mathcal{I}(\ _2) = \mathcal{I}$, for any e ective configurations $_1\triangleright M_1$, $_2\triangleright M_2$ in DπF. Then:

$\mathcal{I} \models\ _1\triangleright M_1 \cong\ _1\triangleright M_1$ if and only if

We start by proving that the derived lts is *closed* over well formed e ective configurations. We prove this with the aid of the following lemma, stating that there is also a special relationship between silent actions and residual networks.

Lemma 4.2.1. Internal transitions do not change the state of the network, unless a kill or a break l process in the configuration itself is consumed. Stated otherwise, if $\triangleright N \xrightarrow{\tau} \triangleright N$ then is either:-

1.
2. l
3. $l \rightsquigarrow k$

Proof. A straightforward induction on the inference of $\triangleright N \xrightarrow{\tau} \triangleright N$. □

Proposition 4.2.2 (Closure). The derived lts given in Definition 3.4.1 forms a binary relation between well-defined e ective configurations. Stated otherwise, if $\vdash N$ and $\triangleright N \xrightarrow{\mu} \triangleright N$ then $\vdash N$.

Proof. By case analysis on the form of μ. We use Proposition 4.1.2 when μ is an external action and Lemma B.0.1 in sub-cases where we need to show that $+n:\mathsf{T}$ is still a valid e ective network. When μ is an internal action, $\mu = \tau$, we use Lemma 4.2.1. □

The next important sanity check for our lts is that our formulation of *internal activity*, namely $\xrightarrow{\tau}$, is in agreement, in some sense, with the reduction semantics.

Proposition 4.2.3 (Reductions correspond to τ-actions).

$\vert$	$\triangleright N \quad \triangleright N$ implies $\triangleright N \xrightarrow{\tau} \triangleright N$ for some $N \quad - N$
$\vert$	$\triangleright N \xrightarrow{\tau} \triangleright N$ implies $\triangleright N \quad \triangleright N$

Proof. The proof for the first clause is by induction on why $\triangleright N \quad \triangleright N$. The proof for the second clause is also by induction. Since the internal transition rule (l-par-comm) is defined in terms of input and output actions, we make use of Lemma 4.1.5 in our induction. □

We now embark on the main task of this section, that of showing that our bisimulation, $\cong$, is contextual. This proof relies heavily on the Composition and Decomposition Lemmas stated below, explaining how actions can be composed of, or decomposed into, other actions. Both Composition and Decomposition Lemmas make use of the following (specific) lemma, which is a slight variation on Proposition 4.1.6; we note that we could not have used Proposition 4.1.6 in this case because the type of the bound input action changes as shown below.

Lemma 4.2.4 (Input actions and the maximal observer view).

| If $\triangleright N \xrightarrow{(\tilde{n}\tilde{K})l:a?(V)} \triangleright N$ then $(\ ) \triangleright N \xrightarrow{(\tilde{n}\tilde{L})l:a?(V)} \triangleright N$ where $\tilde{K} = \tilde{L}/\mathbf{dom}(\ \ )$.

| If $(\ ) \triangleright N \xrightarrow{(\tilde{n}\tilde{L})l:a?(V)} \triangleright N$ and $\mathcal{I}(\ ) \vdash l:\mathbf{alive}$ then $\triangleright N \xrightarrow{(\tilde{n}\tilde{K})l:a?(V)} \triangleright N$ where $\tilde{K} = \tilde{L}/\mathbf{dom}(\ \ )$.

Proof. The proof uses Lemma 4.1.5 to infer the structure of N and the progresses by induction on the structure of N, similar to the proof for Proposition 4.1.6. □

Lemma 4.2.5 (Composition).

| Suppose $\triangleright M \xrightarrow{\mu} \triangleright M'$. If $\vdash N$ for arbitrary system N, then $\triangleright (M \mid N) \xrightarrow{\mu} \triangleright (M' \mid N)$

Hence, by (13), (14), (l-par-comm) and (l-deriv-1) we conclude

$$\triangleright\, M \setminus N \xrightarrow{\tau} \triangleright\, (\nu\, \tilde{n} : \tilde{\mathrm{T}}) M \setminus N$$
$$\triangleright\, N \setminus M \xrightarrow{\tau} \triangleright\, (\nu\, \tilde{n} : \tilde{\mathrm{T}}) N \setminus M$$

as required. □

Lemma 4.2.6 (Decomposition). Suppose $\triangleright\, M \setminus N \xrightarrow{\mu} \triangleright\, M$ where $\vdash_{\mathsf{obs}} M$ or $\vdash_{\mathsf{obs}} N$. Then, one of the following conditions hold:

1. M is $M \setminus N$, where $\triangleright\, M \xrightarrow{\mu} \triangleright\, M$.
2. M is $M \setminus N$ and $\triangleright\, N \xrightarrow{\mu} \triangleright\, N$.
3. M is $(\nu\, \tilde{n} : \tilde{\mathrm{T}}) M \setminus N$, μ is τ,

.95906 Td (00)Tj /R29 14.3462 Tf 7.56001N
09he008
2. MN

Moreover, by (20), (21) and Lemma 4.2.4 we deduce

$$(\ ) \triangleright N \xrightarrow{(\tilde{\nu}\tilde{n}:\tilde{K})l:a?(V)} (\ ) + \tilde{n}:\tilde{T} \triangleright N$$

where $\tilde{K} = \tilde{L}/\mathbf{dom}(\quad)$ as required. □

We now turn our attention to the actual proof for the main proposition of this section, namely that bisimulation, $\cong$, is contextual. We prove this by inductively defining the largest contextual relation whose base element are bisimilar configurations and then show its closure with respect to our derived actions. Based on such a proof, we still require three (specific) lemmas to help us stitch up this proof and guarantee closure. The first lemma is prompted by the first two conditions of the Decomposition Lemma 4.2.6, namely that observing code may alter the state of the network by inducing failure. We thus need the following lemma to guarantee closure.

Lemma 4.2.7. Suppose $_1 \triangleright M_1 \cong {}_2 \triangleright M_2$. Then there exists some M_2, M_2 such that:

| $_2 \triangleright M_2 \stackrel{\widehat{\tau}}{\Longrightarrow} {}_2 \triangleright M_2$ and $(\ _2 \quad l) \triangleright M_2 \stackrel{\tau}{\Longrightarrow} (\ _2 \quad l) \triangleright M_2$ such that $(\ _1 \quad l) \triangleright M_1 \cong (\ _2 \quad l) \triangleright M_2$

| $_2 \triangleright M_2 \stackrel{\widehat{\tau}}{\Longrightarrow} {}_2 \triangleright M_2$ and $(\ _2 \quad l$

Lemma 4.2.9 (Observers and Network extensions). If $+n:\mathtt{U} \vdash_{\mathsf{obs}} O$ where $\vdash_{\mathsf{obs}} \mathtt{U}$, that is n is only linked to locations in the observable part of and thus no hidden state is revealed as a result of the extension, then $+n:\mathtt{T} \vdash_{\mathsf{obs}} O$ for any $\mathtt{T}$ where $\mathtt{U} = \mathtt{T}/\mathbf{dom}(\quad)$.

Proof. The proof progresses by a simple induction on the structure of O. □

We are finally in a position to prove that our bisimulation, $\cong$, is a contextual relation, according to Definition 2.2.5.

Proposition 4.2.10 (Contextuality of Behavioural Equivalence). If two configurations are bisimilar, they are also bisimilar under any context. Stated otherwise, $\mathcal{I} = \quad_1 \triangleright M_1 \quad \cong \quad_2 \triangleright M_2$ implies that for $\mathcal{I} \vdash O, \mathtt{T}$ and n fresh in $\mathcal{I}$ we have:

- $\mathcal{I} = \quad_1 \triangleright M_1 \setminus O \quad \cong \quad_2 \triangleright M_2 \setminus O$ and $\mathcal{I} = \quad_1 \triangleright O \setminus M_1 \quad \cong \quad_2 \triangleright O \setminus M_2$
- $\mathcal{I} + n:\mathtt{T} = \quad_1 + n:\mathtt{T} \triangleright M_1 \quad \cong \quad_2 + n:\mathtt{T} \triangleright M_2$

Proof. The proof progresses by the inductive definition a relation $\mathcal{R}$ as the largest typed relation over configurations satisfying:

$$\mathcal{R} = \left\{ \begin{array}{ll} \left(\quad_1 \triangleright M_1, \quad_2 \triangleright M_2\right) & \quad_1 \triangleright M_1 \cong \quad_2 \triangleright M_2 \\ \left(\quad_1 \triangleright M_1 \setminus O, \quad_2 \triangleright M_2 \setminus O\right) & \\ \left(\quad_1 \triangleright O \setminus M_1, \quad_2 \triangleright O \setminus M_2\right) & \quad_1 \triangleright M_1 \mathcal{R} \end{array} \right.$$

Lemma 4.2.11. If $\Gamma_1 \triangleright M_1 \mathcal{R} \Gamma_2 \triangleright M_2$, then there exist some M_2', M_2'' such that:

- $\Gamma_2 \triangleright M_2 \Longleftarrow^{\hat{\tau}} \Gamma_2 \triangleright M_2'$ and $\Gamma_2 \; l \triangleright M_2 \Longleftarrow^{\tau} \Gamma_2 \; l \triangleright M_2''$, where $\Gamma_1 \; l \triangleright M_1 \mathcal{R} \Gamma_2 \; l \triangleright M_2''$
- $\Gamma_2 \triangleright M_2 \Longleftarrow^{\hat{\tau}} \Gamma_2 \triangleright M_2'$ and $\Gamma_2 \; l \rightsquigarrow k \triangleright M_2 \Longleftarrow^{\tau} \Gamma_2 \; l \rightsquigarrow k \triangleright M_2''$, where $\Gamma_1 \; l \rightsquigarrow k \triangleright M_1 \mathcal{R} \Gamma_2 \; l \rightsquigarrow k \triangleright M_2''$

The proof for the above is by induction on why $\Gamma_1 \triangleright M_1 \mathcal{R} \Gamma_2 \triangleright M_2$; the base case follows from Lemma 4.2.7 and the three inductive cases are straightforward.

To prove that $\mathcal{R}$ is a bisimulation, we take an arbitrary $\mathcal{I} \setminus = \Gamma_1 \triangleright M_1 \; \mathcal{R} \; \Gamma_2 \triangleright M_2$ and any action $\Gamma_1 \triangleright M_1 \xrightarrow{\mu} \Gamma_1 \triangleright M_1'$; we then have to show that $\Gamma_2 \triangleright M_2$ can match this move by performing a weak action $\Gamma_2 \triangleright M_2 \Longrightarrow^{\hat{\mu}} \Gamma_2 \triangleright M_2'$ such that $\mathcal{I} \setminus = \Gamma_1 \triangleright M_1' \; \mathcal{R} \; \Gamma_2 \triangleright M_2'$. The proof progress by induction on why $\mathcal{I} \setminus = \Gamma_1 \triangleright M_1 \; \mathcal{R} \; \Gamma_2 \triangleright M_2$; The first case, that is if $\mathcal{I} \setminus = \Gamma_1 \triangleright M_1 \cong \Gamma_2 \triangleright M_2$ is immediate; the remaining three cases require a bit more work. We here focus on the second case, where

$$\Gamma_1 \triangleright M_1 \setminus O \; \mathcal{R} \; \Gamma_2 \triangleright M_2 \setminus O \text{ because } \mathcal{I} \setminus = \Gamma_1 \triangleright M_1 \; \mathcal{R} \; \Gamma_2 \triangleright M_2 \text{ and } \mathcal{I} \vdash O \tag{23}$$

which is also the most involving and leave the remaining two cases for the interested reader.

We thus assume $\Gamma_1 \triangleright M_1 \setminus O \xrightarrow{\mu} \Gamma_1 \triangleright M_1'$. We decompose this action using the Decomposition Lemma 4.2.6 and focus on the most difficult case, where

$$M_1 \text{ is } (\nu \tilde{n}{:}\tilde{\mathrm{T}}) M_1' \setminus O \text{ , } \mu \text{ is } \tau \text{ and } \Gamma_1 = \Gamma_1' \tag{24}$$

$$\Gamma_1 \triangleright M_1' \vdash \xrightarrow{(\nu \tilde{n}:\tilde{\mathrm{L}})\, l:a!\, V} \Gamma_1' + \tilde{n}{:}\tilde{\mathrm{T}} \triangleright M_1'' \tag{25}$$

$$\Gamma_1 \triangleright O \vdash \xrightarrow{(\nu \tilde{n}:\tilde{\mathrm{K}})\, l:a?(V)} \Gamma_1' + \tilde{n}{:}\tilde{\mathrm{U}} \triangleright O \text{ where } \tilde{\mathrm{U}} = \tilde{\mathrm{T}} / \mathbf{dom}(\Gamma_1') \tag{26}$$

From (23) and (25) we derive the matching weak action

$$\Gamma_2 \triangleright M_2 \Longrightarrow^{(\nu \tilde{n}:\tilde{\mathrm{L}})\, l:a!\, V} \Gamma_2' + \tilde{n}{:}\tilde{\mathrm{W}} \triangleright M_2'' \; \mathcal{R} \; \Gamma_1' + \tilde{n}{:}\tilde{\mathrm{T}} \triangleright M_1'' \tag{27}$$

where we note the different types $\tilde{\mathrm{T}}$ and $\tilde{\mathrm{W}}$ at which the two networks Γ_1' and Γ_2' are updated; there are updates to the hidden part of the networks which we abstract away in the linktype $\tilde{\mathrm{L}}$. From (27) and the hypothesis of (l-deriv-2) we obtain

$$\Gamma_2 \triangleright M_2 \Longleftarrow^{(\nu \tilde{n}:\tilde{\mathrm{W}})\, l:a!\, V} \Gamma_2' + \tilde{n}{:}\tilde{\mathrm{W}} \triangleright M_2''$$

which can be decomposed as

$$ {}_2 \triangleright M_2 \quad \Longrightarrow \quad {}_2 \triangleright M_2 \tag{28}$$

$$ {}_2 \triangleright M_2 \quad \xrightarrow{(\tilde{n}:\tilde{W})l:a!\ V} \quad {}_2 + \tilde{n}:\tilde{W} \triangleright M_2 \tag{29}$$

$$ {}_2 + \tilde{n}:\tilde{W} \triangleright M_2 \quad \Longrightarrow \quad {}_2 + \tilde{n}:\tilde{W} \triangleright M_2 \tag{30}$$

From (28), $\mathcal{I} \vdash O$ and (l-par-ctxt) we get

$$ {}_2 \triangleright M_2 \wr O \Longrightarrow {}_2 \triangleright M_2 \wr O \tag{31}$$

From the fact that $\mathcal{I}(\ {}_1) = \mathcal{I}(\ {}_2)$ and $\mathcal{I}(\ {}_1 + \tilde{n} : \tilde{T}) = \mathcal{I}(\ {}_2 + \tilde{n} : \tilde{W})$ from (27) we know that the visible part of ${}_2$ and ${}_2$ did not change as a result of the silent transitions in (28) and (30) and thus

$$ \mathcal{I}(\ {}_2) = \mathcal{I}(\ {}_2) = \mathcal{I}(\ {}_2) = \mathcal{I}(\ {}_1) \tag{32}$$

and by (32), (26) and Lemma 4.1.6 we get

$$ {}_2 \triangleright O \xrightarrow{(\tilde{n}:\tilde{U})l:a?(V)} {}_2 + \tilde{n}:\tilde{U} \triangleright O \quad \text{where } \tilde{U} = \tilde{W}/\mathbf{dom}(\ {}_2\ ) \tag{33}$$

At this point we note that from (32) and (23) we derive

$$ {}_2 \vdash_{\text{obs}} O \tag{34}$$

and from (33), (34), Lemma 4.2.8 and Lemma 4.2.9 we obtain

$$ \mathcal{I}(\ {}_2 + \tilde{n}:\tilde{U}) \vdash O \quad \text{and} \quad \mathcal{I}(\ {}_2 + \tilde{n}:\tilde{W}) \vdash O \tag{35}$$

Combining the derived action of (29) using (l-deriv-2), the derived action of (33) using (l-deriv-3), (34), and the Composition Lemma 4.2.5, we obtain

$$ {}_2 \triangleright M_2 \wr O \xrightarrow{\tau} {}_2 \triangleright (\nu\, \tilde{n}:\tilde{W}) M_2 \wr O \tag{36}$$

From (30), (35) and (l-par-ctxt) we obtain

$$ {}_2 + \tilde{n}:\tilde{W} \triangleright M_2 \wr O \Longrightarrow {}_2 + \tilde{n}:\tilde{W} \triangleright M_2 \wr O $$

and by applying (l-rest) we get

$$ {}_2 \triangleright (\nu\, \tilde{n}:\tilde{W}) M_2 \wr O \Longrightarrow {}_2 \triangleright (\nu\, \tilde{n}:\tilde{W}) M_2 \wr O \tag{37}$$

and thus by combining (31), (36) and (37) and then applying (l-deriv-1) we obtain the matching move

$$ {}_2 \triangleright M_2 \wr O \stackrel{\tau}{\Longrightarrow} {}_2 \triangleright (\nu\, \tilde{n}:\tilde{W}) M_2 \wr O \tag{38}$$

The only thing remaining is to show that the two residuals are in $\mathcal{R}$, that is

$$ {}_1 \triangleright (\nu\, \tilde{n}:\tilde{T}) M_1 \wr O \quad \mathcal{R} \quad {}_2 \triangleright (\nu\, \tilde{n}:\tilde{W}) M_2 \wr O $$

From (27) we know

$$ \mathcal{I} \models_{\wr} {}_1 + \tilde{n}:\tilde{T} \triangleright M_1 \quad \mathcal{R} \quad {}_2 + \tilde{n}:\tilde{W} \triangleright M_2 \tag{39}$$

and from (35) and (39) we deduce $\mathcal{I} \nvdash O$ and thus from the definition of $\mathcal{R}$ we obtain

$$\mathcal{I} = {}_1 +$$

$\mathtt{T})M \not\rightarrow_{a@l}$ implies $(\ _M + n:\mathtt{T}) \triangleright M \not\rightarrow_{a@l}$. Finally, contextuality is also trivial. As an example, assume $\mathcal{I}(\ _M) \vdash O$ and we have to show that

$$_M \triangleright O \setminus (\nu\, n:\mathtt{T})(M)\ \mathcal{R}\ \ _N \triangleright O \setminus (\nu\, n:\mathtt{U})N.$$

It is clear that $_M + n:\mathtt{T} \vdash_{\mathsf{obs}} O$ and $_N + n:\mathtt{U} \vdash_{\mathsf{obs}} O$ and thus by contextuality of $\cong$, we have $(\ _M + n:\mathtt{T}) \triangleright O \setminus M \cong (\ _N + n:\mathtt{U}) \triangleright O \setminus N$ from which the result follows. □

Our external actions can a ect both the system part of o2.23 0 Td (both)Tj 29.8419

x if and only if $\mathcal{I}(\ ) = \mathcal{I}$. Its implementation is based on the state observing construct ping $l.P\vdash Q$; the sub-process, $verObs_k(x)$, first checks that all *inaccessible* locations in $\mathcal{I}$, expressed as $\mathcal{I} + l\colon\neg$ below, are indeed inaccessible and then checks that the *accessible* locations, expressed as $\mathcal{I} + l : \mathtt{L}$ where $\mathtt{L} \neq \neg$, satisfy the state declared in $\mathcal{I}$, using the sub-process $verLoc_k(x, y_1, y_2, z)$. This last subprocess, goes to the parameterised location x and checks that all its live connections and dead connections correspond to y_1 and y_2 respectively, returning a output on channel z if it is the case. The following lemma formalises the intuition that when run at an appropriate location, $verStat_k(x)$ does satisfy the intended behaviour.

$$verStat_k(x) \qquad (\nu\, sync)\Big(\mathbf{0} \mid verObs_k(sync) \mid verNObs(\mathbf{loc}$$

bound output. In order to complete the proof, we also require the following lemma.

Lemma 4.3.4. $+n : \mathtt{T} \triangleright N$ $+n : \mathtt{T} \triangleright N$ where $n \notin \mathbf{fn}(N)$ i $\vdash N$ and $\triangleright N$ $\triangleright N$

Proof. The proofs are by induction on the structure of N for $\vdash N$ and by induction on the derivations of $\triangleright N$ $\triangleright N$ and $+n : \mathtt{T} \triangleright N$ $+n : \mathtt{T} \triangleright N$. □

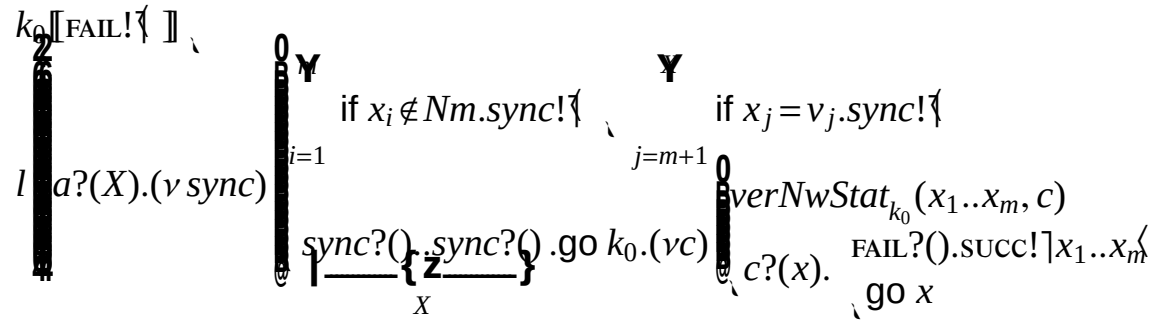
Proposition 4.3.5 (Definability). Assume that for an arbitrary network representation , the network $_+$ denotes:

$$_+ = \quad + k_0 : \mathtt{loc}[\mathtt{a}, \mathbf{dom}(\ _O)], \textsc{succ} : \mathtt{ch}, \textsc{fail} : \mathtt{ch}$$

where k_0, SUCC and FAIL are fresh to $_N$. Thus, for every external action μ and network representation , every non-empty finite set of names Nm where $_N$ Nm, every fresh pair of channel names SUCC, FAIL $\notin Nm$, and every fresh location name $k_0 \notin Nm$ connected to all observable locations in $_O$, there exists a system $T^{\mu}(Nm, \textsc{succ}, \textsc{fail}, k_0)$ with the property that $_+ \vdash_{\mathsf{obs}} T^{\mu}(Nm, \textsc{succ}, \textsc{fail}, k_0)$, such that:

1. $\triangleright N \xrightarrow{\ \mu\ } +\mathbf{bn}(\mu) \triangleright N$ implies
   $_+ \triangleright N \mid T^{\mu}(Nm, \textsc{succ}, \textsc{fail}, k_0) \Longleftarrow\ _+ \triangleright (\nu\, \mathbf{bn}(\mu))\, N \mid k_0[\![\textsc{succ}!\rceil \mathbf{bn}(\mu)]\!]$
2. $_+ \triangleright N \mid T^{\mu}(Nm, \textsc{succ}, \textsc{fail}, k_0) \Longleftarrow\ _+ \triangleright N$,
   where $_+ \triangleright N \not\xrightarrow{\ \mathbf{succ}@k_0\ }$, $_+ \triangleright N \not\xrightarrow{\ \mathbf{fail}@k_0\ }$ implies that
   $N - (\nu\, \mathbf{bn}(\mu))N \mid k_0[\![\textsc{succ}!\rceil \mathbf{bn}(\mu)]\!]$ N $+\mathbf{bn}(\mu) \triangleright N$ implies the
)

is



Subsequently we derive the sequence of reductions

$$\begin{array}{l} {}_{+}\triangleright N \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] \\ \qquad {}_{+}\triangleright N \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\textsc{fail}?().\textsc{succ}!\langle\,\rangle]\!] \\ \qquad\qquad {}_{+}\triangleright N \;\backslash\; k_0[\![\textsc{succ}!\langle\,\rangle]\!] \end{array} \tag{43}$$

Combining the reductions in (40), (42) and (43) we prove the first clause.

For the second clause, the set of barbs ${}_{+}\triangleright N \,/\, \mathbf{succ}@k_0$, ${}_{+}\triangleright N \,/\, \mathbf{fail}@k_0$ can *only* be obtained through the sequence of reductions

$$\begin{array}{ll} {}_{+}\triangleright N \;\backslash\; l[\![\mathsf{kill}]\!] \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] =\!\Rightarrow & (44) \\ \quad {}^{1}_{+}\triangleright N^1 \;\backslash\; l[\![\mathsf{kill}]\!] \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] & \\ \quad\quad {}^{1}_{+}\triangleright N^1 \;\backslash\; l[\![\mathsf{kill}]\!] \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] =\!\Rightarrow & (45) \\ \quad\quad\quad {}^{2}_{+}\triangleright N^2 \;\backslash\; l[\![\mathsf{kill}]\!] \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] & (46) \\ \quad\quad\quad\quad {}^{2}_{+}\;\; l\triangleright N^2 \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] =\!\Rightarrow & (47) \\ \quad\quad\quad\quad\quad {}^{3}_{+}\;\; l\triangleright N^3 \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] & \\ \quad\quad\quad\quad\quad\quad {}^{3}_{+}\;\; l\triangleright N^3 \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\textsc{fail}?().\textsc{succ}!\langle\,\rangle]\!] =\!\Rightarrow & (48) \\ \quad\quad\quad\quad\quad\quad\quad {}^{4}_{+}\;\; l\triangleright N^4 \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\textsc{fail}?().\textsc{succ}!\langle\,\rangle]\!] & \\ \quad\quad\quad\quad\quad\quad\quad\quad {}^{4}_{+}\;\; l\triangleright N^4 \;\backslash\; k_0[\![\textsc{succ}!\langle\,\rangle]\!] =\!\Rightarrow & (49) \\ \quad\quad\quad\quad\quad\quad\quad\quad\quad {}_{+}\triangleright N \;\backslash\; k_0[\![\textsc{succ}!\langle\,\rangle]\!] & \end{array}$$

From (46) and Lemma 4.3.2 we deduce

$$\begin{array}{l} {}^{2}_{+}\triangleright N^2 \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] \vdash^{\mathsf{kill}:l} \\ \qquad\qquad {}^{2}_{+}\;\; l\triangleright N^2 \;\backslash\; k_0[\![\textsc{fail}!\langle\,\rangle]\!] \;\backslash\; k_0[\![\text{ping } l.\vdash\textsc{fail}?().\textsc{succ}!\langle\bullet\,\rangle]\!] \end{array}$$

and by the inductive hypothesis of (l-par-ctxt), the fact that $\mathcal{I}({}^{2}_{+})$ allows $\mathsf{kill} : l$ and Proposition 4.1.6, we derive

$${}^{2}\triangleright N^2 \;\overset{\mathsf{kill}:l}{\Longrightarrow}\; {}^{2}\;\; l\triangleright N^2 \tag{50}$$

From (44), (45), (47), (48) and (49) and (r-par-ctxt) obtain

$$\begin{array}{llll} {}_{+}\triangleright N =\!\Rightarrow & {}^{1}_{+}\triangleright N^1 =\!\Rightarrow & {}^{2}_{+}\triangleright N^2 & \\ {}^{2}\;\; l_{+}\triangleright N^2 =\!\Rightarrow & {}^{3}_{+}\triangleright N^3 =\!\Rightarrow & {}^{4}_{+}\triangleright N^4 =\!\Rightarrow & {}_{+}\triangleright N \end{array} \tag{51}$$

and from (51) and Lemma 4.3.4 we obtain

$$\begin{array}{llll} \triangleright N =\!\Rightarrow & {}^{1}\triangleright N^1 =\!\Rightarrow & {}^{2}\triangleright N^2 & \\ {}^{2}\;\; l\triangleright N^2 =\!\Rightarrow & {}^{3}\triangleright N^3 =\!\Rightarrow & {}^{4}\triangleright N^4 =\!\Rightarrow & \triangleright N \end{array} \tag{52}$$

Finally, using Proposition 4.2.3 to convert the reductions in (52) into weak silent actions and merging these with (50) we obtain as required

$$\triangleright N \;\overset{\mathsf{kill}:l}{=\!\Rightarrow}\; \triangleright N$$

□

The result of Proposition 4.3.5 means that intuitively we can provoke the action $\triangleright N \xleftarrow{\mu}$ $\triangleright N$ by extending with a fresh location k_0 and fresh channels succ and fail and placing N in parallel with $T^{\mu}(Nm, \text{succ}, \text{fail}, k_0)$ for a suitably chosen Nm. But in the case of actions where $\mathbf{bn}(\mu) \neq \emptyset$ we do not get precisely the residual $\triangleright N$ but instead $\nu \mathbf{bn}(\mu)\, N \mid k_0[\![\text{succ}!\langle \mathbf{bn}(\mu) \rangle]\!]$ where $\mathbf{bn}(\mu) = \emptyset$. We therefore state and prove a variant the extrusion lemma in [9, 8], which enables us to recover the residual $\triangleright N$ from $\nu \mathbf{bn}(\mu) N \mid k_0[\![\text{succ}!\langle \mathbf{bn}(\mu) \rangle]\!]$; this lemma uses the preliminary lemma below, which we chose to extract as an important step of the proof.

Lemma 4.3.6. Suppose δ, k_0 are fresh to the systems M, $k[\![P(X)]\!]$. Suppose also that $k \notin \mathsf{C}$. Then:

$$\mathcal{I} = (\nu\, \tilde{n}{:}\tilde{\mathsf{T}})(M \mid k[\![P(\tilde{n})]\!]) \;\cong\; (\nu\, \tilde{n}{:}\tilde{\mathsf{T}})(\nu\, \delta{:}\mathsf{ch})(\nu\, k_0{:}\mathtt{loc}[\mathsf{a}, \mathsf{C}])(M \mid k_0[\![\delta!\langle \tilde{n} \rangle]\!] \mid k_0[\![\delta?(X).\mathtt{go}\, k.P(X)]\!])$$

Proof. We note that the left hand system can be obtained from the right hand system in two reductions, communication on δ and migrating from k_0 to k, that cannot be interfered with by any context. It is easy to come up with a bisimulation proving that the two systems are reduction barbed congruent. □

Lemma 4.3.7 (Extrusion). Suppose $\text{succ}, \text{fail}, k_0$ are fresh to the network representations M, N, M and N. Then

$$\mathcal{I} = \nu + \tilde{n}{:}\tilde{\mathsf{T}} \triangleright M \mid k_0[\![\text{succ}!\langle \tilde{n} \rangle]\!] \cong \nu + \tilde{n}{:}\tilde{\mathsf{U}} \triangleright N \mid k_0[\![\text{succ}!\langle \tilde{n} \rangle]\!] \quad \text{implies} \quad M + \tilde{n}{:}\tilde{\mathsf{T}} \triangleright M \cong N + \tilde{n}{:}\tilde{\mathsf{U}} \triangleright N$$

Proof. We define the relation $\mathcal{R}$ as:

$$\mathcal{R} = \left\{ \left(M + \tilde{n}{:}\tilde{\mathsf{T}} \triangleright M,\; N + \tilde{n}{:}\tilde{\mathsf{U}} \triangleright N \right) \;\middle|\; \nu + \tilde{n}{:}\tilde{\mathsf{T}} \triangleright M \mid k_0[\![\text{succ}!\langle \tilde{n} \rangle]\!] \cong \nu + \tilde{n}{:}\tilde{\mathsf{U}} \triangleright N \mid k_0[\![\text{succ}!\langle \tilde{n} \rangle]\!] \right\}$$

and show that $\mathcal{R}$ satisfies the defining properties of $\cong$. It is obviously reduction closed. We here outline the proof for the barb preserving and contextuality properties.

Suppose $M + \tilde{n}{:}\tilde{\mathsf{T}} \triangleright M \;\mathcal{R}\; N + \tilde{n}{:}\tilde{\mathsf{U}} \triangleright N$ and $M + \tilde{n}{:}\tilde{\mathsf{T}} \triangleright M \downarrow_{a@l}$; we have to show $N + \tilde{n}{:}\tilde{\mathsf{U}} \triangleright N \downarrow_{a@l}$. If $l, a \notin \tilde{n}$ this is straightforward since in this case $\nu + \tilde{n}{:}\tilde{\mathsf{T}} \triangleright (M \mid k_0$M

Since $^{M} \triangleright M \nrightarrow_{a@l}$ it follows that

$$^{M}_{+} + \delta : \mathsf{ch} \triangleright (\nu\, \tilde{n} : \tilde{\mathsf{T}}) M \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \setminus T_a \nrightarrow_{\delta@k_0}$$

From the definition of $\cong$, we know

$$^{M}_{+} + \delta : \mathsf{ch} \triangleright (\nu\, \tilde{n} : \tilde{\mathsf{T}}) M \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \setminus T_a \cong {}^{N}_{+} + \delta : \mathsf{ch} \triangleright (\nu\, \tilde{n} : \tilde{\mathsf{U}}) N \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \setminus T_a$$

and by barb preservation we conclude

$$^{N}_{+} + \delta : \mathsf{ch} \triangleright (\nu\, \tilde{n} : \tilde{\mathsf{U}}) N \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \setminus T_a \nrightarrow_{\delta@k_0}$$

which only be because $^{N} \triangleright N \nrightarrow_{a@l}$ as required.

The case for when $n = l$ is similar, only that instead of T_a we use the system:

$$T_l \quad k_0[\![\mathrm{succ}?(X).(\nu\, k : (\mathbf{dom}(\mathcal{I}_O) \quad X_l))\mathsf{go}\, k, X_l.a?().\mathsf{go}\, k, k_0.\delta!\langle\rangle]\!]$$

This system is similar to T_a with the exception that a specific location k is created so that we short-circuit our route to l, similar to the procedure we used earlier in the definability proof of bound outputs (see Proposition 4.3.5).

We still have to show that $\mathcal{R}$ is contextual. As an example we show that it is preserved by parallel system contexts and leave the simpler case, that for network extensions, to the interested reader. Suppose $\mathcal{I} = {}^{M} \triangleright M \;\; \mathcal{R} \;\; {}^{N} \triangleright N$; we have to show that for arbitrary $k[\![P]\!]$ such that $\mathcal{I} \vdash k[\![P]\!]$ then we have $\mathcal{I} = {}^{M} \triangleright M \setminus k[\![P]\!] \;\; \mathcal{R} \;\; {}^{N} \triangleright N \setminus k[\![P]\!]$.

By definition of $\mathcal{R}$, we have $\mathcal{I} = {}^{M} \triangleright M \;\; \mathcal{R} \;\; {}^{N} \triangleright N$ because

$$\mathcal{I} = {}^{M}_{+} \triangleright (\nu\, \tilde{n} : \tilde{\mathsf{T}}) M \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \cong {}^{N}_{+} \triangleright (\nu\, \tilde{n} : \tilde{\mathsf{U}}) N \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \tag{53}$$

We define the system

$$T_{k[P]} \quad k_0[\![\mathrm{succ}?(X).\mathsf{go}\, k_0.\delta!\langle X \rangle \setminus (X)\mathsf{go}\, k.P]\!]$$

where δ, k_0 are fresh names and $(X)\mathsf{go}\, k.P$ substitutes all occurrences of $\tilde{n}$ in $\mathsf{go}\, k.P$ by the appropriate variables $x_i \quad X$. From $\mathcal{I} \vdash k[\![P]\!]$ we deduce that $\mathcal{I} \quad \vdash T_{k[P]}$ for $\mathcal{I} \quad = \mathcal{I} \; + \delta : \mathsf{ch} + k_0 : \mathtt{loc}[\mathsf{a}, \mathbf{dom}(\mathcal{I}_O)]$ and subsequently, by contextuality of $\cong$ and (53), we obtain

$$\mathcal{I} = {}^{M}_{++} \triangleright M \setminus T_{k[P]} \cong {}^{N}_{++} \triangleright N \setminus T_{k[P]} \tag{54}$$

where

$$\begin{aligned} M &= (\nu\, \tilde{n} : \tilde{\mathsf{T}}) M \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \\ N &= (\nu\, \tilde{n} : \tilde{\mathsf{U}}) N \setminus k_0[\![\mathrm{succ}!\langle \tilde{n} \rangle]\!] \qquad : \mathtt{loc}[\mathsf{a}, \mathbf{dom} \\ {}^{M}_{++} &= {}^{M}_{+} + \delta : \mathsf{ch} + + \quad : \mathtt{loc}[\mathsf{a}, \mathbf{dom} \end{aligned}$$

and by Lemma 4.3.6 and we get

$$\mathcal{I} \setminus = \; {}^{M}_{+} \triangleright (\nu\, \tilde{n}{:}\tilde{\mathrm{T}}) M \setminus k[\![P]\!] \setminus k_0[\![\delta!]\tilde{n}]\!] \;\cong\; {}^{N}_{+} \triangleright (\nu\, \tilde{n}{:}\tilde{\mathrm{T}}) N \setminus k[\![P]\!] \setminus k_0[\![\delta!]\tilde{n}]\!] \quad (56)$$

from which, by definition of $\mathcal{R}$, we derive $\mathcal{I} \setminus = \; {}^{M} \triangleright M \setminus k[\![P]\!] \;\mathcal{R}\; {}^{N} \triangleright N \setminus k[\![P]\!]$ as required. □

Proposition 4.3.8 (Completness).

$$\mathcal{I} \setminus = \; {}^{1} \triangleright M_1 \cong {}^{2} \triangleright M_2 \;\text{ implies }\; \mathcal{I} \setminus = \; {}^{1} \triangleright M_1 \;\widehat{\cong}\; {}^{2} \triangleright M_2$$

Proof. Suppose ${}^{1} \triangleright M_1 \xrightarrow{\mu} {}^{1}_{1} \triangleright M_1$; we must find a move ${}^{2} \triangleright M_2 \;\widehat{\Longrightarrow}\; {}^{2}_{1} \triangleright M_2$ such that ${}^{1}_{1} \triangleright M_1 \cong {}^{2}_{1} \triangleright M_2$. If μ is an internal move then the matching move is obtained from the fact that $\cong$ is reduction closed, together with Proposition 4.2.3. If μ is an external action, then by choosing Nm so that it contains all the free names in $\mathcal{I}_N$ and choosing fresh succ, fail, k_0, from the first part of Proposition 4.3.5 and the assumption ${}^{1} \triangleright M_1 \xrightarrow{\mu} {}^{1}_{1} + \mathbf{bn}(\mu) \triangleright M_1$ we obtain

$${}^{1}_{+} \triangleright M_1 \setminus T^{\mu}(Nm, \mathrm{succ}, \mathrm{fail}, k_0) \Longrightarrow \; {}^{1}_{1+} \triangleright (\nu\, \mathbf{bn}(\mu)) M_1 \setminus k_0[\![\mathrm{succ}!]\mathbf{bn}(\mu)]\!]$$

By contextuality and reduction closure of $\cong$, we know that there is a matching move

$${}^{2}_{+} \triangleright M_2 \setminus T^{\mu}(Nm, \mathrm{succ}, \mathrm{fail}, k_0) \Longrightarrow \; \triangleright N$$

for some $\triangleright N$ such that ${}^{1}_{1+} \triangleright (\nu\, \mathbf{bn}(\mu)) M_1 \setminus k_0[\![\mathrm{succ}!]\mathbf{bn}(\mu)]\!] \cong \triangleright N$. This in turn means that $\triangleright N \not\xrightarrow{\mathbf{succ}@k_0}$ and $\triangleright N \not\xrightarrow{\mathbf{fail}@k_0}$ and so the second part of Proposition 4.3.5 now gives that $\triangleright N \;-\; {}^{2}_{1+} \triangleright (\nu\, \mathbf{bn}(\mu)) M_2 \setminus k_0[\![\mathrm{succ}!]\mathbf{bn}(\mu)]\!]$ for some ${}^{2}_{1+}, M_2$ such that ${}^{2} \triangleright M_2 \;\widehat{\Longrightarrow}\; {}^{2}_{1} + \mathbf{bn}(\mu) \triangleright M_2$. This is the required matching move, since the Extrusion Lemma 4.3.7, gives us the required

$${}^{1}_{1} + \mathbf{bn}(\mu) \triangleright M_1 \cong {}^{2}_{1} + \mathbf{bn}(\mu) \triangleright M_2$$ □

5 Conclusions

We have presented a simple extension of Dπ, in which there is an explicit representation of the underlying network on which processes execute, exhibiting both node and link failures. Our main result is a *fully-abstract* bisimulation equivalence with which we can reason about the behaviour of systems in the presence of dynamic network failures. To the best of our knowledge, this is the first time system bt knowledge, ork failures.

Related work: There have been a number of studies on process behaviour in the presence of *permanent node failure* only, in addition to our starting point [16]. That closest to our work is the already cited [2, 1]; as already mentioned, their approach to developing reasoning tools is also quite di erent from ours. Rather than develop, justify and use bisimulations in the source language of interest, in their case π_l and π_{1l}, they propose a translation into a version of the π-calculus without locations, and use reasoning tools on the translations. But most importantly, they do show that for certain π_{1l} terms, it is su cient to reason on their translations. Elsewhere, permanent location failure with hierarchical dependencies have been studied by Fournet, Gonthier, Levy and Remy in [6]. Berger [3] was the first to study a π-calculus extensTj -432.290ulu746 0 Tsd (also)Tj 26..4549 0 Td
viour72 0.3084 0 T Td (xtensTj gua1.0349 0 Td (on)T

A Notation

Here we give the formal definitions for the various notation we have introduced for extracting information from network representations, and for updating them.

A.1 DπLoc Notation

Recall that for DπLoc a network repsresentation consists of the tuple $\rceil\mathcal{N}, \mathcal{D}\not($, where $\mathcal{N}$ is a set of names known and $\mathcal{D}$ is the set of dead locations. We thus define the following judgements:

$$\vdash a:\mathtt{ch} \stackrel{\text{def}}{=} a \quad \mathcal{N} \qquad \textit{(valid channels)}$$

$$\vdash l:\mathtt{loc[d]} \stackrel{\text{def}}{=} l \quad \mathcal{N} \blacktriangleleft \quad l \quad \mathcal{D} \qquad \textit{(valid dead location)}$$

$$\vdash l:\mathtt{loc[a]} \stackrel{\text{def}}{=} l \quad \mathcal{N} \blacktriangleleft \quad l \notin \quad \mathcal{D} \qquad \textit{(valid live location)}$$

$$\vdash l : \mathbf{alive} \stackrel{\text{def}}{=} \vdash l:\mathtt{loc[a]} \qquad \textit{(live locations)}$$

$$\vdash k \searrow l \stackrel{\text{def}}{=} \vdash k:\mathbf{alive}, l:\mathbf{alive} \qquad \textit{(k accessible from l)}$$

$$\vdash M \stackrel{\text{def}}{=} \mathbf{fn}(M) \quad \mathcal{N} \qquad \textit{(valid systems)}$$

We also define the following operations:

$$+ a:\mathtt{ch} \stackrel{\text{def}}{=} \rceil$$

and . We define

$$\begin{array}{rcll}
\vdash l:\mathbf{alive} & \stackrel{\text{def}}{=} & l \quad \mathbf{dom}(\ O\ \ ) & (\textit{live locations}) \\
\vdash l \rightsquigarrow k & \stackrel{\text{def}}{=} & l \rightsquigarrow k \quad O & (\textit{live link}) \\
\vdash \mathtt{T} & \stackrel{\text{def}}{=} & \mathbf{fn}(\mathtt{T}) \quad \mathcal{N} & (\textit{valid types}) \\
\vdash n:\mathtt{T}, \tilde{n}:\tilde{\mathtt{T}} & \stackrel{\text{def}}{=} & \vdash \mathtt{T} \text{ and } +n:\mathtt{T} \vdash \tilde{n}:\tilde{\mathtt{T}} & \\
\vdash N & \stackrel{\text{def}}{=} & \mathbf{fn}(N) \quad \mathcal{N} & (\textit{valid systems}) \\
\vdash k \searrow l & \stackrel{\text{def}}{=} & O \vdash k \searrow l \text{ or } O \vdash k \searrow l & (\textit{accessibility}) \\
\vdash k \leftsquigarrow l & \stackrel{\text{def}}{=} & O \vdash k \leftsquigarrow l \text{ or } O \vdash k \leftsquigarrow l & (\textit{reachability})
\end{array}$$

$$\blacktriangle \vdash l:\mathbf{alive}, l \rightsquigarrow k, \mathtt{T}, N \stackrel{\text{def}}{=} (\blacktriangle) \vdash l:\mathbf{alive}, l \rightsquigarrow k, \mathtt{T}, N$$

$$\begin{array}{rcll}
\mathcal{I}+n:\mathtt{L} & \stackrel{\text{def}}{=} & \rceil \mathcal{I}_{\mathcal{N}} \quad n\ ,\ \mathcal{I}_O \ \mathbb{L} & (\textit{updates}) \\
\mathcal{I} \vdash l:\mathbf{alive} & \stackrel{\text{def}}{=} & l \quad \mathbf{dom}(\mathcal{I}_O) & (\textit{live locations}) \\
\mathcal{I} \vdash l \rightsquigarrow k & \stackrel{\text{def}}{=} & l \rightsquigarrow k \quad \mathcal{I}_O & (\textit{live link}) \\
\mathcal{I} \vdash \mathtt{T} & \stackrel{\text{def}}{=} & \mathbf{fn}(\mathtt{T}) \quad \mathbf{dom}(\mathcal{I}_O) & (\textit{valid types}) \\
\mathcal{I} \vdash l[\![P]\!] & \stackrel{\text{def}}{=} & \mathbf{fn}(P) \quad \mathcal{I}_{\mathcal{N}} \text{ and } l \quad \mathbf{dom}(\mathcal{I}_O) & (\textit{valid systems}) \\
\mathcal{I} \vdash (\nu\, n:\mathtt{T})N & \stackrel{\text{def}}{=} & \mathcal{I} \vdash \mathtt{T} \text{ and } \mathcal{I}+n:\mathtt{T} \vdash N & \\
\mathcal{I} \vdash N \backslash M & \stackrel{\text{def}}{=} & \mathcal{I} \vdash N \text{ and } \mathcal{I} \vdash M &
\end{array}$$

$$\begin{array}{rcll}
\blacktriangle \vdash_{\mathsf{obs}} l:\mathbf{alive}, l \rightsquigarrow k, \mathtt{T}, N & \stackrel{\text{def}}{=} & \mathcal{I}(\blacktriangle) \vdash l:\mathbf{alive}, l \rightsquigarrow k, \mathtt{T}, N & (\textit{external judgments}) \\
\vdash_{\mathsf{obs}} l:\mathbf{alive}, l \rightsquigarrow k, \mathtt{T}, N & \stackrel{\text{def}}{=} & \mathcal{I}(\ ) \vdash l:\mathbf{alive}, l \rightsquigarrow k, \mathtt{T}, N &
\end{array}$$

Finally we outline a number of operations on types used in reduction rules and transition rules.

$$\begin{array}{rcll}
\mathsf{ch}/\ l_1,..,l_n & \stackrel{\text{def}}{=} & \mathsf{ch} & (\textit{type filtering}) \\
\mathtt{loc[C]}/\ l_1,..,l_n & \stackrel{\text{def}}{=} & \mathtt{loc[C}/\ l_1,..,l_n\ \mathtt{]} &
\end{array}$$

$$\mathsf{inst}(\mathtt{loc[C]}, l, \ , _{\mathsf{N}}$$

B Auxilliary Proofs

We here prove a lemma that is used to show that our lts of 3.3 is closed over valid e ective configurations.

Lemma B.0.1 (Valid Effective Network Updates). If is a valid e ective network, n is fresh in and the type $\mathtt{T}$ is a valid type with respect to , denoted as $\vdash \mathtt{T}$ (see Appendix for definition) then $+n{:}\mathtt{T}$ is a valid e ective network.

Proof. The cases where $\mathtt{T} = \mathtt{ch}$ and $\mathtt{T} = \mathtt{loc}[\mathtt{d},\mathtt{C}]$ are trivial so we focus our attention to the case where $\mathtt{T} = \mathtt{loc}[\mathtt{d},\mathtt{C}]$; at this point, according to Definition 3.3.5, we have two possible subcases:

- If $\mathtt{C} \downarrow$ **dom**($_O$) $=\!\!\uparrow$ then $+n{:}\mathtt{loc}[\mathtt{a},\mathtt{C}]$ has the form $\rceil_N$ n , $_O$, $\mathcal{H}\langle$ where $\mathcal{H}$ = ($l \rightsquigarrow \mathtt{C}$). To prove that this resultant network is a valid e ective network, we have to show that it adheres to the three consistency requirements, defined earlier in Definition 3.3.2:
  1. **dom**($_O$) **loc**($_N$ n). This is immediate from the fact that is valid and thus **dom**($_O$) **loc**($_N$).
  2. **dom**($\mathcal{H}$) **loc**($_N$ n) and that $\mathcal{H}$ is a linkset. The inclusion is obtained from the fact that **dom**( ) **loc**($_N$) and the assumption that **loc**($\mathtt{loc}[\mathtt{a},\mathtt{C}]$) **loc**($_N$). The fact that $\mathcal{H}$ = $l \rightsquigarrow \mathtt{C}$ is a linkset is immediate from the fact that $l \rightsquigarrow \mathtt{C}$ is a component.
  3. **dom**($_O$)$\downarrow$ **dom**($\mathcal{H}$) $=\!\!\uparrow$. This is immediately obtained from the assumptions that **dom**($_O$)$\downarrow$ **dom**( ) $=\!\!\uparrow$, $n \notin$ $_N$ and the condition for this subcase, that is $\mathtt{C} \downarrow$ **dom**($_O$) $=\!\!\uparrow$.
- If ($\mathtt{C} \downarrow$ **dom**($_O$) $\neq\!\!\uparrow$) then $+n{:}\mathtt{loc}[\mathtt{a},\mathtt{C}]$ has the form $\rceil_N$ n , $\mathcal{O}$, $\mathcal{H}\langle$ where $\mathcal{O}$ = $_O$ ($l \rightsquigarrow \mathtt{C}$ ($\leftsquigarrow \mathtt{C}$) and $\mathcal{H}$ = /($\leftsquigarrow \mathtt{C}$). One again, we have to prove that $+n{:}\mathtt{loc}[\mathtt{a},\mathtt{C}]$ satisfies the three consistency conditions:
  1. **dom**($\mathcal{O}$) **loc**($_N$ n) and that $\mathcal{O}$ is a linkset. The proof here progresses similar to the second requirement of the previous subcase.
  2. **dom**($\mathcal{H}$) **loc**($_N$ n) and that $\mathcal{H}$ is a linkset. The proof for the inclusion is a simpler version of the above subcases, while the requirement that $\mathcal{H}$ = / $\leftsquigarrow \mathtt{C}$ is a linkset is obtained from the fact that $\leftsquigarrow \mathtt{C}$ is a component and Lemma 3.3.4.
  3. **dom**($\mathcal{O}$)$\downarrow$ **dom**($\mathcal{H}$) $=\!\!\uparrow$. This is obtained from the assumptions that **dom**($_O$)$\downarrow$ **dom**( ) $=\!\!\uparrow$, $n \notin$ $_N$ and the structure of $\mathcal{O}$ and $\mathcal{H}$. □

References

[1] Roberto M. Amadio. An asynchronous model of locality, failure, and process mobility. In D. Garlan and D. Le Métayer, editors, *Proceedings of the 2nd International Conference on Coordination Languages and Models (COORDINATION'97)*, volume 1282, pages 374–391, Berlin, Germany, 1997. Springer-Verlag.

[2] Roberto M. Amadio and Sanjiva Prasad. Localities and failures. *FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, 14, 1994.

[3] Martin Berger. Basic theory of reduction congruence for two timed asynchronous π-calculi. In *Proc. CONCUR'04*, 2004.

[4] Luca Cardelli. Wide area computation. In *Proceedings of* 26^{th} *ICALP*, Lecture Notes in Computer Science, pages 10–24. Springer-Verlag, 1999.

[5] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[6] Cedric Fournet, Georges Gonthier, Jean Jaques Levy, and Remy Didier. A calculus of mobile agents. *CONCUR 96*, LNCS 1119:406–421, August 1996.

[7] R.J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions (extended abstract). In A. Kreczmar and G. Mirkowska, editors, Proceedings 14^{th} Symposium on *Mathematical Foundations of Computer Science*, MFCS '89, Porąbka-Kozubnik, Poland, August/September 1989, volume 379 of *lncs*, pages 237–248. Springer-Verlag, 1989.

[8] Matthew Hennessy, Massimo Merro, and Julian Rathke. Towards a behavioural theory of access and mobility control in distributed systems. *Theoretical Computer Science*, 322:615–669, 2004.

[9] Matthew Hennessy and Julian Rathke. Typed behavioural equivalences for processes in the presence of subtyping. *Mathematical Structures in Computer Science*, 14:651–684, 2004.

[10] Matthew Hennessy and James Riely. Resource access control in systems of mobile agents. In Uwe Nestmann and Benjamin C. Pierce, editors, *HLCL98: High-Level Concurrent Languages (Nice, France, September 12, 1998)*, volume 16(3), pages 3–17. Elsevier Science Publishers, 1998.

[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] Kohei Honda and Nobuko Yoshida. A uniform type structure for secure information fbw. In *29th Annual Symposium on Principles of Programming Languages*. ACM, January 2002.

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

[15] Nestmann, Fuzzati, and Merro. Modeling consensus in a process calculus. In *CONCUR: 14th International Conference on Concurrency Theory*. LNCS, Springer-Verlag, 2003.

[16] James Riely and Matthew Hennessy. Distributed processes and location failures. *Theoretical Computer Science*, 226:693–735, 2001.

[17] Davide Sangiorgi and David Walker. *The π-calculus*. Cambridge University Press, 2001.

[18] Richard D. Schlichting and Fred B. Schneider. Fail-stop processors: An approach to designing fault-tolerant computing systems. *Computer Systems*, 1(3):222–238, 1983.