On Bisimulation Proofs for the Analysis of Distributed Abstract Machines

Damien Pous

On Bisimulation Proofs for the Analysis of Distributed Abstract Machines *

Damien Pous

ENS Lyon, France

Abstract. We illustrate the use of recent, non-trivial proof techniques for weak bisimulation by analysing a generic framework for the definition of distributed abstract machines based on a message-passing implementation. The definition of the framework comes from previous works on a specific abstract machine; however, its new presentation, as a parametrised process algebra, makes it suitable for a wider range of calculi. A first version of the framework can be analysed using the standard bisimulation up to expansion proof technique. We show that in a second, optimised version, rather complex behaviours appear, for which more sophisticated techniques, relying on termination arguments, are necessary to establish behavioural equivalence.

Introduction

Recently, many calculi encompassing distribution and mobility have been studied as a basis for programming languages. Examples include Join [4], Nomadic Pict [11], Kells [1], Ambients [2], Klaim [7], Seals [3]. The expressive power supplied by the primitives underlying such models raises the question of implementability in a distributed framework. In [10], a distributed abstract machine is defined, to implement the Safe Ambient Calculus [6]: the PAN. The main ingredients in the definition of this machine are *locations* – where local processes are executed – and *forwarders*, that transmit messages between locations. In [5], we defined an optimised version of this machine where useless forwarders can be garbage collected, and less messages are transmitted along the network. We proved that the resulting abstract machine is weak barbed bisimilar to the original one; however due to the lack of adequate up-to techniques or compositionality results, this proof is quite tedious, and cannot easily be used as a basis for further studies.

Motivated by these difficulties, we developed new up-to techniques for weak bisimulation [8]. These techniques improve on previously known techniques; however, they are developed in a completely abstract setting and their applicability has not yet been evaluated beyond rather simple illustrative examples.

Before focusing on behavioural equivalences and proof techniques, we present the first main contribution of this work, the definition of a framework to reason about distributed implementations of process algebras with mobility. In this

* Author's version of the paper published by Springer in Proc. TGC'06, available at `http://dx.doi.org/10.1007/978-3-540-75336-0_10`.

framework, a network is represented by a set of *locations*, or hosts, where arbitrary local processes are executed. The behaviour of local processes is specified by a given LTS, whose labels correspond to the following possible actions:

- sending arbitrary messages to other locations,
- spawning local processes, inside new locations,
- migrating to another location.

While this framework is the basis of the PAN abstract machine [10], we dropped most of the hypotheses that were related to the implementation of an Ambient-based calculus. Therefore, it should be suitable to analyse a rather wide range of calculi (this is discussed in Remark 2.1).

We then move to the analysis of this framework, which serves as support for our second main contribution: illustrating a non-trivial use of recent proof techniques to reason about a rather complex system.

A forwarder from location h to location k acts like a substitution that replaces any occurrence of h by k in the whole network: a message sent to h will actually reach k. Accordingly, we prove that a net with a forwarder *expands* the corresponding substituted net (*expansion* is the standard behavioural preorder that leads to the correct weak bisimulation up to expansion technique [9]). This result allows one to abstract over the communication framework when validating possible optimisations of local processes.

The main drawback of this framework is the creation of forwarder chains, that slow down the communications between local processes. To address this inefficiency, we introduce an optimisation, inspired from [5], that consists in defining a forwarder relocation mechanism, that contracts forwarder chains. Like in [5], this mechanism breaks the initial proof strategy, as the expansion result does no longer hold. We show in details how the techniques we developed in [8] make it possible to give nevertheless a modular proof of correctness, where the bisimulation candidates that we manipulate remain tractable and express only local properties of processes.

Being able to work with small bisimulation candidates is quite important: they are much easier to check, and when a small part of the system is refined, there is hope that only some of the proofs will need to be updated. Even more important is the fact that the relations focus on local properties, since this allows one to write explicitly the slight differences between related processes and to reason syntactically about these.

We actually allude in [8] to an example derived from [5]. It turned out that the development made in a corresponding technical report missed a crucial step in the proof, and, more importantly, contained a mistake. Moreover, the proof presented here for the optimised system is less specific and aims at giving a better illustration of the benefits given by the general techniques of [8], and of their flexibility.

Outline of the paper. In Sect. 1, we introduce our notations and the notions used in the sequel. We define the initial framework in Sect. 2, and we analyse it in

Sect. 3. Section 4 is devoted to the definition of the optimisation, and to the corresponding correctness proof. We conclude with some remarks in Sect. 5.

1 LTS, Bisimilarity

We consider labelled transition systems (LTS) $\langle \mathcal{P}, \mathcal{L}, \rightarrow \rangle$, with domain $\mathcal{P}$, *labels* or *actions* in $\mathcal{L}$ and transition relation $\rightarrow \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$. The elements of $\mathcal{P}$ are called *processes* and are denoted by P, Q in this section. $\mathcal{L}$ will always implicitly contain a distinguished *silent action*, noted τ. We let α, β (resp. a, b) range over actions, in $\mathcal{L}$ (resp. *visible actions*, in $\mathcal{L}\backslash\{\tau\}$).

We let $\mathcal{R}, \mathcal{S}, \mathcal{B}, \mathcal{E}$ range over binary relations (simply called *relations* in the sequel) between processes. We denote respectively by $\mathcal{R}^+, \mathcal{R}^=, \mathcal{R}^\star$ the transitive, reflexive, transitive and reflexive closures of a relation $\mathcal{R}$. $P\mathcal{R}Q$ means $\langle P, Q \rangle \in \mathcal{R}$. The composition of two relations $\mathcal{R}$ and $\mathcal{S}$, written $\mathcal{RS}$, is defined by $\mathcal{RS} \triangleq \{\langle P, Q \rangle / P\mathcal{R}T \text{ and } T\mathcal{S}Q \text{ for some process } T\}$. We also define the inverse of a relation: $\mathcal{R}^{-1} \triangleq \{\langle P, Q \rangle / Q\mathcal{R}P\}$. $\mathcal{I}$ is the identity relation, defined by $\mathcal{I} \triangleq \{\langle P, P \rangle / P \in \mathcal{P}\}$. We say that $\mathcal{R}$ *contains* $\mathcal{S}$ (alternatively, that $\mathcal{S}$ is contained in $\mathcal{R}$), written $\mathcal{S} \subseteq \mathcal{R}$, if $P\mathcal{S}Q$ implies $P\mathcal{R}Q$. Given an action α, the set of transitions along α induces a relation denoted by $\xrightarrow{\alpha}$: $\xrightarrow{\alpha} \triangleq \{\langle P, Q \rangle / \langle P, \alpha, Q \rangle \in \rightarrow\}$. Its inverse is written using a reversed arrow: $\xleftarrow{a} = (\xrightarrow{a})^{-1}$, and similarly for other forms of arrows, defined below.

Definition 1.1 (Termination). *A relation* $\mathcal{R}$ terminates *if there is no infinite sequence* $(P_i)_{i \in \mathbb{N}}$ *such that* $\forall i, P_i \mathcal{R} P_{i+1}$.

The definitions of behavioural equivalences and preorders will make use of the following *weak transition* relations.

Definition 1.2 (Weak transitions).

$$\stackrel{\widehat{\alpha}}{\rightarrow} \triangleq \begin{cases} \stackrel{\tau}{\rightarrow}^= & \text{if } \alpha = \tau \\ \stackrel{a}{\rightarrow} & \text{if } \alpha = a \in \mathcal{L}\backslash\{\tau\} \end{cases} \qquad \begin{array}{l} \stackrel{\alpha}{\Rightarrow} \triangleq \stackrel{\tau}{\rightarrow}^\star \stackrel{\alpha}{\rightarrow} \stackrel{\tau}{\rightarrow}^\star \\ \stackrel{\widehat{\alpha}}{\Rightarrow} \triangleq \stackrel{\tau}{\rightarrow}^\star \stackrel{\widehat{\alpha}}{\rightarrow} \stackrel{\tau}{\rightarrow}^\star \end{array}$$

We can remark the following properties: $\stackrel{\widehat{\tau}}{\Rightarrow} = \stackrel{\tau}{\rightarrow}^\star$, $\stackrel{\tau}{\Rightarrow} = \stackrel{\tau}{\rightarrow}^+$, $\stackrel{\widehat{a}}{\Rightarrow} = \stackrel{a}{\Rightarrow}$ (note in particular the difference between $\stackrel{\widehat{\tau}}{\Rightarrow}$ and $\stackrel{\tau}{\Rightarrow}$).

Definition 1.3 (Evolution of relations). *Let* α *be an action and* $\mathcal{R}, \mathcal{S}$ *two relations. We say that* $\mathcal{R}$ α-evolves *to* $\mathcal{S}$ *if* $P\mathcal{R}Q$, $P \stackrel{\alpha}{\rightarrow} P'$ *implies* $Q \stackrel{\widehat{\alpha}}{\Rightarrow} Q'$ *and* $P'\mathcal{S}Q'$ *for some* Q'. *We say that* $\mathcal{R}$ evolves silently *to* $\mathcal{S}$ *when* $\mathcal{R}$ τ*-evolves to* $\mathcal{S}$, *and that* $\mathcal{R}$ evolves visibly *to* $\mathcal{S}$ *when* $\mathcal{R}$ a*-evolves to* $\mathcal{S}$ *for any* $a \in \mathcal{L}\backslash\{\tau\}$.

Definition 1.4 (Simulation, Bisimulation, Bisimilarity). *Let* $\mathcal{R}$ *be a relation.* $\mathcal{R}$ *is a* simulation *if it evolves to itself. A* bisimulation *is a symmetric simulation.* Bisimilarity*, denoted by* $\approx$*, is the union of all bisimulations.*

2 A Framework for Distributed Computation

We let h, k range over a given set of *locations*. We denote by $[k/h]$ the function that replaces location h by location k. We let σ range over such substitutions, that are naturally extended to the various syntactical categories defined in the sequel. We furthermore assume two sets of *(local) processes* and *messages*, denoted respectively by P, Q and M, N. Processes may contain messages, and vice-versa. In order to represent this, we require two operations on these sets:

- the addition of a message M to a process P, denoted by "$P \,|\, \{M\}$",
- the embedding of a process P into a message, denoted by "$\texttt{reg}\ P$".

These operations are supposed to satisfy the following equations:

$$(P \,|\, \{M\}) \,|\, \{N\} = (P \,|\, \{N\}) \,|\, \{M\} \tag{1}$$

$$P \,|\, \{\texttt{reg}\ (Q \,|\, \{M\})\} = (P \,|\, \{\texttt{reg}\ Q\}) \,|\, \{M\} \tag{2}$$

$$(P \,|\, \{M\})\sigma = P\sigma \,|\, \{M\sigma\} \tag{3}$$

Nets combine localised processes and messages with a destination:

$U ::= \mathbf{0}$	(empty net)	$\mid h[P]$	(process located at h)
$\mid U \parallel U$	(parallel composition)	$\mid h\{M\}$	(pending message)
$\mid (\nu h)U$	(location restriction)	$\mid h \triangleright k$	(forwarder located at h)

Definition 2.1 (Structural congruence).
Structural congruence *is the smallest congruence* $\equiv$ *such that:*

1. *parallel composition* ($\parallel$) *forms an abelian monoid, with neutral element* $\mathbf{0}$*;*
2. $(\nu h)U \equiv (\nu k)(U[k/h])$ *if* k *not free in* U*;*
3. $(\nu h)U \parallel V \equiv (\nu h)(U \parallel V)$ *if* h *not free in* V*;*
4. $(\nu h)(\nu k)U \equiv (\nu k)(\nu h)U$*; and* $(\nu h)\mathbf{0} \equiv \mathbf{0}$.

$\Pi_i U_i$ will denote the parallel composition of the nets U_i. The notation for tuples is $\widetilde{x}$ and $(x, \widetilde{x})$ will denote the addition of an element x to $\widetilde{x}$. Our notations are naturally extended using tuples; for example, $h\{\widetilde{M}\}$ and $\widetilde{h} \triangleright k$ will respectively denote $\Pi_i\ h\{M_i\}$ and $\Pi_i\ h_i \triangleright k$.

Definition 2.2 (Dependency relation). *Let* U *be a net. We call* dependency relation of U *the relation* $\prec_U \triangleq \left\{ \langle h_0, h_n \rangle \ / \ U \equiv V \parallel \Pi_{i<n}\ h_i \triangleright h_{i+1} \right\}$.

An *agent* is either a localised process $h[P]$, or a forwarder $h \triangleright k$. We let a, b range over an arbitrary set of *visible actions*, and we define an LTS over nets:

$$[\textsc{Loc}] \quad \frac{P \xrightarrow{h,a,U} P'}{h[P] \xrightarrow{a} h[P'] \parallel U} \qquad \frac{P \xrightarrow{h,a,\texttt{mig}\ k} P'}{h[P] \xrightarrow{a} h \triangleright k \parallel k\{\texttt{reg}\ P'\}} \quad [\textsc{Mig}]$$

$$[\textsc{Fwd}] \quad h\{M\} \parallel h \triangleright k \xrightarrow{\tau} h \triangleright k \parallel k\{M\} \qquad h\{M\} \parallel h[P] \xrightarrow{\tau} h[P \,|\, \{M\}] \quad [\textsc{Rcv}]$$

$$[\textsc{New}] \quad \frac{U \xrightarrow{\alpha} U' \quad h \text{ not free in } U, U'}{(\nu h)U \xrightarrow{\alpha} (\nu h)U'} \qquad \frac{U \xrightarrow{\alpha} U'}{U \parallel V \xrightarrow{\alpha} U' \parallel V} \quad [\textsc{Par}]$$

$$[\textsc{Cong}] \quad \frac{U \equiv U' \xrightarrow{\alpha} V' \equiv V}{U \xrightarrow{\alpha} V}$$

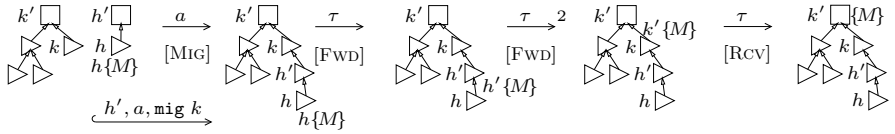

Fig. 1. Nets, Migration, Routing of Messages

This definition depends on a *localised LTS*, describing the behaviour of local processes: $P \xrightarrow{h,a,K} P'$ stands for "process P, when located at h, evolves to P' by performing a visible action a and emitting a *request* K". Such requests correspond to the primitives granted to local processes. They can be either:

- a net U (rule [Loc]), containing messages to send or new localities to spawn;
- or a migration request $\texttt{mig}\ k$ (rule [Mig]): the local process wants to suspend its execution and to send its state to k. In that case, the local process gets replaced by a forwarder that will transmit further messages to k.

The two silent transition rules are concerned with the routing of messages: rule [Fwd] defines the behaviour of forwarders: they transmit the messages; and rule [Rcv] performs the actual reception of a message at its final location. A sequence of transitions is depicted in Fig. 1, where squares and triangles represent respectively localised processes and forwarders; the process located at h' migrates to k and the message $h'\{M\}$ is routed to its final destination, k'.

We assume the two properties below about the localised LTS:

$$\text{If } P \xrightarrow{h,a,K} P', \text{ then for any message } M,\ P \,|\, \{M\} \xrightarrow{h,a,K} P' \,|\, \{M\}. \tag{4}$$

$$P \xrightarrow{h,a,K} P' \quad \text{iff} \quad \text{for any substitution } \sigma,\ P\sigma \xrightarrow{h\sigma,a,K\sigma} P'\sigma. \tag{5}$$

(4) expresses the fact that an additional message should not prevent a process from doing some localised transition. (5) prevents local processes from testing the equality of two locations. In some sense, this means that local processes should not be aware of the implementation details of the framework.

Definition 2.3 (Well-formedness). *A net is well-formed if for any of its reducts U, we have $U \equiv (\nu\widetilde{h})V$ for some $\widetilde{h}, V$ such that:*

1. *any agent of V is located at a location mentioned in $\widetilde{h}$;*
2. *for any location h in $\widetilde{h}$, there is exactly one agent located at h in V; and*
3. *the dependency relation $\prec_V$ is a partial order, whose maximal elements are the locations hosting localised processes.*

In the sequel, we shall often omit the restrictions on locations that should appear in front of a well-formed net $(\nu\widetilde{h})V$: they can be guessed from V.

Hypothesis 2.4. We assume that we are given only well-formed nets.

Remark 2.1 (On the expressiveness of the framework). Locations express only the *logical* distribution of processes. Hence, the second condition in our definition of well-formedness does not rule out the case where several locations are thought of as residing *physically* on the same device. Also, unlike in [10, 5], the processes are not required to be distributed along a tree structure, and there is no constraint on the communication topology: since messages may contain locations, π-calculus-like mobility of links is provided by the model. Independently, migration is *subjective* in our model: the process itself decides to migrate. *Objective* migration mechanisms (like the *passivation* available in the Kell-calculus [1]) may be simulated by using messages to trigger migrations.

The main constraint imposed by the well-formedness hypothesis comes from the third point: the graph of the dependency relation is a forest whose roots are the localised processes, and no cycle of forwarders should be generated. This could happen, for example, if a process located at h migrates to some location pointing to h. We discuss the role of this hypothesis in Sect. 5. One possibility to prevent the creation of such cycles is to define a partial ordering of the local processes, and insure that all migration requests respect this ordering (this is the case in models like Ambients [2], Kells [1], or Seals [3]).

3 Reasoning up to Forwarders

In this section, we validate the behaviour of forwarders, by showing that behavioural equivalence is not sensitive to silent transitions ($\xrightarrow{\tau} \subseteq \approx$) and to the replacement of a forwarder by a substitution.

Even though this property happens to be sufficient for our needs in the sequel (see Sect. 4.2), it does not allow one in general to reason modulo forwarders in other bisimulation proofs: it is well known that weak bisimulation up to $\approx$ is not a correct technique [9]. Therefore, we prove a stronger result using *expansion* ($\succsim$), the standard behavioural preorder that leads to the correct "bisimulation up to $\succsim$" technique. Interestingly, this allows us to use the up to transitivity technique enjoyed by expansion, so that our proof is actually easier.

Definition 3.1 (Expansion). *An* expansion relation *is a relation $\mathcal{R}$ such that for any $\alpha \in \mathcal{L}$, whenever $P\mathcal{R}Q$ we have:*

- *if $P \xrightarrow{\alpha} P'$ then $Q \xRightarrow{\hat{\alpha}} Q'$ and $P'\mathcal{R}Q'$ for some Q',*
- *if $Q \xrightarrow{\alpha} Q'$ then $P \xRightarrow{\hat{\alpha}} P'$ and $P'\mathcal{R}Q'$ for some P'.*

Expansion*, denoted by $\succsim$, is the union of all expansion relations.*

Theorem 3.2 (Bisimulation up to Expansion [9]). *Let $\mathcal{R}$ be a symmetric relation. If $\mathcal{R}$ evolves to $\succsim\mathcal{R}\precsim$, then $\mathcal{R} \subseteq \approx$.*

Theorem 3.3 (Expansion up to Transitivity). *Let $\mathcal{R}$ be a relation. If for any $\alpha \in \mathcal{L}$, whenever $P\mathcal{R}Q$ we have:*

– *if* $P \xrightarrow{\alpha} P'$ *then* $Q \xRightarrow{\widehat{\alpha}} Q'$ *and* $P' \mathcal{R}^\star Q'$ *for some* Q',
– *if* $Q \xrightarrow{\alpha} Q'$ *then* $P \xRightarrow{\widehat{\alpha}} P'$ *and* $P' \mathcal{R} Q'$ *for some* P',

then $\mathcal{R} \subseteq \precsim$.

Notice that transitivity is allowed only on one side in the previous theorem. Nevertheless, this is sufficient for the proof of the following proposition:

Proposition 3.4. *Let* U, V *be two nets. If* $U \xrightarrow{\tau} V$ *then* $U \precsim V$.

Proof. We show that the following relation is an expansion up to transitivity:

$$\mathcal{R} \triangleq \xrightarrow{\tau} \cup \{\langle h\{\mathtt{reg}\ P\} \parallel h\{M\} \parallel W,\ h\{\mathtt{reg}\ (P \mid \{M\})\} \parallel W\rangle\} \quad .$$

We need the up to transitivity technique to analyse the two cases below:

$$\begin{array}{lll}
U \equiv h\{M\} \parallel h[P] \parallel W \qquad \mathcal{R} \qquad h[P \mid \{M\}] \parallel W \equiv V & & [\textsc{Rcv}] \\
U \xrightarrow{a} h\{M\} \parallel h \triangleright k \parallel k\{\mathtt{reg}\ P'\} \parallel W & & [\textsc{Mig}] \\
\quad \mathcal{R}\ h \triangleright k \parallel k\{M\} \parallel k\{\mathtt{reg}\ P'\} \parallel W & & [\textsc{Fwd}] \\
\quad \mathcal{R}\ h \triangleright k \parallel k\{\mathtt{reg}\ P' \mid \{M\}\} \parallel W \xleftarrow{a} V & & [\mathcal{R}, \textsc{Mig}]
\end{array}$$

$$\begin{array}{lll}
U \equiv h\{\mathtt{reg}\ P'\} \parallel h\{M\} \parallel h \triangleright k \qquad \mathcal{R} \qquad h\{\mathtt{reg}\ (P' \mid \{M\})\} \parallel h \triangleright k \equiv V & & \\
U \xrightarrow{\tau} h\{M\} \parallel h \triangleright k \parallel k\{\mathtt{reg}\ P'\} & & [\textsc{Rcv}] \\
\quad \mathcal{R}\ h \triangleright k \parallel k\{M\} \parallel k\{\mathtt{reg}\ P'\} & & [\textsc{Fwd}] \\
\quad \mathcal{R}\ h \triangleright k \parallel k\{\mathtt{reg}\ (P' \mid \{M\})\} \xleftarrow{\tau} V & & [\mathcal{R}, \textsc{Rcv}]
\end{array}$$

The other cases are similar or straightforward. □

The smallest bisimulation relation containing $\xrightarrow{\tau}$ is $\xRightarrow{\widehat{\tau}}$. Hence, proving the weaker result $\xrightarrow{\tau} \subseteq \approx$ without using this expansion-based technique would require to check that $\xRightarrow{\widehat{\tau}}$ is a bisimulation, which is less tractable: while U and V differ only slightly when $U \xrightarrow{\tau} V$, this is no longer the case when $U \xRightarrow{\widehat{\tau}} V$.

We now define a forwarder erasure relation, that replaces a forwarder by the corresponding substitution and we show that this relation is contained in $\precsim$.

Definition 3.5 (Forwarder erasure). *We call* forwarder erasure *the following relation:*

$$\mathcal{E} \triangleq \{\langle (\nu h)(h \triangleright k \parallel U),\ U[k/h]\rangle\} \quad .$$

Proposition 3.6. *Let* U, V *be two nets. If* $U \mathcal{E} V$, *then* $U \precsim V$.

Proof. We check that $\mathcal{E}$ is an expansion relation: let $U = (\nu h)(W \parallel h \triangleright k)$ and $V = W[k/h]$.

– If $U \xrightarrow{\alpha} U'$, by using Hypothesis (5), we obtain $V \xrightarrow{\alpha} V'$ with $U'\mathcal{E}V'$, except in the following case, corresponding to the rule [Fwd]:

$$U \equiv (\nu h)(W' \parallel h\{M\} \parallel h \triangleright k) \quad \xrightarrow{\tau} \quad (\nu h)(W' \parallel h \triangleright k \parallel k\{M\}) \equiv U' \ ,$$

where we just check that $U'\mathcal{E}V$.

– If $V \xrightarrow{\alpha} V'$, again, by using Hypothesis (5), we obtain $U \xrightarrow{\alpha} U'$ with $U'\mathcal{E}V'$, except in the two following cases:

$$\begin{array}{lll}
V \equiv & W'[k/h] \parallel k\{M\} \parallel k[P] & \\
\xrightarrow{\tau} & W'[k/h] \parallel k[P \mid \{M\}] \quad \equiv \quad V' & [\textsc{Rcv}] \\
U \equiv & (\nu h)(W' \parallel h\{M\} \parallel h \triangleright k \parallel k[P]) & \\
\xrightarrow{\tau} & (\nu h)(W' \parallel h \triangleright k \parallel k\{M\} \parallel k[P]) & [\textsc{Fwd}] \\
\xrightarrow{\tau} & (\nu h)(W' \parallel h \triangleright k \parallel k[P \mid \{M\}]) \quad \mathcal{E} \quad V' & [\textsc{Rcv}]
\end{array}$$

$$\begin{array}{lll}
V \equiv & W'[k/h] \parallel k \triangleright k' \parallel k\{M\} & \\
\xrightarrow{\tau} & W'[k/h] \parallel k \triangleright k' \parallel k'\{M\} \quad \equiv \quad V' & [\textsc{Fwd}] \\
U \equiv & (\nu h)(W' \parallel h\{M\} \parallel h \triangleright k \parallel k \triangleright k') & \\
\xrightarrow{\tau} & (\nu h)(W' \parallel h \triangleright k \parallel k\{M\} \parallel k \triangleright k') & [\textsc{Fwd}] \\
\xrightarrow{\tau} & (\nu h)(W' \parallel h \triangleright k \parallel k \triangleright k' \parallel k'\{M\}) \quad \mathcal{E} \quad V' & [\textsc{Fwd}]
\end{array}$$

□

$\mathcal{E}$ and $\xrightarrow{\tau}$ are confluent and terminating relations (for the termination of $\xrightarrow{\tau}$, we rely on the fact that the dependency relation of a well-formed net is a partial order). This allows us to define normal forms of nets, that will be used in Sect. 4:

Definition 3.7 (Normalisation). *We denote respectively by $U_\downarrow$ and $\mathrm{e}(U)$ the normal forms of U w.r.t. $\xrightarrow{\tau}$ and $\mathcal{E}$. We say that a net U is* normal *(resp. $\mathcal{E}$-normal) if $U = U_\downarrow$ (resp. $U = \mathrm{e}(U_\downarrow)$).*

By the two previous propositions, we have that a net expands its $\mathcal{E}$-normal form. This makes it possible to restrict to $\mathcal{E}$-normal nets in bisimulation proofs (see the proof of Theorem 4.17 for an example).

Theorem 3.8. *For any well-formed net U, we have:*

$$U \ \succsim \ \mathrm{e}(U_\downarrow) \ .$$

Proof. By definition of $U_\downarrow$ and $\mathrm{e}(U)$, this follows from Props. 3.4 and 3.6. □

Notice that $(\xrightarrow{\tau} \cup \mathcal{E})$ is terminating and confluent, and that $\mathcal{E}$ preserves $\xrightarrow{\tau}$-normal forms. In particular, we have $\mathrm{e}(U_\downarrow) \equiv \mathrm{e}(U)_\downarrow$. The strategy that we impose in this theorem to compute the normal form of U is hence somehow arbitrary. However, this will ease the development in Sect. 4.

4 Optimisations of the Behaviour of Forwarders

The forwarder chains that are generated along the evolution of a net are the source of inefficiencies. For example, the message $\{M\}$ in Fig. 1 will have to go trough three locations before reaching its final destination. In this section, we define an optimisation of the framework, that contracts such forwarder chains, and we prove the correctness of this optimisation, by showing that simple forwarders, as defined in the previous section, are behaviourally equivalent to the optimised ones.

4.1 Definition of the Optimisation

Optimised nets extend the syntax of nets by

- annotating pending messages with a list of locations: $h\{M\}_{\widetilde{k}}$;
- introducing *blocked forwarders*: $h \lhd k$;
- adding a second kind of messages: *relocation messages* $h\langle \mathsf{go}_{\rhd}\ k\rangle$.

Intuitively, the list that decorates a pending message contains the set of forwarder locations the message did pass through. Messages emitted by the underlying local processes will have an empty list, which will allow us to omit their annotation. Relocation messages are received only by blocked forwarders. Their effect is to redirect such forwarders to a destination closer to the location they indirectly point to.

We shall use the terminology *'simple net'* to denote a net as defined in Sect. 2. Structural congruence is extended to optimised nets in the obvious way. We define over this extended syntax an *optimised LTS*, written $\xrightarrow{\alpha}_{\mathrm{o}}$, by taking the rules of the initial LTS and replacing the silent transition rules ([Fwd] and [Rcv]) with the three rules below.

$$h\{M\}_{\widetilde{h}} \parallel h \rhd k \ \xrightarrow{\tau}_{\mathrm{o}}\ h \lhd k \parallel k\{M\}_{h,\widetilde{h}} \qquad [\mathrm{OFwd}]$$

$$h\{M\}_{\widetilde{h}} \parallel h[P] \ \xrightarrow{\tau}_{\mathrm{o}}\ h[P \mid \{M\}] \parallel \widetilde{h}\langle \mathsf{go}_{\rhd}\ h\rangle \qquad [\mathrm{ORcv}]$$

$$h\langle \mathsf{go}_{\rhd}\ k'\rangle \parallel h \lhd k \ \xrightarrow{\tau}_{\mathrm{o}}\ h \rhd k' \qquad [\mathrm{OUpd}]$$

When a forwarder transmits a message, it registers its location (rule [OFwd]), and enters a *blocked* state so that it will temporarily block further potential messages. Upon reception at the final location, a relocation message is broadcasted to the locations registered in the message (rule [ORcv]). The blocked forwarders located at these locations will finally update their destination accordingly (rule [OUpd]). This behaviour is illustrated in Fig. 2, where reversed, grey triangles correspond to blocked forwarders. Notice that forwarders have to block until they receive the relocation message: otherwise, a timestamps mechanism would be required, so that a forwarder can cleverly chose between two possibly distinct relocation messages.

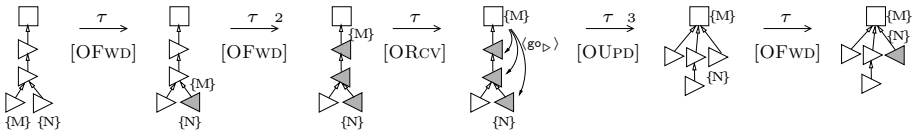

Fig. 2. Optimised Forwarder Behaviour

The definition of dependency relation is adapted by considering all forwarders uniformly, be they blocked or not. We have:

Proposition 4.1. *In the optimised LTS, for any reduct U of a simple, well-formed net, there exist $\widetilde{h}, V$ such that $U \equiv (\nu \widetilde{h})V$ and the following conditions hold:*

1. *the properties required in Def. 2.3 are satisfied;*
2. *for any blocked forwarder $h \triangleleft k$, h appears exactly once in the annotation of a pending message ($h'\{M\}_{\widetilde{h}}$, with $h \in \widetilde{h}$), or as the destination of a relocation message ($h\langle \mathsf{go}_{\triangleright}\ h' \rangle$); in both cases, we have $h \prec_V h'$;*
3. *any location registered in a pending message, or appearing as the target of a relocation message hosts a blocked forwarder.*

In the sequel, we shall implicitly assume that any optimised net that we manipulate satisfies these hypotheses.

4.2 Correctness of the Optimisation

Unlike in the previous section, we cannot rely on expansion-based up-to techniques: neither silent transitions, nor the erasure of forwarders are contained in expansion. This comes from the race conditions introduced by the blocking behaviour of forwarders: for example, in Fig. 2, the message $\{N\}$ has to wait for arrival of $\{M\}$. The very 'controlled' nature of expansion – the right-hand-side process has to be as fast as the left-hand-side process, at each step – cannot take into account the fact that $\{N\}$ is closer to its destination at the end.

However, by proving that in the optimised setting, a net is bisimilar to its $\mathcal{E}$-normal form (Theorem 4.16), the following restricted version of the bisimulation up to $\approx$ technique will be sufficient to prove the equivalence between the two systems (Theorem 4.17): we can restrict to $\xrightarrow{\tau}$-normal forms, so that there is no silent challenge to play.

Theorem 4.2 (Bisimulation up to Bisimilarity). *Let $\mathcal{R}$ be a symmetric relation, if $\mathcal{R}$ evolves silently to itself and visibly to $\approx \mathcal{R} \approx$, then $\mathcal{R} \subseteq \approx$.*

Like in Sect. 3, the smallest bisimulation relations containing $\xrightarrow{\tau}_{\mathrm{o}}$ or $\mathcal{E}$ contain at least $\stackrel{\widehat{\tau}}{\Rightarrow}_{\mathrm{o}}$, so that we need some bisimulation proof technique in order to be able to work with small and local candidate relations. We use for that the following technique from [8].

Definition 4.3 (Controlled relation). *A relation $\mathcal{B}$ is a* controlled relation *if the following conditions hold:*

1. *$\mathcal{B}$ evolves to $\mathcal{B}^\star$,*
2. *$\mathcal{B}^+ \stackrel{\tau}{\Rightarrow}$ terminates,*
3. *$\mathcal{B} \subseteq \approx$.*

Theorem 4.4 (Bisimulation up to a Controlled relation [8]). *Let $\mathcal{B}$ be a controlled relation, if a symmetric relation $\mathcal{R}$ evolves to $\mathcal{B}^\star\mathcal{R}\approx$, then $\mathcal{R} \subseteq \approx$.*

The following proposition will be used to prove the third point of Def. 4.3.

Proposition 4.5. *If $\mathcal{B}$ is a relation evolving to $\mathcal{B}^\star$ and such that $\mathcal{B}^+ \stackrel{\tau}{\Rightarrow}$ terminates, then $\mathcal{B}^\star$ is a simulation.*

We now have enough technical devices to embark in the proof of correctness. We first prove a lemma that will allow us to route messages to their destination.

Lemma 4.6. *Let $U \equiv V \parallel h_0\{M\}_{\widetilde{h}}$ be a net. Then we have:*

$$\begin{aligned} U \equiv\ & V' \parallel h_0\{M\}_{\widetilde{h}} \parallel \Pi_{i<n} F_i(h_i, h_{i+1}) \parallel h_n[P] \\ U \stackrel{\tau}{\Rightarrow}_{\mathrm{o}}\ & V' \parallel \widetilde{h}\langle \mathsf{go}_{\triangleright}\ h_n\rangle \parallel \Pi_{i<n} h_i \triangleright h_n \parallel h_n[P \mid \{\widetilde{N}\} \mid \{M\}] \parallel \widetilde{k} \triangleright h_n \end{aligned}$$

where $F_i(h,k)$ is either:

- *a forwarder: $h \triangleright k$, or*
- *a blocked forwarder, together with a relocation message: $h \triangleleft k' \parallel h\langle \mathsf{go}_{\triangleright}\ k\rangle$, or*
- *a blocked forwarder whose location is registered in a message blocking some other forwarders: $h \triangleleft k' \parallel k\{N\}_{h,\widetilde{k}} \parallel \widetilde{k} \triangleleft \widetilde{k'}$.*

$\widetilde{N}$ and $\widetilde{k}$ are the messages and forwarder locations collected in $\Pi_{i<n} F(h_i, h_{i+1})$.

Proof. The decomposition of U comes from the well-formedness hypothesis. We prove the reduction by induction over n: if $n = 0$ we just apply the rule [ORcv], otherwise we reason by case analysis on the shape of $F_0(h_0, h_1)$:

- a simple forwarder: $h_0 \triangleright h_1$: we transmit the message with rule [OFwd], apply the induction hypothesis (IH), and relocate the forwarder using rule [OUpd]:

$$\begin{aligned} U \stackrel{\tau}{\to}_{\mathrm{o}}\ & V' \parallel h_0 \triangleleft h_1 \parallel h_1\{M\}_{h_0,\widetilde{h}} \parallel \Pi_{0<i<n} F(h_i, h_{i+1}) \parallel h_n[P] && \text{[OFwd]} \\ \stackrel{\tau}{\Rightarrow}_{\mathrm{o}}\ & V' \parallel h_0 \triangleleft h_1 \parallel (h_0, \widetilde{h})\langle \mathsf{go}_{\triangleright}\ h_n\rangle \parallel \Pi_{0<i<n} h_i \triangleright h_n \\ & \qquad \parallel h_n[P \mid \{\widetilde{N}\} \mid \{M\}] \parallel \widetilde{k} \triangleright h_n && \text{(IH)} \\ \stackrel{\tau}{\to}_{\mathrm{o}}\ & V' \parallel h_0 \triangleright h_n \parallel \Pi_{i<n} h_i \triangleright h_n \parallel h_n[P \mid \{\widetilde{N}\} \mid \{M\}] \parallel \widetilde{k} \triangleright h_n && \text{[OUpd]} \end{aligned}$$

- a blocked forwarder with a relocation message: $h_0 \triangleleft k' \parallel h_0\langle \mathsf{go}_{\triangleright}\ h_1\rangle$: we relocate the forwarder with rule [OUpd] and fall back into the previous case.

– $h_0 \triangleleft k' \parallel h_1\{N_1\}_{h_0,\widetilde{k}_1} \parallel \widetilde{k}_1 \triangleleft \widetilde{k}'_1$: we apply the induction hypothesis to the message at h_1, and transmit the initial message trough the relocated forwarder:

$$\begin{aligned} U \overset{\tau}{\Rightarrow}_{\mathrm{o}} \; & V' \parallel h_0\{M\}_{\widetilde{h}} \parallel h_0 \triangleleft h_1 \parallel h_0\langle \mathtt{go}_{\triangleright}\ h_n\rangle \parallel \Pi_{0<i<n} h_i \triangleright h_n \\ & \qquad \parallel h_n[P \mid \{N_1, \widetilde{N}\}] \parallel \widetilde{k}_1 \triangleright \widetilde{k}'_1 \parallel \widetilde{k} \triangleright \widetilde{k}' \qquad \text{(IH)} \\ \overset{\tau}{\to}_{\mathrm{o}} \; & V' \parallel h_0\{M\}_{\widetilde{h}} \parallel \Pi_{i<n} h_i \triangleright h_n \parallel h_n[P \mid \{N_1, \widetilde{N}\}] \parallel \widetilde{k}_1 \triangleright \widetilde{k}'_1 \parallel \widetilde{k} \triangleright \widetilde{k}' \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad [\textsc{oUpd}] \\ \overset{\tau}{\to}_{\mathrm{o}}{}^{3} \; & V' \parallel \widetilde{h}\langle \mathtt{go}_{\triangleright}\ h_n\rangle \parallel \Pi_{i<n} h_i \triangleright h_n \parallel h_n[P \mid \{N_1, \widetilde{N}\} \mid \{M\}] \\ & \qquad \parallel \widetilde{k}_1 \triangleright \widetilde{k}'_1 \parallel \widetilde{k} \triangleright \widetilde{k}' \qquad\qquad [\textsc{oFwd},\textsc{oRcv},\textsc{oUpd}] \end{aligned}$$

□

$\overset{\tau}{\to}_{\mathrm{o}}$ does not commute with visible actions: some relocation of forwarders is involved. To handle this, we introduce the following relation, that allows one to reorganise step by step the forwarder structure of a net.

Definition 4.7. *We denote by $\mathcal{S}$ the* swapping relation*, defined as the symmetric closure of the following relation:*

$$\{\langle h \triangleright h' \parallel h' \triangleright k \parallel U,\ h \triangleright k \parallel h' \triangleright k \parallel U\rangle\} \quad .$$

Our goal is to prove that $(\mathcal{S} \cup \overset{\tau}{\to}_{\mathrm{o}})$ is a controlled relation: this entails $\overset{\tau}{\to}_{\mathrm{o}} \subseteq \approx$, but also makes it possible to give a nice proof of $\mathcal{E} \subseteq \approx$ (Prop. 4.13), by using Theorem 4.4. The three following lemmas establish progressively that $(\mathcal{S} \cup \overset{\tau}{\to}_{\mathrm{o}})$ evolves to $(\mathcal{S} \cup \overset{\tau}{\to}_{\mathrm{o}})^{\star}$, so that this relation satisfies the first point of Def. 4.3. Again, thanks to the up-to technique, we avoid manipulation of complex relations, and focus on nets that differ only slightly (as in Def. 4.7).

Lemma 4.8. *If $U \overset{\tau}{\to}_{\mathrm{o}} V$ and $U \overset{a}{\to}_{\mathrm{o}} U'$ then $U' \overset{\widehat{\tau}}{\Rightarrow}_{\mathrm{o}} \mathcal{S}^{\star} \overset{\widehat{\tau}}{\Leftarrow}_{\mathrm{o}} \overset{a}{\leftarrow}_{\mathrm{o}} V$.*

Proof. It holds that $U' \overset{\tau}{\to}_{\mathrm{o}} \overset{a}{\leftarrow}_{\mathrm{o}} V$, except in the following case:

$$\begin{aligned} U \;\equiv\; & W \parallel h\{M\}_{\widetilde{h}} \parallel h[P] \quad \overset{\tau}{\to}_{\mathrm{o}} \quad W \parallel h[P \mid \{M\}] \parallel \widetilde{h}\langle \mathtt{go}_{\triangleright}\ h\rangle \equiv V \qquad [\textsc{oRcv}] \\ U \overset{a}{\to}_{\mathrm{o}} \; & W \parallel h\{M\}_{\widetilde{h}} \parallel h \triangleright k \parallel k\{\mathtt{reg}\ P'\} \;\equiv\; U' \qquad [\textsc{Mig}] \end{aligned}$$

where we have

$$V \overset{a}{\to}_{\mathrm{o}} \; W \parallel h \triangleright k \parallel k\{\mathtt{reg}\ P' \mid \{M\}\} \parallel \widetilde{h}\langle \mathtt{go}_{\triangleright}\ h\rangle \equiv V' \quad . \qquad [\textsc{Mig}]$$

We reason by case analysis on the agent located at k:

– a localised process $k[Q]$: by routing to k the messages exhibited in U' and V', we obtain:

$$\begin{aligned} U' &\overset{\tau}{\Rightarrow}_{\mathrm{o}} \; W' \parallel (h, \widetilde{h}) \triangleright k \parallel k[Q \mid \{\mathtt{reg}\ P'\} \mid \{M\}] \\ V' &\overset{\tau}{\Rightarrow}_{\mathrm{o}} \; W' \parallel \widetilde{h} \triangleright h \parallel h \triangleright k \parallel k[Q \mid \{\mathtt{reg}\ P' \mid \{M\}\}] \end{aligned}$$

(the only message to route in V' is almost at its final destination, and the relocation message has already been sent to the blocked forwarders located at $\widetilde{h}$, so that the latter gets relocated under h instead of k).
Finally, we relocate these forwarders with n applications of the swapping relation, n being the length of $\widetilde{h}$: $U' \stackrel{\widehat{\tau}}{\Rightarrow}_{\mathrm{o}} \mathcal{S}^n \stackrel{\widehat{f}}{\Leftarrow}_{\mathrm{o}} V'$.

– a forwarder $k \triangleright k'$: like in the previous case, we first route the available messages to their destination, say k'':

$$U' \stackrel{\tau}{\Rightarrow}_{\mathrm{o}} W' \parallel (h, k, \widetilde{h}) \triangleright k'' \parallel k''[Q \mid \{\mathtt{reg}\ P'\} \mid \{M\}] \ ,$$
$$V' \stackrel{\tau}{\Rightarrow}_{\mathrm{o}} W' \parallel \widetilde{h} \triangleright h \parallel h \triangleright k \parallel k \triangleright k'' \parallel k''[Q \mid \{\mathtt{reg}\ P' \mid \{M\}\}] \ .$$

Here we need an additional application of the swapping relation to relocate h to k'', before being able to relocate the forwarders at $\widetilde{h}$: $U' \stackrel{\widehat{\tau}}{\Rightarrow}_{\mathrm{o}} \mathcal{S}^{n+1} \stackrel{\widehat{f}}{\Leftarrow}_{\mathrm{o}} V'$.

– a blocked forwarder $k \triangleleft k'$. We reason like in the previous case by first routing the message that blocks this forwarder to its destination. □

Lemma 4.9. *If $U \mathcal{S} V$ and $U \stackrel{\alpha}{\rightarrow}_{\mathrm{o}} U'$ then $U' \stackrel{\tau}{\Rightarrow}_{\mathrm{o}} \mathcal{S}^\star \stackrel{\widehat{\alpha}}{\Leftarrow}_{\mathrm{o}} V$.*

Proof. It is immediate for visible challenges $\alpha = a$: we have $U' \mathcal{S} \stackrel{a}{\Leftarrow}_{\mathrm{o}} V$. When $\alpha = \tau$, the interesting cases are those where the silent transition $U \stackrel{\tau}{\rightarrow}_{\mathrm{o}} U'$ is the transmission of a message trough one of the two forwarders being swapped:

– $U \equiv W \parallel h\{M\}_{\widetilde{h}} \parallel h \triangleright h' \parallel h' \triangleright k \ \stackrel{\tau}{\rightarrow}_{\mathrm{o}} \ W \parallel h \triangleleft h' \parallel h'\{M\}_{h,\widetilde{h}} \parallel h' \triangleright k \equiv U'$.
By routing the messages, we obtain:

$$U' \stackrel{\tau}{\Rightarrow}_{\mathrm{o}} W' \parallel h \triangleright k' \parallel h' \triangleright k' \parallel k'[Q \mid \{M\}] \ ,$$
$$V \stackrel{\tau}{\Rightarrow}_{\mathrm{o}} W' \parallel h \triangleright k \parallel h' \triangleright k' \parallel k'[Q \mid \{M\}] \equiv V' \ .$$

If $k = k'$, we are done. Otherwise, there is a forwarder $k \triangleright k'$, and we need one application of the swapping relation in order to relocate the forwarder located at h in V'.

– $U \equiv h \triangleright h' \parallel h'\{M\}_{\widetilde{h}} \parallel h' \triangleright k \ \stackrel{\tau}{\rightarrow}_{\mathrm{o}} \ h \triangleright h' \parallel h' \triangleleft k \parallel k\{M\}_{h',\widetilde{h}} \equiv U'$
By routing the messages, we obtain:

$$U' \stackrel{\tau}{\Rightarrow}_{\mathrm{o}} W' \parallel h \triangleright h' \parallel h' \triangleright k' \parallel k'[Q \mid \{M\}] \equiv U'' \ ,$$
$$V \stackrel{\tau}{\Rightarrow}_{\mathrm{o}} W' \parallel h \triangleright k \parallel h' \triangleright k' \parallel k'[Q \mid \{M\}] \equiv V' \ .$$

If $k = k'$, we are done. Otherwise, there is a forwarder $k \triangleright k'$, and we need two applications of the swapping relation in order to relocate the forwarder located at h in both nets:

$$\begin{aligned} U'' \equiv \ & W'' \parallel h \triangleright h' \parallel k \triangleright k' \parallel h' \triangleright k' \parallel k'[Q \mid \{M\}] \\ \mathcal{S} \ & W'' \parallel h \triangleright k' \parallel k \triangleright k' \parallel h' \triangleright k' \parallel k'[Q \mid \{M\}] \\ \mathcal{S} \ & W'' \parallel h \triangleright k \parallel k \triangleright k' \parallel h' \triangleright k' \parallel k'[Q \mid \{M\}] \equiv V' \end{aligned}$$

This analysis also applies for the symmetric cases, where the silent transitions are played by the net with 'flat' forwarders. □

Lemma 4.10. $\xrightarrow{\tau}_{\mathrm{o}}$ *is locally confluent.*

Proof. Using Prop. 4.1, the only critical pair is $U = V \parallel h\{M\}_{\widetilde{h}} \parallel h\{N\}_{\widetilde{k}} \parallel h \triangleright k$,

$$U \xrightarrow{\tau}_{\mathrm{o}} V \parallel h\{M\}_{\widetilde{h}} \parallel h \triangleleft k \parallel k\{N\}_{h,\widetilde{k}} = U_1 \qquad [\textsc{Ofwd}]$$

$$U \xrightarrow{\tau}_{\mathrm{o}} V \parallel h\{N\}_{\widetilde{k}} \parallel h \triangleleft k \parallel k\{M\}_{h,\widetilde{h}} = U_2 \qquad [\textsc{Ofwd}]$$

By using Lemma 4.6 on U_1, we can route the message $\{N\}$ to some location k':

$$U_1 \xRightarrow{\widehat{\tau}}_{\mathrm{o}} V' \parallel h\{M\}_{\widetilde{h}} \parallel h \triangleleft k \parallel (h, \widetilde{k})\langle \mathtt{go}_{\triangleright}\ k' \rangle \parallel k'[P \mid \{N\}] \qquad \text{(Lemma 4.6)}$$

$$\xrightarrow{\tau}_{\mathrm{o}} V' \parallel h\{M\}_{\widetilde{h}} \parallel h \triangleright k' \parallel \widetilde{k}\langle \mathtt{go}_{\triangleright}\ k' \rangle \parallel k'[P \mid \{N\}] \qquad [\textsc{Oupd}]$$

$$\xrightarrow{\tau}_{\mathrm{o}}{}^{3} V' \parallel (\widetilde{k}, \widetilde{h})\langle \mathtt{go}_{\triangleright}\ k' \rangle \parallel h \triangleright k' \parallel k'[P \mid \{N\} \mid \{M\}] \equiv U' \quad [\textsc{Ofwd},\textsc{Orcv},\textsc{Oupd}]$$

The same reasoning about U_2 leads to $U_2 \xRightarrow{\tau}_{\mathrm{o}} U'$. □

Lemma 4.11. $\mathcal{S}^{\star} \xRightarrow{\tau}_{\mathrm{o}}$ *terminates.*

Proof. We call *size* of a net U the triple $s(U) = \langle n, r, l \rangle$, where n is the number of pending messages, r the number of relocation messages, and l the number of forwarders that are not blocked. These triples are ordered lexicographically. We check that $U \mathcal{S} V$ implies $s(U) = s(V)$, and that this size strictly decreases along silent transitions (recall that $\xRightarrow{\tau}_{\mathrm{o}}$ contains at least one transition). □

Proposition 4.12. $(\mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}})$ *is a controlled relation.*

Proof. First we check that $(\mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}})$ satisfies the first two requirements of Def. 4.3: (1) comes from Lemmas 4.8, 4.9, and 4.10; by remarking that $(\mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}})^{+} \xRightarrow{\tau}_{\mathrm{o}} = (\mathcal{S}^{\star} \xRightarrow{\tau}_{\mathrm{o}})^{+}$, Lemma 4.11 gives (2).

For (3), by Prop. 4.5, we have that $(\mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}})^{\star}$ is a simulation. Moreover, $\xleftarrow{\widehat{\tau}}_{\mathrm{o}}$ is a simulation (as is always the case). By combining these two results we obtain that the symmetric relation $(\mathcal{S} \cup \xleftrightarrow{\tau}_{\mathrm{o}})^{\star} = ((\mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}})^{\star} \cup \xleftarrow{\widehat{\tau}}_{\mathrm{o}})^{\star}$ is a simulation, and hence a bisimulation, so that $(\mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}}) \subseteq (\mathcal{S} \cup \xleftrightarrow{\tau}_{\mathrm{o}})^{\star} \subseteq \approx$. □

We now show that $\mathcal{E} \subseteq \approx$. In order to avoid confusion, we consider in the sequel bisimilarity as a relation between *rooted LTSs* that share the same set of labels: $\langle U, \rightarrow \rangle \approx \langle V, \rightsquigarrow \rangle$ will denote the fact that U with labelled transition relation $\rightarrow$, is bisimilar to V, with labelled transition relation $\rightsquigarrow$.

Notice that we do not extend the erasure relation $\mathcal{E}$ to optimised nets; like e(.) it will only be used to reason about simple nets.

Proposition 4.13. *Let U, V be simple nets.*

If $U \mathcal{E} V$ then $\langle U, \rightarrow_{\mathrm{o}} \rangle \approx \langle V, \rightarrow_{\mathrm{o}} \rangle$.

Proof. We show that the symmetric closure of the erasure relation $\mathcal{E}$ is a bisimulation up to the controlled relation $(\mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}})$ (Theorem 4.4).

– If $U \xrightarrow{a}_{\mathrm{o}} U'$, then U' is a simple net, and we check that $V \xrightarrow{a}_{\mathrm{o}} V'$ with $U' \mathcal{E} V'$.
– If $U \xrightarrow{\tau}_{\mathrm{o}} U'$, the interesting case is the following, when $n > 1$:

$$\begin{aligned} U &\equiv (\nu h_0)(W \parallel h_0\{M\} \parallel \Pi_{i<n} h_i \triangleright h_{i+1} \parallel h_n[P]) \\ U &\xrightarrow{\tau}_{\mathrm{o}} U' \equiv (\nu h_0)(W \parallel h_0 \triangleleft h_1 \parallel h_1\{M\}_{h_0} \parallel \Pi_{0<i<n} h_i \triangleright h_{i+1} \parallel h_n[P]) \\ U &\;\mathcal{E}\; V \equiv W[h_1/h_0] \parallel h_1\{M[h_1/h_0]\} \parallel \Pi_{0<i<n} h_i \triangleright h_{i+1} \parallel h_n[P[h_1/h_0]]) \end{aligned}$$

By routing the message in both processes, we obtain:

$$\begin{aligned} U' &\xRightarrow{\tau}_{\mathrm{o}} U'' \equiv (\nu h_0)(W \parallel \Pi_{i<n} h_i \triangleright h_n \parallel h_n[P \mid \{M\}]) \\ V &\xRightarrow{\tau}_{\mathrm{o}} V' \equiv W[h_1/h_0] \parallel \Pi_{0<i<n} h_i \triangleright h_n \parallel h_n[P[h_1/h_0] \mid \{M[h_1/h_0]\}]) \end{aligned}$$

These processes are not related by $\mathcal{E}$, we need first to relocate in U'' the forwarder located at h_0 under h_1, using a 'reversed' application of $\mathcal{S}$:

$$U' \xRightarrow{\tau}_{\mathrm{o}} U'' \;\mathcal{S}\; (\nu h_0)(W \parallel h_0 \triangleright h_1 \parallel \Pi_{0<i<n} h_i \triangleright h_n \parallel h_n[P \mid \{M\}]) \;\mathcal{E}\; V' \xLeftarrow{\tau}_{\mathrm{o}} V \ .$$

– The cases where $V \xrightarrow{\alpha}_{\mathrm{o}} V'$ are handled similarly. □

Remark that we can also prove that $(\mathcal{E} \cup \mathcal{S} \cup \xrightarrow{\tau}_{\mathrm{o}})$ is a controlled relation. This would be useful if we had to reason up to $\mathcal{E}$ on some silent transitions.

Lemmas 4.11 and 4.10 ensure that $\xrightarrow{\tau}_{\mathrm{o}}$ defines a unique normal form for any net (termination of $\mathcal{S}^\star \xRightarrow{\tau}_{\mathrm{o}}$ entails termination of $\xrightarrow{\tau}_{\mathrm{o}}$).

Definition 4.14. *Let U be an optimised net.*
We denote by $U_{\downarrow o}$ the normal form of U w.r.t. $\xrightarrow{\tau}_{\mathrm{o}}$.

Notice that $U_{\downarrow o}$ is always a simple net: it does not contain any blocked forwarder, relocation message, nor pending, annotated messages. Furthermore, we have that an optimised net U is a normal net iff $U = U_{\downarrow o}$.

The normalisation of a simple net by $\xrightarrow{\tau}$ and $\xrightarrow{\tau}_{\mathrm{o}}$ does not necessarily lead to the same net: $U_{\downarrow} \not\equiv U_{\downarrow o}$. However, these nets differ only by some rearrangement of their forwarders: they are related by $\mathcal{S}^\star$. As expressed by the proposition below, in order to obtain the same net, we just need to erase all the forwarders.

Proposition 4.15. *For any simple net U, we have:*

$$\mathrm{e}(U_{\downarrow}) \equiv \mathrm{e}(U_{\downarrow o}) \ .$$

Proof. $U \mathcal{S} V$ entails $\mathrm{e}(U) \equiv \mathrm{e}(V)$, therefore it is sufficient to prove that $U_{\downarrow} \mathcal{S}^\star U_{\downarrow o}$. We proceed by well-founded induction on U, using the termination of $\xrightarrow{\tau}$:

– If U is a normal net, we have $U_{\downarrow} = U = U_{\downarrow o}$.
– If $U \xrightarrow{\tau} U'$, since U is simple, $U \equiv V \parallel h_0\{M\} \parallel \Pi_{i<n} h_i \triangleright h_{i+1} \parallel h_n[P]$ and:

$$\begin{aligned} U &\xrightarrow{\tau} U' \xRightarrow{\tau} V \parallel \Pi_{i<n} h_i \triangleright h_{i+1} \parallel h_n[P \mid \{M\}] \equiv U_1 \\ U &\xRightarrow{\tau}_{\mathrm{o}} V \parallel \Pi_{i<n} h_i \triangleright h_n \parallel h_n[P \mid \{M\}] \equiv U_2 \qquad \text{(Lemma 4.6)} \end{aligned}$$

We check that $U_1 \mathcal{S}^n U_2$, and we have $U_1 \stackrel{\hat{\tau}}{\Rightarrow}_{\mathrm{o}} U_{1\downarrow o}$ so that from Prop. 4.12, $U_2 \stackrel{\hat{\tau}}{\Rightarrow}_{\mathrm{o}} U_2'$ with $U_{1\downarrow o}(\mathcal{S} \cup \stackrel{\tau}{\to}_{\mathrm{o}})^\star U_2'$. Furthermore, since $\mathcal{S}$ preserves normal forms, $U_{1\downarrow o}\mathcal{S}^\star U_2'$, and $U_2' = U_{2\downarrow o}$
Finally, by induction, $U_{1\downarrow}\mathcal{S}^\star U_{1\downarrow o}$ and $U_\downarrow = U_{1\downarrow}\mathcal{S}^\star U_{1\downarrow o}\mathcal{S}^\star U_{2\downarrow o} = U_{\downarrow o}$. □

It follows that a net is bisimilar to its $\mathcal{E}$-normal form, which leads to the final proof of correctness.

Theorem 4.16. *Let U be an optimised net, we have:*

$$\langle U, \to_{\mathrm{o}} \rangle \;\approx\; \langle \mathrm{e}(U_\downarrow), \to_{\mathrm{o}} \rangle \quad .$$

Proof. From Prop. 4.12, $\stackrel{\tau}{\to}_{\mathrm{o}} \subseteq \approx$, and $U \approx U_{\downarrow o}$. We conclude with Props. 4.13 and 4.15: $U_{\downarrow o} \;\approx\; \mathrm{e}(U_{\downarrow o}) \;\equiv\; \mathrm{e}(U_\downarrow)$. □

Theorem 4.17 (Correctness of the optimisation). *For any simple net U, we have:*

$$\langle U, \to_{\mathrm{o}} \rangle \;\approx\; \langle U, \to \rangle \quad .$$

Proof. Using Theorems 4.16 and 3.8, we can suppose w.l.o.g. that U is $\mathcal{E}$-normal.

Let $\mathcal{R} \triangleq \big\{ \langle \langle U, \to_{\mathrm{o}} \rangle \,,\; \langle U, \to \rangle \rangle \;\big/\; U \text{ is } \mathcal{E}\text{-normal} \big\}$. We show that the symmetric closure of $\mathcal{R}$ is a bisimulation up to $\approx$ (Theorem 4.2): since U is normal, there are only visible challenges to play. Suppose $U \stackrel{a}{\to}_{\mathrm{o}} U'$. The LTSs do not differ on visible actions, hence we have $U \stackrel{a}{\to} U'$. U' is not necessarily $\mathcal{E}$-normal, so that $\langle U', \to_{\mathrm{o}} \rangle \,\mathcal{R}\, \langle U', \to \rangle$ does not hold. However, by Theorems 4.16 and 3.8, we have: $\langle U', \to_{\mathrm{o}} \rangle \;\approx\; \langle \mathrm{e}(U'_\downarrow), \to_{\mathrm{o}} \rangle \;\mathcal{R}\; \langle \mathrm{e}(U'_\downarrow), \to \rangle \;\approx\; \langle U', \to \rangle$.

The challenges offered by $\langle U, \to \rangle$ are handled symmetrically. □

5 Concluding Remarks

The bisimilarity proof in [5]. In the GCPAN [5], which is an optimisation of the PAN [10], we actually add counters to the forwarders and garbage collect forwarders whose counter is 0, to which no message will be sent anymore.

We can build on the results presented here to give a complete correctness proof, by validating each optimisation step. Because in PAN local processes satisfy the requirement of our framework, the PAN with relocating forwarders is equivalent to the original machine. Furthermore, adding counters to the machine does not affect this. We then show that the relation that removes a forwarder with counter 0 is a strong bisimulation. Finally, correctness of the GCPAN is established by checking that the identity relation is a bisimulation up to strong bisimilarity relating the previous nets and GCPAN nets.

Forwarder cycles. In this paper, we assumed that no forwarder cycles can appear during the evolution of a net (Hyp. 2.4). However, Theorem 4.17, that states the correctness of the optimisation, holds without this assumption. Indeed, in both systems, all forwarders belonging or pointing to a cycle, and messages routed along such forwarders, will get trapped in the cycle, rendering this part of the net behaviourally equivalent to the empty net:

– In the initial system, any message trapped in a cycle will keep moving indefinitely along the cycle, producing infinitely many silent transitions (the relation $\xrightarrow{\tau}$ remains confluent, but does no longer terminate).
– In the optimised system, messages reaching the cycle will progressively block its forwarders ($\xrightarrow{\tau}_{\mathrm{o}}$ still terminates, but it is no longer confluent: the shape of the final blocked cycle depends on the order of the reductions).

Furthermore, cycles can only be created by the visible rule [Fwd], and we can check that the set of messages and forwarders trapped in a cycle does not depend on the setting we chose. Hence, in the proof of Theorem 4.17, we could safely remove the cycles and lost messages that appear on both sides, the same way as we normalise processes along the bisimulation game.

Acknowledgements. I would like to thank Daniel Hirschkoff, whose great help and comments have been essential during the preparation of this paper.

References

1. P. Bidinger and J.-B. Stefani. The Kell Calculus: Operational Semantics and Type System. In *Proc. of FMOODS '03*, volume 2884 of *LNCS*, pages 109–123. Springer Verlag, 2003.
2. L. Cardelli and A. Gordon. Mobile Ambients. In *Proc. FOSSACS '98*, volume 1378 of *LNCS*, pages 140–155. Springer Verlag, 1998.
3. G. Castagna and F. Zappa Nardelli. The Seal Calculus Revisited. In *Proc. of FSTTCS '02*, volume 2556 of *LNCS*, pages 85–96. Springer Verlag, 2002.
4. C. Fournet, F. Le Fessant, L. Maranget, and A. Schmitt. JoCaml: A Language for Concurrent Distributed and Mobile Programming. In *Proc. of Advanced Functional Programming 2002*, volume 2638 of *LNCS*, pages 129–158. Springer Verlag, 2002.
5. D. Hirschkoff, D. Pous, and D. Sangiorgi. A Correct Abstract Machine for Safe Ambients. In *Proc. COORD '05*, volume 3454 of *LNCS*. Springer Verlag, 2005.
6. F. Levi and D. Sangiorgi. Mobile Safe Ambients. *ACM Trans. on Progr. Lang. and Sys.*, 25(1):1–69, 2003. ACM Press.
7. R. De Nicola, G.L. Ferrari, and R. Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Trans. Software Eng.*, 24(5):315–330, 1998.
8. D. Pous. Up-to Techniques for Weak Bisimulation. In *Proc. 32th ICALP*, volume 3580 of *LNCS*, pages 730–741. Springer Verlag, 2005.
9. D. Sangiorgi and R. Milner. The problem of "Weak Bisimulation up to". In *Proc. 3rd CONCUR*, volume 630 of *LNCS*, pages 32–46. Springer Verlag, 1992.
10. D. Sangiorgi and A. Valente. A Distributed Abstract Machine for Safe Ambients. In *Proc. 28th ICALP*, volume 2076 of *LNCS*. Springer Verlag, 2001.
11. A. Unyapoth and P. Sewell. Nomadic pict: Correct Communication Infrastructure for Mobile Computation. In *Proc. 28th POPL*, pages 116–127. ACM Press, 2001.