On Bisimulation Proofs for the Analysis of Distributed Abstract Machines

Damien Pous

On Bisimulation Proofs for the Analysis of Distributed Abstract Machines [1]

Damien Pous [2]

LIP, ENS Lyon - CNRS - INRIA - UCBL, 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 labelled transition system, 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.

Key words: weak bisimilarity, up-to techniques, abstract machines.

Introduction

Recently, many calculi encompassing distribution and mobility have been studied as a basis for programming languages. Examples include Join [5], Distributed Pi [6], Nomadic Pict [16], Kells [2], Ambients [3], Klaim [10], Seals [4]. The expressive power supplied by the primitives underlying such models raises the question of implementability in a distributed framework. In [14], a distributed abstract machine is defined, to implement the Safe Ambient Calculus [8]: the PAN. The main ingredients in the definition of this

Email address: Damien.Pous@ens-lyon.fr (Damien Pous).

[1] An early version of this paper has appeared in the proceedings of TGC '06, Luca, Italy, November 2006 [12].

[2] This work has been supported by the french project ANR ARASSIA "Modularité Dynamique Fiable".

machine are *locations* – where local processes are executed – and *forwarders*, that transmit messages between locations. In [7], 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 introduced new up-to techniques for weak bisimulation [11]. 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. In this paper, we show how these techniques can be used to develop bisimulation proofs about nontrivial systems, and how to do so in a modular way.

The first contribution of this work is the definition of a framework to define and reason about distributed implementations of process algebras with mobility.

We want to represent the execution of a collection of *local processes*, distributed over a set of *locations* – thought of as asynchronous, independent entities. Each local process may:

- evolve by itself, independently from local processes running at distant sites;
- send arbitrary messages to the local processes hosted in other locations;
- receive such messages;
- spawn new local processes, inside new locations;
- migrate to another location, and redirect further messages to that location.

We abstract over the structure of local processes by representing them by the states of an arbitrary labelled transition system (LTS), whose labels correspond to the above primitives. These local processes, together with the set of messages they exchange, form the input of our framework. We do not make any assumption on the behaviour of local processes, or about the content of messages. Notably, messages may contain locations or processes, so that π-calculus-like name mobility or higher-order messages are allowed.

We then define another family of LTSs, called *abstract nets*, whose labels represent reactions to the primitives: in a sense these labels form the interface of libraries to which local processes can be linked, or equivalently, the interface of environments in which local processes may be executed. Before defining the "body" of such libraries, we show how to "link" them to local processes, by showing how to compose any abstract net (an LTS), with the LTS formed by local processes. Intuitively, the *composite LTS* that we obtain executes the local processes concurrently and react to the actions they request: sending a message, spawning a location, etc... This approach allows us to separate

completely the study of local processes from the study of the library: weak bisimilarity, like other standard behavioural equivalences, is preserved by this composition (Prop. 2.3).

We then study several concrete instantiations of abstract nets, that actually describe the behaviour of the network.

At first, we define a specification, by providing a *reference LTS* that describes the expected behaviour the system using a rather coarse grain semantics. This specification makes use of *forwarders* to record the set of locations whose local processes decided to migrate to another location. The transitions of the LTS are given by means of high-level inference rules that route messages through forwarders in an atomic way.

While the reference is well-suited to reason about the whole system (the composite LTS, containing local processes), it cannot directly be implemented in a distributed way: several locations are involved in order to decide that a single action shall occur. We refine this specification into a simple implementation by defining a *simple LTS*, where internal – local – actions are used to mimic atomic actions of the reference: messages are routed along forwarders in an asynchronous way.

The main drawback of this simple implementation is the persistence of forwarder chains, that slow down the communications between local processes. To address this inefficiency, we introduce an optimisation, inspired from [7]. This optimisation exploits a forwarder relocation mechanism, that contracts forwarder chains. It is expressed as a third *optimised LTS*, that refines the simple one.

While the initial definition of this framework [12] forms the basis of the PAN abstract machine [14], we got rid of all 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 2.7).

Our second main contribution is an illustration of non-trivial use of recent proof techniques to reason about a rather complex system. We achieve this by proving the correctness of the simple implementation, and of its optimisation, that is, that both the simple and the optimised LTS are weakly bisimilar to the reference LTS. The main steps of the proof are summed up in Fig. 1; we explain them below.

A first difficulty comes from the fact that, unlike in [12], we handle the cases where forwarder cycles are created. This occurs for example, when a local process migrates to a location that actually points to its current location. These

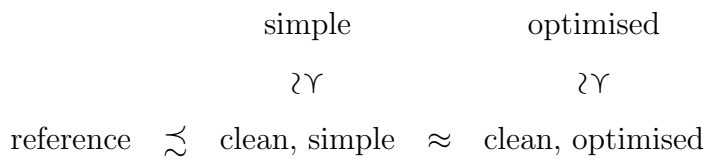

Fig. 1. Main steps in the proof of correctness.

cases can be considered as erroneous behaviours of local processes; however they cannot easily be detected or prevented (at least at our abstract level, where local processes are seen as "black boxes"). Therefore, we prove that our implementations resists to these cases: simple and optimised LTSs remain weakly bisimilar to the reference LTS, even in presence of forwarder cycles. In doing so, we leave the choice of preventing or supporting such cycles to the designer of local process.

Basically, messages sent along forwarder cycles are *lost*: they will never reach a local process. Therefore, we can remove such messages without changing the behaviour of the system. We can express this property using the *expansion preorder* ($\succsim$) [6]: a state of the system expands the state obtained by removing all lost messages. This fact allows us to prove that by "patching" the simple LTS so that it systematically removes lost messages, we obtain a behaviourally equivalent LTS, called *clean*, where lost messages do not exist. Of course, the clean LTS is not realistic from an implementation point of view: detecting that a message is lost may require an analysis of the whole network. This step makes it possible to reason on a simpler system, which is free of the irregularities that appear with lost messages.

This important expansion result can be established for both the simple and the optimised LTS, with little more work in the latter case.

Once the problem has been reduced to the case of clean LTSs, we have to show that the routing of messages, which is achieved by silent transitions, is done correctly in both systems. This involves:

- proving that any message eventually reaches its actual destination, and
- proving that the routing of messages does not affect the behaviour of the system.

The first point is quite easy: for any state, we show that there exists a sequence of internal transitions that brings a given message to its destination. This is almost straightforward in the simple LTS, and requires only little care in the optimised one.

The second point is more demanding: in some sense, we have to show that any internal transition "commutes" with any potential evolution of the system.

In the simple LTS, internal transitions are actually contained in expansion, which makes it possible to use standard up-to techniques. On the contrary, like in [7,12], the relocation mechanism introduced in the optimisation breaks this proof strategy, as the expansion result does no longer hold. We show in details how one of the techniques we developed in [11] makes it possible to give a modular proof of correctness, where the bisimulation candidate that we manipulate remains tractable and expresses only local properties of processes.

Being able to work with small bisimulation candidates is really helpful: 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.

Outline of the paper. In Sect. 1, we introduce our notations and the notions used in the sequel. We define the abstract framework and its simple implementation in Sect. 2. Section 3 is devoted to the definition of the optimisation, and to the corresponding correctness proof.

1 Labelled Transition Systems, Bisimilarity

We assume in this section a set $\mathcal{L}$ of *labels* (or *actions*); this set will be instantiated in different ways in the following sections. Since we will work with weak equivalences, we require that $\mathcal{L}$ contains a distinguished *silent* (or *internal*) action, denoted by τ.

Definition 1.1 (Labelled Transition System, Processes)**.** We call *$\mathcal{L}$-transition system ($\mathcal{L}$-TS)* any pair $\langle \mathcal{P}, \hookrightarrow \rangle$ where $\mathcal{P}$ is an arbitrary set of *states*, and $\hookrightarrow \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$ is a set of *labelled transitions*. $\mathcal{P}$ is called the *domain* of the $\mathcal{L}$-TS.
An *$\mathcal{L}$-process* is a rooted $\mathcal{L}$-TS: a pair $\langle P, \langle \mathcal{P}, \hookrightarrow \rangle \rangle$ where $\langle \mathcal{P}, \hookrightarrow \rangle$ is a $\mathcal{L}$-TS, and the *root* P is a distinguished state of the domain, $\mathcal{P}$.

We write $P \overset{\alpha}{\hookrightarrow} P'$ when $\langle P, \alpha, P' \rangle \in \hookrightarrow$. We shall often denote an $\mathcal{L}$-TS $\langle \mathcal{P}, \hookrightarrow \rangle$ by the set $\hookrightarrow$ of its labelled transitions, when the domain is irrelevant or clear from $\hookrightarrow$. Accordingly, an $\mathcal{L}$-process $\langle P, \langle \mathcal{P}, \hookrightarrow \rangle \rangle$ will be denoted by $\langle P, \hookrightarrow \rangle$, or P, when the $\mathcal{L}$-TS is clear from the context.

Until the end of this section we define several notions that actually depend on $\mathcal{L}$. We do not make this dependency explicit in order to alleviate notations.

We let p, q, r range over $\mathcal{L}$-processes and we let $\mathcal{R}, \mathcal{S}, \mathcal{E}$ range over binary relations between $\mathcal{L}$-processes (simply called *relations* in the sequel). When a relation relates $\mathcal{L}$-processes sharing a unique $\mathcal{L}$-TS, it will be identified with the corresponding binary relation between the states of that $\mathcal{L}$-TS. Let $\mathcal{R}$, $\mathcal{S}$ be two relations; we write $p\ \mathcal{R}\ q$ when $\langle p, q\rangle \in \mathcal{R}$; we denote respectively by $\mathcal{R}^+$, $\mathcal{R}^=$, $\mathcal{R}^\star$ the transitive, reflexive, transitive and reflexive closures of $\mathcal{R}$; the relational composition of $\mathcal{R}$ and $\mathcal{S}$, written $\mathcal{R}\mathcal{S}$, is defined by $\mathcal{R}\mathcal{S} \triangleq \{\langle p, r\rangle \mid p\ \mathcal{R}\ q \text{ and } q\ \mathcal{S}\ r \text{ for some } q\}$; the converse of $\mathcal{R}$ is $\mathcal{R}^{-1} \triangleq \{\langle p, q\rangle \mid q\ \mathcal{R}\ p\}$; $\mathcal{R}$ is symmetric if $\mathcal{R} = \mathcal{R}^{-1}$; 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 $p\ \mathcal{R}\ q$. Finally we denote by $\mathcal{I}$ the reflexive relation.

Definition 1.2 (Transition Relations)**.** Any action $\alpha \in \mathcal{L}$ induces a relation, denoted by $\xrightarrow{\alpha}$:

$$\xrightarrow{\alpha} \triangleq \left\{ \langle \langle P, \hookrightarrow \rangle,\ \langle P', \hookrightarrow \rangle \rangle \mid \text{for any } \mathcal{L}\text{-TS } \hookrightarrow \text{ and states } P, P' \text{ s.t. } P \xrightarrow[\hookrightarrow]{\alpha} P' \right\}.$$

Its converse is written using a reversed arrow: $\xleftarrow{\alpha} = (\xrightarrow{\alpha})^{-1}$, and similarly for other forms of arrows, like the following *weak transition relations*:

$$\xrightarrow{\widehat{\alpha}} \triangleq \begin{cases} \xrightarrow{\tau}{}^{=} & \text{if } \alpha = \tau \\ \xrightarrow{\alpha} & \text{otherwise} \end{cases} \qquad \overset{\alpha}{\Rightarrow} \triangleq \xrightarrow{\tau}{}^{\star} \xrightarrow{\alpha} \xrightarrow{\tau}{}^{\star} \qquad \overset{\widehat{\alpha}}{\Rightarrow} \triangleq \xrightarrow{\tau}{}^{\star} \xrightarrow{\widehat{\alpha}} \xrightarrow{\tau}{}^{\star}$$

We can remark the following properties: $\overset{\widehat{\tau}}{\Rightarrow} = \xrightarrow{\tau}{}^{\star}$, $\overset{\tau}{\Rightarrow} = \xrightarrow{\tau}{}^{+}$ and $\overset{\widehat{\alpha}}{\Rightarrow} = \overset{\alpha}{\Rightarrow}$ when $\alpha \neq \tau$ (note in particular the difference between $\overset{\widehat{\tau}}{\Rightarrow}$ and $\overset{\tau}{\Rightarrow}$).

Definition 1.3 (Simulation, Bisimulation, Bisimilarity)**.** A relation $\mathcal{R}$ is a *simulation* if for any label $\alpha \in \mathcal{L}$ and $\mathcal{L}$-processes p, q, p', we have:

$$p\ \mathcal{R}\ q \text{ and } p \xrightarrow{\alpha} p' \text{ entail } q \overset{\widehat{\alpha}}{\Rightarrow} q' \text{ and } p'\ \mathcal{R}\ q' \text{ for some } q'.$$

A *bisimulation* is a symmetric simulation. *Bisimilarity*, denoted by $\approx$, is the union of all bisimulations.

Proposition 1.4. *Bisimilarity is an equivalence relation, it is the greatest bisimulation.*

By defining bisimilarity as a relation between rooted $\mathcal{L}$-TS ($\mathcal{L}$-processes) rather than between the states of a given $\mathcal{L}$-TS, we gain the ability to relate processes which are associated to different transitions systems. This will be useful in order to compare the different versions of our framework. We finally define *expansion* ($\succsim$), the standard behavioural preorder [1] that leads to the correct "bisimulation up to expansion" technique [13].

Definition 1.5 ((Pre)-Expansion)**.** A *pre-expansion relation* is a relation $\mathcal{R}$ such that for any $\alpha \in \mathcal{L}$, whenever $p\ \mathcal{R}\ q$ we have:

$$\text{if } p \xrightarrow{\alpha} p' \text{ then } q \xrightarrow{\widehat{\alpha}} q' \text{ and } p'\ \mathcal{R}\ q' \text{ for some } q'.$$

An *expansion relation* is a pre-expansion relation $\mathcal{R}$ such that $\mathcal{R}^{-1}$ is a simulation. *Expansion*, denoted by $\precsim$, is the union of all expansion relations.

Proposition 1.6. *Expansion is a preorder contained in bisimilarity, it is the greatest expansion relation.*

We have that $\mathcal{R}$ is a simulation iff $\forall \alpha \in \mathcal{L},\ \xleftarrow{\alpha} \mathcal{R} \subseteq \mathcal{R} \xleftarrow{\widehat{\alpha}}$; accordingly, $\mathcal{R}$ is a pre-expansion iff $\forall \alpha \in \mathcal{L},\ \xleftarrow{\alpha} \mathcal{R} \subseteq \mathcal{R} \xleftarrow{\widehat{\alpha}}$. This kind of concise definitions will be used in sequel to state various up-to techniques involving such kind of "challenges" or "games".

We shall use the notation $\widetilde{x}$ to denote finite multisets of various kind of elements. The empty multiset is denoted by $\emptyset$ and we denote by $x;\widetilde{x}$ (resp. $\widetilde{y};\widetilde{x}$) the addition of an element x (resp. a multiset $\widetilde{y}$) to a multiset $\widetilde{x}$.

Definition 1.7 (Termination)**.** A relation $\mathcal{R}$ *terminates* if there is no infinite sequence $(p_i)_{i\in\mathbb{N}}$ such that $\forall i, p_i\ \mathcal{R}\ p_{i+1}$.

2 A Framework for Distributed Computation

2.1 Abstract Description

We let h, k range over a given set $\mathcal{H}$ of *locations*. We assume a set of *elementary messages*, and we let m, n range over finite multisets of elementary messages, that we simply call *messages*; their set is denoted by $\mathcal{M}$. We let α range over a given set $\mathcal{L}$ of *labels* (containing the distinguished element τ). We let P, Q range over a given set $\mathcal{P}$ of *local processes*.

We define the set $\mathcal{L}_p$ of *process labels* with the following syntax:

$\delta ::= \ \alpha$	(local action)
$\mid h\langle m\rangle$	(message emission)
$\mid (m)$	(message reception)
$\mid \triangleright h$	(migration)
$\mid \nu h[P]$	(location creation)

We assume an $\mathcal{L}_p$-TS with domain $\mathcal{P}$: $\langle \mathcal{P}, \hookrightarrow \rangle$. This is the only $\mathcal{L}_p$-TS we shall

consider, so that a local process $P \in \mathcal{P}$ will implicitly represent the $\mathcal{L}_p$-process $\langle P, \hookrightarrow \rangle$.

The process labels correspond to the primitives we alluded to in the introduction: α corresponds to a standalone transition of a local process, $h\langle m \rangle$ to the emission of message m to location h, (m) to the reception of message m, $\triangleright\, h$ to the migration to some location h, and $\nu h[P]$ to the spawning of a new location h, with local process P.

A *distributed process* is a finite mapping from $\mathcal{H}$ to $\mathcal{P}$, that we write as follows: $h_1 : P_1, \ldots, h_n : P_n$. In this situation, we say that the local process P_i *is hosted at* h_i. The set of distributed processes, ranged over with D, is denoted by $\mathcal{DP}$. In order to represent the execution of distributed processes, we define a notion of abstract net and we show how to compose the states of an abstract net with a distributed process.

Definition 2.1 (Abstract Nets)**.** We call *abstract net* any $\mathcal{L}_n$-process, where $\mathcal{L}_n$ is the set of *net labels* defined below:

$\mu ::= \tau$	(internal action)
$\mid h\langle m \rangle$	(message emission)
$\mid h(m)$	(message reception)
$\mid h \triangleright k$	(migration)
$\mid \nu h$	(location creation)

Net labels closely correspond to process labels: they are actually associated pairwise in order to define the following *composite LTS*, that describes the execution of a distributed process, inside a given $\mathcal{L}_n$-TS:

Definition 2.2 (Composition of Abstract Nets and Distributed Processes)**.** We associate to any $\mathcal{L}_n$-TS $\langle \mathcal{U}, \rightarrow_u \rangle$ the $\mathcal{L}$-TS $\langle \mathcal{U} \times \mathcal{DP}, \mapsto_u \rangle$ where $\mapsto_u$ is defined defined by the inference rules given in Fig. 2 (states of the domain are denoted by $U \circ D$).

By rule [Loc$_\circ$], a local process may evolve on its own; accordingly the net may achieve some internal transitions by rule [Int$_\circ$]. A local process hosted at h can send a message to k via rule [Snd$_\circ$], and receive messages by rule [Rcv$_\circ$]. Rule [New$_\circ$] allows a local process hosted at h to spawn a process Q at new location k. Finally, by rule [Mig$_\circ$], a local process hosted at h may migrate to some location k; in that case, the continuation of the local process (P') is lost. Notice that this is not restrictive: a local process that has to execute some code at k, after the migration (as is often the case in $D\pi$ [6]), can send a message to k before the migration, containing the code to execute.

$$[\textsc{Loc}_\circ]\ \frac{P \overset{\alpha}{\hookrightarrow} P'}{U \circ D, h : P \overset{\alpha}{\mapsto}_u U \circ D, h : P'} \qquad [\textsc{Int}_\circ]\ \frac{U \overset{\tau}{\rightarrow}_u U'}{U \circ D \overset{\tau}{\mapsto}_u U' \circ D}$$

$$[\textsc{Snd}_\circ]\ \frac{U \overset{k\langle m\rangle}{\longrightarrow}_u U' \qquad P \overset{k\langle m\rangle}{\hookleftarrow} P'}{U \circ D, h : P \overset{\tau}{\mapsto}_u U' \circ D, h : P'} \qquad [\textsc{Rcv}_\circ]\ \frac{U \overset{h(m)}{\longrightarrow}_u U' \qquad P \overset{(m)}{\hookleftarrow} P'}{U \circ D, h : P \overset{\tau}{\mapsto}_u U' \circ D, h : P'}$$

$$[\textsc{New}_\circ]\ \frac{U \overset{\nu k}{\longrightarrow}_u U' \qquad P \overset{\nu k[Q]}{\hookleftarrow} P'}{U \circ D, h : P \overset{\tau}{\mapsto}_u U' \circ D, h : P', k : Q} \qquad [\textsc{Mig}_\circ]\ \frac{U \overset{h \triangleright k}{\longrightarrow}_u U' \qquad P \overset{\triangleright k}{\hookleftarrow} P'}{U \circ D, h : P \overset{\tau}{\mapsto}_u U' \circ D, \emptyset}$$

Fig. 2. Transitions for the composite $\mathcal{L}$-TS.

Remark that since the communication is achieved by unification (rule $[\textsc{Rcv}_\text{S}]$ requires that the network and the local process agree on the same message m), we actually impose an "early" semantics [15] to local processes. Adapting this rule to the case where local processes have a "late" semantics is possible; however this would require us to ask for a notion of substitution over local processes. Also notice that while we exchange multisets of elementary messages (this facilitates our notations in the sequel), the early semantics allows a local process to decide to receive only single elementary messages.

The main advantage of this approach, where local processes are strongly separated from the network, is that we can study these two entities separately, as expressed by the following proposition:

Proposition 2.3. *Let $\langle U, \rightarrow_u\rangle$ and $\langle V, \rightarrow_v\rangle$ be two abstract nets, let P, Q be two local processes and let D be a distributed process.*

- *If $\langle U, \rightarrow_u\rangle \approx \langle V, \rightarrow_v\rangle$ then $\langle U \circ D, \mapsto_u\rangle \approx \langle V \circ D, \mapsto_v\rangle$.*
- *If $P \approx Q$ then $\langle U \circ D, h : P, \mapsto_u\rangle \approx \langle U \circ D, h : Q, \mapsto_u\rangle$.*

Proof. We first show that for any states U, U', local processes P, P', distributed process D, and location h, we have:

- $U \overset{\tau}{\rightarrow}_u^\star U'$ entails $U \circ D \overset{\tau}{\mapsto}_u^\star U' \circ D$, and
- $P \overset{\tau}{\hookrightarrow}^\star P'$ entails $U \circ D, h : P \overset{\tau}{\mapsto}_u^\star U \circ D, h : P'$.

It follows easily that the two relations below are bisimulations:

$$\mathcal{R}_1 \triangleq \{\langle U \circ D,\ V \circ D\rangle \mid \forall U, V,\ \text{s.t. } \langle U, \rightarrow_u\rangle \approx \langle V, \rightarrow_v\rangle\} \ ,$$
$$\mathcal{R}_2 \triangleq \{\langle U \circ D, h : P,\ U \circ D, h : Q\rangle \mid \forall U, P, Q,\ \text{s.t. } P \approx Q\} \cup \mathcal{I} \ . \qquad \blacksquare$$

We can show that Prop. 2.3 also holds for stronger behavioural preorders or equivalences like expansion ($\succsim$) or strong bisimilarity [9].

reference nets	*simple nets*	*optimised nets*	
$U ::= \mathbf{0}$	$U ::= \mathbf{0}$	$U ::= \mathbf{0}$	(empty net)
$\mid U \mid U$	$\mid U \mid U$	$\mid U \mid U$	(parallel composition)
$\mid h[m]$	$\mid h[m]$	$\mid h[m]$	(real location)
$\mid h \triangleright k$	$\mid h \triangleright k$	$\mid h \triangleright k$	(forwarder)
	$\mid h\langle m \rangle$	$\mid h\langle m \rangle_{\widetilde{k}}$	(pending message)
		$\mid h \not\triangleright k$	(blocked forwarder)
		$\mid h\langle \triangleright k \rangle$	(relocation message)

Fig. 3. Syntax of reference, simple and optimised nets.

2.2 Specification of the Expected Behaviour

While the abstract view we have introduced allows us to define several execution models, it does not specify their expected behaviour: only some of all possible $\mathcal{L}_n$-TS, each corresponding to a possible behaviour, are of interest. In order to fix the expected behaviour we define a "reference $\mathcal{L}_n$-TS"; valid (interesting) execution models will correspond to $\mathcal{L}_n$-TS which are bisimilar to this reference $\mathcal{L}_n$-TS.

The syntax of *nets* is given in Fig. 3. At first we only consider *reference nets*: the two other extensions of this syntax will be studied later on. We implicitly reason modulo associativity and commutativity of the parallel composition ($\mid$) which admits $\mathbf{0}$ as a neutral element. Therefore, a reference net is basically a finite multiset where each element is either

- a real location $h[m]$ containing a message m (recall that a message is actually a collection of elementary messages) – in the composite $\mathcal{L}$-TS (Def. 2.2), a local process will be running at such locations;
- or a forwarder $h \triangleright k$ that will redirect any message for h to k.

Therefore, locations are independent entities that host some "agents" which have to be uniquely identified. This is what we express using the following well-formedness property:

Definition 2.4 (Well-formedness)**.** A reference net U is *well-formed* if any location $h \in \mathcal{H}$ appears at most once as a real location ($h[m]$) or as the source of a forwarder ($h \triangleright k$) in U.
For any reference net U, we call *locations defined in* U – denoted by $l(U)$ – the set of locations that appear at such positions in U.

Well-formed nets could thus be defined as finite mappings, associating an agent to the element of a finite subset of $\mathcal{H}$. We prefer our explicitly concurrent

$$[\textsc{Rcv}]\ \frac{}{h[m;n] \mid U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} h[n] \mid U} \qquad [\textsc{Snd}]\ \frac{}{h[n] \mid U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} h[m;n] \mid U}$$

$$[\textsc{Kil}]\ \frac{h \notin l(U)}{U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} U} \qquad [\textsc{Fwd}]\ \frac{U \xrightarrow{k\langle m\rangle}_{\mathrm{r}} U'}{h \triangleright k \mid U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} h \triangleright k \mid U'}$$

$$[\textsc{New}]\ \frac{h \notin l(U)}{U \xrightarrow{\nu h}_{\mathrm{r}} h[\emptyset] \mid U} \qquad [\textsc{Mig}]\ \frac{U \xrightarrow{k\langle m\rangle}_{\mathrm{r}} U'}{h[m] \mid U \xrightarrow{h \triangleright k}_{\mathrm{r}} h \triangleright k \mid U'}$$

Fig. 4. Transitions of reference nets.

syntax, which we find both easier to manipulate and more intuitive.

We can now define the dynamics of our specification:

Definition 2.5 (Reference $\mathcal{L}_n$-TS)**.** The *reference $\mathcal{L}_n$-TS* is $\langle \mathcal{U}_{\mathrm{r}}, \rightarrow_{\mathrm{r}} \rangle$, where $\mathcal{U}_{\mathrm{r}}$ is the set of well-formed reference nets, and $\rightarrow_{\mathrm{r}}$ is defined by the inference rules of Fig. 4.

The following lemma validates our definition, and allows us to consider only well-formed nets in the sequel.

Lemma 2.6. *Well-formed nets* are preserved by $\rightarrow_r$ *, i.e., if* $U \xrightarrow{\mu}_r U'$ *and* U *is well-formed, then* U' *is well-formed. Hence,* $\langle \mathcal{U}_r, \rightarrow_r \rangle$ *is an* $\mathcal{L}_n$*-TS.*

We briefly describe the transitions of reference nets (Fig. 4): when a real location contains some message m, the latter can be received by the local process hosted at that location (rule [Rcv], recall that messages m and n are multiset of elementary messages). When a message is sent to a real location, it is added to the other elementary messages of that location (rule [Dst]); by rule [Kil], any message sent on a location which is not defined just disappears; when a message is sent to a location hosting a forwarder $h \triangleright k$, it is redirected to k by rule [Fwd]: by successive applications of this rule, it will either disappear, or be added into some real location. Notice that since the forwarder is removed in the premise of rule [Fwd], any message trapped in a forwarder cycle will eventually be removed by rule [Kil]. Rule [New] allows one to add a new real location, provided that it is not yet defined. Finally a real location $h[m]$ may migrate to some location k by rule [Mig]; in that case, the message m is first routed to k (m contains the elementary messages that reached h, but that the local process hosted at h has not yet consumed), and the real location is replaced by a forwarder $h \triangleright k$.

This reference $\mathcal{L}_n$-TS is a rather high-level one: it does not use internal transitions, and the routing of messages through forwarders is achieved in one atomic

step, by successive applications of rule [Fwd]. We can moreover remark that the transitions are deterministic: if $U \xrightarrow{\mu}_{\mathrm{r}} U_1$ and $U \xrightarrow{\mu}_{\mathrm{r}} U_2$, then $U_1 = U_2$. As hinted in the introduction, this makes a distributed implementation of this $\mathcal{L}_n$-TS hardly feasible.

Remark 2.7 (On the expressiveness of the framework)**.** Locations express only the *logical* distribution of processes. Hence, the well-formedness condition does not rule out the case where several locations are thought of as residing *physically* on the same device. Also, unlike in [14,7], 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. Remark also that migration is *subjective* in our model: local processes decide to migrate by themselves. *Objective* migration mechanisms (like the *passivation* available in the Kell-calculus [2]) may be simulated by using messages to trigger migrations.

2.3 A Simple Implementation of the Specification

We now define another $\mathcal{L}_n$-TS, which is more tractable from an implementation point of view. We validate this implementation by proving that it is bisimilar to the reference $\mathcal{L}_n$-TS.

We consider *simple nets* whose syntax – which is given in Fig. 3 – extends that of reference nets by adding pending messages. They will be used to track the intermediate steps corresponding to the routing of pending messages trough forwarders.

We extend Def. 2.4 to simple nets by ignoring pending messages and we can define the corresponding dynamics:

Definition 2.8 (Simple $\mathcal{L}_n$-TS)**.** The *simple $\mathcal{L}_n$-TS* is $\langle \mathcal{U}_{\mathrm{s}}, \rightarrow_{\mathrm{s}} \rangle$, where $\mathcal{U}_{\mathrm{s}}$ is the set of well-formed simple nets, and $\rightarrow_{\mathrm{s}}$ is defined in Fig. 5.

Lemma 2.9. *Well-formedness of simple nets is preserved by $\rightarrow_s$; hence, $\langle \mathcal{U}_s, \rightarrow_s \rangle$ is an $\mathcal{L}_n$-TS.*

Rules [$\mathrm{Rcv_s}$] and [$\mathrm{New_s}$] are unchanged, rule [$\mathrm{Snd_s}$] is simplified: we just add the message to the net, as a pending message. The two silent transition rules are concerned with the routing of messages: rule [$\mathrm{Fwd_s}$] defines the behaviour of forwarders: they transmit the messages; rule [$\mathrm{Dst_s}$] performs the actual reception of a message at a real location. Notice that rule [$\mathrm{Mig_s}$], which may seem different from [Mig], is actually the same, due to the new definition of $\xrightarrow{h\langle m \rangle}_{\mathrm{s}}$. Finally, there is no rule corresponding to rule [Kil]: messages whose destination is "undefined" persist as garbage in simple nets.

$$[\textsc{Rcv}_\textsc{s}]\ \frac{}{h[m;n] \mid U \xrightarrow{h(m)}_\textsc{s} h[n] \mid U} \qquad [\textsc{Snd}_\textsc{s}]\ \frac{}{U \xrightarrow{h\langle m\rangle}_\textsc{s} h\langle m\rangle \mid U}$$

$$[\textsc{Dst}_\textsc{s}]\ \frac{}{h\langle m\rangle \mid h[n] \mid U \xrightarrow{\tau}_\textsc{s} h[m;n] \mid U}$$

$$[\textsc{Fwd}_\textsc{s}]\ \frac{}{h\langle m\rangle \mid h \triangleright k \mid U \xrightarrow{\tau}_\textsc{s} h \triangleright k \mid k\langle m\rangle \mid U}$$

$$[\textsc{Mig}_\textsc{s}]\ \frac{}{h[m] \mid U \xrightarrow{h \triangleright k}_\textsc{s} h \triangleright k \mid k\langle m\rangle \mid U} \qquad [\textsc{New}_\textsc{s}]\ \frac{h \notin l(U)}{U \xrightarrow{\nu h}_\textsc{s} h[\emptyset] \mid U}$$

Fig. 5. Transitions of simple nets.

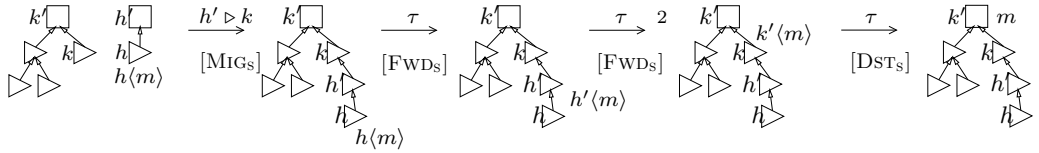

Fig. 6. Migration and routing of messages in simple nets.

A sequence of transitions is depicted in Fig. 6, where squares and triangles respectively represent real locations and forwarders; location h' migrates to k and the message $h'\langle m\rangle$ is routed to k', its final destination.

We can remark that the rules defining the simple $\mathcal{L}_n$-TS have no premise, and that each of these rules affects only one location. This is what makes this $\mathcal{L}_n$-TS well-suited for a distributed implementation: the programs running at each location can be executed asynchronously.

2.4 Correctness of the Simple Implementation

In the sequel, we denote by $\Pi_{i\in I}\ U_i$ the parallel composition of some nets $(U_i)_{i\in I}$.

Definition 2.10 (Destination, Lost Locations, Clean Nets)**.** We say that h *has destination* k in a simple net U if there exist some l, m, V and $(h_i)_{i\leq l}$ with $h = h_0$, $k = h_l$ such that: $U = V \mid \Pi_{i<l}\ h_i \triangleright h_{i+1} \mid h_l[m]$. We say that h *is lost in* U if h has no destination in U.
A simple net U is called *clean* if all its pending messages are sent on locations that have a destination in U. We denote by $\lfloor \mathcal{U}_\textsc{s} \rfloor$ the set of clean simple nets.

Intuitively, a location is lost in U when it does not belong to $l(U)$ or when it belongs to a cycle of forwarders. Notice that any reference net is clean, and that thanks to the well-formedness condition, each location has at most one

destination in a given simple net. We have the following properties:

Lemma 2.11. *(1) If h is lost in U, and $U \xrightarrow{\mu}_s U'$, then h is lost in U'.*
(2) h is lost in $U = h \triangleright k \mid V$, if and only if k is lost in V.
(3) If h is lost in a reference net U then for any message m, $U \xrightarrow{h\langle m\rangle}_r U$.
(4) If h has destination k in a reference net of the form $U \mid k[n]$, then for any message m, $U \mid k[n] \xrightarrow{h\langle m\rangle}_r U \mid k[m; n]$.

Proof. (1) Straightforward.
(2) We check that h has destination k' in U iff k has destination k' in V.
(3) We proceed by induction on the size of U. Since h is lost in U, either
- $h \notin l(U)$, and $U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} U$ by rule [KIL];
- or $U = h \triangleright k \mid V$, by (2), k is lost in V, so that we have $V \xrightarrow{k\langle m\rangle}_{\mathrm{r}} V$ by induction. We conclude by applying the rule [FWD]: $U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} U$.

(4) We have $U = V \mid \Pi_{i<l}\ h_i \triangleright h_{i+1}$ with $k = h_l$, and we can proceed by induction on l. ■

In the sequel, we shall say that a pending message has destination k (in a net) when it is emitted on a location which has destination k (in that net). Similarly, we shall say that a pending message is lost when it is emitted on a lost location. This is justified by the following result:

Lemma 2.12. *Let h be a lost location in a simple net U. For any message m, we have:*

$$U \mid h\langle m\rangle \gtrsim U \ .$$

Proof. We check that $\mathcal{R} \triangleq \{\langle U \mid h\langle m\rangle, U\rangle \mid U \in \mathcal{U}_{\mathrm{s}},\ h \text{ lost in } U\}$ is an expansion relation. We obviously have that $U \xrightarrow{\mu}_{\mathrm{s}} U'$ entails $U \mid h\langle m\rangle \xrightarrow{\mu}_{\mathrm{s}} U' \mid h\langle m\rangle$, and Lemma 2.11(1) ensures that h is still lost in U'. For the other direction, the only non-trivial case is when $U = V \mid h \triangleright k$, $U \mid h\langle m\rangle \xrightarrow{\tau}_{\mathrm{s}} U \mid k\langle m\rangle$. In that case, by Lemma 2.11(2), k is lost in V, so that $U \mid k\langle m\rangle\ \mathcal{R}\ U$: the right-hand-side process does not move. ■

At this point, we can show directly that the relation $\xrightarrow{\tau}_{\mathrm{s}}$ is contained in $\gtrsim$. Together with Lemma 2.12, this would suffice to prove Theorem 2.23 below. We however introduce a somewhat artificial step in the proof, that will be useful in Sect. 3.

The converse of Lemma 2.11(1) does not hold: by rule [MIG$_{\mathrm{S}}$], it is possible to introduce forwarder cycles, so that some locations may become lost locations along the visible action $h \triangleright k$. This entails that the set $\lfloor \mathcal{U}_{\mathrm{s}} \rfloor$ of clean simple

nets is not preserved by $\rightarrow_{\mathrm{s}}$ (rule [$\textsc{Snd}_{\mathrm{S}}$] is also problematic, as it may add arbitrary pending messages to the net).

For any simple net U, we denote by $\lfloor U \rfloor$ the clean net obtained from U by removing all messages pending on lost locations. We can use this "normalisation" function to define an $\mathcal{L}_n$-TS over clean simple nets ($\lfloor \mathcal{U}_{\mathrm{s}} \rfloor$). This $\mathcal{L}_n$-TS is not realistic from an implementation point of view, as it is able to detect lost messages; however, we will only use it to reason about the simple $\mathcal{L}_n$-TS, by showing that both systems are behaviourally equivalent.

Definition 2.13 (Clean $\mathcal{L}_n$-TS)**.** The *clean $\mathcal{L}_n$-TS* is $\left\langle \lfloor \mathcal{U}_{\mathrm{s}} \rfloor, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle$, where $\rightarrow_{\lfloor \mathrm{s} \rfloor}$ is defined by the following rule:

$$\frac{U \xrightarrow{\mu}_{\mathrm{s}} U'}{U \xrightarrow{\mu}_{\lfloor \mathrm{s} \rfloor} \lfloor U' \rfloor}$$

The following lemma shows that the use of the cleaning function is only required after a migration or the emission of a message (rules [$\textsc{Fwd}_{\mathrm{S}}$], [$\textsc{Mig}_{\mathrm{S}}$]):

Lemma 2.14. *For any $U \in \lfloor \mathcal{U}_s \rfloor$, and $\mu \notin \{h \triangleright k; h\langle m \rangle\}$, if $U \xrightarrow{\mu}_s U'$ then U' is clean: $U \xrightarrow{\mu}_{\lfloor s \rfloor} U' = \lfloor U' \rfloor$.*

Lemma 2.12 entails that $\langle U, \rightarrow_{\mathrm{s}} \rangle \succsim \langle \lfloor U \rfloor, \rightarrow_{\mathrm{s}} \rangle$: any simple net U expands its cleaned form, seen as a state of the simple $\mathcal{L}_n$-TS. To be able to reason only about clean nets, we actually need to show that the result remains true when we consider $\lfloor U \rfloor$ as a state of the clean $\mathcal{L}_n$-TS (Prop. 2.16 below). In order to obtain this result, we will need the following standard up to technique [15]. It basically states that we can use expansion to rewrite the left-hand-side process, when playing the games required by an expansion relation:

Technique 2.15 (Expansion up to Expansion)**.** Let $\mathcal{R}$ be a relation such that for any $\alpha \in \mathcal{L}$, $\overset{\alpha}{\leftarrow} \mathcal{R} \subseteq \overset{\widehat{\alpha}}{\leftarrow} \succsim \mathcal{R}$, and $\mathcal{R} \overset{\alpha}{\rightarrow} \subseteq \overset{\widehat{\alpha}}{\Rightarrow} \succsim \mathcal{R}$.
Then $\succsim \mathcal{R}$ is an expansion relation, and $\mathcal{R} \subseteq \succsim$.

Proposition 2.16. *For any simple net U, we have*

$$\langle U, \rightarrow_s \rangle \succsim \left\langle \lfloor U \rfloor, \rightarrow_{\lfloor s \rfloor} \right\rangle \ .$$

Proof. Since for any simple net U, $\langle U, \rightarrow_{\mathrm{s}} \rangle \succsim \langle \lfloor U \rfloor, \rightarrow_{\mathrm{s}} \rangle$, and $\lfloor U \rfloor$ is a clean net, it suffices to show that $\langle U, \rightarrow_{\mathrm{s}} \rangle \succsim \left\langle U, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle$ for any clean net U.

We apply Technique 2.15 to the following relation:

$$\mathcal{R} \triangleq \left\{ \left\langle \langle U, \rightarrow_{\mathrm{s}} \rangle , \ \left\langle U, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \right\rangle \mid U \in \lfloor \mathcal{U}_{\mathrm{s}} \rfloor \right\} \ .$$

- If $\langle U, \rightarrow_{\mathrm{s}} \rangle \xrightarrow{\mu} \langle U', \rightarrow_{\mathrm{s}} \rangle$, then $\left\langle U, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \xrightarrow{\mu} \left\langle \lfloor U' \rfloor, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle$ and we check that $\langle U', \rightarrow_{\mathrm{s}} \rangle \gtrsim \langle \lfloor U' \rfloor, \rightarrow_{\mathrm{s}} \rangle \, \mathcal{R} \left\langle \lfloor U' \rfloor, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle$.
- If $\left\langle U, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \xrightarrow{\mu} \left\langle U', \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle$, then $\langle U, \rightarrow_{\mathrm{s}} \rangle \xrightarrow{\mu} \langle V, \rightarrow_{\mathrm{s}} \rangle$ with $\lfloor V \rfloor = U'$ and we check that $\langle V, \rightarrow_{\mathrm{s}} \rangle \gtrsim \langle U', \rightarrow_{\mathrm{s}} \rangle \, \mathcal{R} \left\langle U', \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle$. ■

Now that we have a clean $\mathcal{L}_n$-TS where any pending message has a destination, we can examine the correctness of the routing of pending messages. We first show that internal transitions are deterministic:

Lemma 2.17. *The relation $\xrightarrow{\tau}_{\lfloor s \rfloor}$ is terminating and confluent.*

Proof. For any pending message $h\langle m \rangle$ of a clean net U, define its weight as the natural number $l+1$, where l is given by Def. 2.10 (since U is clean, h has a destination in U); then let the size of U be the sum of the weights of all its pending messages. This size strictly decreases along silent transitions, $\xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor}$.

Thanks to the well-formedness condition, $\xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor}$ has no critical pair, so that it is confluent. ■

Notice on the contrary that $\xrightarrow{\tau}_{\mathrm{s}}$ does not terminate, due to the potential presence of pending messages in forwarder cycles. We denote by $U_{\downarrow}$ the normal form w.r.t. $\xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor}$ of any clean net U. Remark that for any simple net U, $U = \lfloor U \rfloor_{\downarrow}$ if and only if U is actually a reference net.

The following lemma states the correctness of the routing algorithm: given a reference net U, if the routing of a message by the reference $\mathcal{L}_n$-TS leads to a net V, then the clean $\mathcal{L}_n$-TS can do so, by performing some silent transitions. Conversely, if a pending message is added to U so that we obtain a clean net V, the normalisation of V yields exactly the same simple net as the one we obtain with the reference $\mathcal{L}_n$-TS.

Lemma 2.18. *Let U be a reference net.*

(1) If $U \xrightarrow{h\langle m \rangle}_r V$, then $U \xrightarrow{h\langle m \rangle}_{\lfloor s \rfloor} \overset{\widehat{\tau}}{\Rightarrow}_{\lfloor s \rfloor} V$.
(2) If $U \xrightarrow{h\langle m \rangle}_{\lfloor s \rfloor} V$, then $U \xrightarrow{h\langle m \rangle}_r V_{\downarrow}$.

Proof. (1) If h is lost in U, then by Lemma 2.11(3) $U = V$, and we have $U \xrightarrow{h\langle m \rangle}_{\lfloor \mathrm{s} \rfloor} \lfloor h\langle m \rangle \mid U \rfloor = \lfloor U \rfloor = U$. Otherwise, h has some destination in U, so that

$$\begin{aligned} U &= U' \mid \Pi_{i<l} \; h_i \triangleright h_{i+1} \mid h_l[n] && \text{(with } h = h_0\text{)} \\ V &= U' \mid \Pi_{i<l} \; h_i \triangleright h_{i+1} \mid h_l[m;n], && \text{(Lemma 2.11(4))} \end{aligned}$$

and we check that $U \xrightarrow{h\langle m\rangle}_{\lfloor \mathrm{s} \rfloor} h\langle m\rangle \mid U \stackrel{\widehat{\tau}}{\Rightarrow}_{\lfloor \mathrm{s} \rfloor} V$, by an induction on l.

(2) If h is lost in U, then $V = \lfloor h\langle m\rangle \mid U \rfloor = U$, and $U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} U$ by Lemma 2.11(3). Otherwise,

$$U = U' \mid \Pi_{i<l}\ h_i \triangleright h_{i+1} \mid h_l[n] \qquad \text{(with } h = h_0\text{)}$$
$$U \xrightarrow{h\langle m\rangle}_{\lfloor \mathrm{s} \rfloor} V = h\langle m\rangle \mid U \ \stackrel{\widehat{\tau}}{\Rightarrow}_{\lfloor \mathrm{s} \rfloor}\ U' \mid \Pi_{i<l}\ h_i \triangleright h_{i+1} \mid h_l[m;n] = V'.$$

where U' and thus V' are necessarily reference nets. By Lemma 2.11(4), $U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} V'$, and since V' is a normal form of V, $V_{\downarrow} = V'$. ■

We now have to prove that silent transitions do not change the behaviour of the system, i.e., that silent transitions are contained in bisimilarity. We actually show a stronger result: silent transitions are contained in expansion; as explained below, this allows us to use the following up-to technique, so that the proof remains tractable. Technique 2.19 below actually defines two up-to techniques, the second will be used in Sect. 3.

Technique 2.19 (Expansion up to Transitivity)**.**
Let $\mathcal{R}$ be a relation such that for any $\alpha \in \mathcal{L}$, $\stackrel{\alpha}{\leftarrow}\ \mathcal{R} \subseteq \stackrel{\widehat{\alpha}}{\leftarrow}\ \mathcal{R}^\star$.
If $\mathcal{R}^{-1}$ is a simulation or $\mathcal{R}$ is symmetric, then $\mathcal{R} \subseteq \succsim$.

Proof. The first hypothesis allows us to show that $\mathcal{R}^\star$ is a pre-expansion relation. If $\mathcal{R}^{-1}$ is a simulation, then $(\mathcal{R}^\star)^{-1} = (\mathcal{R}^{-1})^\star$ is a simulation. If $\mathcal{R}$ is symmetric, then we have $(\mathcal{R}^\star)^{-1} = \mathcal{R}^\star$ which is a pre-expansion relation, and hence a simulation. Therefore, $\mathcal{R}^\star$ is an expansion relation, and $\mathcal{R} \subseteq \mathcal{R}^\star \subseteq \succsim$. ■

Lemma 2.20. *The relation $\xrightarrow{\tau}_{\lfloor s \rfloor}$ is contained in expansion.*

Proof. We apply Technique 2.19 to $\mathcal{R} = \xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor} \cup \mathcal{R}'$, where

$$\mathcal{R}' = \{\langle h\langle m\rangle \mid h\langle n\rangle \mid U,\ h\langle m;n\rangle \mid U\rangle \mid U \in \lfloor \mathcal{U}_{\mathrm{s}} \rfloor,\ h \text{ not lost in } U\}$$

The two cases where we need transitivity are the following visible and silent left-to-right challenges:

- $U = h\langle m\rangle \mid h[n] \mid U_0 \quad \xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor} \quad h[m;n] \mid U_0 = V$ by rule [$\textsc{Dst}_{\mathrm{s}}$], and $U \xrightarrow{h \triangleright k}_{\lfloor \mathrm{s} \rfloor} U' = h\langle m\rangle \mid h \triangleright k \mid k\langle n\rangle \mid U_0$ by rule [$\textsc{Mig}_{\mathrm{s}}$]. We have:

$$\begin{array}{lll} V \xrightarrow{h \triangleright k}_{\lfloor \mathrm{s} \rfloor} V' = h \triangleright k \mid k\langle m;n\rangle \mid U_0 & & [\textsc{Mig}_{\mathrm{s}}] \\ U' \quad \mathcal{R} \quad h \triangleright k \mid k\langle m\rangle \mid k\langle n\rangle \mid U_0 \quad \mathcal{R} \quad V' \ ; & & [\textsc{Fwd}_{\mathrm{s}}],(\mathcal{R}') \end{array}$$

- $U = h\langle m\rangle \mid h\langle n\rangle \mid h \triangleright k \mid U_0 \quad \mathcal{R}' \quad h\langle m; n\rangle \mid h \triangleright k \mid U_0 = V$ and
  $U \xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor} U' = h\langle m\rangle \mid h \triangleright k \mid k\langle n\rangle \mid U_0$ by rule [FWD$_\mathrm{S}$]. In this case we have:

$$V \xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor} V' = h \triangleright k \mid k\langle m; n\rangle \mid U_0 \qquad [\mathrm{FWD_S}]$$
$$U' \quad \mathcal{R} \quad h \triangleright k \mid k\langle m\rangle \mid k\langle n\rangle \mid U_0 \quad \mathcal{R} \quad V' \ . \qquad [\mathrm{FWD_S}],(\mathcal{R}')$$

■

The smallest bisimulation relation containing $\xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor}$ contains at least $\overset{\widehat{\tau}}{\Rightarrow}_{\lfloor \mathrm{s} \rfloor}$. Hence, proving the weaker result $\xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor} \subseteq \approx$ without using this expansion-based technique would require to check that some relation containing $\overset{\widehat{\tau}}{\Rightarrow}_{\lfloor \mathrm{s} \rfloor}$ is a bisimulation ("bisimilarity up to transitivity" is not a valid technique [13]). This could be tedious: while U and V differ only slightly when $U \xrightarrow{\tau}_{\lfloor \mathrm{s} \rfloor} V$, this is no longer the case when $U \overset{\widehat{\tau}}{\Rightarrow}_{\lfloor \mathrm{s} \rfloor} V$.

Corollary 2.21. *For any clean net U we have:*

$$\left\langle U, \rightarrow_{\lfloor s \rfloor} \right\rangle \succsim \left\langle U_{\downarrow}, \rightarrow_{\lfloor s \rfloor} \right\rangle \ .$$

We now have enough technical devices to give the correctness proof of the implementation w.r.t. the specification. Thanks to the "expansion up to expansion" technique (Technique 2.15), we can work with a very simple candidate relation: syntactical identity between reference nets, equipped with clean simple transitions on one side, and with reference transitions on the other side:

Lemma 2.22. *For any reference net U, we have:*

$$\left\langle U, \rightarrow_{\lfloor s \rfloor} \right\rangle \succsim \langle U, \rightarrow_r \rangle$$

Proof. We apply Technique 2.15 to the following relation:

$$\mathcal{R} = \left\{ \left\langle \left\langle U, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle, \ \langle U, \rightarrow_{\mathrm{r}} \rangle \right\rangle \mid U \in \mathcal{U}_{\mathrm{r}} \right\} \ .$$

We consider the different challenges along a label μ:

- μ cannot be τ, since reference nets are in normal form;
- if $\mu = \nu h$ or $\mu = h(m)$, we have $U \xrightarrow{\mu}_{\lfloor \mathrm{s} \rfloor} V$ iff $U \xrightarrow{\mu}_{\mathrm{r}} V$;
- if $\mu = h\langle m\rangle$, we apply Lemma 2.18:
  - if $U \xrightarrow{h\langle m\rangle}_{\lfloor \mathrm{s} \rfloor} V$, then we have $U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} V_{\downarrow}$ and by Cor. 2.21, we obtain $\left\langle V, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \succsim \left\langle V_{\downarrow}, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \mathcal{R} \ \langle V_{\downarrow}, \rightarrow_{\mathrm{r}} \rangle$ ($V_{\downarrow}$ is a reference net);
  - if $U \xrightarrow{h\langle m\rangle}_{\mathrm{r}} V$, then we have $U \overset{\widehat{h\langle m\rangle}}{\Longrightarrow}_{\lfloor \mathrm{s} \rfloor} V$ and $\left\langle V, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \mathcal{R} \ \langle V, \rightarrow_{\mathrm{r}} \rangle$.
- The case $\mu = h \triangleright k$ relies on the previous one, and is very similar. ■

Theorem 2.23. *For any simple net U, we have:*

$$\langle U, \rightarrow_s \rangle \succsim \left\langle \lfloor U \rfloor_\downarrow, \rightarrow_r \right\rangle \ .$$

Proof. By Prop. 2.16, Cor. 2.21 and Lemma 2.22, we have:

$$\langle U, \rightarrow_{\mathrm{s}} \rangle \succsim \left\langle \lfloor U \rfloor, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \succsim \left\langle \lfloor U \rfloor_\downarrow, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \succsim \left\langle \lfloor U \rfloor_\downarrow, \rightarrow_{\mathrm{r}} \right\rangle$$

(recall that $\lfloor U \rfloor_\downarrow$ is a reference net). ■

3 Validating an Optimisation

The forwarder chains that are generated along the evolution of a net are the source of inefficiencies. For example, the message m in Fig. 6 will have to go trough three locations before reaching its final destination. In this section, we define an optimised $\mathcal{L}_n$-TS, that contracts such forwarder chains, and we prove the correctness of this optimisation by showing that simple nets are bisimilar to optimised nets.

3.1 Optimised Nets

The syntax of *optimised nets* is given in Fig. 3; it extends the syntax of simple nets by

- annotating pending messages with a list of locations: $h\langle m \rangle_{\widetilde{k}}$;
- introducing *blocked forwarders*: $h\not\triangleright$;
- adding a second kind of messages: *relocation messages* $h\langle \triangleright k \rangle$.

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.

Definition 3.1 (Well-formedness for Optimised Nets)**.** An optimised net U is *well-formed* if:

(1) any location $h \in \mathcal{H}$ appears at most once as a real location ($h[m]$), as the source of a forwarder ($h \triangleright k$) in U, or as the source of a blocked forwarder ($h\not\triangleright$);

$$[\textsc{Rcv}_{\mathrm{o}}]\ \frac{}{h[m;n] \mid U \xrightarrow{h(m)}_{\mathrm{o}} h[n] \mid U} \qquad [\textsc{Snd}_{\mathrm{o}}]\ \frac{}{U \xrightarrow{h\langle m\rangle}_{\mathrm{o}} h\langle m\rangle_{\emptyset} \mid U}$$

$$[\textsc{Mig}_{\mathrm{o}}]\ \frac{}{h[m] \mid U \xrightarrow{h \triangleright k}_{\mathrm{o}} h \triangleright k \mid k\langle m\rangle_{\emptyset} \mid U} \qquad [\textsc{New}_{\mathrm{o}}]\ \frac{h \notin l(U)}{U \xrightarrow{\nu h}_{\mathrm{o}} h[\emptyset] \mid U}$$

$$[\textsc{Dst}_{\mathrm{o}}]\ \frac{}{h\langle m\rangle_{\widetilde{k}} \mid h[n] \mid U \xrightarrow{\tau}_{\mathrm{o}} h[m;n] \mid \widetilde{k}\langle \triangleright h\rangle \mid U}$$

$$[\textsc{Fwd}_{\mathrm{o}}]\ \frac{}{h\langle m\rangle_{\widetilde{k}} \mid h \triangleright k \mid U \xrightarrow{\tau}_{\mathrm{o}} h\not\triangleright \mid k\langle m\rangle_{h;\widetilde{k}} \mid U}$$

$$[\textsc{Upd}_{\mathrm{o}}]\ \frac{}{h\langle \triangleright k\rangle \mid h\not\triangleright \mid U \xrightarrow{\tau}_{\mathrm{o}} h \triangleright k \mid U}$$

Fig. 7. Transitions of optimised nets.

(2) for any blocked forwarder $h\not\triangleright$, h appears exactly once in the annotation of a pending message ($k\langle m\rangle_{\widetilde{h}}$, with $h \in \widetilde{h}$), or as the destination of a relocation message ($h\langle \triangleright k\rangle$);
(3) any location registered in a pending message or appearing as the target of a relocation message hosts a blocked forwarder.

According to (1), our definition of *defined locations* ($l(U)$) is extended by taking blocked forwarders into account.

Definition 3.2 (Optimised $\mathcal{L}_n$-TS)**.** The *optimised $\mathcal{L}_n$-TS* is $\langle \mathcal{O}, \rightarrow_{\mathrm{o}} \rangle$, where $\mathcal{O}$ is the set of well-formed optimised nets, and $\rightarrow_{\mathrm{o}}$ is defined in Fig. 7.

Proposition 3.3. *Any simple well-formed net in the sense of Def. 2.4 is well-formed in the sense of Def. 3.1. Well-formed optimised nets are preserved by $\rightarrow_o$, so that $\langle \mathcal{O}, \rightarrow_o \rangle$ is an $\mathcal{L}_n$-TS.*

This allows us to consider only well-formed nets in the sequel.

We describe the rules below; in comparison to the simple $\mathcal{L}_n$-TS, "visible" rules [Rcv], [Snd], [Mig] and [New] are left unchanged, the two "silent" rules [$\textsc{Fwd}_{\mathrm{s}}$] and [$\textsc{Dst}_{\mathrm{s}}$] are updated and the rule [$\textsc{Upd}_{\mathrm{o}}$] is new. In rule [$\textsc{Dst}_{\mathrm{o}}$], we use $\widetilde{h}\langle \triangleright k\rangle$ to denote $\Pi_{h\in\widetilde{h}}\ h\langle \triangleright k\rangle$. In the sequel, we shall furthermore use $\widetilde{h} \triangleright k$ and $\widetilde{h}\not\triangleright$ to respectively denote $\Pi_{h\in\widetilde{h}}\ h \triangleright k$ and $\Pi_{h\in\widetilde{h}}\ h\not\triangleright$.

When a forwarder transmits a message (rule [$\textsc{Fwd}_{\mathrm{o}}$]), it registers its location 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 [$\textsc{Dst}_{\mathrm{o}}$]). The blocked forwarders located at these locations will then update their destina-

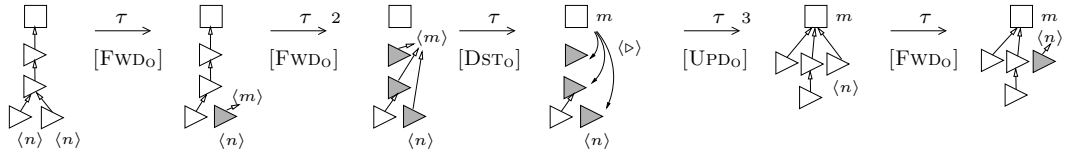

Fig. 8. Optimised Forwarder Behaviour

tion accordingly (rule $[\text{UPD}_\text{O}]$). This behaviour is illustrated in Fig. 8, where 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.

Remark 3.4 (Further possible optimisations)**.** We could go beyond the optimisation presented here: for example, pending messages waiting behind a blocked forwarder could be packed together so that they can be sent at once, when the blocked forwarder gets relocated:

$$[\text{PCK}_\text{O}]\ \frac{}{h\langle m\rangle_{\widetilde{h}} \mid h\langle n\rangle_{\widetilde{k}} \mid h\not\triangleright \mid U \xrightarrow{\tau}_\text{o} h\langle m;n\rangle_{\widetilde{h};\widetilde{k}} \mid h\not\triangleright \mid U}$$

We could also decompose rule $[\text{DST}_\text{O}]$, in order to make the broadcast of the relocation message explicit and asynchronous:

$$[\text{DST}_\text{O}]\ \frac{}{h\langle m\rangle_{k;\widetilde{k}} \mid h[n] \mid U \xrightarrow{\tau}_\text{o} h\langle m\rangle_{\widetilde{k}} \mid h[n] \mid k\langle\triangleright h\rangle \mid U}$$

$$[\text{DST'}_\text{O}]\ \frac{}{h\langle m\rangle_{\emptyset} \mid h[n] \mid U \xrightarrow{\tau}_\text{o} h[m;n] \mid U}$$

However, rather than finding the best implementation of the specification, the aim of this paper is to give a methodology for the analysis of such distributed systems. Therefore, we prefer sticking to our rather simple optimisation, so that hopefully our methodology does not get lost into technical details.

We now prove the correctness of the optimisation. As explained in the introduction, this requires us to handle forwarder cycles (Sect. 3.2), and to validate the algorithm for routing pending messages (Sect. 3.3).

3.2 Handling Forwarder Cycles

We start by extending Def. 2.10 to optimised nets:

Definition 3.5 (Destination, Lost Locations, Clean Nets)**.** We say that h *has destination* k in an optimised net U if there exist l, m, V and $(h_i)_{i\leq l}$ with $h = h_0$, $k = h_l$ s.t. $U = \Pi_{i<n}\ F_i \mid h_l[m] \mid V$, where for all $i < l$, F_i is either:

- a forwarder: $h_i \triangleright h_{i+1}$, or
- a blocked forwarder, together with a relocation message: $h_i \not\triangleright \mid h_i\langle \triangleright h_{i+1}\rangle$, or
- a blocked forwarder whose location is registered in a message blocking some other forwarders: $h_i \not\triangleright \mid \widetilde{k} \not\triangleright \mid h_{i+1}\langle n\rangle_{h_i;\widetilde{k}}$.

We say that h *is lost in* U if h has no destination in U.
An optimised net U is called *clean* if all its pending messages are sent on locations that have a destination in U. We denote by $\lfloor \mathcal{O} \rfloor$ the set of clean optimised nets.

Again, well-formedness ensures that each location has at most one destination in a given net. We have the following properties:

Lemma 3.6. *(1) If h is lost in U, and $U \xrightarrow{\mu}_o U'$, then h is lost in U'.*
(2) In each of the following nets, h is lost if and only if k is lost:

$$(a)\ h \triangleright k \mid U\ , \qquad (b)\ h \not\triangleright \mid h\langle \triangleright k\rangle \mid U\ , \qquad (c)\ h\langle m\rangle_{k,\widetilde{k}} \mid U\ .$$

We can no longer directly remove lost messages: we have to take care of the forwarders that are blocked by these messages. We introduce for that the following relation:

Definition 3.7. We call *cleaning relation* the following relation over $\mathcal{O}$:

$$\mathcal{E} \triangleq \left\{ \left\langle U \mid \widetilde{h} \not\triangleright \mid h\langle m\rangle_{\widetilde{h}}\ ,\ U \mid \widetilde{h} \triangleright h \right\rangle \;\middle|\; h \text{ lost in } \left(U \mid \widetilde{h} \not\triangleright \mid h\langle m\rangle_{\widetilde{h}}\right) \right\}\ .$$

Notice that V is well-formed whenever $U\ \mathcal{E}\ V$ and U is well-formed, and that $\mathcal{E}$ preserves destinations and lost locations. We show that this relation is contained in expansion by using the second version of the "expansion up to expansion" technique (Technique 2.19)

Lemma 3.8. *The cleaning relation is contained in expansion.*

Proof. Let $\mathcal{E}'$ be the symmetric closure of the following relation:

$$\left\{ \left\langle U,\ U_0 \mid \Pi_{h\in\widetilde{h}} h \triangleright k_h \right\rangle \;\middle|\; U = U_0 \mid \widetilde{h} \not\triangleright \mid h\langle m\rangle_{\widetilde{h}} \text{ and } h, (k_h)_{h\in\widetilde{h}} \text{ are lost in } U \right\}$$

$\mathcal{E}$ is clearly contained in $\mathcal{E}'$; and $\mathcal{E}'$ preserves well-formed nets, destinations and lost locations. We show that $\mathcal{E}'$ is a candidate to Technique 2.19. Since $\mathcal{E}'$ is symmetric, it suffices to show that this is a "pre-expansion up to transitivity". The well-formedness condition allows us to rule out a lot of cases, the remaining interesting cases are the following:

- $U = U_0 \mid \widetilde{h} \not\triangleright \mid h\langle m\rangle_{\widetilde{h}} \mid h \triangleright k\ \ \mathcal{E}'\ \ V = U_0 \mid \Pi_{h\in\widetilde{h}} h \triangleright k_h \mid h \triangleright k$ and $U \xrightarrow{\tau}_{\mathrm{o}} U' = U_0 \mid (h;\widetilde{h}) \not\triangleright \mid k\langle m\rangle_{h;\widetilde{h}}$ by rule [$\textsc{Fwd}_{\mathrm{o}}$]. We just check that

$U' \; \mathcal{E}' \; V$: the right-hand-side process does not move.

- $U = U_0 \mid \widetilde{k}\not\triangleright \mid k\langle n\rangle_{\widetilde{k}} \mid (k;\widetilde{h})\not\triangleright \mid h\langle m\rangle_{k;\widetilde{h}}$
  $\mathcal{E}' \quad V = U_0 \mid \widetilde{k}\not\triangleright \mid k\langle n\rangle_{\widetilde{k}} \mid k \triangleright k' \mid \Pi_{h\in\widetilde{h}} h \triangleright k_h$
  and $V \xrightarrow{\tau}_{\mathrm{o}} V' = U_0 \mid (k;\widetilde{k})\not\triangleright \mid k'\langle n\rangle_{k;\widetilde{k}} \mid \Pi_{h\in\widetilde{h}} h \triangleright k_h$ by rule [FWD$_{\mathrm{O}}$]. In this case, we have $V \; \mathcal{E}'^2 \; V'$:

$$V \quad \mathcal{E} \quad U_0 \mid (k;\widetilde{k}) \triangleright k' \mid \Pi_{h\in\widetilde{h}} h \triangleright k_h \quad \mathcal{E}' \quad V' \; .$$

  Hence the left-hand-side process (U) does not move, and we apply $\mathcal{E}'$ three times in order to relate U and V': $U \; \mathcal{E}' \; V \; \mathcal{E}'^2 \; V'$. ■

The cleaning relation is confluent and terminating; we denote by $\lfloor U \rfloor$ the normal form of an optimised net U w.r.t. $\mathcal{E}$ (on simple nets, this function coincides with that defined in Sect. 2.4). We have:

Corollary 3.9. *For any optimised net U, we have:*

$$\langle U, \rightarrow_o \rangle \gtrsim \langle \lfloor U \rfloor, \rightarrow_o \rangle \quad .$$

Like in Sect. 2.3, we use this "cleaning function", to define a optimised $\mathcal{L}_n$-TS over clean optimised nets ($\lfloor \mathcal{O} \rfloor$), which is equivalent to the initial one:

Definition 3.10 (Clean Optimised $\mathcal{L}_n$-TS)**.** The *clean optimised $\mathcal{L}_n$-TS* is $\left\langle \lfloor \mathcal{O} \rfloor, \rightarrow_{\lfloor \mathrm{o} \rfloor} \right\rangle$, where $\rightarrow_{\lfloor \mathrm{o} \rfloor}$ is defined by the following rule:

$$\frac{U \xrightarrow{\mu}_{\mathrm{o}} U'}{U \xrightarrow{\mu}_{\lfloor \mathrm{o} \rfloor} \lfloor U' \rfloor}$$

Proposition 3.11. *For any optimised net U, we have:*

$$\langle U, \rightarrow_o \rangle \gtrsim \left\langle \lfloor U \rfloor, \rightarrow_{\lfloor o \rfloor} \right\rangle \quad .$$

Proof. Identical to the proof of Prop. 2.16. ■

Like in Sect. 2.4, the cleaning function is actually only used upon migration and emission of messages:

Lemma 3.12. *For any $U \in \lfloor \mathcal{O} \rfloor$, and $\mu \notin \{h \triangleright k; h\langle m \rangle\}$, if $U \xrightarrow{\mu}_o U'$ then U' is clean: $U \xrightarrow{\mu}_{\lfloor o \rfloor} U' = \lfloor U' \rfloor$.*

We now can work with clean optimised nets, and validate the routing of pending messages without bothering with lost messages.

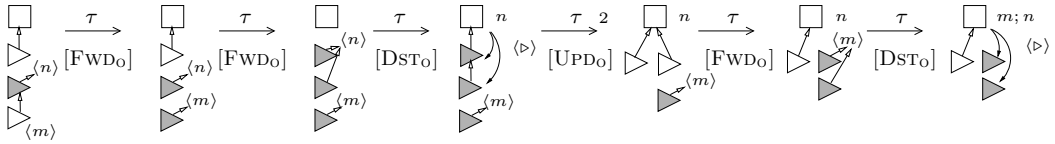

Fig. 9. Unblocking a forwarder in order to route a given message.

3.3 Validating the Routing Algorithm

Like in Sect. 2.4, the smallest bisimulation relation containing $\xrightarrow{\tau}_{\lfloor o \rfloor}$ contains at least $\widehat{\Rightarrow}_{\lfloor o \rfloor}$, so that we need some bisimulation proof technique in order to be able to work with small and local candidate relations.

However, unlike in the previous section, we cannot rely on expansion-based up-to techniques: silent transitions are not contained in expansion. This comes from the race conditions introduced by the blocking behaviour of forwarders: for example, in Fig. 8, the message n has to wait for the 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.

We will actually use the following technique, from [11], that relies on a termination argument in order to allow the "up to transitivity" technique:

Technique 3.13 (Simulation up to Transitivity, Constrained)**.**
Let $\mathcal{R}$ be a relation such that $\forall \alpha \in \mathcal{L}, \overset{\alpha}{\leftarrow} \mathcal{R} \subseteq \mathcal{R}^\star \overset{\widehat{\alpha}}{\Leftarrow}$.
If $\mathcal{R}^+ \overset{\tau}{\Rightarrow}$ terminates, then $\mathcal{R}^\star$ is a simulation.

Since we work with clean nets, any pending message has a destination; we first prove a lemma that allows us to route these messages to their destination. While this was almost trivial in the simple $\mathcal{L}_n$-TS (Lemma 2.18(1)), here, as depicted on Fig. 9, in order to route a given message m to its destination, we first have to route any message that blocks some forwarder between the message and its destination. As a side effect, this involves some reconfiguration of the forwarders tree.

Lemma 3.14. *Let $U = V \mid h_0\langle m_0 \rangle_{\widetilde{h}}$ be a (clean) optimised net. By Def. 3.5, $U = V' \mid h_0\langle m_0 \rangle_{\widetilde{h}} \mid \Pi_{i<l} F_i \mid h_n[n]$, and we have:*

$$U \overset{\tau}{\Rightarrow}_{\lfloor o \rfloor} \; V' \mid \widetilde{h}\langle \triangleright h_l \rangle \mid \Pi_{i<l} h_i \triangleright h_l \mid h_l[m_0; m; n] \mid \widetilde{k} \triangleright h_l$$

where n and $\widetilde{k}$ are the messages and forwarder locations collected in $\Pi_{i<l} F_i$.

Proof. We proceed by induction over l: if $l = 0$ we simply apply rule [DST$_\text{O}$], otherwise we reason by case analysis on the shape of F_0:

- a simple forwarder: $h_0 \triangleright h_1$: we transmit the message with rule [FWD$_{\mathrm{O}}$], apply the induction hypothesis (IH), and relocate the forwarder using rule [UPD$_{\mathrm{O}}$]:

$$\begin{aligned}
U &\xrightarrow{\tau}_{\lfloor \mathrm{o} \rfloor} V' \mid h_0 \not\triangleright \mid h_1\langle m_0 \rangle_{h_0,\widetilde{h}} \mid \Pi_{0<i<l} F_i \mid h_l[n] && [\mathrm{FWD_O}] \\
&\overset{\tau}{\Rightarrow}_{\lfloor \mathrm{o} \rfloor} V' \mid h_0 \not\triangleright \mid (h_0, \widetilde{h})\langle \triangleright h_l \rangle \mid \Pi_{0<i<l} h_i \triangleright h_l \mid h_l[m_0; m; n] \mid \widetilde{k} \triangleright h_l && (\mathrm{IH}) \\
&\xrightarrow{\tau}_{\lfloor \mathrm{o} \rfloor} V' \mid \widetilde{h} \triangleright h_l \mid h_0 \triangleright h_l \mid \Pi_{i<l} h_i \triangleright h_l \mid h_l[m_0; m; n] \mid \widetilde{k} \triangleright h_l && [\mathrm{UPD_O}]
\end{aligned}$$

- a blocked forwarder with a relocation message: $h_0 \not\triangleright \mid h_0\langle \triangleright h_1 \rangle$: we relocate the forwarder with rule [UPD$_{\mathrm{O}}$] and fall back into the previous case.
- $(h_0; \widetilde{k}_1) \not\triangleright \mid h_1\langle m_1 \rangle_{h_0,\widetilde{k}_1}$: we apply the induction hypothesis to the message at h_1, and transmit the initial message trough the relocated forwarder:

$$\begin{aligned}
U = V' &\mid h_0\langle m_0 \rangle_{\widetilde{h}} \mid (h_0; \widetilde{k}_1) \not\triangleright \mid h_1\langle m_1 \rangle_{(h_0;\widetilde{k}_1)} \mid \Pi_{0<i<l} F_i \mid h_l[n] \\
&\overset{\tau}{\Rightarrow}_{\lfloor \mathrm{o} \rfloor} V' \mid h_0\langle m_0 \rangle_{\widetilde{h}} \mid (h_0; \widetilde{k}_1) \not\triangleright \mid (h_0; \widetilde{k}_1)\langle \triangleright h_l \rangle \mid \Pi_{0<i<l} h_i \triangleright h_l \\
&\qquad \mid h_l[m_1; m; n] \mid \widetilde{k} \triangleright h_l && (\mathrm{IH}) \\
&\overset{\tau}{\Rightarrow}_{\lfloor \mathrm{o} \rfloor} V' \mid h_0\langle m_0 \rangle_{\widetilde{h}} \mid (h_0; \widetilde{k}_1) \triangleright h_l \mid \Pi_{0<i<l} h_i \triangleright h_l \mid h_l[m_1; m; n] \mid \widetilde{k} \triangleright h_l \\
& && [\mathrm{UPD_O}] \\
&\overset{\tau}{\rightarrow}{}^{3}_{\lfloor \mathrm{o} \rfloor} V' \mid \widetilde{h}\langle \triangleright h_l \rangle \mid \Pi_{i<l} h_i \triangleright h_l \mid h_l[m_0; m_1; m; n] \mid (h_0; \widetilde{k}_1; \widetilde{k}) \triangleright h_l \\
& && [\mathrm{FWD_O}, \mathrm{DST_O}, \mathrm{UPD_O}]
\end{aligned}$$

■

Despite these modifications of forwarder trees, the routing of messages is deterministic (we shall prove that it terminates in the sequel). Notice that this is not true in presence of pending messages whose location is in a forwarder cycle: in that case the way the cycle gets blocked may depend on the order of the routing steps.

Lemma 3.15. $\xrightarrow{\tau}_{\lfloor o \rfloor}$ *is locally confluent.*

Proof. By well-formedness, the only critical pair is the following:

$$\begin{aligned}
&U = V \mid h\langle m \rangle_{\widetilde{h}} \mid h\langle n \rangle_{\widetilde{k}} \mid h \triangleright k \\
&U \xrightarrow{\tau}_{\lfloor \mathrm{o} \rfloor} V \mid h\langle m \rangle_{\widetilde{h}} \mid h \not\triangleright \mid k\langle n \rangle_{h,\widetilde{k}} = U_1 && [\mathrm{FWD_O}] \\
&U \xrightarrow{\tau}_{\lfloor \mathrm{o} \rfloor} V \mid h\langle n \rangle_{\widetilde{k}} \mid h \not\triangleright \mid k\langle m \rangle_{h,\widetilde{h}} = U_2 && [\mathrm{FWD_O}]
\end{aligned}$$

By using Lemma 3.14 on U_1, we can route the message n to some location k':

$$\begin{aligned}
U_1 &\overset{\widehat{\tau}}{\Rightarrow}_{\lfloor \mathrm{o} \rfloor} V' \mid h\langle m \rangle_{\widetilde{h}} \mid h \not\triangleright \mid (h, \widetilde{k})\langle \triangleright k' \rangle \mid k'[n; m'] && (\text{Lemma 3.14}) \\
&\xrightarrow{\tau}_{\lfloor \mathrm{o} \rfloor} V' \mid h\langle m \rangle_{\widetilde{h}} \mid h \triangleright k' \mid \widetilde{k}\langle \triangleright k' \rangle \mid k'[n; m'] && [\mathrm{UPD_O}] \\
&\xrightarrow{\tau}{}^{3}_{\lfloor \mathrm{o} \rfloor} V' \mid (\widetilde{k}, \widetilde{h})\langle \triangleright k' \rangle \mid h \triangleright k' \mid k'[m; n; m'] = U' && [\mathrm{FWD_O}, \mathrm{DST_O}, \mathrm{UPD_O}]
\end{aligned}$$

The same reasoning about U_2 leads to $U_2 \overset{\tau}{\Rightarrow}_{\lfloor \mathrm{o} \rfloor} U'$. ■

However, due to forwarders relocations, $\xrightarrow{\tau}_{\lfloor o \rfloor}$ does not commute with visible actions. To handle this, we introduce the following relation, that allows one to reorganise step by step the forwarder structure of a net:

Definition 3.16. We denote by $\mathcal{S}$ the *swapping relation*, defined as the symmetric closure of the following relation over $\lfloor \mathcal{O} \rfloor$:

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

Notice that $\mathcal{S}$ preserves destinations and lost locations. Our goal is to prove that $(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})$ is a candidate relation for Technique 3.13. This will allow us to prove that both $\mathcal{S}$ and $\xrightarrow{\tau}_{\lfloor o \rfloor}$ are contained in bisimilarity. The two following lemmas establish progressively that $(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})$ is a "simulation up to transitivity".

Lemma 3.17. *If $U \xrightarrow{\tau}_{\lfloor o \rfloor} V$ and $U \xrightarrow{\mu}_{\lfloor o \rfloor} U'$, then $U' \mathrel{\widehat{\xrightarrow{\tau}}}_{\lfloor o \rfloor} \mathcal{S}^\star \mathrel{\widehat{\xleftarrow{\tau}}}_{\lfloor o \rfloor} \mathrel{\widehat{\xleftarrow{\mu}}}_{\lfloor o \rfloor} V$.*

Proof. The case $\mu = \tau$ is given by the local confluence of $\xrightarrow{\tau}_{\lfloor o \rfloor}$ (Lemma 3.15); otherwise, it holds that $U' \xrightarrow{\tau}_{\lfloor o \rfloor} \xleftarrow{\mu}_{\lfloor o \rfloor} V$, except in the following case:

$$U = W \mid h\langle m \rangle_{\widetilde{h}} \mid h[n] \quad \xrightarrow{\tau}_{\lfloor o \rfloor} \quad W \mid h[m;n] \mid \widetilde{h}\langle \triangleright h \rangle = V \qquad [\textsc{Dst}_o]$$
$$U \xrightarrow{h \triangleright k}_{\lfloor o \rfloor} W \mid h\langle m \rangle_{\widetilde{h}} \mid h \triangleright k \mid k\langle n \rangle = U' \qquad [\textsc{Mig}_o]$$

where we have

$$V \xrightarrow{h \triangleright k}_{\lfloor o \rfloor} W \mid h \triangleright k \mid k\langle m;n \rangle \mid \widetilde{h}\langle \triangleright h \rangle = V' \quad . \qquad [\textsc{Mig}_o]$$

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

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

  $$U' \mathrel{\widehat{\xrightarrow{\tau}}}_{\lfloor o \rfloor} W' \mid (h, \widetilde{h}) \triangleright k \mid k[m;n;n']$$
  $$V' \mathrel{\widehat{\xrightarrow{\tau}}}_{\lfloor o \rfloor} W' \mid \widetilde{h} \triangleright h \mid h \triangleright k \mid k[m;n;n']$$

  (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 l applications of the swapping relation, l being the length of $\widetilde{h}$: $U' \mathrel{\widehat{\xrightarrow{\tau}}}_{\lfloor o \rfloor} \mathcal{S}^l \mathrel{\widehat{\xleftarrow{\tau}}}_{\lfloor o \rfloor} V'$.
- A forwarder $k \triangleright k'$: like in the previous case, we first route the available

messages to their destination, say k'':

$$U' \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} W' \mid (h, k, \widetilde{h}) \triangleright k'' \mid k''[m; n; n'] \ ,$$
$$V' \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} W' \mid \widetilde{h} \triangleright h \mid h \triangleright k \mid k \triangleright k'' \mid k''[m; n; n'] \ .$$

Here we need an additional application of the swapping relation to relocate h to k'', before being able to relocate the forwarders at $\widetilde{h}$:

$$U' \stackrel{\widehat{\tau}}{\Rightarrow}_{\lfloor o \rfloor} \mathcal{S}^{l+1} \stackrel{\widehat{\tau}}{\Leftarrow}_{\lfloor o \rfloor} V' \ .$$

- A blocked forwarder $k \not\triangleright$. We reason like in the previous case by first routing the message that blocks this forwarder to its destination. ■

Lemma 3.18. *If $U \ \mathcal{S} \ V$ and $U \stackrel{\mu}{\rightarrow}_{\lfloor o \rfloor} U'$ then $U' \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} \mathcal{S}^\star \stackrel{\widehat{\mu}}{\Leftarrow}_{\lfloor o \rfloor} V$.*

Proof. It is immediate if $\mu \neq \tau$: we have $U' \ \mathcal{S} \stackrel{\mu}{\leftarrow}_{\lfloor o \rfloor} V$. When $\mu = \tau$, the interesting cases are those where the silent transition $U \stackrel{\tau}{\rightarrow}_{\lfloor o \rfloor} U'$ is the transmission of a message trough one of the two forwarders being swapped:

- $U = W \mid h\langle m \rangle_{\widetilde{h}} \mid h \triangleright h' \mid h' \triangleright k \ \stackrel{\tau}{\rightarrow}_{\lfloor o \rfloor} \ W \mid h \not\triangleright \mid h'\langle m \rangle_{h, \widetilde{h}} \mid h' \triangleright k = U'$.
  By routing the messages, we obtain:

$$U' \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} W' \mid h \triangleright k' \mid h' \triangleright k' \mid k'[m; n] = U'' \ ,$$
$$V \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} W' \mid h \triangleright k' \mid h' \triangleright k \mid k'[m; n] = V' \ .$$

  If $k = k'$, we are done: $U'' = V'$. Otherwise, there is a forwarder $k \triangleright k'$ in W', and we need one application of the swapping relation in order to relocate the forwarder located at h' in V'.
- $U = h \triangleright h' \mid h'\langle m \rangle_{\widetilde{h}} \mid h' \triangleright k \ \stackrel{\tau}{\rightarrow}_{\lfloor o \rfloor} \ h \triangleright h' \mid h' \not\triangleright \mid k\langle m \rangle_{h', \widetilde{h}} = U'$
  By routing the messages, we obtain:

$$U' \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} W' \mid h \triangleright h' \mid h' \triangleright k' \mid k'[m; n] = U'' \ ,$$
$$V \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} W' \mid h \triangleright k \mid h' \triangleright k' \mid k'[m; n] = V' \ .$$

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

$$\begin{aligned} U'' = W'' \mid h \triangleright h' \mid k \triangleright k' \mid h' \triangleright k' \mid k'[m; n] \\ \mathcal{S} \ W'' \mid h \triangleright k' \mid k \triangleright k' \mid h' \triangleright k' \mid k'[m; n] \\ \mathcal{S} \ W'' \mid h \triangleright k \mid k \triangleright k' \mid h' \triangleright k' \mid k'[m; n] = V' \end{aligned}$$

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

We now have to prove the termination property for Technique 3.13. Since $(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})^+ \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor} \subseteq \mathcal{S}^\star \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor}$, the following lemma will suffice:

Lemma 3.19. *$\mathcal{S}^\star \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor}$ 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 $\stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor}$ contains at least one transition). ■

Proposition 3.20. *For any $U, V \in \lfloor \mathcal{O} \rfloor$, if $U \; \mathcal{S} \; V$ or $U \xrightarrow{\tau}_{\lfloor o \rfloor} V$, then we have:*

$$\left\langle U, \rightarrow_{\lfloor o \rfloor} \right\rangle \approx \left\langle V, \rightarrow_{\lfloor o \rfloor} \right\rangle \ .$$

Proof. By applying Technique 3.13 to $(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})$, we obtain that $(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})^\star$ is a simulation. Moreover, $\widehat{\xleftarrow{\tau}}_{\lfloor o \rfloor}$ is also a simulation (as is always the case for reversed silent transitions). By combining these two results we obtain that the symmetric relation $(\mathcal{S} \cup \xleftrightarrow{\tau}_{\lfloor o \rfloor})^\star = ((\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})^\star \cup \widehat{\xleftarrow{\tau}}_{\lfloor o \rfloor})^\star$ is a simulation, and hence a bisimulation. We finally have $(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor}) \subseteq (\mathcal{S} \cup \xleftrightarrow{\tau}_{\lfloor o \rfloor})^\star \subseteq \approx$. ■

Notice that Technique 3.13 plays a crucial role in the proof of the above proposition: it allows us to focus on the "local" relations $\mathcal{S}$ and $\xrightarrow{\tau}_{\lfloor o \rfloor}$, which relate nets that differ only slightly, so that we can reason about their small syntactical differences. In contrast, if we had to prove the previous proposition directly, we would have to show that $(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})^\star$ is a simulation; which is really hard as this relation relates completely different nets.

Lemmas 3.19 and 3.15 ensure that $\xrightarrow{\tau}_{\lfloor o \rfloor}$ defines a unique normal form for any clean optimised net U (termination of $\mathcal{S}^\star \stackrel{\tau}{\Rightarrow}_{\lfloor o \rfloor}$ entails termination of $\xrightarrow{\tau}_{\lfloor o \rfloor}$); we denote by $U_{\downarrow o}$ this normal form. Notice that $U_{\downarrow o}$ is always a reference net: it does not contain any blocked forwarder, relocation message, nor pending, annotated messages.

The normalisation of a clean simple net by $\xrightarrow{\tau}_{\lfloor s \rfloor}$ and $\xrightarrow{\tau}_{\lfloor o \rfloor}$ does not necessarily lead to the same net: $U_\downarrow \neq U_{\downarrow o}$. However, these nets differ only by some rearrangement of their forwarders, they are related by $\mathcal{S}^\star$:

Lemma 3.21. *For any clean simple net U, we have:*

$$U_\downarrow \; \mathcal{S}^\star \; U_{\downarrow o} \ .$$

Proof. We proceed by well-founded induction on U, using the termination of the relation $\xrightarrow{\tau}_{\lfloor s \rfloor}$:

- If U is a reference net, we have $U_\downarrow = U = U_{\downarrow o}$.
- If $U \xrightarrow{\tau}_{\lfloor s \rfloor} U'$, since U is clean and simple, we have

$$\begin{aligned} & U = V \mid h_0\langle m \rangle \mid \Pi_{i<l}\ h_i \triangleright h_{i+1} \mid h_l[n] \\ & U \xrightarrow{\tau}_{\lfloor s \rfloor} U' \overset{\tau}{\Rightarrow}_{\lfloor s \rfloor} V \mid \Pi_{i<l}\ h_i \triangleright h_{i+1} \mid h_l[m;n] = U_1 && [\text{FWD}_s, \text{DST}_s] \\ & U \overset{\tau}{\Rightarrow}_{\lfloor o \rfloor} V \mid \Pi_{i<l}\ h_i \triangleright h_l \mid h_l[m;n] = U_2 && (\text{Lemma 3.14}) \end{aligned}$$

  We check that $U_1\ \mathcal{S}^l\ U_2$, and we have $U_1 \overset{\widehat{\tau}}{\Rightarrow}_{\lfloor o \rfloor} U_{1\downarrow o}$ so that from Prop. 3.20, $U_2 \overset{\widehat{\tau}}{\Rightarrow}_{\lfloor o \rfloor} U_2'$ with $U_{1\downarrow o}(\mathcal{S} \cup \xrightarrow{\tau}_{\lfloor o \rfloor})^\star U_2'$.
  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 in the clean, optimised $\mathcal{L}_n$-TS, any clean simple net is bisimilar to its normal form w.r.t. $\xrightarrow{\tau}_{\lfloor s \rfloor}$ (this correspond to Corollary 2.21 in the clean, simple $\mathcal{L}_n$-TS):

Corollary 3.22. *For any clean simple net U, we have:*

$$\left\langle U, \rightarrow_{\lfloor o \rfloor} \right\rangle \approx \left\langle U_{\downarrow o}, \rightarrow_{\lfloor o \rfloor} \right\rangle \approx \left\langle U_\downarrow, \rightarrow_{\lfloor o \rfloor} \right\rangle \ .$$

Proof. By Lemma 3.21, $U \overset{\widehat{\tau}}{\Rightarrow}_{\lfloor o \rfloor} U_{\downarrow o}\ \mathcal{S}^\star\ U_\downarrow$; we conclude with Prop. 3.20. ■

Since Corollary 3.22 does not give an expansion result, we cannot use the standard bisimulation up-to expansion technique to give the final proof of correctness. Instead, we use the following restricted form of the "bisimulation up to bisimilarity" technique (in its unrestricted form, this technique is not correct [13]):

Technique 3.23 (Bisimulation up to Bisimilarity on Visible Actions)**.**
Let $\mathcal{R}$ be a relation such that:

- $\xleftarrow{\tau} \mathcal{R} \subseteq \mathcal{R} \overset{\widehat{\tau}}{\Leftarrow}$ and $\mathcal{R} \xrightarrow{\tau} \subseteq \overset{\widehat{\tau}}{\Rightarrow} \mathcal{R}$; and
- $\xleftarrow{a} \mathcal{R} \subseteq \approx\mathcal{R}\approx \overset{a}{\Leftarrow}$ and $\mathcal{R} \xrightarrow{a} \subseteq \overset{a}{\Rightarrow} \approx\mathcal{R}\approx$ for any visible action $a \in \mathcal{L} \setminus \{\tau\}$.

Then $\approx\mathcal{R}\approx$ is a bisimulation, and $\mathcal{R} \subseteq \approx$.

Using this technique, we can consider only reference nets, that do not contain any pending messages, and normalise them whenever they do a visible action that brings some pending messages. Since reference nets do not give rise to

silent transitions, even when considered in the simple or optimised $\mathcal{L}_n$-TSs, the restriction of the previous technique to visible challenges is not problematic.

Lemma 3.24. *For any reference net U, we have:*

$$\left\langle U, \rightarrow_{\lfloor o \rfloor} \right\rangle \approx \left\langle U, \rightarrow_{\lfloor s \rfloor} \right\rangle \ .$$

Proof. We apply Technique 3.23 to the following relation:

$$\mathcal{R} \triangleq \left\{ \left\langle \left\langle U, \rightarrow_{\lfloor \mathrm{o} \rfloor} \right\rangle, \ \left\langle U, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \right\rangle \mid U \in \mathcal{U}_{\mathrm{r}} \right\} \ .$$

For any reference net U, since U does not contain any pending message, we have $U \xrightarrow{\mu}_{\lfloor \mathrm{o} \rfloor} V$ iff $U \xrightarrow{\mu}_{\lfloor \mathrm{s} \rfloor} V$. However, while V is a clean simple net, it is not necessarily a reference net, so that we do not have $\left\langle V, \rightarrow_{\lfloor \mathrm{o} \rfloor} \right\rangle \mathcal{R} \left\langle V, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle$. Nevertheless, $V_{\downarrow}$ is a reference net, and by Corollaries 3.22 and 2.21, we have:

$$\left\langle V, \rightarrow_{\lfloor \mathrm{o} \rfloor} \right\rangle \approx \left\langle V_{\downarrow}, \rightarrow_{\lfloor \mathrm{o} \rfloor} \right\rangle \mathcal{R} \left\langle V_{\downarrow}, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \precsim \left\langle V, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle \ .$$

We use bisimilarity to rewrite the left-hand-side process; this is allowed by Technique 3.23: we are necessarily in a visible challenge. ■

By following the steps depicted in Fig. 1, we can finally prove the correctness of the implementation w.r.t. the specification:

Theorem 3.25. *For any optimised net U, we have:*

$$\langle U, \rightarrow_o \rangle \approx \left\langle \lfloor U \rfloor_{\downarrow o}, \rightarrow_r \right\rangle \ .$$

Proof. $\lfloor U \rfloor_{\downarrow \mathrm{o}}$ is a reference net, therefore, we have:

$$\begin{aligned}
\langle U, \rightarrow_{\mathrm{o}} \rangle &\succsim \left\langle \lfloor U \rfloor, \rightarrow_{\lfloor \mathrm{o} \rfloor} \right\rangle && \text{(Proposition 3.11)} \\
&\approx \left\langle \lfloor U \rfloor_{\downarrow \mathrm{o}}, \rightarrow_{\lfloor \mathrm{o} \rfloor} \right\rangle && \text{(Corollary 3.22)} \\
&\approx \left\langle \lfloor U \rfloor_{\downarrow \mathrm{o}}, \rightarrow_{\lfloor \mathrm{s} \rfloor} \right\rangle && \text{(Lemma 3.24)} \\
&\succsim \left\langle \lfloor U \rfloor_{\downarrow \mathrm{o}}, \rightarrow_{\mathrm{r}} \right\rangle \ . && \text{(Theorem 2.23)}
\end{aligned}$$

■

Acknowledgements

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

References

[1] S. Arun-Kumar and M. Hennessy. An Efficiency Preorder for Processes. *Acta Informatica*, 29(9):737–760, 1992.

[2] 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.

[3] L. Cardelli and A. Gordon. Mobile Ambients. In *Proc. FOSSACS '98*, volume 1378 of *LNCS*, pages 140–155. Springer Verlag, 1998.

[4] 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.

[5] 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.

[6] M. Hennessy. *A Distributed Pi-Calculus*. Cambridge University Press, 2007.

[7] 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.

[8] F. Levi and D. Sangiorgi. Mobile Safe Ambients. *ACM Trans. on Progr. Lang. and Sys.*, 25(1):1–69, 2003. ACM Press.

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

[10] 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.

[11] D. Pous. Up-to Techniques for Weak Bisimulation. In *Proc. 32th ICALP*, volume 3580 of *LNCS*, pages 730–741. Springer Verlag, 2005.

[12] D. Pous. On bisimulation proofs for the analysis of distributed abstract machines. In *Proc. TGC '06*, 2006. (to appear).

[13] 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.

[14] D. Sangiorgi and A. Valente. A Distributed Abstract Machine for Safe Ambients. In *Proc. 28th ICALP*, volume 2076 of *LNCS*. Springer Verlag, 2001.

[15] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[16] A. Unyapoth and P. Sewell. Nomadic pict: Correct Communication Infrastructure for Mobile Computation. In *Proc. 28th POPL*, pages 116–127. ACM Press, 2001.