Math. Struct. in Comp. Science (2008), *vol.* 18, *pp.* 107–143.
doi:10.1017/S0960129507006585 Printed in the United Kingdom

Multiple-Labelled Transition Systems for nominal calculi and their logics[†]

ROCCO DE NICOLA and MICHELE LORETI

Dipartimento di Sistemi e Informatica, Viale Morgagni, 65 50134 Firenze, Italy

Received 21 May 2007

In Memory of Sauro Tulipani

Action-labelled transition systems (LTSs) have proved to be a fundamental model for describing and proving properties of concurrent systems. In this paper we introduce Multiple-Labelled Transition Systems (MLTSs) as generalisations of LTSs that enable us to deal with system features that are becoming increasingly important when considering languages and models for network-aware programming. MLTSs enable us to describe not only the actions that systems can perform but also their usage of resources and their handling (creation, revelation ...) of names; these are essential for modelling changing evaluation environments. We also introduce MoMo, which is a logic inspired by Hennessy–Milner Logic and the μ-calculus, that enables us to consider state properties in a distributed environment and the impact of actions and movements over the different sites. MoMo operators are interpreted over MLTSs and both MLTSs and MoMo are used to provide a semantic framework to describe two basic calculi for mobile computing, namely μKLAIM and the asynchronous π-calculus.

1. Introduction

A well-established and successful approach to modelling and verifying the properties of concurrent systems uses process algebras and temporal logics as a basis. Concurrent systems are specified as terms of a process description language, while properties are specified as temporal logic formulae. Labelled transition systems are associated with terms via a set of structural operational semantics rules, and model checking is used to determine whether the transition systems associated with terms enjoy the property specified by the temporal formulae.

A *Labelled Transition System* (LTS) consists of a set of states $\mathcal{S}$, a set of transition labels $\mathcal{L}$ and a transition relation $\rightarrow \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$. *States* correspond to the configurations that the system can reach. *Labels* describe the actions that the system can perform to interact with the environment. *Transition relations* describe the system's evolution as determined by the execution of specific actions. Temporal logic formulae are a mix of logical operators and modal operators. The former are the usual boolean operators, while the latter are the operators that permit reasoning about the system's evolution in time and for dealing with the LTS's dynamic aspects. The success of these two formalisms has led to interesting results and to the development of general tools and theorems for

[†] The work presented in this paper was partially supported by EU Project Software Engineering for Service-Oriented Overlay Computers (SENSORIA, contract IST-3-016004-IP-09).

minimisation and animation of LTSs and for model checking satisfaction of temporal formulae over them.

In the last ten years, and stimulated by new applications of network-aware programming, several new formalisms based on process algebras, but with new constructs for modelling network topology, name passing, resources usage and mobility, have been proposed. Here, we will just mention the Distributed Join-calculus (Fournet *et al.* 1996), the Distributed π-calculus (Hennessy and Riely 2002), the Ambient calculus (Cardelli and Gordon 2000), the Seal calculus (Castagna *et al.* 2005), Nomadic Pict (Wojciechowski and Sewell 1999) and KLAIM (De Nicola *et al.* 1998).

The new primitives of these distributed nominal calculi, are such that not only actions, but also names and resources play a central role. This makes it more difficult to use LTSs, operational semantics and classical temporal logics to deal with the new calculi. To model some of their distinguishing features, such as *name scoping* or *process and data distribution*, transition labels had to be *enriched* to carry information not only about the actions performed, but also about the state of the system. This interplay has made the development of a general approach to distributed nominal calculi more difficult, and we have witnessed the development of a number of theories tailored to specific calculi that cannot be easily generalised.

A side effect of this lack of homogeneity has been the proposal of many different temporal logics devised for reasoning about the different calculi. Two variants of HML (*Hennessy–Milner Logic* (Hennessy and Milner 1985)) for specifying and verifying properties of π-calculus processes were introduced in Milner *et al.* (1993) and in Dam (1996). Cardelli and Gordon (2006) introduced a spatial and modal logic for specifying and verifying properties of Mobile Ambients. On the other hand, *MobileUnity* (McCann and Roman 1998) and *MobAdtl* (Ferrari *et al.* 2002) are two program logics designed specifically for specifying and reasoning about mobile systems by exploiting a Unity-like proof system.

We feel that it is important to come up with a general operational model that can play the same role for distributed nominal calculi that LTSs have played for process algebras. Such an operational model should allow us to capture all main aspects of the new class of systems in a natural way and should represent the natural basis for interpreting temporal logics for describing system properties. A central role in the new set of mobile calculi is played by *names*. Names can be *public*, that is, known by the environment where the system is executing, or *private*. Systems, during execution, can *discover* some of the private names. Thus, the behaviour of terms of mobile calculi cannot be described without taking into account the names they *use*, and classical LTS do not have a natural and explicit handling of names.

In this paper we propose a variant of LTSs, which we call *Multiple-Labelled Transition Systems* (MLTSs), as a candidate general operational model. MLTSs are equipped with a number of transition relations that capture the different aspects of system behaviour, namely *action execution*, *resource creation and consumption* and *name revelation*. An MLTS consists of:

— a set of *states*, which describe system configurations;

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

— a set of *resources*, which model data (for example, the values exchanged over a channel), computational environments (for example, the locations where processes can be executed) and network links (which can be used for interaction);
— a set of *transition labels*;
— a *naming structure*, which enables us to associate public (known by the environment) names with each state;
— three transition relations:
  – the *computation relation* describes the interaction with the environment;
  – the *resource relation* describes resource usage;
  – the *revelation relation* describes the names revealed to the environment.

In order to model the properties of an MLTS, we introduce a temporal logic (MoMo), which consists of a number of basic operators to be used to describe specific properties/behaviours of mobile and distributed systems. Thus, together with the usual logical connectives and the operators for minimal and maximal fixed points, MoMo is equipped with operators for describing dynamic behaviours (*temporal properties*), for modelling resource management (*state properties*), for the handling of names (*nominal properties*), and for controlling mobile processes (*mobility properties*).

To show the usefulness of our proposal, we show how MLTSs can be used to describe the operational semantics of two formalisms with opposite objectives, namely μKLAIM (Bettini *et al.* 2003) and the asynchronous π-calculus ($A\pi$) (Boudol 1992; Honda and Tokoro 1991). μKLAIM is a simplified version of KLAIM, which is an experimental programming language that supports a programming paradigm where both processes and data can be moved across different computing environments, and which relies on the use of explicit localities. $A\pi$ is the generally recognised minimal common denominator of calculi for mobility. We also introduce two dialects of MoMo for reasoning about the two calculi under consideration.

The rest of the paper is organised as follows. In Section 2, we introduce our general MLTS model, while in Sections 3 and 4 we show how MLTSs can be used to describe the operational semantics of $A\pi$ and μKLAIM, respectively. The modal logic for mobility is presented in Section 5, and its dialects are discussed in Section 6, where we also specify a few properties of a simple client–server system. Section 7 contains some results about the relationships between the equivalence induced on $A\pi$ by MoMo and the classical notion of bisimulation. The final section contains some concluding remarks.

2. Multiple-Labelled Transition Systems

In this section we introduce our variant of LTSs, which will be used as a general framework for describing the semantics of mobile calculi. We will begin by introducing a few basic definitions.

Definition 2.1. Let $\mathcal{N}$ be a set of names and X be a subset of $\mathcal{N}$. A *name substitution* is a function $\sigma : \mathcal{N} \to \mathcal{N}$, where:

— $\{y_1/x_1, \ldots, y_n/x_n\}$ denotes the substitution that maps x_i into y_i and is the identity on the other names;

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

— $\sigma_1 \cdot \sigma_2$ is the composition of σ_1 and σ_2;
— $\sigma_1 \backslash X$ is the substitution σ_2 such that

$$\sigma_2(n) = \begin{cases} n & \text{if } n \in X \\ \sigma_1(n) & \text{otherwise;} \end{cases}$$

— $\varnothing$ denotes the identity substitution;
— Σ denotes the set of substitutions σ.

Definition 2.2. With $X \subseteq \mathcal{N}$ and $\sigma \in \Sigma$, we define $\sigma(X)$ to be the image of X with respect to σ, namely

$$\sigma(X) = \{x' | \exists x \in X : \sigma(x) = x'\}.$$

Definition 2.3. A *naming structure* for a set S is a triple $\mathbb{N} = \langle \mathcal{N}, \eta, \circ \rangle$ such that:

— $\mathcal{N}$ is a countable set of *names*
— $\eta : S \to 2^{\mathcal{N}}$ is the *naming function*
— $\circ : S \times \Sigma \to S$ is the *renaming function*

where, for each $s \in S$ and $\sigma \in \Sigma$, if we use $s \circ \sigma$ to denote $\circ(s, \sigma)$, we have

— $\eta(s)$ is finite
— $s \circ \varnothing = s$
— $\eta(s \circ \sigma) = \sigma(\eta(s))$.

Intuitively, a naming structure for S enables us to consider the elements of S as containers of names. For each element s in S, $\eta(s)$ gives the set of names that appear in s, while the application of an element s to a substitution σ ($s \circ \sigma$) yields the element of S obtained by renaming each name in s according to σ.

If one considers the set of π-calculus processes Proc where

— Ch is the set of channel names
— $fn(P) \subseteq$ Ch yields the free-names in P
— $P \circ \sigma$ returns the process obtained from P by replacing each free name in P as determined by substitution σ,

then the triple $\langle$Ch, $fn, \cdot\rangle$ is a naming structure for Proc.

A Multiple-Labelled Transition System, MLTS for short, consists of a set of *states* $\mathcal{S}$, a set of *resources* $\mathcal{R}$, a set of *transition labels* $\mathcal{L}$ and a naming structure $\langle N, \eta, \sigma \rangle$ for each of $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{L}$. *States* describe the configurations of a system modelled by an MLTS. *Naming structures* label each state by a set of names. These are the names that are public (known by the environment) when the system is in that state. *Resources* are the necessary prerequisites for system evolutions. *Transition labels* typically identify the actions a system can perform to interact with the environment. We also assume that $\mathcal{L}$ contains a special distinct transition label τ denoting internal/silent evolution of a system.

MLTSs provide information about:

— the actions that a system can perform to interact with the environment;
— a system's reactions to creation/deletion of new *resources*; and
— a system's handling of privite/public names.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

The modelling via an MLTS is based on three different transition relations:

— the *computation relation* $\longrightarrow \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$;
— the *resource relation* $\cdots\!\!\rightarrow \subseteq \mathcal{S} \times (\{\oplus, \ominus\} \times \mathcal{R}) \times \mathcal{S}$;
— the *revelation relation* $\hookrightarrow \subseteq \mathcal{S} \times \mathcal{N} \times \mathcal{S}$.

As already mentioned, the three transition relations of an MLTS enable us to model a system's behaviour with respect to three different aspects. The *computation relation* plays the same role as the transition relation in an LTS in that it describes the actions a system can perform to interact with the environment. The *resource relation* describes how a system evolves when resources are created or consumed. If r is a resource, we have that $s_1 \overset{\ominus r}{\cdots\!\!\rightarrow} s_2$ is possible if r is available at state s_1; after r is consumed state s_2 is reached. Similarly, we have $s_1 \overset{\oplus r}{\cdots\!\!\rightarrow} s_2$ if s_2 can be obtained from s_1 by adding resource r. The *revelation relation* describes the capability of a system to reveal a private name to the environment.

To guarantee the correct management of names, we need to require some specific properties for the three transition relations, namely that they have to be preserved by *name permutations*, that is, by bijective substitutions.

Definition 2.4. Let $\mathbb{N} = \langle \mathcal{N}, \eta, \circ \rangle$ be a naming structure for $A_1, \ldots, A_n$. We say that a relation $\mathscr{R} \subseteq A_1 \times \ldots \times A_n$ is *preserved by name permutations* if and only if for each *name permutation* σ,

$$(a_1, \ldots, a_n) \in \mathscr{R} \Leftrightarrow (a_1 \circ \sigma, \ldots, a_n \circ \sigma) \in \mathscr{R}\,.$$

Definition 2.5. A *Multiple-Labelled Transition System* is a 7-tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{R}, \mathcal{L}, \mathbb{N}, \longrightarrow, \cdots\!\!\rightarrow, \hookrightarrow \rangle$$

where:

— $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{L}$ are countable sets of *states*, *resources* and *transition labels*;
— $\mathbb{N} = \langle \mathcal{N}, \eta, \circ \rangle$ is a naming structure for $\mathcal{S}$, $\mathcal{R}$, $\mathcal{L}$ and $\mathcal{N}$; and
— the three relations

$$\begin{aligned} \longrightarrow &\subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}; \\ \cdots\!\!\rightarrow &\subseteq \mathcal{S} \times (\{\oplus, \ominus\} \times \mathcal{R}) \times \mathcal{S} \\ \hookrightarrow &\subseteq \mathcal{S} \times \mathcal{N} \times \mathcal{S} \end{aligned}$$

are preserved by name permutations.

In the rest of this paper we will use $\mathcal{M}^{\mathcal{S}}$, $\mathcal{M}^{\mathcal{R}}$, $\mathcal{M}^{\mathcal{L}}$ and $\mathcal{M}^{\mathbb{N}}$ to denote the set of states, the set of resources, the set of transition labels and the naming structure of an MLTS $\mathcal{M}$, respectively.

In the next two sections we will show how MLTSs can be used for describing the behaviour of asynchronous π-calculus and μKLAIM.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

3. Asynchronous π-calculus

In this section we consider the asynchronous π-calculus ($A\pi$) (Boudol 1992; Honda and Tokoro 1991), which is a subset of the π-calculus in which there is no output prefixing.

If Ch is a countable set of channels (whose elements are ranged over by $a, b, \ldots$), we let $\text{Proc}_{A\pi}$ be the set of $A\pi$ processes ($P, Q, \ldots$) defined by the following syntax:

$$\begin{aligned} P, Q &::= \bar{a}b \mid G \mid P|Q \mid \nu a.P \mid !G \\ G &::= \mathbf{0} \mid a(b).P \mid \tau.P \mid G + G \end{aligned}$$

where G denotes *guarded processes*.

In $A\pi$, processes interact by exchanging messages over channels.

The process $\bar{a}b$ is used to indicate the availability of message b over channel a; it models non-blocking outputs.

A *guarded process* G can be:

— $\mathbf{0}$, the deadlocked process;
— $a(b).P$, the process that retrieves a value over channel a and then behaves like process P, and where the name b is bound to the retrieved value;
— $\tau.P$, the process that performs an internal action and then behaves like P;
— $G_1 + G_2$, the process that can non-deterministically behave like G_1 or G_2.

Processes are composed via *parallel composition*: $P|Q$ describes a system composed from two components (specified by P and Q, respectively) that can proceed independently and can interact via shared channels.

Private names are defined using *restriction*: $\nu a.P$ denotes the fact that a is a private name in P.

Infinite behaviours are modelled via process *replication* ($!G$). This can be thought of as an infinite composition of G ($G|G|\ldots$).

Note that both input prefixing $b(a).P$ and name restriction $\nu a.P$ act as binders for name a within P. In the rest of the paper we shall use $fn(P)$ to denote the set of names free in P. We will also write $P =_\alpha Q$ to denote the fact that P and Q are equal up-to renaming of bound names. For instance, $a(x).\bar{b}(x) =_\alpha a(y).\bar{b}(y)$ and $\nu a.\bar{b}a =_\alpha \nu c.\bar{b}c$. Let P be a process and σ a name substitution, we write $P \circ \sigma$ to mean the process obtained from P by replacing every free name a with $\sigma(a)$.

Terms that intuitively represent the same process are identified by means of standard *structural congruence* $\equiv$. This relation is defined as the smallest congruence relation over $A\pi$ processes induced by the laws in Table 1. The structural laws express the fact that $|$ is commutative and associative and that the empty process can always be safely removed from or added to a parallel composition. Structural equivalence also states that $!P$ is equivalent to $!P|P$, and that if a does not occur in Q, then $\nu a.(P|Q)$ is equivalent to $\nu a.P|Q$. In the rest of the paper, elements in $\text{Proc}_{A\pi}$ will be considered equal up to structural equivalence.

We will now introduce a simple $A\pi$ specification for a simple system, which we will use as a running example in the rest of the paper.

Example 3.1. A *proxy* is a system such that, given two channels a and b, if appropriate, it will emit the values read from channel a on channel b.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 1. PROC$_{A\pi}$ *Structural equivalence*

$$P|\mathbf{0} \equiv P \qquad P|Q \equiv Q|P \qquad (P|Q)|R \equiv P|(Q|R)$$

$$!P \equiv P|!P \qquad \frac{a \notin fn(Q)}{\nu\, a.(P|Q) \equiv (\nu\, a.P)|Q}$$

Table 2. *Labelled transition system for* $A\pi$

$$\tau.P \overset{\tau}{\longmapsto} P \qquad a(b).P \overset{ac}{\longmapsto} P[c/b] \qquad \overline{a}b \overset{\overline{a}b}{\longmapsto} \mathbf{nil}$$

$$\frac{P \overset{\overline{a}b}{\longmapsto} P' \quad a \neq b}{\nu\, b.P \overset{\overline{a}(b)}{\longmapsto} P'} \qquad \frac{P \overset{\lambda}{\longmapsto} P' \quad a \notin n(\lambda)}{\nu\, a.P \overset{\lambda}{\longmapsto} \nu\, a.P'} \qquad \frac{P \overset{\overline{a}b}{\longmapsto} P' \quad Q \overset{ab}{\longmapsto} Q'}{P|Q \overset{\tau}{\longmapsto} P'|Q'}$$

$$\frac{P \overset{\overline{a}(b)}{\longmapsto} P' \quad Q \overset{ab}{\longmapsto} Q' \quad b \notin fn(Q)}{P|Q \overset{\tau}{\longmapsto} \nu\, b.P'|Q'} \qquad \frac{G \overset{\lambda}{\longmapsto} P}{G + G' \overset{\lambda}{\longmapsto} P} \qquad \frac{G \overset{\lambda}{\longmapsto} P}{!G \overset{\lambda}{\longmapsto} P|!G}$$

$$\frac{P \overset{\lambda}{\longmapsto} P' \quad fn(Q) \cap bn(\alpha) = \varnothing}{P|Q \overset{\lambda}{\longmapsto} P'|Q'}$$

A possible $A\pi$ specification for this system is

$$\text{PROXY}_1 = !a(x).\tau.\overline{b}x$$

where process PROXY$_1$ models the system as the infinite replication of a process that first reads a value x from channel a, then performs some internal actions, and, finally, emits x on channel b.

The operational semantics of $A\pi$ processes is described in Amadio *et al.* (1998) by means of the LTS relation ($\longmapsto$), which is defined in Table 2, where it is assumed that $\longmapsto$ is closed with respect to the structural congruence relation $\equiv$. This means that

$$\frac{P \equiv Q \quad Q \overset{\lambda}{\longmapsto} Q' \quad Q' \equiv P'}{P \overset{\lambda}{\longmapsto} P'}\,.$$

As usual, bound and free names are considered, and we have $n(\lambda) = fn(\lambda) \cup bn(\lambda)$ where

$$fn(\tau) = \varnothing \quad fn(\overline{a}(b)) = \{a\} \quad fn(\overline{a}b) = fn(a(b)) = \{a, b\}$$
$$bn(\tau) = \varnothing \quad bn(\overline{a}(b)) = \{b\} \quad bn(\overline{a}b) = bn(a(b)) = \varnothing.$$

The transition relation $\longmapsto$ of Amadio *et al.* (1998) makes use of labels of the form

$$\lambda ::= \tau \mid \overline{a}b \mid \overline{a}(b) \mid a(b)\,,$$

and describes behaviours by considering different aspects at the same time. Labels τ and $a(b)$ are used to describe computations of a process: $P \overset{\tau}{\longmapsto} P'$ if P can perform an internal synchronisation and then behaves like P'; while $P \overset{a(b)}{\longmapsto} P'$ if P behaves like

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 3. $\mathcal{M}_{A\pi}$: *transition relations*

$$\tau.P \xrightarrow{\tau} P \qquad a(b).P|\overline{a}c \xrightarrow{\tau} P[c/b]$$

$$\frac{P \xrightarrow{\tau} P'}{P|Q \xrightarrow{\tau} P'|Q} \qquad \frac{P \xrightarrow{\tau} P'}{\nu\, a.P \xrightarrow{\tau} \nu\, a.P'}$$

$$P \overset{\oplus\overline{a}b}{\cdots\!\!\rightarrow} P|\overline{a}b \qquad P|\overline{a}b \overset{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P$$

$$\frac{a \neq b}{\nu\, a.(P \parallel \overline{b}a) \hookrightarrow^{a} P \parallel \overline{b}a}$$

P' after value b is retrieved from channel a. The same relation is used to describe the state/spatial configuration of a process: $P \overset{\overline{a}b}{\longmapsto} P'$ if $P \equiv P'|\overline{a}b$. Moreover, if the value b is private in P ($P \equiv \nu\, b.(P'|\overline{a}b)$), P evolves to P' with a label $\overline{a}(b)$ denoting the fact that b is bound in P' and that b is communicated in some way to the environment, so b is no longer private in P'.

We shall now see how MLTSs can be used to describe the behaviour of $A\pi$ processes in such a way that computations, spatial configurations and name revelations are modelled separately. We will define an MLTS (named $\mathcal{M}_{A\pi}$) that can be used as a semantic model for $A\pi$ processes. The set of states in $\mathcal{M}_{A\pi}$ will be the set of all $A\pi$ processes. Since processes during their computations consume and produce values over channels, we consider our resources to be the set of terms denoting the availability of messages over channels.

The computation relation will only consider internal actions and synchronisation. For this reason, the set of transition labels in $\mathcal{M}_{A\pi}$ will contain τ labels only.

We let $\mathcal{M}_{A\pi}$ be the MLTS defined by

$$\langle \textsc{Proc}_{A\pi}, \textsc{Res}_{A\pi}, \textsc{Lab}_{A\pi}, \mathbb{N}_{A\pi}, \longrightarrow, \cdots\!\!\rightarrow, \hookrightarrow \rangle \tag{1}$$

where:

— $\textsc{Res}_{A\pi} = \{\overline{a}b | a, b \in \textsc{Ch}\}$;
— $\textsc{Lab}_{A\pi} = \{\tau\}$;
— $\mathbb{N}_{A\pi} = \langle \textsc{Ch}, fn, \circ \rangle$;
— $\longrightarrow$, $\cdots\!\!\rightarrow$ and $\hookrightarrow$ are the transition relations induced by the rules of Table 3, plus those induced by the closure of the relations under $\equiv$ as defined in Table 1.

The computation relation describes internal computation of a system caused by internal moves ($\tau.P$) or by process synchronisation. The resource relation staes that starting from a process P, a resource $\overline{a}b$ can be created leading to process $P|\overline{a}b$. Conversely, a resource $\overline{a}b$ can be consumed in $P \equiv Q|\overline{a}b$ to obtain Q. Finally, a process P reveals a name a, if this is available over a public channel b. Intuitively, this means that the environment is able to know the *private name*.

To show that the $\mathcal{M}_{A\pi}$ defined above is indeed an MLTS, we need to prove that $\mathbb{N}_{A\pi}$ is a named structures for states, resources and transition labels, and that all the transition relations considered are preserved by name permutation.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

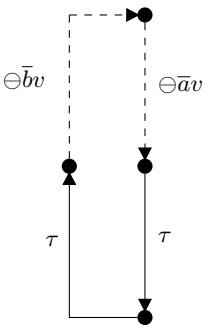

Fig. 1. A portion of the MLTS for $\textsc{Proxy}_1$ of Example 3.1

Lemma 3.2. $\mathbb{N}_{A\pi}$ is a naming structure for $\textsc{Proc}_{A\pi}$, $\textsc{Res}_{A\pi}$, $\textsc{Lab}_{A\pi}$ and $\textsc{Ch}$.

Proof. The thesis follows easily by noting that:

— for each process P and substitution σ, we have $fn(P)$ is finite, $P \circ \varnothing = P$ and $fn(P \circ \sigma) = \sigma(fn(P))$;
— for each $a \in \textsc{Ch}$, we have $fn(a) = \{a\}$ and $a \circ \sigma = \sigma(a)$;
— $fn(\tau) = \varnothing$ and $\tau \circ \sigma = \tau$. □

Lemma 3.3. $\longrightarrow$, $\cdots\!\!\rightarrow$ and $\hookrightarrow$ are preserved by name permutation.

Proof. We prove the claim for $\longrightarrow$; the proofs for $\cdots\!\!\rightarrow$ and $\hookrightarrow$ are similar.

Let σ be a permutation, we have to prove that for each P and Q, if $P \xrightarrow{\tau} Q$, then $P \circ \sigma \xrightarrow{\tau} Q \circ \sigma$.

We proceed by induction on the derivation for $P \xrightarrow{\tau} Q$.

Base of induction: We can distinguish two cases:

— $P = \tau.Q$
— $P = a(b).R|\overline{a}c$ with $Q = R[c/b]$.

In both the cases, for each permutation σ, $P \circ \sigma \xrightarrow{\tau} Q \circ \sigma$.

Induction hypothesis: For each P and Q, if $P \xrightarrow{\tau} Q$ is derivable in i steps, then for each permutation σ, $P \circ \sigma \xrightarrow{\tau} Q \circ \sigma$.

Induction step: Let $P \xrightarrow{\tau} Q$ be derivable in $i+1$ steps. We can distinguish three cases:

— $P = P'|R$, $P' \xrightarrow{\tau} Q'$ and $Q = Q'|R$;
— $P \equiv P'$, $P' \xrightarrow{\tau} Q'$ and $Q' \equiv Q$;
— $P = \nu\, a.P'$ $(\sigma(a) = a)$, $P' \xrightarrow{\tau} Q'$ and $Q = \nu\, a.Q'$;

In these cases, we use the induction hypothesis for each permutation σ to get $P' \circ \sigma \xrightarrow{\tau} Q' \circ \sigma$. Thus the thesis follows easily from the fact that for each R and S:

— $(S|R) \circ \sigma = (S \circ \sigma)|(R \circ \sigma)$;
— $\nu\, a.S \circ \sigma = \nu\, a.S \circ \sigma$ $(\sigma(a) = a)$; and
— if $R \equiv S$, then for each σ, $R \circ \sigma \equiv S \circ \sigma$. □

In Figure 1, we present a portion of the MLTS describing the behaviour of $\textsc{Proxy}_1$ (see Example 3.1), where dashed arrows denote *resource* transitions. Note that only a subset

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

of the possible transitions is shown in the figure; because v could be any of the admissible values and resources could be added at any state.

We now consider the following lemma, which enables us to establish a strong correspondence between the LTS semantics of Amadio *et al.* (1998) and the MLTS semantics presented in this paper.

Lemma 3.4. The following assertions hold:

1 $P \stackrel{\tau}{\longmapsto} P' \Leftrightarrow P \stackrel{\tau}{\longrightarrow} P'$.
2 $P \stackrel{\overline{a}b}{\longmapsto} P' \Leftrightarrow P \stackrel{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P'$.
3 $P \stackrel{\overline{a}(b)}{\longmapsto} P' \Leftrightarrow \exists P''.P \hookrightarrow^{b} P'' \stackrel{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P'$.

Proof.

1 $P \stackrel{\tau}{\longmapsto} P'$ if and only if, either
   — $P \equiv \nu\, \widetilde{a}.(\tau.Q_1|Q_2)$ and $P' = \nu\, \widetilde{a}.Q_1|Q_2$; or
   — $P \equiv \nu\, \widetilde{a}.(\overline{a}b|a(c).Q_1|Q_2)$ and $P' \equiv \nu\, \widetilde{a}.(Q_1[c/b]|Q_2)$.
   But the above holds if and only if $P \stackrel{\tau}{\longrightarrow} P'$.
2 $P \stackrel{\overline{a}b}{\longmapsto} P'$ if and only if $P \equiv \overline{a}b|P'$. Moreover, $P \equiv \overline{a}bP'$ if and only if $P \stackrel{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P'$.
3 $P \stackrel{\overline{a}(b)}{\longmapsto} P'$ if and only if $P \equiv \nu\, b.(\overline{a}b|P')$ and $b \neq a$. However, this holds if and only if $P \hookrightarrow^{b} \overline{a}b|P'$ and $\overline{a}b|P' \stackrel{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P'$. □

4. μKLAIM

KLAIM (De Nicola *et al.* 1998) is a formalism that can be used to model and program mobile systems. It has been designed to provide programmers with primitives for handling the physical distribution, scoping and mobility of processes. KLAIM is based on process algebras but makes use of Linda-like asynchronous communication and models distribution *via* multiple shared tuple spaces (Carriero and Gelernter 1989; Gelernter 1985; Gelernter 1989). Tuple spaces and processes are distributed over different localities and the classical Linda operations are indexed by the location of the tuple space they operate on.

For simplicity, we shall use a simplified version of KLAIM, called μKLAIM (see, for example, Bettini *et al.* (2003)). The main differences between KLAIM and μKLAIM is that the former allows high-level communication (processes can be used as tuple fields) while the latter only permits the remote evaluation of processes. Moreover, the simpler language does not make any distinction between physical and logical localities, and does not need allocation environments.

A μKLAIM system, called a *net*, is a set of *nodes*, each of which is identified by a *locality*. *Localities* can be seen as the addresses of nodes. We shall use LOC to denote the set of localities $l, l_1, \ldots$. Every node has a computational component (a set of processes running in parallel) and a data component (a tuple space). Processes interact with each other either locally or remotely, by inserting and withdrawing tuples from tuple spaces.

A tuple is a sequence of *actual* fields. Each actual field can be either a locality (l), a value v from the (finite) set of *basic values Val* (not specified here), or a *variable* x from

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 4. μKLAIM *syntax*

N	::=		NETS	P	::=		PROCESSES
		$\mathbf{0}$	*empty net*			$\mathbf{nil}$	*Empty process*
	|	$l :: C$	*located component*		|	$act.P$	*Action Prefixing*
	|	$N_1 \parallel N_2$	*net composition*		|	$P \mid P$	*Par. Composition*
	|	$\nu\, l.N$	*name restriction*		|	$P + P$	*Nondet. Choice*
					|	X	*Recursion Variable*
C	::=	$P \mid \langle t \rangle$	COMPONENTS		|	$\mathbf{rec}X.P$	*Recursion*
					|	$\nu\, l.P$	*Name restriction*
act	::=		ACTIONS				
		$\mathbf{out}(t)@l$	*output*	t	::=	$f \mid f, t$	TUPLES
	|	$\mathbf{eval}(P)@l$	*migration*	f	::=	$l \mid v \mid x$	FIELDS
	|	$\mathbf{in}(T)@l$	*input*				
				T	::=	$F \mid F, T$	TEMPLATES
				F	::=	$f \mid !l \mid !x$	TEMP. FIELDS

the set of *variables Var*. Tuples are retrieved from tuple spaces via *pattern matching* using *templates* (T). Templates are sequences of *actual* and *formal* fields. The latter are *variables* that will get a value when a tuple is retrieved, and are distinguished by a '!' before the variable name.

The *pattern-matching* function *match* is defined in Table 5. The meaning of the rules is straightforward: a template matches against a tuple if both have the same number of fields and the corresponding fields match; two values (localities) match only if they are identical, while formal fields match any value of the same type. A successful matching returns a substitution function σ associating the variables contained in the formal fields of the template with the values contained in the corresponding actual fields of the accessed tuple.

The syntax of μKLAIM nets is defined in the first part of Table 4. Term $\mathbf{0}$ denotes the *empty net*, that is, the net that does not contain any node. Terms $l :: P$ (*located process*) and $l :: \langle t \rangle$ (*located tuple*) are used to describe basic μKLAIM nodes: the former states that process P is running at l, and the latter that the tuple space located at l contains tuple $\langle t \rangle$. μKLAIM nets are obtained by parallel composition ($\parallel$) of located processes and tuples. Finally, $\nu\, l.N$ states that l is a private name within N.

The following term denotes a net consisting of two nodes, named l_1 and l_2.

$$l_1 :: P_1 \parallel l_1 :: P_2 \parallel l_2 :: (Q_1 | Q_2) \parallel l_2 :: \langle t_1 \rangle \parallel l_2 :: \langle t_2 \rangle$$

Processes P_1 and P_2 are running at l_1 while Q_1 and Q_2 are running at l_2. The tuple space located at l_2 contains tuples $\langle t_1 \rangle$ and $\langle t_2 \rangle$, while that located at l_1 is empty

The syntax of μKLAIM processes is defined in the second part of Table 4, where: **nil** stands for the process that cannot perform any actions; $P_1 | P_2$ stands for the parallel composition of P_1 and P_2; and $act.P$ stands for the process that executes action act then behaves like P. Possible actions are $\mathbf{out}(t)@l$, $\mathbf{in}(T)@l$ and $\mathbf{eval}(P)@l$.

Action $\mathbf{out}(t)@l$ adds t to the tuple space located at l. Action $\mathbf{eval}(P)@l$ spawns a process P at locality l. Action $\mathbf{in}(T)@l$ is used to retrieve tuples from tuple spaces. Unlike **out**, this is a blocking operation: the computation is blocked until a tuple matching

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 5. *Matching rules*

$$(\mathrm{M}_1)\ \ match(l,l) = [] \qquad (\mathrm{M}_2)\ \ match(!\,l_1, l_2) = \left[l_2/l_1\right]$$

$$(\mathrm{M}_3)\ \ \frac{match(F,l) = \sigma_1 \quad match(T,t) = \sigma_2}{match(\ (F,T)\ ,\ (l,t)\ ) = \sigma_1 \cdot \sigma_2}$$

Table 6. *Structural congruence*

$N_1 \parallel N_2 \equiv N_2 \parallel N_1$	$(N_1 \parallel N_2) \parallel N_3 \equiv N_1 \parallel (N_2 \parallel N_3)$
$l :: C \equiv l :: C \parallel l :: \mathbf{nil}$	$l :: \mathbf{rec}X.P \equiv l :: P[\mathbf{rec}X.P/X]$
$l :: (P_1 \mid P_2) \equiv l :: P_1 \parallel l :: P_2$	$\dfrac{l \notin fn(N_2)}{\nu\, l.(N_1 \parallel N_2) \equiv (\nu\, l.N_1) \parallel N_2}$
$N_1 \parallel \mathbf{0} \equiv N_1$	$\nu\, l.\mathbf{0} \equiv \mathbf{0}$
$\dfrac{N_1 =_\alpha N_2}{N_1 \equiv N_2}$	$\dfrac{l_2 \neq l_1}{l_1 :: \nu\, l_2.P \equiv \nu\, l_2.l_1 :: P}$

template T is found in the tuple space located at l. When such a tuple t is found, it is removed from the tuple space and the continuation process is closed with substitution $\sigma = match(T,t)$, which replaces the formals in T with the corresponding values in t. For instance, if $T = (!u, 4)$ and $et = (l, 4)$, then $match(T,t) = [l/u]$. For this reason, $\mathbf{in}(T)@l.P$ acts as a binder for variables in the formal fields of T. Finally, $\nu\, l.P$ declares a new name l that will be used in P, so $\nu\, l.N$ and $\nu\, l.P$ act as binders for l in N and P, respectively.

In the rest of the paper, we will use $\tilde{l}$ to denote a finite sequence of names. Moreover, if $\tilde{l} = l_1, \ldots, l_n$, we will write $\nu\, \tilde{l}.N$ for $\nu\, l_1 \ldots \nu\, l_n..N$, and use $\{\tilde{l}\}$ to denote the set $\{l_1, \ldots, l_n\}$. We will use $fn(\gamma)$, where γ is a generic syntactic term, to denote the set of free names in γ. Finally, we will write $P_1 =_\alpha P_2$ whenever P_1 is equal to P_2 up to renaming of bound names and variables.

In the rest of this section we will use the *structural congruence* $\equiv$ to identify terms that, intuitively, represent the same net. It is defined as the smallest congruence relation over nets that satisfies the laws in Table 6. The structural laws express the following facts:

— $\parallel$ is commutative and associative;
— α-equivalent processes give rise to equivalent nodes;
— the null process and the empty net can always be safely removed/added;
— a process identifier can be replaced by the body of its definition; and
— it is always possible to transform a parallel of co-located processes into a parallel over nodes.

Note that the commutativity and associativity of '|' is derived from the commutativity and associativity of '$\parallel$' and from the fact that $l :: (P \mid Q) \equiv l :: P \parallel l :: Q$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 7. *A simple* μKLAIM *system*

$$
\begin{array}{rcl}
Printer & :: & \mathbf{rec}X.\mathbf{in}(!f\mathit{rom})@\mathit{Printer}. \\
 & & \quad (X|\mathbf{out}(f\mathit{rom})@\mathit{PrintServer}.\mathbf{nil} \\
 & & \\
\mathit{PrintServer} & :: & \langle \mathit{PrintSlot} \rangle \\
 & | & \langle \mathit{PrintSlot} \rangle \\
 & | & \mathbf{rec}X.\mathbf{in}(\mathit{Print}, !f\mathit{rom})@\mathit{PrintServer}. \\
 & & \quad (X|\mathbf{out}(f\mathit{rom})@\mathit{Printer}. \\
 & & \qquad \mathbf{in}(f\mathit{rom})@\mathit{PrintServer}. \\
 & & \qquad \mathbf{out}(\mathit{PrintOk})@f\mathit{rom}. \\
 & & \qquad \mathbf{out}(\mathit{PrintSlot})@\mathit{PrintServer}.\mathbf{nil})
\end{array}
$$

Example 4.1 (Proxy specification in μKLAIM). The proxy described in Example 3.1 can be specified in μKLAIM as a system consisting of two locations identified by localities l_a and l_b. When a tuple (matching $!l$) is stored in the tuple space located at l_a, this is retrieved and stored in the tuple space located at l_b. Two of the possible μKLAIM implementations of such a system are:

$$\text{KPROXY}_1 = l_a :: \mathbf{rec}X.\mathbf{in}(!l)@l_a.\mathbf{out}(l)@l_b.X \parallel l_b :: \mathbf{nil} \tag{2}$$

$$\text{KPROXY}_2 = l_a :: \mathbf{rec}X.\mathbf{in}(!l)@l_a.\mathbf{eval}(\mathbf{out}(l)@l_b.\mathbf{nil})@l_b.X \parallel l_b :: \mathbf{nil} \tag{3}$$

In KPROXY_1, located at l_a, there is a process that waits for a tuple t matching $!l$ and then stores t at l_b. In KPROXY_2, the process located at l_a withdraws a tuple matching $!l$ and then spawns at l_b a process that adds the retrieved tuples locally.

Example 4.2 (A print server). In Table 7 we show how μKLAIM can be used to model a simple print server with two μKLAIM nodes: *PrintServer* and *Printer*. Located at *PrintServer* there is a process that waits for a print request. Each such request contains the locality from which it has been sent. When a request appears in the tuple space at *PrintServer*, the process sends the document to the printer ($\mathbf{out}(f\mathit{rom})@\mathit{Printer}$), waits for the printing signal ($\mathbf{in}(f\mathit{rom})@\mathit{PrintServer}$) and sends an ack ($\mathbf{out}(\mathit{PrintOk})@f\mathit{rom}$) to the client. To send a print request, a print client has to retrieve a *PrintSlot* from the tuple space located at *PrintServer*, where two *PrintSlot*s are available.

We now present the operational semantics of μKLAIM nets in terms of MLTSs. We will define an MLTS $\mathcal{M}_{\mu K}$ by singling out a set of states describing the configurations that μKLAIM nets can reach; a set of labels describing the actions nets can perform to interact with the environment; and a set of resources used/produced by μKLAIM nets in their computation. We will then define the three transition relations that describe: the actions a net can perform; how a net reacts when resources are created or consumed; and when a net reveals private names. The MLTS-based semantics proposed here is just a different presentation of the one, based on LTS, proposed in De Nicola and Loreti (2005). Indeed, it is easy to establish a correspondence between the transition relations of De Nicola and Loreti (2005) and those proposed here.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 8. $\mathcal{M}_{\mu K}$: *computation relation*

$$l_1 :: \mathbf{out}(t)@l_2.P \xrightarrow{l_1 : t \rhd l_2} l_1 :: P \parallel l_2 :: \langle t \rangle \qquad \frac{\sigma = match(T,t)}{l_1 :: \mathbf{in}(T)@l_2.P \xrightarrow{l_1 : t \lhd l_2} l_1 :: P\{\sigma\}}$$

$$l_1 :: \mathbf{eval}(Q)@l_2.P \xrightarrow{l_1 : Q \blacktriangleright l_2} l_1 :: P \parallel l_2 :: Q$$

$$\frac{N_1 \xrightarrow{l_1 : et \rhd l_2} N_1'}{N_1 \parallel l_2 :: C \xrightarrow{\tau} N_1' \parallel l_2 :: C} \qquad \frac{N_1 \xrightarrow{l_1 : t \lhd l_2} N_1'}{N_1 \parallel l_2 :: \langle t \rangle \xrightarrow{\tau} N_1'}$$

$$\frac{N_1 \xrightarrow{l_1 : Q \blacktriangleright l_2} N_1'}{N_1 \parallel l_2 :: C \xrightarrow{\tau} N_2 \parallel l_2 :: C}$$

$$\frac{N_1 \xrightarrow{\lambda} N_2 \quad \lambda' = \lambda / l}{v\, l.N_1 \xrightarrow{\lambda'} v\, l.N_2} \ \lambda' \text{ is } \textit{relevant} \qquad \frac{N_1 \xrightarrow{\lambda} N_2}{N_1 \parallel N \xrightarrow{\lambda} N_2 \parallel N}$$

The set of states in $\mathcal{M}_{\mu K}$ coincides with the set NET of μKLAIM nets. Transition labels have a complex structure and contain information about the actions performed by located processes. Moreover, the set of resources $\text{RES}_{\mu K}$ will contain locality names and tuples with an indication of their location.

Let $\mathcal{M}_{\mu K}$ be the MLTS defined by

$$\langle \text{NET}, \text{RES}_{\mu K}, \text{LAB}_{\mu K}, \mathbb{N}_{\mu K}, \longrightarrow, \dashrightarrow, \hookrightarrow \rangle$$

where:

— $\text{RES}_{\mu K}$ is the set of resources r defined by the syntax

$$r ::= l \mid \langle t \rangle @ l\,.$$

— $\text{LAB}_{\mu K}$ is the set of transition labels λ defined using the grammar

$$\lambda ::= \tau \mid l_1 : t \rhd l_2 \mid l_1 : t \lhd l_2 \mid l_1 : P \blacktriangleright l_2\,.$$

— $\mathbb{N}_{\mu K} = \langle \text{LOC}, fn, \circ \rangle$.

— $\longrightarrow$, $\dashrightarrow$ and $\hookrightarrow$ are the transition relations, closed with respect to $\equiv$, induced by the rules of Table 8 and Table 9.

Resource $\langle t \rangle @ l$ indicates the presence of tuple t in the tuple space located at l, and l indicates that location l does indeed exist.

Label τ denotes silent transitions while $l_1 : t \rhd l_2$, $l_1 : t \lhd l_2$ and $l_1 : P \blacktriangleright l_2$ denote, respectively, the fact that a process located at l_1:

— inserts tuple t in the tuple space located at l_2;
— withdraws tuple t from the tuple space located at l_2;
— spawns process P to be evaluated at l_2.

The computation relation ($\longrightarrow$) respects the intuitive meaning of μKLAIM operations. A net N_1 reduces to N_1' with label $l_1 : t \rhd l_2$ if there is a process located at l_1 that performs

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 9. $\mathcal{M}_{\mu K}$ *: resource relation and revelation relation*

$$l :: \langle t \rangle \overset{\ominus\langle t\rangle@l}{\cdots\!\!\rightarrow} l :: \mathbf{nil} \qquad l :: C \overset{\ominus l}{\cdots\!\!\rightarrow} l :: C$$

$$\frac{N_1 \overset{\ominus r}{\cdots\!\!\rightarrow} N_1' \quad l \notin r}{\nu\, l.N_1 \overset{\ominus r}{\cdots\!\!\rightarrow} \nu\, l.N_1'} \qquad \frac{N_1 \overset{\ominus r}{\cdots\!\!\rightarrow} N_1'}{N_1 \parallel N_2 \overset{\ominus r}{\longrightarrow} N_1' \parallel N_2}$$

$$N_1 \overset{\oplus\langle t\rangle@l}{\cdots\!\!\rightarrow} N_1 \parallel l :: \langle t \rangle \qquad N_1 \overset{\oplus l}{\cdots\!\!\rightarrow} N_1 \parallel l :: \mathbf{nil}$$

$$\frac{l_1 \neq l_2 \quad l_2 \in t}{\nu\, l_1.(N \parallel l_2 :: \langle t \rangle) \hookrightarrow^{l_1} N \parallel l_2 :: \langle t \rangle}$$

out$(t)@l_2$. The same net N_1 evolves with a τ when it is composed in parallel with node l_2. Similar considerations apply for input and eval.

The rule for restriction $(\nu\, l.N_1)$ relies on name hiding: if λ is a transition label, λ/l denotes the transition label obtained from λ by *hiding* location l, namely by replacing l with a special distinct locality $\star$ denoting the *unknown* location.

Definition 4.3.

1 Let $\star$ be a special distinct name in Loc. We let

$$l_1/l_2 = \begin{cases} \star & \text{if } l_1 = l_2 \\ l_1 & \text{otherwise.} \end{cases}$$

2 λ/l is defined inductively as follows:

— $\tau/l = \tau$

— $(l_1 : t \rhd l_2)/l = (l_1/l) : (t[\star/l]) \rhd (l_2/l)$

— $(l_1 : t \lhd l_2)/l = (l_1/l) : (t[\star/l]) \lhd (l_2/l)$

— $(l_1 : P \blacktriangleright l_2)/l = (l_1/l) : P[\star/l] \blacktriangleright (l_2/l)$.

A transition label is *relevant* if it is τ or it takes effect on a visible/public location. Formally, we have the following definition.

Definition 4.4. A transition λ is *relevant* if and only if:

— $\lambda = \tau$; or

— λ is $l_1 : t \rhd l_2$, $l_1 : t \lhd l_2$ or $l_1 : P \blacktriangleright l_2$ and $l_2 \neq \star$.

If N_1 reduces with λ to N_1', and λ/l is relevant, $\nu\, l.N_1$ reduces with λ/l to $\nu\, l.N_1'$.

The resource relation $(\cdots\!\!\rightarrow)$ is as expected. If in N_1 a tuple space located at l contains tuple t, then N_1 evolves with $\ominus\langle t\rangle@l$ by removing such a tuple. Conversely, N_1 evolves with $\oplus\langle t\rangle@l$ by adding tuple t to the tuple space located at l. Reductions concerning $\ominus l$ and $\oplus l$ are similar. If N_1 contains a node named l, then $N_1 \xrightarrow{\ominus l} N_1$, while $N_1 \xrightarrow{\oplus l} N_1 \parallel l :: \mathbf{nil}$. Note that in the case of $\ominus l$ nothing is removed from N_1: resource l is used without being consumed.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Finally, a net N_1 reveals a private name l if there exists a tuple t containing l located in a *public* tuple space, that is, a tuple space located at a non-private node.

As with the $A\pi$, we now have to prove that $\mathbb{N}_{\mu K}$ is a naming structure for NET, $\text{RES}_{\mu K}$, $\text{LAB}_{\mu K}$ and LOC, and that $\longrightarrow$, $\cdots\!\!\blacktriangleright$ and $\hookrightarrow$ are preserved by name (locality) permutation.

Lemma 4.5. $\mathbb{N}_{\mu K}$ is a naming structure for NET, $\text{LAB}_{\mu K}$, $\text{RES}_{\mu K}$ and LOC.

Lemma 4.6. $\longrightarrow$, $\cdots\!\!\blacktriangleright$ and $\hookrightarrow$ are preserved by name (locality) permutation.

The proofs for the lemmas above follow along the same lines as the proofs of Lemmas 3.2 and 3.3.

5. MoMo: a modal logic for mobility

In this section, we present MoMo logic, which is a modal logic that enables us to specify properties of MLTSs. With MoMo we lay the basis for defining a common logical framework for specifying properties of mobile calculi whose semantics is given in terms of an MLTS. A variant of MoMo that was specifically designed for specifying and verifying properties of KLAIM systems was introduced in De Nicola and Loreti (2005).

MoMo contains four groups of logical operators:

— The *kernel fragment* contains standard first-order logic operators and *recursive formulae* that enable us to describe recursive properties.

The remaining fragments rely on the three transition relations contained in an MLTS.

— *State formulae* are used to assert properties concerning the allocation of resources in a system; their interpretation function will be defined in terms of the relation $\cdots\!\!\blacktriangleright$.

— *Temporal formulae*, which are interpreted by considering the computation relation $\longrightarrow$, describe the action a system can perform to interact with the environment. These are parametrised with respect to a set $\mathcal{A}$ of *label predicates* that will be instantiated for the different calculi.

— The properties of names, such as freshness or revelation, are specified using *nominal formulae*, whose semantics are given using the relation $\hookrightarrow$.

In the rest of this section we describe and motivate each fragment of the logic separately, and discuss formulae satisfaction. Models for MoMo formulae are pairs of the form $\langle \mathcal{M}, \mathbb{A} \rangle$ where $\mathcal{M}$ is an MLTS and $\mathbb{A}$ is a function that maps label predicates to sets of transition labels belonging to $\mathcal{M}^{\mathcal{L}}$. The formal syntax and semantics of MoMo will be defined in Section 5.5.

5.1. *Kernel fragment*

This fragment of the logics is standard. The propositional part of this fragment contains *True* (**T**), *Conjunction* ($\phi_1 \wedge \phi_2$) and *Negation* ($\neg\phi$). The interpretation of this part of the logic is as expected. Every state satisfies *True*, a state satisfies $\phi_1 \wedge \phi_2$ if and only if both ϕ_1 and ϕ_2 are satisfied, and $\neg\phi$ is satisfied by each state that does not satisfy ϕ.

The kernel fragment also contains formulae for specifying recursive properties: *maximal fixed point* and *logical variables*. Intuitively, a state s satisfies $\nu\kappa.\phi$ if s satisfies $\phi[\nu\kappa.\phi/\kappa]$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Formally, the interpretation of $\nu\kappa.\phi$ is defined as the maximal fixed point of the interpretation of ϕ.

If $\mathbb{M}[\![]\!]$ is the interpretation function of the proposed logic (which is formally defined later), the set of nets satisfying $\nu\kappa.\phi$ is defined as the maximal fixed point of the function $\mathscr{F}_\phi : 2^{\mathcal{M}^S} \to 2^{\mathcal{M}^S}$ defined as follows:

$$\mathscr{F}_\phi(\mathcal{S}) \stackrel{\text{def}}{=} \mathbb{M}[\![\phi]\!][\kappa \mapsto \mathcal{S}]$$

where $\mathbb{M}[\![\phi]\!][\kappa \mapsto \mathcal{S}]$ identifies the set of states satisfying ϕ when it is assumed that each state in $\mathcal{S}$ satisfies κ.

To guarantee well-definedness of the interpretation function, it is assumed that in every formula $\nu\kappa.\phi$, the logical variable κ is positive within ϕ (that is, it appears within the scope of an even number of negations). This enables us to define $\mathbb{M}[\![\cdot]\!]$ as the composition of monotonic functions in $2^{\mathcal{M}^S} \to 2^{\mathcal{M}^S}$; since $2^{\mathcal{M}^S}$ is a complete lattice, Tarski's Fixed Point Theorem guarantees the existence of a unique *maximal* fixed point for $\mathbb{M}[\![\phi]\!]$.

5.2. *State properties*

This fragment of the logic is intended to be used for describing properties concerning the availability of resources and a system's reactions to the placement of new resources in a state.

Typical properties one could specify using this fragment of the logic are:

'Two instances of resource ρ are available.' (†)

'Eventually after the insertion of resource ρ_1, resource ρ_2 will be available.' (‡)

An analysis of the resources present in a given state can be described by means of the *consumption* operator $\rho \to \phi$. Intuitively, a state s satisfies $\rho \to \phi$ if resource ρ is available at s and, once it is removed, formula ϕ is satisfied.

By using the consumption operator, property (†) above would be rendered as

$$\rho \to \rho \to \mathbf{T}.$$

Production properties are used to state properties depending on the availability of new resources. These properties are specified by using the *production* operator $\rho \leftarrow \phi$, which is satisfied by every state that satisfies ϕ after a resource ρ is produced. Hence, a state s_1 satisfies $\rho \leftarrow \phi$ if there exists s_2 such that $s_1 \overset{\oplus\ \rho}{\cdots\!\!\blacktriangleright} s_2$ and s_2 satisfies ϕ.

Property (‡) can be formalised in MoMo as

$$\rho_1 \leftarrow \mathit{Eventually}(\rho_2 \to \mathbf{T})$$

where $\mathit{Eventually}(\phi)$ is a derivable operator, which we use as a macro in MoMo, asserting that sooner or later a state satisfying ϕ will be reached.

Note that the production operator is reminiscent of the *separating implication* of *Separation Logic* (Reynolds 2002). This operator allows one to describe the properties of *shared memories* extended with a set of values satisfying a given specification.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

5.3. *Temporal properties*

Temporal properties are specified using *diamond* operator $\langle\cdot\rangle\phi$. This is a variant of the HML diamond operator that, instead of being indexed by a set of transition labels, is indexed by a label predicate α from a given set of *label predicates* $\mathcal{A}$. The interpretation of modal operator $\langle\cdot\rangle\phi$ relies on function $\mathbb{A}$ that, for each label predicate α, yields the set of transition labels satisfying α.

Formally, a state s_1 satisfies $\langle\alpha\rangle\phi$ if and only if there exist s_2 and λ such that $\lambda \in \mathbb{A}(\alpha)$, $s_1 \xrightarrow{\lambda} s_2$ and s_2 satisfies ϕ.

5.4. *Nominal properties*

In this section we introduce operators for dealing with name properties. They will enable us to describe properties of the form *'there exists a name that is* used *in a specific way'* or *'a private name is used accordingly to a given protocol'*. Below, we shall consider four different operators: *quantification*; *revelation*; *name freshness* and *name matching*.

Name quantification ($\exists n.\phi$) enables us to quantify properties with respect to names – a state s satisfies $\exists n.\phi$ if there exists n' such that s satisfies $\phi[n'/n]$.

Name revelation ($n \circledR \phi$) is used to describe properties concerning the disclosure of private names. A state satisfies $n \circledR \phi$ if n can be uncovered (revealed) before reaching a state satisfying ϕ. A state s_1 reveals n if there exists s_2 such that $s_1 \hookrightarrow^n s_2$. Hence, s_1 satisfies $n \circledR \phi$ if and only if there exists s_2 such that $s_1 \hookrightarrow^n s_2$ and s_2 satisfies ϕ.

Name Freshness ($\not|n\phi$) acts as a quantifier over all names that do not occur free either in the formula ϕ or in the state. It is an adaptation of the Gabbay–Pitts quantifier $\not|l\phi$ (Gabbay and Pitts 1999). The fresh name quantifier, when used with state formulae (and, in particular, with the production operator), enables us to assert properties related to the creation of new resources.

Given the MLTS $\mathcal{M}$, where $\mathcal{M}^{\mathbb{N}} = \langle \mathcal{N}, \eta, \circ \rangle$, a state $s \in \mathcal{M}^{\mathcal{S}}$ satisfies $\not|n\phi$ if and only if there exists n' that is not in $\eta(s)$ and does not appear in ϕ such that $\phi[n'/n]$ is satisfied by s.

Name matching ($\{n_1 = n_2\}$) enables us to verify whether two names are equal. Namely, a state satisfies $\{n_1 = n_2\}$ if n_1 and n_2 refer to the same name.

5.5. *Syntax and Semantics of* MoMo

The actual syntax of MoMo is summarised in Table 10 and its interpretation function is formally defined in Table 11.

MoMo formulae are interpreted with respect to a pair $\langle \mathcal{M}, \mathbb{A} \rangle$, where $\mathcal{M}$ is an MLTS and $\mathbb{A}$ is a function that maps label predicates to sets of transition labels. Formulae interpretation function $(\mathbb{M}[\![]\!]^{\langle \mathcal{M}, \mathbb{A} \rangle})$ is parametrised with respect to such a pair. We will omit explicit reference to $\langle \mathcal{M}, \mathbb{A} \rangle$ whenever it is clear from the context.

Formulae are interpeted under a function, called the *logical environment* and denoted by ε, that associates to each logical variable a set of states. We use ε_0 to denote the logical environment such that, for each logical variable κ, $\varepsilon_0(\kappa) = \varnothing$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 10. *The syntax of formulae*

$\phi ::=$	FORMULAE
$\mathbf{T}$	*true*
$\mid \phi \wedge \phi$	*conjunction*
$\mid \neg\phi$	*negation*
$\mid \nu\kappa.\phi$	*maximal fixed point*
$\mid \kappa$	*logical variable*
$\mid \rho \rightarrow \phi$	*consumption*
$\mid \rho \leftarrow \phi$	*production*
$\mid \langle\alpha\rangle\phi_2$	*diamond*
$\mid \exists n.\phi$	*name quantification*
$\mid \{n_1 = n_2\}$	*name matching*
$\mid n \text{ Ⓡ } \phi$	*revelation*
$\mid \diagup\!\!\diagup n\phi$	*fresh name quantification*

Table 11. *The interpretation of formulae*

Kernel Fragment

$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\mathbf{T}]\!]\varepsilon = S$$
$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\neg\phi]\!]\varepsilon = S - \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon$$
$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi_1 \wedge \phi_2]\!]\varepsilon = \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi_1]\!]\varepsilon \cap \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi_2]\!]\varepsilon$$
$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\nu\kappa.\phi]\!]\varepsilon = \nu\mathcal{F}$$
$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\kappa]\!]\varepsilon = \varepsilon(\kappa)$$

where:

$$\mathcal{F}(x) = \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon \cdot [\kappa \mapsto x]\}$$

State Formulae

$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\rho \rightarrow \phi]\!]\varepsilon = \{s_1 | s_1 \overset{\ominus\rho}{\cdots\!\rightarrow} s_2 \text{ and } s_2 \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon\}$$
$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\rho \leftarrow \phi]\!]\varepsilon = \{s_1 | s_1 \overset{\oplus\rho}{\cdots\!\rightarrow} s_2 \text{ and } s_2 \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon\}$$

Temporal Formulae

$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\langle\alpha\rangle\phi]\!]\varepsilon = \left\{ s_1 | \exists s_2 \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon.s_1 \xrightarrow{\lambda} s_2,\ \lambda \in \mathbb{A}(\alpha)) \right\}$$

Nominal Properties

$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\exists n.\phi]\!]\varepsilon = \bigcup_{n' \in \mathcal{N}} \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi[n'/n]]\!]\varepsilon$$
$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![n \text{ Ⓡ } \phi]\!]\varepsilon = \left\{ s_1 | \exists s_2 \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon.s_1 \hookrightarrow^n s_2 \right\}$$
$$\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\diagup\!\!\diagup n\phi]\!]\varepsilon = \{s | \exists n'.n' \notin fn(\phi) \cup \eta(s) \text{ and } s \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi[n'/n]]\!]\varepsilon\}$$

The interpretation function $\mathbb{M}[\![]\!]^{\langle\mathcal{M},\mathbb{A}\rangle}$ takes a formula ϕ and a *logical environment* ε and yields the set of states in $\mathcal{M}$ satisfying ϕ, that is, the set of state that are *models* of ϕ.

We shall use $\mathbb{L}_{\mathcal{A}}$ to denote the set of MoMo formulae that use label predicates belonging to $\mathcal{A}$.

The following lemma guarantees that names can be consistently replaced in nets and formulae while preserving the satisfaction relation.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Table 12. *Derivable operators*

$\mu\kappa.\phi \stackrel{\text{def}}{=} \neg\nu\kappa.\neg\phi[\neg\kappa/\kappa]$	$\phi_1 \vee \phi_2 \stackrel{\text{def}}{=} \neg(\neg\phi_1 \wedge \neg\phi_2)$
$\rho \Rightarrow \phi \stackrel{\text{def}}{=} \neg\rho \rightarrow \neg\phi$	$[\alpha]\,\phi \stackrel{\text{def}}{=} \neg\langle\alpha\rangle\neg\phi$
$\forall n.\phi \stackrel{\text{def}}{=} \neg\exists n.\neg\phi$	$\{n_1 \neq n_2\} \stackrel{\text{def}}{=} \neg\{n_1 = n_2\}$

Lemma 5.1. Let $\mathcal{M}$ be an MLTS, and $\mathcal{M}^{\mathbb{N}} = \langle \mathcal{N}, \eta, \circ\rangle$, for every state $s \in \mathcal{M}^{\mathcal{S}}$, formula ϕ and names n_1 and n_2 in $\mathcal{N}$, we have $s \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon$ if and only if $s \circ \{n_2/n_1\} \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi[n_2/n_1]]\!]$.

Proof. The proof follows easily by induction on the structure of the syntax of formulae thanks to the fact that all the transition relations in $\mathcal{M}$ are preserved under name permutations. □

In the rest of this paper we will use $\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]$ to denote $\mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]\varepsilon_0$, and $s \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi$ will be used to indicate that $s \in \mathbb{M}^{\langle\mathcal{M},\mathbb{A}\rangle}[\![\phi]\!]$.

In Table 12 we present a set of fairly standard derivable operators, which we will use as macros of the logic in the examples in the rest of the paper.

6. MoMo dialects for the analysis of mobile and distributed systems

In this section we present two different dialects of MoMo that eanble us to specify and verify properties of mobile and distributed systems specified using $A\pi$ and μKLAIM. To define each of these dialects we will single out a set of label predicates ($\mathcal{A}$) and an interpretation function ($\mathbb{A}$) that identifies the set of transition labels satisfying a given label predicate.

6.1. MoMo$_{A\pi}$

Since we have only considered a single transition label (τ) in $\mathcal{M}_{A\pi}$, we will just consider a single label predicate $\surd$. Moreover, we shall use the trivial interpretation function $\mathbb{A}_{A\pi}[\![\surd]\!] = \{\tau\}$.

MoMo$_{A\pi}$ syntax can be summarised as follows (propositional and recursive formulae are omitted):

$$\phi \;::=\; \ldots \mid \overline{a}b \rightarrow \phi \mid \overline{a}b \leftarrow \phi \mid \langle\surd\rangle\phi \mid \exists a.\phi \mid a \,®\, \phi \mid \between a\phi\,.$$

The interpretation of these formulae, which is obtained by considering $\mathcal{M}_{A\pi}$ as a model in the definition of Table 11, can be described informally as follows:

— $P \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \overline{a}b \rightarrow \phi$ if and only if $P \equiv \overline{a}b|Q$ and $Q \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \phi$;
— $P \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \overline{a}b \leftarrow \phi$ if and only if $P|\overline{a}b \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \phi$;
— $P \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \langle\surd\rangle\phi$ if and only if $P \stackrel{\tau}{\longrightarrow} Q$ and $Q \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \phi$;
— $P \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \exists a.\phi$ if and only if there exists $b \in fn(P)$ such that $P \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \phi[b/a]$;
— $P \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} a \,®\, \phi$ if and only if $P \equiv \nu\, a.(Q|\overline{b}a)$ $(a \neq b)$ and $Q|\overline{b}a \models_{\langle\mathcal{M}_{A\pi},\mathbb{A}_{A\pi}\rangle} \phi$;

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

— $P \models_{\langle \mathcal{M}_{A\pi}, \mathbb{A}_{A\pi} \rangle} \not|a\phi$ if and only if there exists $b \notin fn(P)$ and $b \notin fn(\phi)$ such that $P \models_{\langle \mathcal{M}_{A\pi}, \mathbb{A}_{A\pi} \rangle} \phi[b/a]$.

We will now show how the proposed logic can be used to specify properties of the system described in Example 3.1. For this system, we may be interested in guaranteeing the following properties:

PROP1 Every message sent over channel a will eventually be emitted over channel b.
PROP2 The order of messages is preserved by the proxy.

The first property can be rephrased as *if we let c be a* new *name, then if c is made available over channel a (by creating resource* $\overline{a}c$*), then sooner or later resource* $\overline{b}c$ *can be consumed*. In the logic, this can be formalised as

$$\phi_1 \stackrel{\text{def}}{=} \not|c\overline{a}c \leftarrow Evn(\overline{b}c \rightarrow \mathbf{true}) \tag{4}$$

where

$$Evn(\phi) \stackrel{\text{def}}{=} \nu\kappa.\phi \vee ([\surd]\kappa \wedge \langle\surd\rangle\mathbf{true})\,.$$

It easy to prove that PROXY_1 satisfies (4).

Property **PROP2** can be rephrased as *if we let c and d be two names, then it is not the case that if d is sent over a after c, then c is emitted over b after d*. This property can be rendered in $\text{MOMO}_{A\pi}$ as follows:

$$\phi_1 \stackrel{\text{def}}{=} \neg\overline{a}c \leftarrow \langle\surd\rangle\overline{a}c \Rightarrow \mathbf{false} \wedge \overline{a}d \leftarrow \phi_2 \tag{5}$$

where

$$\phi_2 \stackrel{\text{def}}{=} \nu\kappa.\overline{b}c \Rightarrow \mathbf{false} \wedge (\overline{b}d \rightarrow \mathbf{true} \vee)\langle\surd\rangle\kappa\,.$$

Now we have that process PROXY_1 does not satisfy formula (5), because there is no synchronisation between two different instances of the replicated process.

Both **PROP1** and **PROP2** would be satisfied by a different implementation of the $A\pi$ proxy:

$$\text{PROXY}_2 \stackrel{\text{def}}{=} \nu\, c.(a(x).\tau.(\overline{b}x|\overline{c}(c)))|(!c(y).a(x).\tau.(\overline{b}x|\overline{c}(y)))$$

where a new istance of the replicated process can start only when the previous one is terminated. This behaviour is guaranteed by the synchronisation over the private name c.

6.2. $\text{MOMO}_{\mu K}$

Let $\mathcal{A}_{\mu K}$ be the set of label predicates α defined as follows:

— $\surd$, identifying τ labels;
— $l_1 : t \trianglelefteq l_2$, identifying the execution of an input;
— $l_1 : t \trianglerighteq l_2$, corresponding to an output action;
— $l_1 : \phi \blacktriangleright\!\underline{\triangleright} l_2$, characterising the remote evaluation of a process.

The interpretation of $\surd$, $l_1 : t \trianglelefteq l_2$ and $l_1 : t \trianglerighteq l_2$ is trivial: they are satisfied by τ, $l_1 : t \triangleleft l_2$ and $l_1 : t \triangleright l_2$, respectively. The interpretation of $l_1 : \phi \blacktriangleright\!\underline{\triangleright} l_2$ is more complicated. Intuitively, a transition label λ satisfies $l_1 : \phi \blacktriangleright\!\underline{\triangleright} l_2$ if $\lambda = l_1 : P \blacktriangleright l_2$ and $l_2 :: P$ satisfies ϕ. Hence,

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

to define $\mathbb{A}_{\mu K}$ (the interpretation of $\mathcal{A}_{\mu K}$), we need a mutual recursion because we make use of $\mathbb{M}^{\langle \mathcal{M}_{\mu K}, \mathbb{A}_{\mu K}\rangle}$ when defining $\mathbb{A}_{\mu K}$. Indeed, the interpretation function of $\mathcal{A}_{\mu K}$ is the greatest lower bound of a chain of interpretation functions that depends on the depth of temporal formulae.

Definition 6.1. Let $depth(\phi)$ and $depth(\alpha)$ be defined inductively as follows:

— $depth(\mathbf{T}) = depth(\kappa) = depth(\{n_1 = n_2\}) = 0$;
— $depth(\neg\phi) = depth(\nu\kappa.\phi) = depth(\rho \rightarrow \phi) = depth(\rho \leftarrow \phi) = depth(\exists n.\phi) = depth(n \,®\, \phi) = depth(|/|n\phi) = depth(\phi)$
— $depth(\langle\alpha\rangle\phi) = \max\{depth(\alpha), depth(\phi)\}$
— $depth(\phi_1 \wedge \phi_2) = \max\{depth(\phi_1), depth(\phi_2)\}$
— $depth(\alpha) = \begin{cases} 1 + depth(\phi) & \text{if } \alpha = l_1 : \phi \blacktriangleright \underline{\ } l_2 \\ 0 & \text{otherwise.} \end{cases}$

Definition 6.2. Let $\mathbb{A}[\![\alpha]\!] = \bigcap_{i\in\mathbb{N}} \mathbb{A}^i_{\mu K}[\![\alpha]\!]$ where for all $i \geqslant 0$:

— $\mathbb{A}^i_{\mu K}[\![\surd]\!] = \{\tau\}$;
— $\mathbb{A}^i_{\mu K}[\![l_1 : t \trianglelefteq l_2]\!] = \{l_1 : t \lhd l_2\}$;
— $\mathbb{A}^i_{\mu K}[\![l_1 : t \trianglerighteq l_2]\!] = \{l_1 : t \rhd l_2\}$;
— $\mathbb{A}^0_{\mu K}[\![l_1 : \phi \blacktriangleright \underline{\ } l_2]\!] = \{l_1 : P \blacktriangleright l_2 | l_2 :: P \in \textsc{Net}\}$;
— $\mathbb{A}^{i+1}_{\mu K}[\![l_1 : \phi \blacktriangleright \underline{\ } l_2]\!] = \begin{cases} \{l_1 : P \blacktriangleright l_2 | l_2 :: P \in \mathbb{M}^{\mathbb{A}^i}[\![\phi]\!]\varepsilon_0\} & \text{if } depth(\phi) \leqslant i \\ \{l_1 : P \blacktriangleright l_2 \mid l_2 :: P \in \textsc{Net}\} & \text{otherwise.} \end{cases}$

To guarantee the well-definedness of $\mathbb{A}$, we have to prove, for each α, that $\{\mathbb{A}^i_{\mu K}[\![\alpha]\!]\}_{i\in\mathbb{N}}$ is a chain in the complete lattice $2^{\textsc{Lab}_{\mu K}}$. Namely, that for each i, we have $\mathbb{A}^{i+1}_{\mu K}[\![\alpha]\!] \subseteq \mathbb{A}^i_{\mu K}[\![\alpha]\!]$.

Lemma 6.3. The following assertions hold:

— For each α and for each i and j such that $depth(\alpha) \leqslant i$ and $depth(\alpha) \leqslant j$, we have

$$\mathbb{A}^i_{\mu K}[\![\alpha]\!] = \mathbb{A}^j_{\mu K}[\![\alpha]\!].$$

— For each ϕ and for each i and j such that $depth(\phi) \leqslant i$ and $depth(\phi) \leqslant j$, we have

$$\mathbb{M}^{\mathbb{A}^i}[\![\phi]\!] = \mathbb{M}^{\mathbb{A}^j}[\![\phi]\!].$$

Proof. The proof is by induction on $depth(\alpha)$ and $depth(\phi)$.

Base of induction: It is easy to prove that if $depth(\alpha) = 0$, then for each i and j

$$\mathbb{A}^i_{\mu K}[\![\alpha]\!] = \mathbb{A}^j_{\mu K}[\![\alpha]\!].$$

Similarly, if $depth(\phi) = 0$, then ϕ does not contain label predicates of the form $l_1 : \psi \blacktriangleright \underline{\ } l_2$. Hence,

$$\mathbb{M}^{\mathbb{A}^i}[\![\phi]\!] = \mathbb{M}^{\mathbb{A}^j}[\![\phi]\!].$$

Induction hypothesis: We assume that:

— For each α, with $depth(\alpha) \leqslant k$, and for each i and j such that $depth(\alpha) \leqslant i$ and $depth(\alpha) \leqslant j$, we have

$$\mathbb{A}^i_{\mu K}[\![\alpha]\!] = \mathbb{A}^j_{\mu K}[\![\alpha]\!].$$

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

— For each ϕ with $depth(\phi) \leqslant k$, and for each i and j such that $depth(\phi) \leqslant i$ and $depth(\phi) \leqslant j$, we have

$$\mathbb{M}^{\mathbb{A}^i}[\![\phi]\!] = \mathbb{M}^{\mathbb{A}^j}[\![\phi]\!].$$

Induction step: Let α be such that $depth(\alpha) = k+1$. Hence, $\alpha = l_1 : \psi \blacktriangleright \underline{\ } l_2$ for some l_1, l_2 and ψ such that $depth(\psi) = k$. For each $i \geqslant k+1$, we have

$$\mathbb{A}^i_{\mu K}[\![\alpha]\!] = \left\{ l_1 : P \blacktriangleright l_2 | l_2 :: P \in \mathbb{M}^{\mathbb{A}^{i-1}}[\![\psi]\!] \right\} = (A).$$

By the induction hypothesis, we have that for each $j \geqslant k+1$,

$$(A) = \left\{ l_1 : P \blacktriangleright l_2 | l_2 :: P \in \mathbb{M}^{\mathbb{A}^{j-1}}[\![\psi]\!] \right\} = \mathbb{A}^j_{\mu K}[\![\alpha]\!].$$

Similarly, let ϕ be such that $depth(\phi) = k+1$. Without lost of generality we let $\phi = \langle\alpha\rangle\psi$ with $depth(\alpha) = k+1$ and $depth(\psi) \leqslant k$. Let $i \geqslant k+1$. We have

$$\mathbb{M}^{\mathbb{A}^i}[\![\langle\alpha\rangle\psi]\!] = \{N | \exists \lambda, N' : N \xrightarrow{\lambda} N', \lambda \in \mathbb{A}^i_{\mu K}[\![\alpha]\!], \text{ and } N' \in \mathbb{M}^{\mathbb{A}^i}[\![\psi]\!]\}.$$

Using the induction hypothesis for each $j \geqslant k$, we have $\mathbb{M}^{\mathbb{A}^j}[\![\psi]\!] = \mathbb{M}^{\mathbb{A}^j}[\![\psi]\!]$. Moreover, we have shown that, if $depth(\alpha) = k+1$, then for each $j \geqslant k+1$ $\mathbb{A}^i_{\mu K}[\![\alpha]\!] = \mathbb{A}^j_{\mu K}[\![\alpha]\!]$. Thus, $\mathbb{M}^{\mathbb{A}^i}[\![\langle\alpha\rangle\psi]\!] = \mathbb{M}^{\mathbb{A}^j}[\![\langle\alpha\rangle\psi]\!]$, and the statement follows. □

Lemma 6.4. For each $i \geqslant 0$ and for each α, we have $\mathbb{A}^{i+1}_{\mu K}[\![\alpha]\!] \subseteq \mathbb{A}^i_{\mu K}[\![\alpha]\!]$.

Proof. We can distinguish three cases:

— If $depth(\alpha) \leqslant i$, then for each i and j, $\mathbb{A}^{i+1}_{\mu K}[\![\alpha]\!] = \mathbb{A}^i_{\mu K}[\![\alpha]\!]$ (Lemma 6.3).

— If $depth(\alpha) = i+1$, then $\alpha = l_1 : \phi \blacktriangleright \underline{\ } l_2$ for some l_1, l_2 and ϕ such that $depth(\phi) = i$. We have:

$$\mathbb{A}^{i+1}_{\mu K}[\![\alpha]\!] = \{l_1 : P \blacktriangleright l_2 | l_2 :: P \in \mathbb{M}^{\mathbb{A}^i}[\![\phi]\!]\varepsilon_0\} \subseteq \{l_1 : P \blacktriangleright l_2 | l_2 :: P \in \textsc{Net}\} = \mathbb{A}^i_{\mu K}[\![\alpha]\!].$$

— If $depth(\alpha) > i+1$, then we have

$$\mathbb{A}^{i+1}_{\mu K}[\![\alpha]\!] = \mathbb{A}^i_{\mu K}[\![\alpha]\!] = \{l_1 : P \blacktriangleright l_2 | l_2 :: P \in \textsc{Net}\}.$$ □

The *state fragment* of MoMo$_{\mu K}$ ($r \to \phi$ and $r \leftarrow \phi$) relates formulae satisfaction to the creation or consumption of resources and enables us to describe the *spatial* properties of a system. On the other hand, the *temporal fragment* ($\langle\alpha\rangle\phi$) enables us to describe properties concerning *interaction protocols* among system components and to *control* properties of agents migrating over the net.

We can use MoMo$_{\mu K}$ to specify properties of the proxy specifications in μKLAIM presented in Example 4.1. As with the $A\pi$ specification, we are interested in guaranteeing the following properties:

PROP1 Every message sent over channel a will eventually be emitted over channel b.

PROP2 The order of messages is preserved by the proxy.

These can be rendered in MoMo$_{\mu K}$ using the following formulae:

$$|\!\not\rightarrow l\langle l\rangle @ l_a \leftarrow Evn(\langle l\rangle @ l_b \to \mathbf{true}) \tag{6}$$

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

where

$$Evn(\phi) \stackrel{\text{def}}{=} \nu\kappa.\phi \vee ([\surd]\,\kappa \wedge \langle\surd\rangle \mathbf{true})$$

$$\neg\langle l_1\rangle@l_a \leftarrow \langle\surd\rangle\langle l_1\rangle@l_a \Rightarrow \mathbf{false} \wedge \langle l_2\rangle@l_a \leftarrow \phi_2 \tag{7}$$

where

$$\phi_2 \stackrel{\text{def}}{=} \nu\kappa.\langle l_1\rangle@l_b \Rightarrow \mathbf{false} \wedge (\langle l_2\rangle@l_b \rightarrow \mathbf{true} \vee)\langle\surd\rangle\kappa\,.$$

It easy to prove that KPROXY$_2$ (see specification (2) in Section 4) satisfies both (6) and (7) while KPROXY$_1$ (see specification (3) in Section 4) satisfies only (6).

Note that the formalisation of properties PROP1 and PROP2 in MOMO$_{\mu K}$ (6 and 7) and MOMO$_{A\pi}$ (4 and 5) coincide apart from the resource specification. Indeed, while in the $A\pi$ specification a resource is a channel with a value ($\overline{a}c$), in μKLAIM it is a located tuple ($\langle t\rangle@l$). We thus have that the properties of a generic proxy can be described in MOMO while ignoring specific system implementations.

MOMO$_{\mu K}$ can also be used for specifying and verifying properties of Example 4.2. For instance, a possible property to establish is that the print server handles all requests. In other words, one could require that if a client retrieves a print slot and sends a document for printing, then sooner or later the document will be printed and the client will be notified. This property can be specified by the following formula:

$$\begin{aligned}
\phi \stackrel{\text{def}}{=}\ & (PrintSlot)@PrintServer \rightarrow && (1)\\
& l \leftarrow && (2)\\
& (l, \text{"}test\text{"})@PrintServer \leftarrow && (3)\\
& Evn(PrintOk@l \rightarrow \mathbf{T}) && (4)
\end{aligned}$$

This can be read as:

1 Let ($PrintSlot$) be a tuple available at $PrintServer$,
2 and let l be the locality of a print client,
3 if (l,'$test$') is inserted at $PrintServer$,
4 then eventually tuple ($PrintOK$) will be at l.

7. Logical and behavioural equivalences

In the previous section we showed how dialects of MOMO logics can be used for specifying and verifying properties of mobile and distributed applications specified using calculi like $A\pi$ and μKLAIM.

An alternative method for proving properties of process calculi is the one based on behavioural equivalences that requires us to provide a concrete and an abstract specification of the behaviour of a given system and then establish that they are 'indistinguishable' under appropriate assumptions. Equivalences also turn out to be important because they would enable us to determine when parts of a system can be *replaced* without changing the behaviour of the whole specification. If one considers the proxy specified in Example 3.1, one could imagine that the τ action, which is performed after a message is retrieved over channel a, models an internal behaviour and the system could be refined with an

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

alternative implementation:

$$\textsc{Proxy}_3 = !v\, c.a(x).\bar{c}x|c(y).\bar{b}y\,.$$

Obviously, formulae satisfaction also induces an equivalence on the interpretation model and two system descriptions are equivalent if (and only if) they satisfy the same set of formulae. A classical result relating modal logic and behavioural equivalence was given in Hennessy and Milner (1985) and shows that the equivalence induced by a very simple modal logic coincides with the so-called bisimulation equivalence.

Correspondence results between differently characterised relations are important in many ways. They help us to gain confidence in the proposed formalisation, enable us to use minimised models when checking formulae satisfaction and help us in studying the properties of the more convenient variant of the equivalent specification.

In this section we shall provide a behavioural characterisation of the logical equivalence induced by MoMo on an MLTS, and show that it coincides with a natural adaptation of the bisimulation relations built on the three transition relations of our model.

The proposed behavioural equivalence is defined over the states of an MLTS. We first define three relations that characterise the bisimulations over the three relations associated with each MLTS.

Definition 7.1. Let $\mathcal{M}$ be an MLTS.

1 A relation $\mathscr{R} \subseteq \mathcal{M}^{\mathcal{S}} \times \mathcal{M}^{\mathcal{S}}$ is a resource preserving bisimulation if and only if $\mathscr{R}$ is symmetric and for each $(s_1, s_2) \in \mathscr{R}$ and $r \in \mathcal{M}^{\mathcal{R}}$, we have:
   — If $s_1 \overset{\oplus r}{\cdots\!\!\rightarrow} s_1'$, then there exists s_2' such that $s_2 \overset{\oplus r}{\cdots\!\!\rightarrow} s_2'$ and $(s_1', s_2') \in \mathscr{R}$.
   — If $s_1 \overset{\ominus r}{\cdots\!\!\rightarrow} s_1'$, then there exists s_2' such that $s_2 \overset{\ominus r}{\cdots\!\!\rightarrow} s_2'$ and $(s_1', s_2') \in \mathscr{R}$.
   We use $\sim_{\mathcal{R}}$ to denote the largest resource preserving bisimulation.
2 A relation $\mathscr{R} \subseteq \mathcal{M}^{\mathcal{S}} \times \mathcal{M}^{\mathcal{S}}$ is a revelation bisimulation if and only if $\mathscr{R}$ is symmetric and for each $(s_1, s_2) \in \mathscr{R}$ and $n \in \mathcal{M}^{\mathcal{N}}$, we have:
   — If $s_1 \hookrightarrow^n s_1'$, then there exists s_2' such that $s_2 \hookrightarrow^n s_2'$ and $(s_1', s_2') \in \mathscr{R}$.
   We use $\sim_{\mathcal{N}}$ to denote the largest revelation bisimulation.
3 A relation $\mathscr{R} \subseteq \mathcal{M}^{\mathcal{S}} \times \mathcal{M}^{\mathcal{S}}$ is an $\mathscr{A}$-parameterised bisimulation whenever $\mathscr{A} \subseteq \mathcal{M}^{\mathcal{L}} \times \mathcal{M}^{\mathcal{L}}$, if and only if $\mathscr{R}$ is symmetric and for each $(s_1, s_2) \in \mathscr{R}$ and $\lambda_1 \in \mathcal{M}^{\mathcal{L}}$, we have:
   — If $s_1 \xrightarrow{\lambda_1} s_1'$, then there exist λ_2 and s_2' such that $s_2 \xrightarrow{\lambda_2} s_2'$, $(\lambda_1, \lambda_2) \in \mathscr{A}$ and $(s_1', s_2') \in \mathscr{R}$;
   We use $\sim_{\mathcal{L}}^{\mathscr{A}}$ to denote the largest $\mathscr{A}$-parameterised bisimulation.

The required bisimulation for an MLTS is obtained as the intersection of the three relations we have just introduced.

Definition 7.2. Let $\mathcal{M}$ be an MLTS and $\mathscr{A} \subseteq \mathcal{M}^{\mathcal{L}} \times \mathcal{M}^{\mathcal{L}}$. We define $\sim^{\mathscr{A}} = \sim_{\mathcal{R}} \cap \sim_{\mathcal{N}} \cap \sim_{\mathcal{L}}^{\mathscr{A}}$.

Let $\mathcal{M}$ be an MLTS and $\mathscr{A}$ be a subset of $\mathcal{M}^{\mathcal{L}} \times \mathcal{M}^{\mathcal{L}}$. We will now show how $\sim^{\mathscr{A}}$ can characterise the behavioural equivalence induced by formulae satisfaction $\mathbb{L}_{\mathcal{A}}$. We will first introduce the following technical lemma.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Lemma 7.3. If $\mathscr{A}_1 \subseteq \mathscr{A}_2$, then for each s_1 and s_2, $s_1 \sim^{\mathscr{A}_1} s_2 \Rightarrow s_1 \sim^{\mathscr{A}_2} s_2$.

Proof. We prove that if $\mathscr{A}_1 \subseteq \mathscr{A}_2$, then $\sim^{\mathscr{A}_1}$ is an $\mathscr{A}_2$-parameterised bisimulation.

Let $s_1 \sim^{\mathscr{A}_1} s_2$. Then, if $s_1 \xrightarrow{\lambda_1} s_1'$, there exists λ_2 and s_2' such that $s_2 \xrightarrow{\lambda_2} s_2'$, $(\lambda_1, \lambda_2) \in \mathscr{A}_1$ and $s_1' \sim^{\mathscr{A}_1} s_2'$. Since $\mathscr{A}_1 \subseteq \mathscr{A}_2$, we have $(\lambda_1, \lambda_2) \in \mathscr{A}_2$ and $\sim^{\mathscr{A}_1}$ is an $\mathscr{A}_2$-parameterised bisimulation and $\sim^{\mathscr{A}_1} \subseteq \sim^{\mathscr{A}_2}$. □

We now show that for each s_1 and s_2, if $s_1 \sim^{\mathscr{A}} s_2$, then s_1 and s_2 satisfy the same set of formulae in $\mathbb{L}_{\mathcal{A}}$, under the assumption that relation $\mathscr{A}$ is *respectful of* interpretation $\mathbb{A}$, where 'respectful of' is defined as follows.

Definition 7.4. Let $\mathscr{A} \subseteq \mathcal{M}^{\mathcal{L}} \times \mathcal{M}^{\mathcal{L}}$, $\mathcal{A}$ be a set of label predicates and $\mathbb{A} : \mathcal{A} \to 2^{\mathcal{M}^{\mathcal{L}}}$ be the interpretation function. We say that

— $\mathscr{A}$ is respectful of $\mathbb{A}$ if and only if $\forall \alpha \in \mathcal{A}$,

$$(\lambda_1, \lambda_2) \in \mathscr{A} \textbf{ if and only if } \lambda_1 \in \mathbb{A}[\![\alpha]\!] \Leftrightarrow \lambda_2 \in \mathbb{A}[\![\alpha]\!]\,.$$

Theorem 7.5. Let $\mathcal{M}$ be an MLTS, $\mathcal{A}$ be a set of label predicates, $\mathbb{A} : \mathcal{A} \to 2^{\mathcal{M}^{\mathcal{L}}}$ be the interpretation function, and $\mathscr{A} \subseteq \mathcal{M}^{\mathcal{L}} \times \mathcal{M}^{\mathcal{L}}$ be respectful of $\mathbb{A}$. Then

$$s_1 \sim^{\mathscr{A}} s_2 \Longrightarrow \forall \phi \in \mathbb{L}_{\mathcal{A}}.s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \text{ iff } s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi\,.$$

Proof. We prove the result for the recursion-free fragment of the logic, denoted by $\mathbb{L}^{-}_{\mathcal{A}}$. Indeed, it is easy to prove that

$$\forall \phi \in \mathbb{L}_{\mathcal{A}}.s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \Leftrightarrow s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$$

if and only if

$$\forall \phi \in \mathbb{L}^{-}_{\mathcal{A}}.s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \Leftrightarrow s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi\,.$$

Let:

— $\mathscr{R}_{\mathbb{L}} = \{(s_1, s_2) | \forall \phi \in \mathbb{L}^{-}_{\mathcal{A}}.s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \Leftrightarrow s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle}\}$;
— $size(\phi)$ be defined inductively as follows:

- $size(\mathbf{T}) = 0$
- $size(\phi_1 \wedge \phi_2) = 1 + \max\{size(\phi_1), size(\phi_2)\}$
- $size(\neg\phi) = 1 + size(\phi)$
- $size(\rho \to \phi) = 1 + size(\phi)$
- $size(\rho \leftarrow \phi) = 1 + size(\phi)$
- $size(\langle\alpha\rangle\phi) = 1 + size(\phi)$
- $size(\exists n.\phi) = 1 + size(\phi)$
- $size(\{n_1 = n_2\}) = 0$
- $size(n \circledR \phi) = 1 + size(\phi)$
- $size(\not|n\phi) = 1 + size(\phi)$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

We will prove by induction on $size(\phi)$ that for each $\phi \in \mathbb{L}^{-}_{\mathcal{A}}$, if $s_1 \sim^{\mathscr{A}} s_2$, then $s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$ if and only if $s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$.

Base of induction: We have to prove that for each ϕ such that $size(\phi) = 0$, if $s_1 \sim^{\mathscr{A}} s_2$, then $s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$ if and only if $s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$. We can distinguish two cases:

— $\phi = \mathbf{T}$;

— $\phi = \{n_1 = n_2\}$.

In both cases the statement follows trivially.

Induction hypothesis: We assume that for each ϕ such that $size(\phi) \leqslant n$, if $s_1 \sim^{\mathscr{A}} s_2$, then $s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$ if and only if $s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$.

Induction step: Let ϕ be such that $size(\phi) = n + 1$. We prove that if $s_1 \sim^{\mathscr{A}} s_2$, then $s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$ if and only if $s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi$. We can distinguish the following cases according to the syntax of ϕ:

— $\phi = \phi_1 \wedge \phi_2$:

$$\begin{aligned} s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi_1 \wedge \phi_2 &\Leftrightarrow s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi_1 \text{ and } s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi_2 \\ &\Leftrightarrow s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi_1 \text{ and } s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi_2 \\ &\qquad \text{for the induction hypothesis} \\ &\Leftrightarrow s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi_1 \wedge \phi_2 . \end{aligned}$$

— $\phi = \rho \rightarrow \phi$:

$$\begin{aligned} s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \rho \rightarrow \phi &\Leftrightarrow \exists s'_1 . s_1 \overset{\ominus \rho}{\cdots\cdots\rightarrow} s'_1 \text{ and } s'_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \\ &\Leftrightarrow \exists s'_2 . s_2 \overset{\ominus \rho}{\cdots\cdots\rightarrow} s'_2 \text{ and } s'_1 \sim^{\mathscr{A}} s'_2 \\ &\qquad \text{since } \sim^{\mathscr{A}} \text{ is a resource preserving bisimulation} \\ &\Leftrightarrow s'_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \\ &\qquad\qquad \text{for the induction hypothesis} \\ &\Leftrightarrow s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \rho \rightarrow \phi . \end{aligned}$$

— $\phi = \rho \leftarrow \phi$:

$$\begin{aligned} s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \rho \leftarrow \phi &\Leftrightarrow \exists s'_1 . s_1 \overset{\oplus \rho}{\cdots\cdots\rightarrow} s'_1 \text{ and } s'_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \\ &\Leftrightarrow \exists s'_2 . s_2 \overset{\oplus \rho}{\cdots\cdots\rightarrow} s'_2 \text{ and } s'_1 \sim^{\mathscr{A}} s'_2 \\ &\qquad \text{since } \sim^{\mathscr{A}} \text{ is a resource preserving bisimulation} \\ &\Leftrightarrow s'_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \\ &\qquad\qquad \text{for the induction hypothesis} \\ &\Leftrightarrow s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \rho \leftarrow \phi . \end{aligned}$$

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

— $\phi = \langle\alpha\rangle\phi$:

$$\begin{aligned} s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \langle\alpha\rangle\phi &\Leftrightarrow \exists s_1'.s_1 \xrightarrow{\lambda_1} s_1', \lambda \in \mathbb{A}[\![\alpha]\!] \\ &\quad \text{and } s_1' \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi \\ &\Leftrightarrow \exists \lambda_2, s_2'.s_2 \xrightarrow{\lambda_2} s_2', \lambda_2 \in \mathbb{A}[\![\alpha]\!] \\ &\quad \text{and } s_1' \sim^{\mathscr{A}} s_2' \end{aligned}$$

since $\sim^{\mathscr{A}}$ is an $\mathscr{A}$-parametrised bisimulation and $\mathscr{A}$ is respectful of $\mathbb{A}$

$$\begin{aligned} &\Leftrightarrow s_2' \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi \\ &\qquad\qquad \text{for the induction hypothesis} \\ &\Leftrightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \langle\alpha\rangle\phi\,. \end{aligned}$$

— $\phi = \exists n.\phi$:

$$\begin{aligned} s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \exists n.\phi &\Rightarrow s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi[n'/n] \quad \text{for some } n' \in \mathcal{M}^{\mathcal{N}} \\ &\Rightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi[n'/n] \quad \text{for the induction hypothesis} \\ &\Rightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \exists n.\phi\,. \end{aligned}$$

— $\phi = n \,®\, \phi$:

$$\begin{aligned} s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} n \,®\, \phi &\Leftrightarrow s_1 \hookrightarrow^n s_1' \text{ and } s_1' \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi \\ &\Leftrightarrow s_2 \hookrightarrow^n s_2' \text{ and } s_1' \sim^{\mathscr{A}} s_2' \\ &\qquad\qquad \text{since } s_1' \sim^{\mathscr{A}} s_2' \text{ is a revelation bisimulation} \\ &\Leftrightarrow s_2' \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi \qquad\qquad \text{for the induction hypothesis} \\ &\Leftrightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle} n \,®\, \phi\,. \end{aligned}$$

— $\phi = \not|n\phi$:

$$\begin{aligned} s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \not|n\phi &\Leftrightarrow s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi[n'/n] \quad \text{for some } n' \notin \eta_{\mathcal{S}}(s_1) \\ &\Leftrightarrow s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi[n''/n] \quad \text{for some } n'' \notin \eta_{\mathcal{S}}(s_1) \sup \eta_{\mathcal{S}}(s_2) \\ &\qquad\qquad \text{and } n'' \text{ not occurring in } \phi \text{ (see Lemma 5.1)} \\ &\Leftrightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi[n''/n] \quad \text{for the induction hypothesis} \\ &\Leftrightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \not|n\phi\,. \end{aligned}$$

— $\phi = \neg\phi$:

$$\begin{aligned} s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \neg\phi &\Leftrightarrow s_1 \not\models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi \\ &\Leftrightarrow s_2 \not\models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi \qquad \text{for the induction hypothesis} \\ &\Leftrightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \neg\phi\,. \end{aligned}$$

□

Let $\mathscr{R}_{\mathbb{L}} \subseteq \mathcal{M}^{\mathcal{S}} \times \mathcal{M}^{\mathcal{S}}$ be the symmetric relation defined by

$$\mathscr{R}_{\mathbb{L}} = \{(s_1, s_2) | \forall \phi \in \mathbb{L}_{\mathcal{A}}^{-}.s_1 \models_{\langle\mathcal{M},\mathbb{A}\rangle} \phi \Leftrightarrow s_2 \models_{\langle\mathcal{M},\mathbb{A}\rangle}\}\,.$$

We want to prove that $\mathscr{R}_{\mathbb{L}}$ is a subset of $\sim^{\mathscr{A}}$.

First, we have to guarantee that $\mathcal{M}$ is *tractable*, namely that for each state in $\mathcal{M}^{\mathcal{S}}$, only a finite set of next states has to be considered.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Definition 7.6. An MLTS $\mathcal{M}$ is *image-finite* with respect to the interpretation function $\mathbb{A} : \mathcal{A} \to 2^{\mathcal{M}^{\mathcal{L}}}$ if and only if for each state s_1, resource r, name n and label predicate α, the following sets are finite:

— $\{s_2 | s_1 \overset{\oplus r}{\cdots\!\rightarrow} s_2\}$
— $\{s_2 | s_1 \overset{\ominus r}{\cdots\!\rightarrow} s_2\}$
— $\{s_2 | s_1 \hookrightarrow^n s_2\}$
— $\{s_2 | \exists \lambda \in \mathbb{A}[\![\alpha]\!].s_1 \xrightarrow{\lambda} s_2\}$.

Moreover, we also need $\mathbb{A}$ and $\mathscr{A}$ to *discriminate* between the different transitions performed at each state.

Definition 7.7. Let $\mathcal{M}$ be an MLTS, $\mathscr{A} \subseteq \mathcal{M}^{\mathcal{L}} \times \mathcal{M}^{\mathcal{L}}$, $\mathcal{A}$ be a set of label predicates and $\mathbb{A} : \mathcal{A} \to 2^{\mathcal{M}^{\mathcal{L}}}$ be the interpretation function. We say that $\mathscr{A}$ and $\mathbb{A}$ are *transition respectful* in $\mathcal{M}$ if and only if for each $s \in \mathcal{M}^{\mathcal{S}}$ and $\lambda \in \mathcal{M}^{\mathcal{L}}$ there exists α in $\mathcal{A}$ such that:

— $\lambda \in \mathbb{A}[\![\alpha]\!]$
— for each λ' such that $s \xrightarrow{\lambda'}$,

$$(\lambda, \lambda') \notin \mathscr{A} \Leftarrow \lambda' \notin \mathbb{A}[\![\alpha]\!]$$

where $s \xrightarrow{\lambda'}$ if there exists s' such that $s \xrightarrow{\lambda'} s'$.

The following theorem enables us to guarantee that if s_1 and s_2 satisfy the same set of formulae, then s_1 is behaviourally equivalent to s_2.

Theorem 7.8. Let $\mathcal{M}$ be an MLTS, $\mathcal{A}$ be a set of label predicates, $\mathbb{A} : \mathcal{A} \to 2^{\mathcal{M}^{\mathcal{L}}}$ be the interpretation function and $\mathscr{A} \subseteq \mathcal{M}^{\mathcal{L}} \times \mathcal{M}^{\mathcal{L}}$ such that:

1 $\mathscr{A}$ is respectful of $\mathbb{A}$ in $\mathcal{M}$;
2 $\mathscr{A}$ and $\mathbb{A}$ are transition respectful in $\mathcal{M}$;
3 $\mathcal{M}$ is *image-finite* with $\mathcal{A}$ and $\mathbb{A}$.

Then,

$$s_1 \sim^{\mathscr{A}} s_2 \Leftarrow\!= \forall \phi \in \mathbb{L}_{\mathcal{A}}.s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi \text{ iff } s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi\,.$$

Proof. We will prove that $\mathscr{R}_{\mathbb{L}}$ is a resource preserving bisimulation, a revelation bisimulation and an $\mathscr{A}$-parameterised bisimulation.

Since $\sim_{\mathcal{R}}$, $\sim_{\mathcal{N}}$ and $\sim_{\mathcal{L}}^{\mathscr{A}}$ are the the largest resource preserving bisimulations, revelation bisimulations and $\mathscr{A}$-parameterised bisimulations, respectively, it follows that

$$\mathscr{R}_{\mathbb{L}} \subseteq \sim_{\mathcal{R}} \cap \sim_{\mathcal{N}} \cap \sim_{\mathcal{L}}^{\mathscr{A}} = \sim^{\mathscr{A}}.$$

We now give the formal proof that $\mathscr{R}_{\mathbb{L}}$ is an $\mathscr{A}$-parameterised bisimulation: we will omit the similar proofs that $\mathscr{R}_{\mathbb{L}}$ is a resource preserving bisimulation.

Let $(s_1, s_2) \in \mathscr{R}_{\mathbb{L}}$ and $s_1 \xrightarrow{\lambda_1} s_1'$ for some λ_1. We have to prove that there exists λ_2 and s_2' such that $(\lambda_1, \lambda_2) \in \mathscr{A}$, $s_2 \xrightarrow{\lambda_2} s_2'$ and $(s_1', s_2') \in \mathscr{R}_{\mathbb{L}}$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

We suppose that for each λ_2 and s_2' such that $(\lambda_1, \lambda_2) \in \mathscr{A}$, we have $(s_1', s_2') \notin \mathscr{R}_{\mathbb{L}}$. Let

$$X = \left\{ s_2' | \exists \lambda_2 . (\lambda_1, \lambda_2) \in \mathscr{A} \text{ and } s_2 \xrightarrow{\lambda_2} s_2' \right\}.$$

Since $\mathcal{M}$ is *image-finite*, X is finite. For each $s \in X$ there exists ϕ_s such that $s_1' \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \phi_s$ while $s \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \neg \phi_s$. If X is non-empty, we let

$$\phi_X = \bigwedge_{s \in X} \phi_s .$$

And if X is empty, we let $\phi_X = \mathbf{T}$.

$\mathscr{A}$ and $\mathbb{A}$ are transition respectful in $\mathcal{M}$. Hence, there exists α such that:

— $\lambda_1 \in \mathbb{A}[\![\alpha]\!]$; and

— for each λ such that $s \xrightarrow{\lambda}$, if $(\lambda_1, \lambda) \notin \mathscr{A}$, then $\lambda \notin \mathbb{A}[\![\alpha]\!]$.

We have that $s_1 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \langle \alpha \rangle \phi_X$ while $s_2 \models_{\langle \mathcal{M}, \mathbb{A} \rangle} \neg \langle \alpha \rangle \phi_X$. Hence, $(s_1, s_2) \notin \mathscr{R}_{\mathbb{L}}$, which contradicts the assumptions, so the statement is proved. □

7.1. *Logical characterisation of bisimulation for Aπ*

In this section we will investigate the equivalence induced by the satisfaction of formulae in $\mathbb{L}_{A\pi}$ and their relationships with another well-studied behavioural equivalence between $A\pi$ processes, *viz*. the asynchronous bisimulation presented in Amadio *et al.* (1998). First we note the following result.

Lemma 7.9.

— $\mathcal{M}_{A\pi}$ is *image-finite* with respect to $\mathcal{A}_{A\pi}$ and $\mathbb{A}_{A\pi}$.

— $\mathscr{A}_{A\pi} = \{(\tau, \tau)\}$ and $\mathbb{A}_{A\pi}$ are *transition respectful* in $\mathcal{M}_{A\pi}$.

*Proof. Image-finite*ness of $\mathcal{M}_{A\pi}$ with respect to $\mathcal{A}_{A\pi}$ and $\mathbb{A}_{A\pi}$ follows easily from the following:

— For each P and $\overline{a}b$, we have $\{P' | P \overset{\oplus \overline{a}b}{\cdots\cdots\rightarrow} P'\} = \{P | \overline{a}b\}$.

— For each P and $\overline{a}b$, we have $\{P' | P \overset{\ominus \overline{a}b}{\cdots\cdots\rightarrow} P'\}$ is $\{Q\}$ if $P \equiv Q \parallel \overline{a}b$, otherwise it is the empty set.

— For each P and a, we have $\{P' | P \hookrightarrow^a P'\} = \{P' | \exists P'' . P \equiv \nu\, a . P' \text{ and } P' \equiv P'' | \overline{a}b\}$, which is finite since in P only a finite number of bound names can occur.

— For each P and α, we have $\{P' | \exists \lambda . \lambda \in \mathbb{A}[\![\alpha]\!] \text{ and } P \xrightarrow{\lambda} P'\}$ is equal to $\{P' | P \xrightarrow{\tau} P'\}$, which is finite because in P only a finite number of synchronisations can occur.

Moreover, $\mathscr{A}_{A\pi} = \{(\tau, \tau)\}$. Hence, for each P and λ_1, we have $\lambda_1 \in \mathbb{A}_{A\pi}[\![\surd]\!]$, while for each $\lambda_2 \in \text{Lab}_{A\pi}$, we have $(\lambda_1, \lambda_2) \in \mathscr{A}_{A\pi}$. Hence, $\mathscr{A}_{A\pi}$ and $\mathbb{A}_{A\pi}$ are transition respectful in $\mathcal{M}_{A\pi}$. □

Moreover, we will show that the equivalence $\sim^{\{(\tau,\tau)\}}$ defined over $A\pi$ processes coincides with the asynchronous bisimulation ($\sim_a$) defined in Amadio *et al.* (1998). This means that $\mathbb{L}_{\mathcal{A}_{A\pi}}$ characterises asynchronous bisimulation.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Definition 7.10. A symmetric relation $\mathscr{R}$ on $A\pi$ terms is a strong $o\tau$-bisimulation if $P\mathscr{R}Q$, $P \overset{\alpha}{\longmapsto} P'$, α is not an input action, and $bn(\alpha) \cap fn(Q) = \varnothing$ implies $Q \overset{\alpha}{\longmapsto} Q'$ and $P'\mathscr{R}Q'$. Let $\sim_{o\tau}$ be the largest $o\tau$-bisimulation.

Definition 7.11. A relation $\mathscr{R}$ is an asynchronous bisimulation if it is an $o\tau$-bisimulation and whenever $P\mathscr{R}Q$ and $P \overset{ab}{\longmapsto} P'$, we have either:

— $Q \overset{ab}{\longmapsto} Q'$ and $P'\mathscr{R}Q'$; or
— $Q \overset{\tau}{\longmapsto} Q'$ and $P'\mathscr{R}(Q'|\overline{a}b)$.

Definition 7.12. $\sim_a$ is the largest asynchronous bisimulation.

Definition 7.13. Let $\sim_1$ be the largest relation $\mathscr{R}$ such that $\mathscr{R}$ is an $o\tau$-bisimulation and $P\mathscr{R}Q$ implies $(P|\overline{a}b)\mathscr{R}(Q|\overline{a}b)$ for all $\overline{a}b$.

Theorem 7.14. $\sim_a = \sim_1$

Proof. See Amadio *et al.* (1998). □

Let $P = !\tau.\mathbf{0}$ and $Q = !\tau.\mathbf{0}|a(b).\overline{a}b$. We have that $P \sim_a Q$. Note that these processes can be distinguished using the modal logics defined in Milner *et al.* (1993), Dam (1996) and Caires (2004).

Theorem 7.15. $\sim^{\{(\tau,\tau)\}} \subseteq \sim_a$

Proof. We first prove that $\sim^{\{(\tau,\tau)\}}$ is an $o\tau$-bisimulation. We have to show that if $P\mathscr{R}Q$, then $P \overset{\alpha}{\longmapsto} P'$, α is not an input action, and $bn(\alpha) \cap fn(Q) = \varnothing$ implies $Q \overset{\alpha}{\longmapsto} Q'$ and $P'\mathscr{R}Q'$.

Thanks to Lemma 3.4, it easy to prove that if $P \sim^{\{(\tau,\tau)\}} Q$, then:

— $P \xrightarrow{\tau} P' \Rightarrow \exists Q'.Q \xrightarrow{\tau} Q'$ and $P' \sim^{\{(\tau,\tau)\}} Q'$.
— $P \overset{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P' \Rightarrow \exists Q'.Q \overset{\ominus\overline{a}b}{\cdots\!\!\rightarrow} Q'$ and $P' \sim^{\{(\tau,\tau)\}} Q'$.
— $P \hookrightarrow^b P'' \overset{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P' \Rightarrow \exists Q', Q''.Q \hookrightarrow^b Q'' \overset{\ominus\overline{a}b}{\cdots\!\!\rightarrow} Q'$ and $P'' \sim^{\{(\tau,\tau)\}} Q''$, $P' \sim^{\{(\tau,\tau)\}} Q'$.

We now have to prove that for each $\overline{a}b$, if $P \sim^{\{(\tau,\tau)\}} Q$, then $P|\overline{a}b \sim^{\{(\tau,\tau)\}} Q|\overline{a}b$. For each $\overline{a}b$, we have $P \overset{\oplus\overline{a}b}{\cdots\!\!\rightarrow} P|\overline{a}b$ and $Q \overset{\oplus\overline{a}b}{\cdots\!\!\rightarrow} Q|\overline{a}b$. Since $P \sim^{\{(\tau,\tau)\}} Q$, it follows that $P|\overline{a}b \sim^{\{(\tau,\tau)\}} Q|\overline{a}b$ and that $\sim^{\{(\tau,\tau)\}}$ is an asynchronous bisimulation.

Since $\sim_a$ is the largest asynchronous bisimulation, this implies $\sim^{\{(\tau,\tau)\}} \subseteq \sim_a$. □

Theorem 7.16. $\sim_a \subseteq \sim^{\{(\tau,\tau)\}}$

Proof. We prove that $\sim_a$ is a resource preserving bisimulation, a revelation bisimulation and a $\{(\tau,\tau)\}$-parameterised bisimulation. We have to show that if $P \sim_a Q$, then:

1 If $P \overset{\oplus\overline{a}b}{\cdots\!\!\rightarrow} P'$, then there exists Q' such that $Q \overset{\oplus\overline{a}b}{\cdots\!\!\rightarrow} Q'$ and $P' \sim_a Q'$.
2 If $P \overset{\ominus\overline{a}b}{\cdots\!\!\rightarrow} P'$, then there exists Q' such that $Q \overset{\ominus\overline{a}b}{\cdots\!\!\rightarrow} Q'$ and $P' \sim_a Q'$.
3 If $P \hookrightarrow^a P'$, then there exists Q' such that $Q \hookrightarrow^a Q'$ and $P' \sim_a Q'$.
4 If $P \xrightarrow{\tau} P'$, then there exists Q' such that $Q \xrightarrow{\tau} Q'$ and $P' \sim_a Q'$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

The above follow directly from the definition of $\sim_a$ by noting that:

1 If $P \overset{\oplus \overline{a}b}{\cdots\cdots\rightarrow} P'$, then $P' \equiv P|\overline{a}b$.
2 If $P \overset{\ominus \overline{a}b}{\cdots\cdots\rightarrow} P'$, then $P \overset{\overline{a}b}{\longmapsto} P'$.
3 If $P \hookrightarrow^b P'$, then $P' = \overline{a}b|P''$ $(a \neq b)$ and $P \overset{\overline{a}(b)}{\longmapsto} P''$.
4 If $P \overset{\tau}{\longrightarrow} P'$, then $P \overset{\tau}{\longmapsto} P'$. □

7.2. *A bisimulation for* $\mathcal{M}_{\mu K}$

In this section we introduce an equivalence between μKLAIM transition labels ($\mathscr{A}_{\mu K}$). This will be used to define a bisimulation for $\mathcal{M}_{\mu K}$ ($\sim^{\mathscr{A}_{\mu K}}$) that agrees with the equivalence induced by the satisfaction of formulae in $\mathbb{L}_{\mu K}$.

We will define $\mathscr{A}_{\mu K}$ as the *inf* of a chain of equivalences ($\mathscr{A}^i_{\mu K}$), each of which is respectful of the corresponding interpretation $\mathbb{A}^i$ (see Definition 6.2).

Definition 7.17. Let $\mathscr{A}^i_{\mu K}$ be the subset of $\text{LAB}_{\mu K} \times \text{LAB}_{\mu K}$ defined inductively as follows:

— $(\lambda_1, \lambda_2) \in \mathscr{A}^0_{\mu K}$ if and only if either:
  – $\lambda_1 = \lambda_2$; or
  – $\lambda_1 = l_1 : P \blacktriangleright l_2$ and $\lambda_2 = l_1 : Q \blacktriangleright l_2$ for some l_1, l_2, P and Q.

— $(\lambda_1, \lambda_2) \in \mathscr{A}^{i+1}_{\mu K}$ if and only if either:
  – $\lambda_1 = \lambda_2$; or
  – $\lambda_1 = l_1 : P \blacktriangleright l_2$ and $\lambda_2 = l_1 : Q \blacktriangleright l_2$ for some l_1, l_2, P and Q, and $l_2 :: P \sim^{\mathscr{A}^i_{\mu K}} l_2 :: Q$.

We will now show that each $\mathscr{A}^i_{\mu K}$ is respectful of $\mathbb{A}^i$ (see Definition 6.2), and that $\mathscr{A}^i_{\mu K}$ and $\mathbb{A}^i$ are transition respectful in $\mathcal{M}_{\mu K}$. This allows us to prove (using Theorems 7.5 and 7.8) that for each i:

$$N_1 \sim^{\mathscr{A}^i_{\mu K}} N_2 \text{ if and only if } N_1 \models_{\mathbb{A}^i} \phi \Leftrightarrow N_2 \models_{\mathbb{A}^i} \phi\,.$$

Indeed, the following Lemma holds.

Lemma 7.18. $\mathcal{M}_{\mu K}$ is *image-finite* with respect to $\mathbb{A}_{\mu K}$.

Proof. It is easy to prove that for each N_1:

— $\{N_2 | N_1 \overset{\oplus r}{\cdots\cdots\rightarrow} N_2\}$
— $\{N_2 | N_1 \overset{\ominus r}{\cdots\cdots\rightarrow} N_2\}$
— $\{N_2 | N_1 \hookrightarrow^n N_2\}$
— $\{N_2 | \exists \lambda \in \mathbb{A}[\![\alpha]\!].N_1 \overset{\lambda}{\longrightarrow} N_2\}$. □

Lemma 7.19. For each i:

— $\mathscr{A}^i_{\mu K}$ is respectful of $\mathbb{A}^i$.
— $\mathscr{A}^i_{\mu K}$ and $\mathbb{A}^i$ are transition respectful in $\mathcal{M}_{\mu K}$.

Proof. The proof is by induction on i.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Base of induction (*i*=0): If $(\lambda_1, \lambda_2) \in \mathscr{A}^0_{\mu K}$, then either $\lambda_1 = \lambda_2 \in \{\tau, l_1 : t \rhd l_2, l_1 : t \lhd l_2\}$ (for some l_1, l_2 and t), or $\lambda_1 = l_1 : P \blacktriangleright l_2$ and $\lambda_2 = l_1 : Q \blacktriangleright l_2$ (for some l_1, l_2, P and Q).
In the first case we let α be $\surd$, $l_1 : t \underline{\rhd} l_2$ or $l_1 : t \underline{\lhd} l_2$, respectively: $\mathbb{A}^0_{\mu K}[\![\alpha]\!] = \{\lambda_1\}$ and for each $\alpha' \neq \alpha$, $\lambda_1 \notin \mathbb{A}^0_{\mu K}[\![\alpha']\!]$.
In the second case we have that for each ϕ, $\lambda_1, \lambda_2 \in \mathbb{A}^0_{\mu K}[\![l_1 : \phi \underline{\blacktriangleright} l_2]\!]$, while neither λ_1 nor λ_2 belongs to $\mathbb{A}^0_{\mu K}[\![\alpha]\!]$ if α is $\surd$, $l_1 : t \underline{\rhd} l_2$ or $l_1 : t \underline{\lhd} l_2$.
Hence, $(\lambda_1, \lambda_2) \in \mathscr{A}^0_{\mu K}$ if and only if for each $\alpha \in \mathcal{A}_{\mu K}$ we have $\lambda_1 \in \mathbb{A}^0_{\mu K}[\![\alpha]\!] \Leftrightarrow \lambda_1 \in \mathbb{A}^0_{\mu K}[\![\alpha]\!]$.
We now have to prove that for each N_1 and λ_1 there exists α such that:

— $\lambda_1 \in \mathbb{A}^0_{\mu K}[\![\alpha]\!]$; and

— for each λ such that $N_1 \xrightarrow{\lambda}$, if $(\lambda_1, \lambda) \notin \mathscr{A}^0_{\mu K}$, then $\lambda_2 \notin \mathbb{A}^0_{\mu K}[\![\alpha]\!]$.

The statement follows easily by considering $\alpha = \surd$ if $\lambda_1 = \tau$, $\alpha = l_1 : t \underline{\rhd} l_2$ if $\lambda_1 = l_1 : t \rhd l_2$, $\alpha = l_1 : t \underline{\lhd} l_2$ if $\lambda_1 = l_1 : t \lhd l_2$ and $\alpha = l_1 : \mathbf{T} \underline{\blacktriangleright} l_2$ if $\lambda_1 = l_1 : t \lhd l_2$.

Induction hypothesis: For each $i \leqslant n$:

— $\mathscr{A}^i_{\mu K}$ is respectful of $\mathbb{A}^i$.

— $\mathscr{A}^i_{\mu K}$ and $\mathbb{A}^i$ are transition respectful in $\mathcal{M}_{\mu K}$.

Induction step: Let $(\lambda_1, \lambda_2) \in \mathscr{A}^{n+1}_{\mu K}$. We have that either $\lambda_1 = \lambda_2$ and there exists α such that $\mathbb{A}^{n+1}_{\mu K}[\![\alpha]\!] = \{\lambda_1\}$ (and for each $\alpha' \neq \alpha$ $\lambda_1 \notin \mathbb{A}^{n+1}_{\mu K}[\![\alpha']\!]$), or $\lambda_1 = l_1 : P \blacktriangleright l_2$, $\lambda_2 = l_1 : Q \blacktriangleright l_2$ and $l_2 :: P \sim^{\mathscr{A}^n_{\mu K}} l_2 :: Q$.
In the first case the statement follows directly from definition of $\mathbb{A}^{n+1}$.
In the second case, using the induction hypothesis, we have that for each ϕ, $l_2 :: P \in \mathbb{M}^{\mathscr{A}^n_{\mu K}}[\![\phi]\!]$ if and only if $l_2 :: Q \in \mathbb{M}^{\mathscr{A}^n_{\mu K}}[\![\phi]\!]$. Hence, for each ϕ such that $depth(\phi) \leqslant n$, we have $l_1 : P \blacktriangleright l_2 \in \mathbb{A}^{n+1}_{\mu K}[\![\phi]\!] \Leftrightarrow l_1 : Q \blacktriangleright l_2$
We now prove that for each N_1 and λ_1 there exists an α such that:

— $\lambda_1 \in \mathbb{A}^{n+1}_{\mu K}[\![\alpha]\!]$; and

— for each λ such that $N_1 \xrightarrow{\lambda}$, if $(\lambda_1, \lambda) \notin \mathscr{A}^{n+1}_{\mu K}$, then $\lambda_2 \notin \mathbb{A}^{n+1}_{\mu K}[\![\alpha]\!]$.

If λ_1 is one of τ, $l_1 : t \rhd l_2$ or $l_1 : t \lhd l_2$ (for some l_1, t or l_2), the statement follows easily by considering $\surd$, $l_1 : t \underline{\rhd} l_2$ or $l_1 : t \underline{\lhd} l_2$.
If $\lambda_1 = l_1 : P \blacktriangleright l_2$, we let X be the finite set defined by

$$X = \{l_1 : Q \blacktriangleright l_2 | \exists N_2 : N_1 \xrightarrow{\lambda_2} N_2 \text{ and } l_2 :: P \sim^{\mathscr{A}^n_{\mu K}} l_2 :: Q\}.$$

From the induction hypotheses, for each $\lambda = l_1 : Q \blacktriangleright l_2 \in X$ there exists ϕ_λ such that $l_2 :: P \models_{\mathbb{A}^n} \phi$ and $l_2 :: Q \models_{\mathbb{A}^n} \neg\phi$. We let $\alpha = l_1 : \phi_X \underline{\blacktriangleright} l_2$ where $\phi_X = \bigwedge_{\lambda \in X} \phi_\lambda$. Therefore:

— $\lambda_1 \in \mathbb{A}^{n+1}_{\mu K}[\![\alpha]\!]$; and

— for each λ such that $N_1 \xrightarrow{\lambda}$, if $(\lambda_1, \lambda_2) \notin \mathscr{A}^{n+1}_{\mu K}$, then $\lambda_2 \notin \mathbb{A}^{n+1}_{\mu K}[\![\alpha]\!]$. □

Let $\mathscr{A}_{\mu K}$ be the greatest lower bound of the chain $\mathscr{A}^i_{\mu K}$ in the CPO $2^{\mathrm{LAB}_{\mu K} \times \mathrm{LAB}_{\mu K}}$. Since $\langle 2^{\mathrm{LAB}_{\mu K} \times \mathrm{LAB}_{\mu K}}, \subseteq \rangle$ is a complete lattice, all we have to do to guarantee the well-definedness of $\mathscr{A}_{\mu K}$ is prove that for each i we have $\mathscr{A}^{i+1}_{\mu K} \subseteq \mathscr{A}^i_{\mu K}$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Lemma 7.20. For each i we have $\mathscr{A}^{i+1}_{\mu K} \subseteq \mathscr{A}^{i}_{\mu K}$.

Proof. The thesis follows directly from the definition of $\mathscr{A}^{i}_{\mu K}$ and Lemma 6.3. □

Definition 7.21. We define $\mathscr{A}_{\mu K} = \bigcap_i \mathscr{A}^{i}_{\mu K}$.

The following lemmas enable us to guarantee that $\mathscr{A}_{\mu K}$ is respectful of $\mathbb{A}_{\mu K}$ and that $\mathscr{A}_{\mu K}$ and $\mathbb{A}_{\mu K}$ are transition respectful in $\mathcal{M}_{\mu K}$. Hence, $\sim^{\mathscr{A}_{\mu K}}$ characterises the equivalence induced by the satisfaction of formulae in $\mathbb{L}_{\mu K}$.

Lemma 7.22. $\mathscr{A}_{\mu K}$ is respectful of $\mathbb{A}_{\mu K}$.

Proof. We have

$$\begin{aligned}(\lambda_1,\lambda_2) \in \mathscr{A}_{\mu K} &\Rightarrow \forall i.(\lambda_1,\lambda_2) \in \mathscr{A}^{i}_{\mu K}\\ &\Rightarrow \forall i.\forall \alpha.\lambda_1 \in \mathbb{A}^{i}_{\mu K}[\![\alpha]\!] \Leftrightarrow \lambda_2 \in \mathbb{A}^{i}_{\mu K}[\![\alpha]\!]\\ &\Rightarrow \forall \alpha.(\forall i \lambda_1 \in \mathbb{A}^{i}_{\mu K}[\![\alpha]\!]) \Leftrightarrow (\forall i \lambda_2 \in \mathbb{A}^{i}_{\mu K}[\![\alpha]\!])\\ &\Rightarrow \forall \alpha.\lambda_1 \in \mathbb{A}[\![\alpha]\!] \Leftrightarrow \lambda_2 \in \mathbb{A}[\![\alpha]\!]\,.\end{aligned}$$

Moreover,

$$\begin{aligned}(\lambda_1,\lambda_2) \notin \mathscr{A}_{\mu K} &\Rightarrow \exists k.(\lambda_1,\lambda_2) \in \mathscr{A}^{k}_{\mu K}\\ &\Rightarrow \exists k.\exists \alpha.\lambda_1 \in \mathbb{A}^{i}_{\mu K}[\![\alpha]\!] \wedge \lambda_2 \notin \mathbb{A}^{i}_{\mu K}[\![\alpha]\!]\\ &\Rightarrow \exists \alpha.\lambda_1 \in \mathbb{A}[\![\alpha]\!] \wedge \lambda_2 \notin \mathbb{A}[\![\alpha]\!]\,.\end{aligned}$$

□

Lemma 7.23. If $N_1 \not\sim^{\mathscr{A}^{i}_{\mu K}} N_2$, there exist k and ϕ such that $depth(\phi) \leqslant k$ and $N_1 \models_{\mathbb{A}^k} \phi$ $N_2 \models_{\mathbb{A}^k} \neg\phi$.

Proof. If $N_1 \not\sim^{\mathscr{A}^{i}_{\mu K}} N_2$, there exists ϕ such that $N_1 \models_{\mathbb{A}^i} \phi$ $N_2 \models_{\mathbb{A}^i} \neg\phi$.

If $i > 0$ for each P and ψ such that $depth(\psi) \leqslant i-1$, then $l_1 : P \blacktriangleright l_2 \in \mathbb{A}^{i}_{\mu K}[\![l_1 : \psi \underline{\blacktriangleright} l_2]\!]$, and it follows that $N_1 \models_{\mathbb{A}^i} \phi'$ $N_2 \models_{\mathbb{A}^i} \neg\phi'$. Where ϕ' is obtained from ϕ by replacing each label predicate $l_1 : \psi \underline{\blacktriangleright} l_2$ ($depth(\psi) \geqslant i$) with $l_1 : \mathbf{T} \underline{\blacktriangleright} l_2$. We have that $depth(\phi') \leqslant i+1$, and, moreover, $N_1 \models_{\mathbb{A}^{i+1}} \phi'$ $N_2 \models_{\mathbb{A}^{i+1}} \neg\phi'$.

If $i = 0$, we let ϕ' be the formula obtained from ϕ by replacing each label predicate $l_1 : \psi \underline{\blacktriangleright} l_2$ with $l_1 : \mathbf{T} \underline{\blacktriangleright} l_2$. We have that $depth(\phi') \leqslant 1$, and, moreover, $N_1 \models_{\mathbb{A}^1} \phi'$ $N_2 \models_{\mathbb{A}^1} \neg\phi'$. □

Lemma 7.24. $\mathscr{A}_{\mu K}$ and $\mathbb{A}_{\mu K}$ are transition respectful in $\mathcal{M}_{\mu K}$.

Proof. We now prove that for each N_1 and λ_1 there exists an α such that:

— $\lambda_1 \in \mathbb{A}[\![\alpha]\!]$; and

— for each λ_2 and N_2 such that $N_2 \xrightarrow{\lambda_2} N_2'$ and $(\lambda_1,\lambda_2) \notin \mathscr{A}_{\mu K}$, we have $\lambda_2 \notin \mathbb{A}^{n+1}_{\mu K}[\![\alpha]\!]$.

If λ_1 is one of τ, $l_1 : t \rhd l_2$ or $l_1 : t \lhd l_2$ (for some l_1, t or l_2), the statement follows easily by considering $\surd$, $l_1 : t \underline{\rhd} l_2$ or $l_1 : t \underline{\lhd} l_2$.

If $\lambda_1 = l_1 : P \blacktriangleright l_2$, we let X be the finite set defined by

$$X = \{l_1 : Q \blacktriangleright l_2 | \exists N_2 : N_1 \xrightarrow{l_1 : Q \blacktriangleright l_2} N_2 \text{ and } l_2 :: P \not\sim^{\mathscr{A}_{\mu K}} l_2 :: Q\}\,.$$

There exists a k such that for each $l_1 : Q \blacktriangleright l_2 \in X$, $l_2 :: P \not\sim^{\mathscr{A}_{\mu K}} l_2 :: Q$. For each $\lambda = l_1 : Q \blacktriangleright l_2 \in X$ there exists ϕ_λ such that $l_2 :: P \models_{\mathbb{A}^k} \phi$, while $l_2 :: Q \models_{\mathbb{A}^k} \neg\phi$.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

Let $\phi_X = \bigcap_{\lambda \in X} \phi_\lambda$. We have that $l_1 : P \blacktriangleright l_2 \in \mathbb{A}^k_{\mu K}[\![l_1 : \phi_X \underline{\blacktriangleright} l_2]\!]$, while for each $\lambda \in X$, $\lambda \not\in \mathbb{A}^k_{\mu K}[\![l_1 : \phi_X \underline{\blacktriangleright} l_2]\!]$. Thus, $l_1 : P \blacktriangleright l_2 \in \mathbb{A}[\![l_1 : \phi_X \underline{\blacktriangleright} l_2]\!]$, while for each $\lambda \in X$, $\lambda \not\in \mathbb{A}[\![l_1 : \phi_X \underline{\blacktriangleright} l_2]\!]$. □

We can now state the final theorem of this section.

Theorem 7.25. $\sim^{\mathscr{A}_{\mu K}}$ characterises the equivalence induced by formulae satisfaction.

Proof. The statement follows directly from Lemmas 7.22 and 7.24 using Theorems 7.5 and 7.8. □

8. Conclusions and future work

In this paper we have proposed a variant of LTSs, which we have called *Multiple-Labelled Transition Systems* (MLTSs), as a candidate general operational model for distributed calculi with names and mobility. To show the usefulness of our proposal, we used MLTSs to describe the operational semantics of two formalisms, namely the asynchronous π-calculus ($A\pi$) (Boudol 1992; Honda and Tokoro 1991) and μKLAIM (Bettini *et al.* 2003), that have the opposite objectives of expressivity and usability.

For modelling the properties of MLTSs, we introduced a temporal logic (MoMo) that consists of a small set of operators to be used to describe specific properties/behaviours of mobile and distributed systems. Together with the usual logical connectives and the operators for minimal and maximal fixed points, the logic is equipped with operators for: describing dynamic behaviours (*temporal properties*); modelling resource management (*state properties*); taking into account name handling (*nominal properties*); and controlling mobile processes (*mobility properties*).

We have also studied the relationships between the equivalences induced on $A\pi$ by MoMo and by bisimulation.

The main limitation of the proposed approach is that in order to establish system properties we need to provide detailed descriptions of the whole systems under consideration. Obviously, this is a very strong assumption for wide area networks, since it is very often the case that only some components of the system are known; and one has only a limited knowledge of the overall context in which the component is operating. Nevertheless, one would like to guarantee that components behave well whenever the context guarantees specific resources or interactions.

For this reason, we plan to set up a framework for specifying contexts for distributed calculi (whose semantics is specified in term of MLTSs). By means of contexts, we will be able to provide abstract specifications of a given system and avoid the need to describe all of its components in full. Indeed, some of these components could be known or implemented only at a later stage. Then, the implemented component can be removed from the context and added to the implemented part, thus performing a *concretion* operation. We aim to set up a framework that would guarantee the preservation of formulae satisfaction at each stage of refinement if the introduced implementation *agrees* with the original specification.

https://doi.org/10.1017/S0960129507006585 Published online by Cambridge University Press

We also plan to use MLTSs to describe the behaviour of other calculi with explicit notions of distribution and mobility, and, in particular, for modelling emerging calculi for Service Oriented Architecture, such as SCC (Boreale *et al.* 2006). This will enable us to specify and verify the properties of distributed service architecture using MoMo.

Acknowledgements

We would like to thank Diego Latella and Mieke Massink for suggestions, and members of the SENSORIA projects for interesting discussions.

References

Amadio, R. M., Castellani, I. and Sangiorgi, D. (1998) On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science* **195** (2) 291–324.

Bettini, L., Bono, V., De Nicola, R., Ferrari, G., Gorla, D., Loreti, M., Moggi, E., Pugliese, R., Tuosto, E. and Venneri, B. (2003) The klaim project: Theory and practice. In: Global Computing. *Springer-Verlag Lecture Notes in Computer Science* **2874** 88–150.

Boreale, M., Bruni, R., Caires, L., De Nicola, R., Lanese, I., Loreti, M., Martins, F., Montanari, U., Ravara, A., Sangiorgi, D., Vasconcelos, V. and Zavattaro, G. (2006) SCC: a service centered calculus. In: Web Services and Formal Methods, Third International Workshop, WS-FM 2006. *Springer-Verlag Lecture Notes in Computer Science* **4184** 38–57.

Boudol, G. (1992) Asynchrony and the π-calculus (note). Rapport de Recherche 1702, INRIA Sophia-Antipolis.

Caires, L. (2004) Behavioral and spatial observations in a logic for the pi-calculus. In: Walukiewicz, I. (ed.) FoSSaCS. *Springer-Verlag Lecture Notes in Computer Science* **2987** 72–89.

Cardelli, L. and Gordon, A. D. (2000) Mobile ambients. *Theoretical Computer Science* **240** (1) 177–213.

Cardelli, L. and Gordon, A. D. (2006) Ambient logic. *Mathematical Structures in Computer Science* (to appear).

Carriero, N. and Gelernter, D. (1989) Linda in Context. *Communications of the ACM* **32** (10) 444–458.

Castagna, G., Vitek, J. and Nardelli, F. Z. (2005) The Seal Calculus. *Information and Computation* **201** (1) 1–54.

Dam, M. (1996) Model checking mobile processes. *Journal of Information and Computation* **129** (1) 35–51.

De Nicola, R., Ferrari, G. L. and Pugliese, R. (1998) KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering* **24** (5) 315–330.

De Nicola, R. and Loreti, M. (2005) MoMo: A modal logic for reasoning about mobility. In: de Boer, F. S., Bonsangue, M. M., Graf, S. and de Roever, W. P. (eds.) Proc. of FMCO 2004, Revised Lectures. *Springer-Verlag Lecture Notes in Computer Science* **3657** 95–119.

Ferrari, G., Montangero, C., Semini, L. and Semprini, S. (2002) Mark, a reasoning kit for mobility. *Automated Software Engineering* **9** 137–150.

Fournet, C., Gonthier, G., Levy, J. J., Maranget, L. and Remy, D. (1996) A Calculus of Mobile Agents. In: Montanari, U. and Sassone, V. (eds.) Proc. of 7th Int. Conf. on Concurrency Theory (CONCUR'96). *Springer-Verlag Lecture Notes in Computer Science* **1119** 406–421.

Gabbay, M. and Pitts, A. M. (1999) A new approach to abstract syntax involving binders. *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society 214–224.

Gelernter, D. (1985) Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems* **7** (1) 80–112.

Gelernter, D. (1989) Multiple Tuple Spaces in Linda. In: Odijk, E., Rem, M. and Syre, J.-C. (eds.) Proceedings, PARLE '89. *Springer-Verlag Lecture Notes in Computer Science* **366** 20–27.

Hennessy, M. and Milner, R. (1985) Algebraic laws for nondeterminism and concurrency. *Journal of the ACM* **32** (1) 137–161.

Hennessy, M. and Riely, J. (2002) Resource access control in systems of mobile agents. *Information and Computation* **173** (1) 82–120.

Honda, K. and Tokoro, M. (1991) An object calculus for asynchronous communication. In: America, P. (ed.) European Conference on Object-Oriented Programming (ECOOP'91). *Springer-Verlag Lecture Notes in Computer Science* **512** 133–147.

McCann, P. and Roman, G.-C. (1998) Compositional programming abstraction for mobile computing. *IEEE Transactions on Software Engineering* **24** (2) 97–110.

Milner, R., Parrow, J. and Walker, D. (1993) Modal logics for mobile processes. *Theoretical Computer Science* **114** 149–171.

Reynolds, J. C. (2002) Separation logic: A logic for shared mutable data structures. In: *Proceedings of 17th IEEE Symposium on Logic in Computer Science*, IEEE Computer Society 55–74.

Wojciechowski, P. T. and Sewell, P. (1999) Nomadic Pict: Language and infrastructure design for mobile agents. In: *ASA/MA*, IEEE Computer Society 2–12.