Formal modeling and quantitative analysis of KLAIM-based mobile systems. — Full Version[*]—

Rocco De Nicola
Dip. Sistemi e Informatica, Univ. di Firenze
v. Lombroso 6/17, I50134 Firenze, Italy
denicola@dsi.unifi.it

Diego Latella
C.N.R., Istituto di Scienze e Tecnologie dell'Informazione - A. Faedo
Via Moruzzi, 1 I56124 Pisa, Italy
Diego.Latella@isti.cnr.it

Mieke Massink
C.N.R., Istituto di Scienze e Tecnologie dell'Informazione - A. Faedo
Via Moruzzi, 1 I56124 Pisa, Italy
Mieke.Massink@isti.cnr.it

Aug. 24, 2004

Abstract

KLAIM is an experimental language designed for modeling and programming distributed systems composed of *mobile* components where distribution awareness and dynamic system architecture configuration are key issues, like most web-based service implementations. In this paper we propose STOCKLAIM, a STOchastic extension of cKLAIM, the *core* subset of KLAIM. cKLAIM includes process distribution, process mobility, and asynchronous communication. The extension makes it possible to *integrate* the modeling of *quantitative* aspects of *mobile* systems—e.g. performance—with the *functional* specification of such systems. At the conceptual level, our proposal is based on the assumption that any *action* of a process in a cKLAIM network takes some *time* to be executed. At the technical level, the time associated to the execution of an action is determined by a continuous random variable with an *exponential* distribution. Consequently, process actions in STOCKLAIM are equipped with the *rate* of such distributions. We present a formal operational semantics of STOCKLAIM, which associates a labeled transition system to each STOCKLAIM network and a translation from such labeled transition systems to Continuous Time Markov Chains for quantitative analysis. We also show how STOCKLAIM can be used by means of two simple examples: namely a distributed mobile service and the spreading of a virus.

1 Introduction

Components of modern widely distributed ubiquitous systems are characterized by highly dynamic behaviour and have to deal with changes of the network environment and its heterogeneity. This is a major result of the dramatic recent change which made computers from isolated devices to powerful, interconnected, interacting components of large complex systems, often referred to as *global computers*. Such

[*]This work has been partially funded by Project EU-IST IST-2001-32747 Architectures for Mobility (AGILE), http://www.pst.informatik.uni-muenchen.de/projekte/agile/.

systems, and the applications running on them, are characterized by features which were absent, or hidden on purpose, in previous generation systems, like *distribution awareness* and *code mobility*. The World Wide Web is an example of such a global system. In order to capture these aspects in a systematic way, specification languages have been developed that allow designers to address key concepts such as *locality* and *movement* of data, processes or devices explicitly. KLAIM (*Kernel Language for Agents Interaction and Mobility*, [9, 4]) is an experimental language designed for modeling and programming distributed systems composed of mobile components interacting via multiple distributed tuple spaces. The KLAIM interaction model builds over, and extends, Linda's one of single shared tuple space [11]. In [9] it is shown how KLAIM can be used for modeling, as well as programming, mobile code applications, i.e. applications whose distinct feature is the exploitation of code mobility. In particular, KLAIM supports all major paradigms relevant in such a context, namely *remote evaluation*, *mobile agents* and *code on demand*.

In this paper we address a first step towards the extension of KLAIM with stochastic features. In particular, we focus on (a variant of) cKLAIM, the *Core* subset of KLAIM, first introduced in [13] and described also in [4], which includes process distribution, process mobility, and asynchronous communication of names through shared located repositories (tuples).

The extension of cKLAIM that we propose and call STOCKLAIM, makes it possible to *integrate* the modeling of *quantitative* aspects of mobile systems—e.g. performance—with the *functional* specification of such systems. The operational semantics of our language associates a Labeled Transition System (LTS) to each STOCKLAIM network specification. The associated LTS defines in turn a stochastic process [19]—and in particular a Continuous Time Markov Chain (CTMC)—which can be used for checking stochastic properties of the behaviour of the network.

CTMCs provide a modeling framework which has proved extremely useful for practical analysis of quantitative aspects of system behaviour. Moreover, in recent years proper stochastic extensions of temporal logics have been proposed and efficient algorithms for checking the satisfiability of formulae of such logics on CTMCs (i.e. stochastic model checkers) have been implemented [18, 20, 23, 6]. It is finally worth pointing out that there is a strong connection between traditional (i.e. qualitative) model-checking and stochastic model-checking, which brings to a sound integration of formal modeling and analysis of functional (qualitative) and non-functional (quantitative) aspects of system behaviour. Such integration in the context of mobile systems is our main long term goal.

cKLAIM can be used for specifying *networks* as finite collections of nodes that may host processes and data. The central ingredients of cKLAIM are names; a countable set of names is assumed ($l, l', \ldots, u, u', \ldots,$ $A, A', \ldots$ are used for denoting such names) that provide the abstract counterpart of the set of communicable objects and can be used as localities, basic variables or process variables.

Each network node is singled out by a name that indicates its locality. Processes are the active computational units and may be executed concurrently either at the same locality or at different localities. They are built up from the terminated process $\mathbf{nil}$ and from a set of basic actions by using prefixing, parallel composition and recursion. Basic actions permit removing/adding data from/to node repositories, activating new threads of execution and creating new nodes. cKLAIM has four different basic actions; and three of them explicitly indicate the (possibly remote) locality where they will take effect. With an output action $\mathbf{out}\ l' \ @\ l$ a process can write the datum l' in repository l. With an input action $\mathbf{in}\ T\ @\ l$ a process can withdraw a datum from repository l. Processes can be written to/withdrawn from a repository as well. The action $\mathbf{eval}\ Q\ @\ l$ spawns process Q at repository l and action $\mathbf{newloc}\ u$ serves for creating a new node; thus providing a means for modeling dynamic network architectures. Action $\mathbf{newloc}\ u$ is not indexed with an address because it always acts locally.

The basic idea underlying our extension is rather simple. Our modeling assumption is that any action a of a cKLAIM process *takes* some *time* to be executed. The time taken by a particular action a for being executed is determined by a random variable. In the context of this paper we restrict such random variables to *exponentially* distributed ones. This restriction is quite common when dealing with quantitative system modeling due to the mathematical tractability of exponential distributions. Consequently (efficient) analytical methods and automatic tools exist for reasoning about system models based on exponential distributions. Moreover, exponential distributions can be used for approximating general distributions, like, e.g. deterministic ones. Finally, exponential distributions form the basis for the definition of CTMCs.

The parameter which completely characterizes an exponentially distributed random variable is its *rate*

λ, which is a positive real number. [1] Consequently, we equip each action a with a rate λ and call the resulting pair a *stochastic* action. The intended meaning of (a, λ) is that the time taken for the complete execution of action a is a random variable distributed as $EXP(\lambda)$.

After having discussed in Sect. 2 existing work on stochastic languages for mobility, in Sect. 3 we define the syntax and static semantics requirements of STOCKLAIM, together with an informal explanation of its operational semantics. The formal definition of the operational semantics is given in Sect. 4. The semantics associates each STOCKLAIM network to a LTS; in Sect. 4 also a translation from such LTSs to CTMCs is defined. Applications of STOCKLAIM are given in Sect. 5 by means of two illustrative examples—namely a distributed network service which exploits mobility for load balancing and the spreading of a network virus—while some conclusions and an outline of future research are presented in Sect. 6.

2 Related Work

In our proposal, at a conceptual level, we follow essentially the same approach as Priami in [22] where he extends the π-Calculus with stochastic features. There is however a key difference between our work and the above mentioned one. In fact, the basic model of interaction of π-Calculus processes is *synchronous*, while that of KLAIM processes is *asynchronous*. Synchronization of actions with exponentially distributed durations poses non-trivial problems to the compositional definition of the operational semantics when the intuition on action execution times is to be preserved by composition operators. For an interesting discussion on the subject we refer the reader to [5].

As we shall see in the sequel, the choice of using an asynchronous model of interaction as the underlying model for stochastic behaviour allows for a rather simple definition of the operational semantics. Moreover it preserves a direct relation between the rates assigned to actions in specifications and those assigned to them in the automata models associated to such specifications. Such a relation is more involving in approaches based on synchronous models of interaction due to rate/probability normalization procedures required by such models.

Of course, the above advantages come at the price of dropping component synchronization as a primitive interaction mechanism. However, experience has shown that many fundamental behavioural aspects of mobile, cooperating agents in distributed networks can be satisfactorily described and analyzed by relying on asynchronous models of interaction [9, 3].

On a more technical level, another peculiarity of our approach is the fact that the definition of the operational semantics of the language is based on a structural congruence which *includes*, among others, *commutativity* and *associativity* of (network and) process parallel composition, non-deterministic choice, and *absorption*. The use of such "coarser" structural congruences greatly simplifies the definition of the operational semantics of locality-based, KLAIM-like languages. By using approaches which rule out commutativity and associativity of parallel and choice operators, one cannot easily exploit the locality-based pattern matching style which is typical of KLAIM-like languages operational semantics definition.

In [10] a probabilistic discrete- (resp. continuous-) time extension of full KLAIM has been proposed. Basically, all sources of non-determinism in the notation have been enriched with probabilistic information. In particular, (process) choice and parallel composition operators have been replaced by their probabilistic counterparts and, in the discrete-time case, probabilities have been added also to the network nodes used in network composition. Intuitively, the probability attached to a node is related to the scheduling criteria at the global network level and extends the scheduling probability defined by the process parallel composition operator at the node level. In the continuous time case, rates of exponential distributions are associated to nodes, which are related to the execution time of any action in the node. Finally, the mappings of logical to physical names (i.e. KLAIM allocation environments) have been replaced by mappings from logical names to probability distributions on physical names. Our proposal is orthogonal to this approach in the sense that non-deterministic and parallel operators are left unchanged while *specific* rates are associated to *each* action, so that the former are features of the specific actions rather than of the node where the actions are

[1] Recall that a real-valued random variable X is exponentially distributed with rate λ—written $EXP(\lambda)$—if the probability of X being at most t, i.e. Prob$(X \leq t)$, is $1 - e^{-\lambda t}$ if $t \geq 0$ and is 0 for $t < 0$, where t is a real number. The expected value of X is λ^{-1}. Exponentially distributed random variables enjoy the so called *memoryless property*, i.e. Prob$(X > t+t' \mid X > t)$ =Prob$(X > t')$, for $t, t' \geq 0$.

$$\begin{array}{lll} N & ::= & \textsc{Networks} \\ & | & l :: \langle l' \rangle \\ & | & l :: \langle P \rangle \\ & | & l :: P \\ & | & N \parallel N \end{array} \qquad \begin{array}{lll} P & ::= & \textsc{Processes} \\ & & \mathbf{nil} \\ & | & (a, r).P \\ & | & P + P \\ & | & P \mid P \\ & | & A \end{array}$$

$$\begin{array}{lll} a & ::= & \textsc{Actions} \\ & & \mathbf{out}\ \ell' \mathbin{@} \ell \\ & | & \mathbf{out}\ P \mathbin{@} \ell \\ & | & \mathbf{in}\ T \mathbin{@} \ell \\ & | & \mathbf{eval}\ P \mathbin{@} \ell \\ & | & \mathbf{newloc}\ u \end{array} \qquad \begin{array}{lll} T & ::= & \textsc{Terms} \\ & & l \\ & | & !u \\ & | & !A \end{array}$$

Table 1: Syntax of STOCKLAIM

executed. This gives rise to a clean semantic model which directly reflects the modeling choices expressed at the specification level, whereas the probabilities of different alternatives of choice, parallel, or network compositions are *derived* on the basis of the *race condition* principle [22]. In the proposal of [10] the specifier has several different conceptual tools and related linguistic constructs for expressing probabilistic information. On the other hand there is a certain interference among such concepts which results in several normalization steps. As a result, the relationship between the specific probability/rate values used in a specification and those resulting in the associated semantical structure can be quite complicated.

In [16] a probabilistic extension of the *asynchronous* π-Calculus is proposed, which does not address time and continuous distributions.

Finally, in [12] PEPA nets are proposed, where mobile code is modeled by expressions of the stochastic process algebra PEPA which play the role of tokens in (stochastic) Petri nets. The Petri net of a PEPA net models the architecture of the net, which to our understanding, is a static one. A PEPA expression can move from a place to another one if there is a transition from the first place to the second. A proper synchronization mechanism between PEPA expressions and Petri nets is provided in order to fire transitions (i.e. to move code).

We are not aware of other proposals for stochastic/probabilistic calculi for mobile systems.

3 Syntax of STOCKLAIM

Let $\mathcal{L}$, ranged over by $l, l', l_1, \ldots$, be a set of *localities*, $\mathcal{U}$, ranged over by $u, u', u_1, \ldots$, a set of *locality variables*, $\mathcal{A}$, ranged over by $A, A', A_1, B, \ldots$ a set of *process variables*, and $\mathcal{R}$, ranged over by $r, r', r_1, w, x, y \ldots$ a set of *rate names*. We assume that the above sets are mutually disjoint. Moreover, we let ℓ, ℓ' range over $\mathcal{L} \cup \mathcal{U}$.

The syntax of STOCKLAIM, given in Table 1, is exactly the same as that of cKLAIM, except that processes have a richer *action prefix* operator, and for the addition of an explicit choice composition operator. Moreover, processes can be stored to (resp. retrieved from) localities by means of **out** (resp. **in**) actions, in much the same way as data. A *network* node $l :: \langle l' \rangle$ intuitively means that value $\langle l' \rangle$ is stored, or located, in node, or locality, l. Similarly, for process P, $l :: \langle P \rangle$ means that P is stored in l as a piece of data. On the other hand $l :: P$ indicates that process P is running in locality l. Complex networks are built from simpler ones by means of the *network parallel operator* $\parallel$. Given network N, the set of values located in locality l coincides with all those l' and P such that $l :: \langle l' \rangle$ or $l :: \langle P \rangle$ occurs in N. The set of processes running in locality l coincides with all those P such that $l :: P$ occurs in N. The intuition behind action prefix $(a, r).P$ is that the execution time of action a is distributed exponentially with rate specified by *rate name* r. Rate names are mapped to rate values by means of binding functions, which are (partial) functions from $\mathcal{R}$ to $\mathbb{R}^+$. The main reason for using rate *names* instead of just rates is related to the way we deal with the *race condition* semantics for the non-deterministic choice operator $(P_1 + P_2)$ and interleaving one $(P_1 \mid P_2)$.

As we mentioned in Sect.1, STOCKLAIM networks can be mapped to CTMCs. The presence of a choice operator for processes facilitates the specification of stochastic processes since each choice expression essentially corresponds to a state of the underlying CTMC with as many transitions as those of the components of the choice (and of possibly nested parallel compositions).

A typical problem in the definition of stochastic process calculi, due to the race condition principle, is that an expression like $(a, \lambda).P + (a, \lambda).P$ should *not* be identified with the expression $(a, \lambda).P$—as it would be the case in non-stochastic process calculi. In fact, to an external observer, the first process should appear twice as fast as the second one (i.e. it should be equated to $(a, 2\lambda).P)^2$. There are several ways for dealing with the problem of not identifying a process offering a choice of two equal components with one of the components. One way is to use *proved transition systems* as in [22] that permit distinguishing left and right components by labelling transitions starting from derivations labelled with the actual proof of process transitions. Unfortunately, such an approach is not naturally compatible with a definition of the operational semantics based on a structural congruence which includes, among others, commutativity and associativity of $\|, +, |$ and absorption. As we shall see, such structural congruences greatly simplify the definition of the operational semantics since they allow the full exploitation of locality-based pattern matching in the application of the deduction rules. Consequently, in this paper we prefer not to use proved transition systems and to require that the rate names occurring in any network expression be distinct. Actually it would suffice to require that the rate names of the initial steps of the components of choice expressions be distinct and similarly for all rate names of the components of process and node parallel compositions. But, we prefer a more homogeneous approach. Clearly, care is needed to guarantee name uniqueness in presence of process replication and migration during execution. This choice allows us to keep the definition of the operational semantics as simple as possible, focusing more on the main issues of mobility and stochastic behaviour than on the technicalities of such a definition. Moreover, the use of rate names and binding functions instead of actual rate values permits re-using a network specification, with different bindings, for several validation sessions, as we will see in the examples in Sect.5. Finally, the separation of rates from rate names facilitates the future extension of our calculus with *rate variables*.

Recursive behaviours are modelled via *process definitions*; it is assumed that each identifier A has a *single defining equation* of the form $A \stackrel{\Delta}{=} P$ where P may contain occurrences of A and other process names. It is also assumed that occurrences of A on the right part are always guarded, i.e. prefixed by a stochastic action.

Finally, a term T can be either a locality constant l or a parameter, i.e. a locality variable u or a process variable A. Parameters are identified by prefixing them by an exclamation mark.

4 Semantics of STOCKLAIM

In this section, the operational semantics of STOCKLAIM is defined as well as the translation from the resulting LTSs to CTMCs.

4.1 STOCKLAIM Operational Semantics

The operational semantics of STOCKLAIM is an orthogonal extension of the one of cKLAIM as presented, e.g., in [4, 13]. The transition relation is defined over (network) configurations, i.e. triples (L, β, N)—henceforth written as $L, \beta \vdash N$—where L is a finite set of localities, $\beta : \mathcal{R} \mapsto \mathbb{R}^+$ is a mapping from rate names to rates, with $(dom\,\beta)$—the domain of β—also finite, and N a STOCKLAIM network expression.

^{2}An expression like $(a, \lambda).P + (b, \mu).Q$ is interpreted as a *race condition* between a and b. This, intuitively can be interpreted as follows: when such a process is executed, *both* a and b are enabled and their actual execution times are given by a sample of $EXP(\lambda)$ and $EXP(\mu)$ respectively. The action with the smallest execution time is actually executed. From standard theory we know that for independent random variables X and Y respectively in $EXP(\lambda)$ and in $EXP(\mu)$ the random variable $\textsc{min}(X, Y)$ is exponentially distributed with rate $\lambda + \mu$.

Let $(\mathsf{Loc}\, N)$ denote the set of all localities occurring free [3] in N and $(\mathsf{Rat}\, N)$ be the set of all rate names occurring in N. We require $(\mathsf{Loc}\, N) \subseteq L$ and $(\mathsf{Rat}\, N) \subseteq (dom\, \beta)$. Finally, we require that all rate names occurring in N be *distinct* and that for expressions of the form $\mathbf{in}\ !A\ @\ l'.P$, (i) there exists *at most one* free occurrence of A in P which is not the first argument of an **out** or **eval** operator, and (ii) there exists no defining equation for A.

The Structural Congruence is the smallest relation satisfying the laws given in Table 2. The main difference with those of cKLAIM is the addition of the laws for commutativity (CO+) and associativity (AS+) for non-deterministic choice, a law for its neutral element (NE+), and a law (REN) for rate name remaning. The law for rate renaming (REN) states that the rate names occurring in N can be replaced by means of a rate names substitution θ. We use the usual notation for syntactical substitution, namely $N\theta$, where θ is a total function in $\mathcal{R} \to \mathcal{R}$. Of course it is required that (i) the substitution does not interfere with the current binding (i.e. $(dom\, \beta) \cap (rng\, \theta) = \emptyset$, where $(rng\, \theta)$ is the *range* of θ), (ii) rate names uniqueness is preserved in $N\theta$ (i.e. θ must be injective), and (iii) the binding in the configuration where the substitution has been applied is defined for the new names and gives the same rates as for the old ones (i.e. the new binding is the composition of the old binding and the inverse of θ).

The Reduction Relation $\longrightarrow$ is the smallest relation induced by the rules of Table 3. Let $\mathcal{C}$, ranged over by $c, c', c_1, \ldots$ be the set of all standard *representatives* of the equivalence classes on configurations induced by the Structural Congruence Laws. We abstract here from the way in which such representatives are chosen; a possibility could be taking the *smallest* elements, where we can use set inclusion for locality sets and binding(-domain)s, and lexicographic order for network terms. For configuration $c \in \mathcal{C}$, let $\mathcal{D}\ c$ be the smallest set such that (i) $c \in \mathcal{D}\ c$, and (ii) if $c' \in \mathcal{D}\ c$, $c'' \in \mathcal{C}$, $r \in \mathcal{R}$ and $(c', r, c'') \in \longrightarrow$ can be derived using the rules and laws of tables 3 and 2, then also $c'' \in \mathcal{D}\ c$.

The operational semantics of a network N_0, with rate names defined by binding β_0, associates a LTS, $TS(N_0, \beta_0) =_{\mathrm{def}} (C, \Lambda, \longrightarrow, c_0)$ to N_0 with β_0. $C =_{\mathrm{def}} \mathcal{D}\ c_0$ is the set of states of the LTS, where the initial state of the LTS, $c_0 \in \mathcal{C}$, is the standard representative of $(\mathsf{Loc}\, N_0), \beta_0 \vdash N_0$; $\Lambda \subseteq \mathcal{R}$ is the set of labels (i.e. rate names) of the LTS and $\longrightarrow \subseteq C \times \Lambda \times C$ is its transition relation, as deduced by the Reduction Rules and the Congruence Laws. As usual, $c \xrightarrow{r} c'$ stands for $(c, r, c') \in \longrightarrow$; moreover, if c is the state $L, \beta \vdash N$, we let β_c denote the binding β of c.

Let us briefly comment on some of the rules. Rule (OUTL), resp. (OUTP), models the dispatching of a name, resp. a process, at a (possibly remote) locality. In order to preserve uniqueness of rate names, when executing action $(\mathbf{out}\ Q\ @\ l', r).P$ process Q' is stored instead of Q; Q' is obtained from Q by means of function RN, defined in Fig. 1, which *renames* all rate names in Q into fresh names; function RN returns also a new binding where the fresh names are bound to the same rates as those the original ones were bound to. Notice that, in the definition of function RN, rate name uniqueness is guaranteed by means of function $\mathbf{choose}$ and by a proper sequentialization of the application of RN to the components of (process) parallel and non-deterministic composition; such sequentialization is achieved by means of β''. The selection criterion of $\mathbf{choose}$ is immaterial here and is easily implementable due to finiteness of the domain of binding functions. Rate names renaming takes place also for process spawning (EVA) and instantiation (PIN) for similar reasons.

Rule (INL), resp. (INP) models retrieval and removal of names, resp. processes, from given localities. Action $\mathbf{in}\ T\ @\ l'$ is a blocking action that can be performed only if the required datum is present at the chosen locality l'. Moreover, if the argument (T) is a locality variable ($!u$), resp. a process variable ($!A$), the retrieved datum is used to replace all free occurrences of u, resp. A, in the rest of the process executing the action; instead, if the argument is a locality constant the only effect is its removal from the target node.

Rule (NLC) models the creation of a new node, with its fresh name; indeed the expected result of $\mathbf{newloc}\ u$ is a fresh name, i.e. a name not present in the set of all used names, L; function $\mathbf{choose}$ is used to choose such a new element of $\mathcal{L} \setminus L$, the selection criterion being immaterial here. The new name is then added to the set of used names, L. Rule (EVA) is used to model the spawning of the argument process at the intended locality; there it will run concurrently with the processes already present. The remaining rules are standard and are used to deal with parallel composition, nonderministic choice and to take advantage of structural congruence. For the rest, the rules should be self-explanatory.

[3]The formal definition of free and bound names in cKLAIM can be found in [13].

(CO$\parallel$)	$L, \beta \vdash N_1 \ \parallel\ N_2 \equiv L, \beta \vdash N_2 \ \parallel N_1$
(AS$\parallel$)	$L, \beta \vdash N_1 \ \parallel\ (N_2 \ \parallel\ N_3) \equiv$ $L, \beta \vdash (N_1 \ \parallel\ N_2) \ \parallel\ N_3$
(NE$\vert$)	$L, \beta \vdash l :: P \equiv L, \beta \vdash l :: P \mid \mathbf{nil}$
(CO+)	$L, \beta \vdash l :: P_1 + P_2 \equiv L, \beta \vdash l :: P_2 + P_1$
(AS+)	$L, \beta \vdash l :: P_1 \ + \ (P_2 \ + \ P_3) \equiv$ $L, \beta \vdash l :: (P_1 \ + \ P_2) \ + \ P_3$
(NE+)	$L, \beta \vdash l :: P \equiv L, \beta \vdash l :: P + \mathbf{nil}$
(CLO)	$L, \beta \vdash l :: P_1 \mid P_2 \equiv L, \beta \vdash l :: P_1 \parallel l :: P_2$
(REN)	$L, \beta \vdash N \equiv L, (\beta \circ \theta^{-1}) \vdash N\theta$ for any $\theta : \mathcal{R} \to \mathcal{R}$ total, injective, and such that $(rng\, \theta) \cap (dom\, \beta) = \emptyset$

Table 2: Structural Congruence of STOCKLAIM

$$
\begin{array}{lll}
\mathsf{RN}(\mathbf{nil}, \beta) & =_{\mathrm{def}} & (\mathbf{nil}, \beta) \\[1em]
\mathsf{RN}((a, r).P, \beta) & =_{\mathrm{def}} & ((a, r').P', \beta') \text{ where} \\
 & & r' = \mathbf{choose}\ r_1 \in \mathcal{R} \setminus (dom\, \beta) \\
 & & (P', \beta') = \mathsf{RN}(P, \beta[\beta(r)/r']) \\[1em]
\mathsf{RN}(P\ op\ Q, \beta) & =_{\mathrm{def}} & (P'\ op\ Q', \beta'),\ op \in \{+, |\}, \text{ where} \\
 & & (P', \beta'') = \mathsf{RN}(P, \beta) \\
 & & (Q', \beta') = \mathsf{RN}(Q, \beta'') \\[1em]
\mathsf{RN}(A, \beta) & =_{\mathrm{def}} & (A, \beta), \mathbf{if}\ \ A \triangleq P
\end{array}
$$

Figure 1: Definition of function RN

(OUTL) $L, \beta \vdash l :: (\mathbf{out}\ l'' \ @\ l', r).P \parallel l' :: P' \xrightarrow{r}$
$L, \beta \vdash l :: P \parallel l' :: P' \parallel l' :: \langle l'' \rangle$

(OUTP) $L, \beta \vdash l :: (\mathbf{out}\ Q \ @\ l', r).P \parallel l' :: P' \xrightarrow{r}$
$L, \beta' \vdash l :: P \parallel l' :: P' \parallel l' :: \langle Q' \rangle$
where $(Q', \beta') = \mathsf{RN}(Q, \beta)$

(INL) $L, \beta \vdash l :: (\mathbf{in}\ T \ @\ l', r).P \parallel l' :: \langle l'' \rangle \xrightarrow{r}$
$L, \beta \vdash l :: P\sigma \parallel l' :: \mathbf{nil}$
where $\sigma = \begin{cases} [l''/u], & \mathbf{if} \quad T = !u \\ \epsilon, & \mathbf{if} \quad T = l'' \end{cases}$

(INP) $L, \beta \vdash l :: (\mathbf{in}\ (!A) \ @\ l', r).P \parallel l' :: \langle P' \rangle \xrightarrow{r}$
$L, \beta \vdash l :: P[P'/A] \parallel l' :: \mathbf{nil}$

(NLC) $L, \beta \vdash l :: (\mathbf{newloc}\ u, r).P \xrightarrow{r}$
$L \cup \{l'\}, \beta \vdash\ l :: P[l'/u] \ \parallel\ l' :: \mathbf{nil}$
where $l' = \mathbf{choose}\ l_1 \in \mathcal{L} \setminus L$

(EVA) $L, \beta \vdash l :: (\mathbf{eval}\ Q \ @\ l', r).P \parallel l' :: P' \xrightarrow{r}$
$L, \beta' \vdash l :: P \parallel l' :: P' \mid Q'$
where $(Q', \beta') = \mathsf{RN}(Q, \beta)$

(PIN) $$\frac{L, \beta' \vdash l :: P' \xrightarrow{r} L', \beta'' \vdash N}{L, \beta \vdash l :: A \xrightarrow{r} L', \beta'' \vdash N}$$
where $A \triangleq P$ and $(P', \beta') = \mathsf{RN}(P, \beta)$

(CHO) $$\frac{L, \beta \vdash l :: P_1 \parallel N \xrightarrow{r} L', \beta' \vdash l :: P' \parallel N'}{L, \beta \vdash l :: P_1 + P_2 \parallel N \xrightarrow{r} L', \beta' \vdash l :: P' \parallel N'}$$

(PAR) $$\frac{L, \beta \vdash N_1 \xrightarrow{r} L', \beta' \vdash N'}{L, \beta \vdash N_1 \parallel N_2 \xrightarrow{r} L', \beta' \vdash N' \parallel N_2}$$

(STC) $$\frac{\begin{array}{c} L, \beta \vdash N \equiv L_1, \beta_1 \vdash N_1, \\ L_1, \beta_1 \vdash N_1 \xrightarrow{r} L_2, \beta_2 \vdash N_2, \\ L_2, \beta_2 \vdash N_2 \equiv L', \beta' \vdash N' \end{array}}{L, \beta \vdash N \xrightarrow{r} L', \beta' \vdash N'}$$

Table 3: Reduction Rules of STOCKLAIM

4.2 From LTSs to CTMCs

As we already mentioned in Sect. 1, in order to apply numerical analysis techniques for studying quantitative aspects of (mobile) systems, and especially in order to use stochastic model checking, it is necessary to obtain a CTMC from the LTS associated to a STOCKLAIM network expression. This in turn requires the LTS be finite. Consequently, henceforth we will take into consideration only finite LTSs. There are several ways for assuring finiteness of the LTS automatically generated from higher level specifications, like process algebras. Some relay on syntactical restrictions, like avoiding certain (combinations of) operators, and they have been studied extensively in the context of traditional process algebra. Others, typically used in the context of verification tools design, are based on the introduction of constraints on certain kinds of resources, e.g. buffer sizes and data value domains, in the definition of the operational semantics. The latter approach seems to be most suitable for STOCKLAIM. For instance a limit can be imposed on the maximum number of values which can be stored in a single node. We leave the details of these issues for further study.

CTMCs have been extensively studied in the literature (a comprehensive treatment can be found in [19]; we suggest [15] for a gentle introduction). For the purposes of the present paper it suffices to recall that a CTMC $\mathcal{M}$ is a pair $(\mathcal{S}, \mathbf{R})$ where $\mathcal{S}$ is a finite set of *states* and $\mathbf{R} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the *rate matrix*. The rate matrix characterizes the transitions between the states of $\mathcal{M}$. If $\mathbf{R}(s, s') \neq 0$ then it is possible that a transition from state s to state s' takes place, and the probability of "taking" such a transition within time t, is $1 - e^{-\mathbf{R}(s,s')\cdot t}$. If $\mathbf{R}(s, s') = 0$ then no such a transition can take place[4]. Finally, we would like to point out that the traditional definition of CTMCs does not include self-loops, i.e. transitions from a state to itself. On the other hand, the presence of such self-loops does not alter standard analysis techniques (e.g. transient and steady state ones) and turn out to be useful when addressing model-checking CTMCs [1], therefore we will allow them in this paper.

Given a network N with binding β and assuming $TS(N, \beta) = (C, \Lambda, \longrightarrow, c_0)$ finite, the CTMC $(\mathcal{S}, \mathbf{R})$ associated to N with binding β, denoted by $CTMC(N, \beta)$, is defined as follows: the set $\mathcal{S}$ of states coincides with C, and for all $c, c' \in C$

$$\mathbf{R}(c, c') \quad =_{\text{def}} \quad \begin{cases} \sum_{r \in T_{c,c'}} (\beta_c\ r) & \textbf{if} \ \ T_{c,c'} \neq \emptyset \\ 0 & \textbf{if} \ \ T_{c,c'} = \emptyset \end{cases}$$

where $T_{c,c'} \ =_{\text{def}} \ \{r' \mid c \xrightarrow{r'} c'\}$

5 Examples of Modeling and Analysis

In this section we show how STOCKLAIM can be used for modeling stochastic aspects of mobile systems. In fact we also give an idea of how stochastic model-checking would work for STOCKLAIM.

A complete model-checking framework for a modeling language requires the availability of a proper temporal logic for the specification of the requirements against which models are to be checked. Such a logic should provide specific modalities for the primitive notions which the modeling language is built upon, besides (or extending) the purely temporal ones. So, in the case of STOCKLAIM we should address both the stochastic features and mobility. We are currently developing a logic for STOCKLAIM which addresses both issues by means of integrating notions of ACSL [17] with concepts of the KLAIM logic [4]. ACSL is an action-based variant of CSL, the Continuous Stochastic Logic, as proposed in [2]. CSL is the input logic for the Erlangen-Twente Markov Chain Model Checker, ETMCC [18]. For the examples in the present paper, we use a simple customization of CSL which allows us to express limited aspects of mobility, namely the fact that a certain piece of data (resp. process) is stored (resp. running) at a certain locality. We do this by using atomic propositions of the form $A@l$ and $\langle l' \rangle @l$ where A is a process identifier and l, l' localities. A state s in the CTMC corresponding to a LTS will satisfy, i.e. will be labelled by $A@l$ (resp. $\langle l' \rangle @l$) if and only if the associated configuration c in the LTS (belongs to the same congruence class

[4]The reader should be warned that the above intuitive interpretation is correct only if there is only one transition originating from s. If this is not the case, then a *race* condition arises among all transitions originating from s.

which contains a configuration which) is of the form $L, \beta \vdash N$ and $l :: A$ (resp. $l :: \langle l' \rangle$) is a sub-expression of N.

Despite the above limitation, our examples should illustrate the benefits of a formal approach to modeling and analysis of stochastic aspects of mobile systems.

CSL is a stochastic variant of the well-known Computational Tree Logic (CTL, see e.g. [7]). CTL permits stating properties of *states*, and of *paths*. CSL extends CTL with two probabilistic operators that refer to the steady-state and the transient behavior of the system under consideration. While the steady-state operator refers to the probability of the system being, in the long run, in any of the *states* belonging to a given set (specified by a state-formula), the transient operator allows us to refer to the probability of the occurrence of particular *paths* in the CTMC. In order to express the time-span of specific paths, the path-operators until $\mathcal{U}$ and next X are extended with a parameter that specifies a time-interval. Let I be an interval on the real line, p a probability value and $\bowtie$ an ordering operator on $\mathbb{R}$, i.e. $\bowtie \in \{<, \leq, \geq, >\}$. The syntax of CSL is:

State-formulas

$$\Phi ::= \alpha \;\Big|\; \neg\Phi \;\Big|\; \Phi \vee \Phi \;\Big|\; \mathcal{S}_{\bowtie p}(\Phi) \;\Big|\; \mathcal{P}_{\bowtie p}(\varphi)$$

$\mathcal{S}_{\bowtie p}(\Phi)$	: prob. that Φ holds in steady state is $\bowtie p$
$\mathcal{P}_{\bowtie p}(\varphi)$	: prob. that path-formula φ holds is $\bowtie p$

Path-formulas

$$\varphi ::= X^I\,\Phi \;\Big|\; \Phi\mathcal{U}^I\,\Phi$$

$X^I\,\Phi$	: next state is reached at time $t \in I$ and fulfills Φ
$\Phi\mathcal{U}^I\,\Psi$	: Φ holds along path until Ψ holds at $t \in I$

The meaning of atomic propositions (α), negation ($\neg$) and disjunction ($\vee$) is standard; using these operators, other boolean operators such as conjunction ($\wedge$), implication ($\Rightarrow$), true (TRUE) and false (FALSE), and so forth, can be defined, as usual. The state-formula $\mathcal{S}_{\bowtie p}(\Phi)$ asserts that the steady-state probability for the set of states satisfying Φ, the Φ-states, meets the bound $\bowtie p$. $\mathcal{P}_{\bowtie p}(\varphi)$ asserts that the probability measure of the set of paths satisfying φ meets the bound $\bowtie p$. The operator $\mathcal{P}_{\bowtie p}(.)$ replaces the usual CTL path quantifiers $\exists$ and $\forall$. In CTL, the state-formula $\exists\varphi$ is valid in state s if there *exists* some path starting in s and satisfying φ and $\forall\varphi$ is valid if *all* paths satisfy φ. In CSL, the formula $\mathcal{P}_{\geq 1}(\varphi)$ holds if *almost all* paths satisfy φ. Moreover, clearly $\exists\varphi$ holds whenever $\mathcal{P}_{>0}(\varphi)$ holds. Thus, qualitative as well as stochastic properties can be expressed in CSL[5].

In CTL, a path satisfies an until-formula $\Phi\mathcal{U}\Psi$ if there is a state on the path where Ψ holds, and at every preceding state on the path, if any, Φ holds. The CSL counterpart, $\Phi\mathcal{U}^I\,\Psi$ is satisfied by a path if Ψ holds at time $t \in I$ and at every preceding state on the path, if any, Φ holds. In CSL, temporal operators like $\Diamond$, $\Box$ and their real-time variants $\Diamond^I$ or $\Box^I$ can be derived, e.g., $\mathcal{P}_{\bowtie p}(\Diamond^I\,\Phi) = \mathcal{P}_{\bowtie p}(\text{TRUE}\ \mathcal{U}^I\,\Phi)$ and $\mathcal{P}_{\geq p}(\Box^I\,\Phi) = \mathcal{P}_{<1-p}(\Diamond^I\,\neg\Phi)$. The untimed next- and until-operators are obtained by $X\Phi = X^I\Phi$ and $\Phi_1\mathcal{U}\Phi_2 = \Phi_1\,\mathcal{U}^I\,\Phi_2$ for $I = [0, \infty)$.

Four different types of performance and dependability measures can be expressed in CSL, viz. steady-state measures, transient-state measures, path-based measures, and nested measures.

The ETMCC model checker [18] is a prototype tool that supports the verification of CSL-properties over CTMCs. The model checker takes as input a model file with a textual representation of a CTMC, a label file associating each state to the atomic propositions that hold in that state and a given accuracy. ETMCC is based on sparse matrix representations of CTMCs. Alternative model checkers for CSL include PRISM [20], Prover [23] and the APNN (Abstract Petri Net Notation) toolbox [6].

In the following we present two simple examples with the purpose of showing how STOCKLAIM can be used in conjunction with a stochastic model-checker.

[5]We recall that in the context of probabilistic program/model verification, a qualitative property is one which does not involve numeric probabilities, except probability 1; in such a context, a qualitative property is satisfied by a system if the probability of the set of computations which satisfy the property amounts to 1, i.e. the property is satisfied by *almost all* computations (see e.g. [21, 8, 14]).

$\mathtt{L} :: \langle\mathtt{GO}\rangle \parallel \mathtt{L} :: \langle\mathtt{LF}\rangle \parallel \mathtt{L} :: \mathtt{Srv}|\mathtt{Usr} \parallel \mathtt{R} :: \langle\mathtt{RF}\rangle$
where:

$$\begin{array}{lll}
\mathtt{Usr} & \triangleq & (\mathbf{in}\ \mathtt{GO}\ @\ \mathtt{L}, \mathtt{u}).\mathtt{UsrAct} \\
\mathtt{UsrAct} & \triangleq & (\mathbf{out}\ \mathtt{S1}\ @\ \mathtt{L}, \mathtt{u1}).\mathtt{Usr} + \\
 & & (\mathbf{out}\ \mathtt{S2}\ @\ \mathtt{L}, \mathtt{u2}).\mathtt{Usr} \\
\mathtt{Srv} & \triangleq & (\mathbf{in}\ \mathtt{S1}, \mathtt{LF}\ @\ \mathtt{L}, \mathtt{s1}).\mathtt{SrvAct1} + \\
 & & (\mathbf{in}\ \mathtt{S2}, \mathtt{LF}\ @\ \mathtt{L}, \mathtt{s2}).\mathtt{SrvAct2} \\
\mathtt{SrvAct1} & \triangleq & (\mathbf{eval}\ \mathtt{Agt1}\ @\ \mathtt{L}, \mathtt{sa1}).\mathtt{SrvGo} \\
\mathtt{SrvAct2} & \triangleq & (\mathbf{eval}\ \mathtt{Agt2}\ @\ \mathtt{L}, \mathtt{sa2}).\mathtt{SrvGo} \\
\mathtt{SrvGo} & \triangleq & (\mathbf{out}\ \mathtt{GO}\ @\ \mathtt{L}, \mathtt{sg}).\mathtt{Srv} \\
\mathtt{Agt1} & \triangleq & (\mathbf{out}\ \mathtt{LF}\ @\ \mathtt{L}, \mathtt{a1}).\mathbf{nil} \\
\mathtt{Agt2} & \triangleq & (\mathbf{in}\ \mathtt{RF}\ @\ \mathtt{R}, \mathtt{a2}).\mathtt{Amr} \\
\mathtt{Amr} & \triangleq & (\mathbf{eval}\ \mathtt{Ar}\ @\ \mathtt{R}, \mathtt{amr}).\mathbf{nil} \\
\mathtt{Ar} & \triangleq & (\mathbf{out}\ \mathtt{LF}\ @\ \mathtt{L}, \mathtt{ar}).\mathtt{Ac} \\
\mathtt{Ac} & \triangleq & (\mathbf{out}\ \mathtt{RF}\ @\ \mathtt{R}, \mathtt{ac}).\mathbf{nil}
\end{array}$$

Figure 2: DMS defi nition

5.1 A Distributed Mobile Service

Figure 2 shows the StocKlaim specifi cation of a simple network service, DMS, which exploits mobility for balancing the usage of network resources. We consider a very simple confi guration of the distributed service, consisting of two localities, namely L, for *local node*, and R, for *remote node*. Moreover, in order to simplify the specifi cation and the associated transition system, we use a slightly extended version of the StocKlaim input action **in** $T\ @\ l$, where T can also be a pair of locality constants l_1, l_2 which matches the data l_1 and l_2 in the node l of a network like $N \parallel l :: \langle l_1\rangle \parallel l :: \langle l_2\rangle$.

There is a user Usr running locally, continuously sending requests to the local server Srv. For each request, the server spawns a specifi c agent which will take care of servicing it. Requests of service S1 are processed locally while those of service S2 require also special computing resources which are located remotely, so they are initiated locally but completed remotely, which requires the migration of the related agent, Agt2, to R. Each agent terminates as soon as the processing of the request it is in charge of fi nishes. The user is allowed to send a new request for a service only after the server has dispatched the previous one; this is achieved by means of the token GO. At any point in time, at most one agent is running at each node (but two agents can run in parallel, one at L and one at R); this is achieved by means of tokens LF and RF. The LTS of the distributed mobile service, $TS(DMS)$ is shown in the table in Fig. 3. States are numbered from 1 to 44 and the number of each state is given in the fi rst column. The second (resp. third) column gives the current tuple at locality L (resp. R). Finally the fourth column lists, for each state, the transitions originating from it; each transition is represented as a pair *(rate-name,target-state)*. For the sake of readability, in the table, expressions like $l :: \langle l_1\rangle \parallel l :: \langle l_2\rangle \parallel l :: P$ are represented by $\langle l_1\rangle \mid \langle l_2\rangle \mid P$ in the column of locality l; only the network component of confi gurations is shown, and the same names used in the specifi cation are used instead of those occurring in the representatives, after renaming, according to the LTS defi nition.

Below we give some examples of quantitative and qualitative properties of the DMS model that can be verifi ed by means of the ETMCC model checker. The fi rst issue we address is resource usage. More specifi cally, we are interested in knowing a lower bound for the fraction or percentage of usage of the computing resources of node L. This can be computed as the dual of an upper bound for the probability that *no* process is running at locality L, in the long run, which corresponds to an upper bound for the steady state probability of having value ⟨LF⟩ in locality L. We performed our analysis on $CTMC(N_0, \beta_0)$ where N_0 is the network expression of Fig.2 and $\beta_0\, u = 10$, $\beta_0\, u1 = 2$, $\beta_0\, u2 = 8$, $\beta_0\, s1 = \beta_0\, s2 = \beta_0\, sg = \beta_0\, a1 = \beta_0\, a2 = \beta_0\, amr = \beta_0\, ar = 10$, $\beta_0\, sa1 = 8$, $\beta_0\, sa2 = 5$, and $\beta_0\, ac = 2$. We analyzed the case

s#	L	R	succ
1	⟨GO⟩|⟨LF⟩|Srv|Usr	⟨RF⟩	(u, 2)
2	⟨LF⟩|Srv|UsrAct	⟨RF⟩	(u1, 3), (u2, 4)
3	⟨S1⟩|⟨LF⟩|Srv|Usr	⟨RF⟩	(s1, 5)
4	⟨S2⟩|⟨LF⟩|Srv|Usr	⟨RF⟩	(s2, 6)
5	SrvAct1|Usr	⟨RF⟩	(sa1, 7)
6	SrvAct2|Usr	⟨RF⟩	(sa2, 8)
7	SrvGo|Usr|Agt1	⟨RF⟩	(sg, 9), (a1, 10)
8	SrvGo|Usr|Agt2	⟨RF⟩	(sg, 11), (a2, 12)
9	⟨GO⟩|Srv|Usr|Agt1	⟨RF⟩	(u, 13), (a1, 1)
10	⟨LF⟩|SrvGo|Usr	⟨RF⟩	(sg, 1)
11	⟨GO⟩|Srv|Usr|Agt2	⟨RF⟩	(u, 14), (a2, 15)
12	SrvGo|Usr|Amr	**nil**	(sg, 15), (amr, 16)
13	Srv|UsrAct|Agt1	⟨RF⟩	(u1, 17), (u2, 18), (a1, 2)
14	Srv|UsrAct|Agt2	⟨RF⟩	(u1, 19), (u2, 20), (a2, 21)
15	⟨GO⟩|Srv|Usr|Amr	**nil**	(u, 21), (amr, 22)
16	SrvGo|Usr	Ar	(sg, 22), (ar, 23)
17	⟨S1⟩|Srv|Usr|Agt1	⟨RF⟩	(a1, 3)
18	⟨S2⟩|Srv|Usr|Agt1	⟨RF⟩	(a1, 4)
19	⟨S1⟩|Srv|Usr|Agt2	⟨RF⟩	(a2, 24)
20	⟨S2⟩|Srv|Usr|Agt2	⟨RF⟩	(a2, 25)
21	Srv|UsrAct|Amr	**nil**	(u1, 24), (u2, 25), (amr, 26)
22	⟨GO⟩|Srv|Usr	Ar	(u, 26), (ar, 27)
23	⟨LF⟩|SrvGo|Usr	Ac	(sg, 27), (ac, 10)
24	⟨S1⟩|Srv|Usr|Amr	**nil**	(amr, 28)
25	⟨S2⟩|Srv|Usr|Amr	**nil**	(amr, 29)
26	Srv|UsrAct	Ar	(u1, 28), (u2, 29), (ar, 30)
27	⟨GO⟩|⟨LF⟩|Srv|Usr	Ac	(u, 30), (ac, 1)
28	⟨S1⟩|Srv|Usr	Ar	(ar, 31)
29	⟨S2⟩|Srv|Usr	Ar	(ar, 32)
30	⟨LF⟩|Srv|UsrAct	Ac	(u1, 31), (u2, 32), (ac, 2)
31	⟨S1⟩|⟨LF⟩|Srv|Usr	Ac	(s1, 33), (ac, 3)
32	⟨S2⟩|⟨LF⟩|Srv|Usr	Ac	(s2, 34), (ac, 4)
33	SrvAct1|Usr	Ac	(sa1, 35), (ac, 5)
34	SrvAct2|Usr	Ac	(sa2, 36), (ac, 6)
35	SrvGo|Usr|Agt1	Ac	(sg, 37), (a1, 23), (ac, 7)
36	SrvGo|Usr|Agt2	Ac	(sg, 38), (ac, 8)
37	⟨GO⟩|Srv|Usr|Agt1	Ac	(u, 39), (a1, 27), (ac, 9)
38	⟨GO⟩|Srv|Usr|Agt2	Ac	(u, 40), (ac, 11)
39	Srv|UsrAct|Agt1	Ac	(u1, 41), (u2, 42), (a1, 30), (ac, 13)
40	Srv|UsrAct|Agt2	Ac	(u1, 43), (u2, 44), (ac, 14)
41	⟨S1⟩|Srv|Usr|Agt1	Ac	(a1, 31), (ac, 17)
42	⟨S2⟩|Srv|Usr|Agt1	Ac	(a1, 32), (ac, 18)
43	⟨S1⟩|Srv|Usr|Agt2	Ac	(ac, 19)
44	⟨S2⟩|Srv|Usr|Agt2	Ac	(ac, 20)

Figure 3: DMS LTS

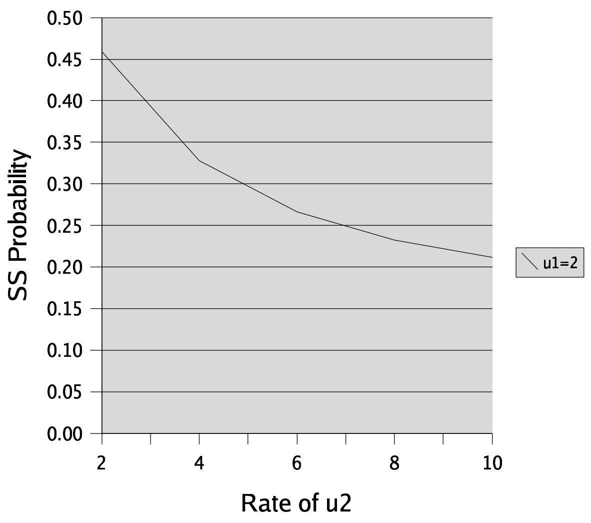

Figure 4: Results for L

when the upper bound is 0.3 by running ETMCC on the CTMC with the CSL formula $\mathcal{S}_{<0.3}(\langle \mathtt{LF} \rangle @ \mathtt{L})$, where the states of the CTMC satisfying atomic proposition $\langle \mathtt{LF} \rangle @ \mathtt{L}$ are those corresponding to the states 1, 2, 3, 4, 10, 23, 27, 30, 31 and 32 of the LTS. It has been found that the formula holds (in the initial state). It is worth pointing out here that, for formula $\mathcal{S}_{\bowtie p}(\Phi)$, ETMCC actually computes and reports also the actual probability that Φ holds "in the long run". So we can study such probability as a function of the rates specifi ed in the input model. Consequently we can vary β in $CTMC(N_0, \beta)$, thus getting different CTMCs and study them. In Fig. 4 the study is performed for all those β with $\beta\, u2$ varying from 2 to 10 and $\beta\, r = \beta_0\, r$ otherwise. From the curve it can be seen that the probability of not using the resources in L decreases when $u2$ increases, which means that, on the average, the percentage of usage of the resources in L increases when the rate of the requests for Service S2 increases. With CSL also qualitative properties can be expressed. For example, the following property specifi es that it is possible that both agents are running at the same time at L:

$$\mathcal{P}_{>0}(\Diamond(\mathtt{Agt1}@\mathtt{L} \wedge (\mathtt{Agt2}@\mathtt{L} \vee \mathtt{Amr}@\mathtt{L})))$$

Verifi cation with ETMCC shows that this property is not satisfi ed by any state (i.e. almost all paths originating from any state satisfy $\neg(\Diamond(\mathtt{Agt1}@\mathtt{L} \wedge (\mathtt{Agt2}@\mathtt{L} \vee \mathtt{Amr}@\mathtt{L})))$. A further property shows that a new service request for Service S1 or S2 can be issued when a previous request is still being processed at locality R.

$$\mathcal{P}_{>0}(\Diamond((\langle \mathtt{S1} \rangle @ \mathtt{L} \vee \langle \mathtt{S2} \rangle @ \mathtt{L}) \wedge \mathtt{Ar}@\mathtt{L}))$$

This property is satisfi ed by all states.

5.2 A virus

This example has been inspired by a similar one in [10]. Although our example is slightly simpler than the one presented in [10], we are able to show some quantitative results which we obtained by means of model-checking, while the treatment of the example in [10] is essentially limited to system specifi cation. We model the *spreading* of a virus in a network. A network is modeled as a set of nodes and the virus running on a node can move arbitrarily from the node to a subset of adjacent nodes, infecting them. At each node, an operating system runs, which upon receiving the virus, can either run it or suppress it. In this paper, for the sake of simplicity we consider simple networks which are in fact grids of $n \times m$ nodes. Each node is connected with its four neighbors (north, south, east, west), except for border nodes, which lack some connections in the obvious way (e.g. the nodes on the east border have no east connection). Moreover, in order to keep the state space at a size which permits us to graphically represent it in this

$$
\begin{array}{lll}
\mathtt{V}_{ij} & \triangleq & (\mathtt{n}_{ij}, \textbf{out}\ \mathtt{V}_{i-1j}\ @\ \mathtt{l}_{i-1j}).\textbf{nil} + \\
& & \text{/* alternative present only for } i > 1 \text{ /*} \\
& & (\mathtt{s}_{ij}, \textbf{out}\ \mathtt{V}_{i+1j}\ @\ \mathtt{l}_{i+1j}).\textbf{nil} + \\
& & \text{/* alternative present only for } i < n \text{ /*} \\
& & (\mathtt{e}_{ij}, \textbf{out}\ \mathtt{V}_{ij+1}\ @\ \mathtt{l}_{ij+1}).\textbf{nil} + \\
& & \text{/* alternative present only for } j < m \text{ /*} \\
& & (\mathtt{w}_{ij}, \textbf{out}\ \mathtt{V}_{ij-1}\ @\ \mathtt{l}_{ij-1}).\textbf{nil} + \\
& & \text{/* alternative present only for } j > 0 \text{ /*}
\end{array}
$$

$$
\begin{array}{lll}
\mathtt{O}_{ij} & \triangleq & (\mathtt{u}_{ij}, \textbf{in}\ !\mathtt{C}\ @\ \mathtt{l}_{ij}).\mathtt{X}_{ij} + \\
& & \text{/* the received virus is undetected and will run/*} \\
& & (\mathtt{d}_{ij}, \textbf{in}\ !\mathtt{C}\ @\ \mathtt{l}_{ij}).\mathtt{O}_{ij} \\
& & \text{/* the received virus is detected and suppressed/*} \\
\mathtt{X}_{ij} & \triangleq & (\mathtt{r}_{ij}, \textbf{eval}\ \mathtt{C}\ @\ \mathtt{l}_{ij}).\mathtt{O}_{ij} \\
& & \text{/* the virus is activated /*}
\end{array}
$$

Figure 5: Specification of an infected network

paper, we assume that the virus can move only to *one* adjacent node. Finally, we refrain from modeling aspects of the virus other than the way it replicates in the network. In particular we do not consider the local effects of the virus and we make the virus die as soon as it has infected one of the neighbors of its locality.

The specification schema of the virus and the operating system running at each node is given in Fig. 5, where a network is conventionally represented as a $n \times m$ matrix of localities l_{ij}. For the verification, we chose $n = m = 3$ with the following initial state: $\mathtt{l}_{11}::\mathtt{O}_{11} \parallel \mathtt{l}_{11}::\langle \mathtt{V}_{11} \rangle$, while $\mathtt{l}_{ij}::\mathtt{O}_{ij}$ for $1 \leq i, j \leq 3$ with $i \neq 1$ or $j \neq 1$. The resulting LTS is shown in Fig. 8 where, for the sake of readability we represent each state of the network as a square box with nine sectors. A blank sector ij corresponds to $\mathtt{l}_{ij}::\mathtt{O}_{ij}$, i.e. a normally running node; a black triangle in sector ij corresponds to the expression $\mathtt{l}_{ij}::\mathtt{O}_{ij} \parallel \mathtt{l}_{ij}::\langle \mathtt{V}_{ij} \rangle$, i.e. a node which has been infected; a thick $\times$ in sector ij corresponds to $\mathtt{l}_{ij}::\mathtt{X}_{ij}$, i.e. a node which is going to run the virus; and $\mathtt{l}_{ij}::\mathtt{O}_{ij} \parallel \mathtt{l}_{ij}::\mathtt{V}_{ij}$—i.e. the virus is running at the node—is represented by a black sector ij.

There are several issues that can be analyzed. First of all we can study the probability that the virus is running at node $\mathtt{l}_{ij}$ within t time-units after the infection of node $\mathtt{l}_{11}$; for instance we can check if such a probability is smaller than a given upper bound p. This question becomes more interesting when we define the rates associated to the detection (resp. undetection) of the virus in such a way that the operating systems of the localities on the diagonal from bottom-left to top-right—$\mathtt{O}_{31}$, $\mathtt{O}_{22}$, and $\mathtt{O}_{13}$—have a relatively high rate of detection and can be considered as a firewall to protect the nodes $\mathtt{l}_{32}$, $\mathtt{l}_{33}$, and $\mathtt{l}_{23}$.

We can now express the above property in CSL for locality $\mathtt{l}_{33}$ and $p = 0.2$ as follows:

$$\mathcal{P}_{\leq 0.2}(\neg(\mathtt{V}_{33}@\mathtt{l}_{33})\,\mathcal{U}^{\leq t}\,\mathtt{V}_{33}@\mathtt{l}_{33})$$

The ETMCC model checker gives as a result the list of states where the formula holds. Moreover, as for the case of steady state probabilities, the tool provides also, for each state, the actual probability that starting from such a state, the formula $\neg(\mathtt{V}_{33}@\mathtt{l}_{33})\,\mathcal{U}^{\leq t}\,\mathtt{V}_{33}@\mathtt{l}_{33}$ is satisfied. In Fig. 6 the probability to reach, from the initial state, a state where the virus is running in locality $\mathtt{l}_{33}$ is presented for time values ranging from 1 to 10 with $\beta_0\, e_{ij} = \beta_0\, n_{ij} = \beta_0\, s_{ij} = \beta_0\, w_{ij} = \beta_0\, r_{ij} = 2$ for $1 \leq i, j \leq 3$, $\beta_0\, d_{31} = \beta_0\, d_{22} = \beta_0\, d_{13} = 10$, and $\beta_0\, d_{ij} = 1$ otherwise, $\beta_0\, u_{31} = \beta_0\, u_{22} = \beta_0\, u_{13} = 1$, and $\beta_0\, u_{ij} = 10$ otherwise. We performed similar analysis for different values of the detection (resp. undetection) rates of the firewall. In particular for d_{31}, d_{22}, d_{13} and u_{31}, u_{22}, u_{13} range over $[1, \ldots, 10]$, with $d_{(4-i)i} + u_{(4-i)i}$ constant for $1 \leq i \leq 3$ (and equal to 11). For the sake of readability, in Fig. 7 we show the results only for $d_{31}, d_{22}, d_{13} \in \{1, 6, 10\}$ and $u_{31}, u_{22}, u_{13} \in \{1, 5, 10\}$. The results clearly indicate that for high detection rates the probability for locality l_{33} to run the virus within a certain time interval is lower.

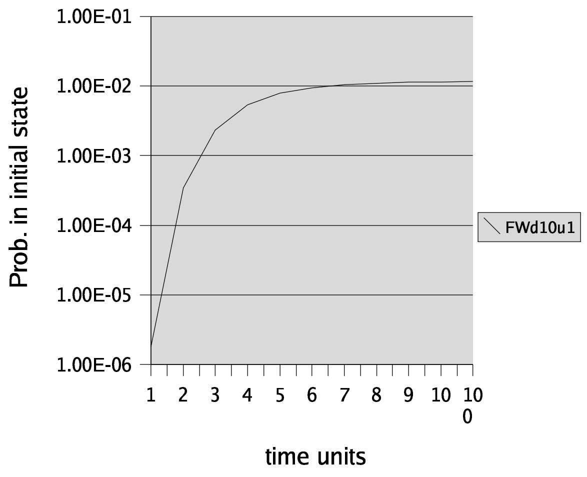

Figure 6: Probability that virus reaches $\mathtt{l}_{33}$ after t time-units

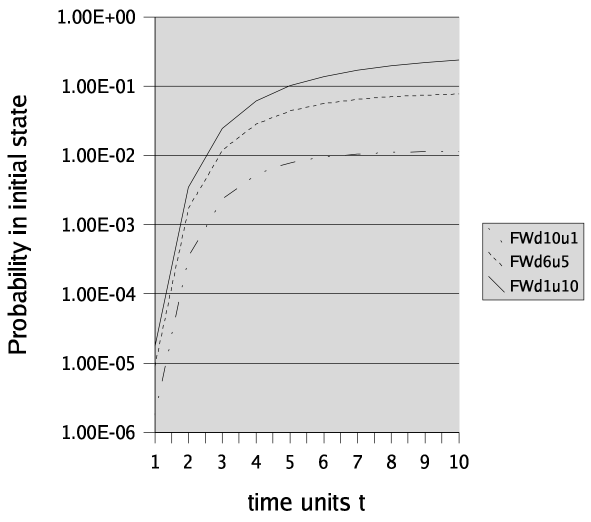

Figure 7: Results for Firewalls with different detection capability

Finally we show two examples of qualitative properties. Both properties clearly show the limitations of this, simplified, model of the spreading of a virus in a network. In STOCKLAIM more realistic models can easily be specified. Their analysis requires adequate tool support, not vailable at the time of writing the present paper, such as the automatic generation of a CTMC from a STOCKLAIM specification, in order to generate their considerably larger state-spaces.

The first property states that it is never the case that the virus is running at two different localities in the network at the same time. Actually this property is a conjunction of many properties, each of them stating that a certain pair of localities cannot both have a virus running at the same time.

$$\bigwedge_{\substack{1 \leq i,y,j,z \leq 3 \\ (i \neq y \vee j \neq z)}} \mathcal{P}_{>0}(\Diamond(\mathtt{V_{ij}@l_{ij}} \wedge \mathtt{V_{yz}@l_{yz}}))$$

For example, verification shows that the formula $\mathcal{P}_{>0}(\Diamond(\mathtt{V_{13}@l_{13}} \wedge \mathtt{V_{21}@l_{21}}))$ is not satisfied in any state.

The second qualitative property shows that we can always reach a state in which the network is free of viruses forever.

$$\mathcal{P}_{>0}(\Diamond\mathcal{P}_{\leq 0}(\Diamond(\bigvee_{1 \leq i,j \leq 3} \mathtt{V_{ij}@l_{ij}})))$$

This highly desired property is satisfied in every state. The outer-most path-formula in the last formula is satisfied only in state 28.

6 Conclusions and Future Work

In this paper we introduced STOCKLAIM, a stochastic extension of cKLAIM, that makes it possible to integrate the modeling of quantitative and qualitative aspects of mobile systems. The starting point of our proposal is to use continuous random variables with exponential distributions for modeling action durations.

We presented a formal operational semantics for STOCKLAIM that associates a labelled transition system to each STOCKLAIM network and showed how it can be transformed into a Continuous Time Markov Chain (CTMC). We worked out two small examples, one modeling a distributed mobile service and another modeling the spreading of a virus through a network. We analyzed some of the qualitative and quantitative aspects of these examples such as resource usage and velocity of spreading of a virus.

The results in this paper show the viability of the approach and give a first impression of its practical usefulness when addressing quantitative aspects of mobile systems. In particular, they show that the choice of an asynchronous model of computation for the base-language greatly simplifies the definition of the operational semantics of the stochastic extension. Such definition is further simplified by the fact that the auxiliary structural congruence includes, among others, associativity and commutativity of parallel and non-deterministic operators. These advantages are not restricted to KLAIM, but can be exploited for other languages based on an asynchronous model of interaction, s.a. Linda-based languages.

The ideas proposed in this paper give rise to a whole range of related interesting research questions. We address some of them briefly:

1. *Logics for mobility & stochastic behaviour.* STOCKLAIM provides an explicit way for addressing quantitative issues of mobility and locality in specifications of dynamic wide area networks. The examples presented in this paper show that the analysis of such issues can be addressed only in an indirect way by means of stochastic model checking of CSL formulas. Therefore STOCKLAIM needs to be equipped with a proper logic for the expression of both stochastic notions and mobility issues. Obviously, good starting points seem to be the logic(s) for KLAIM [4] and CSL [1, 18].

2. *Tools.* Given the complexity of the state space of more realistic models for quantitative aspects of mobility, a first necessary step is the development of a tool that generates CTMCs from cKLAIM expressions. This tool could be designed as a front-end to the ETMCC model checker. A further

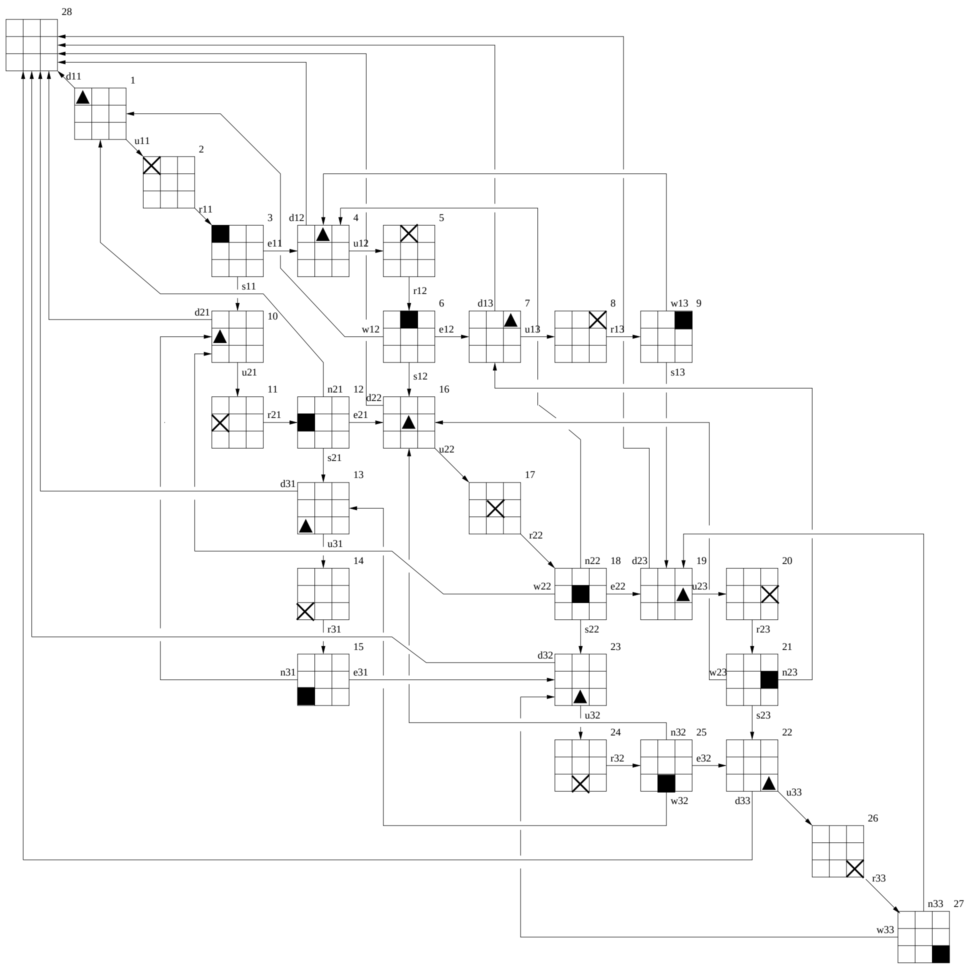

Figure 8: LTS for the specification in Fig. 5

step is the development of a model checker for the new temporal logic for mobility and stochastic behaviour.

3. *Extending* KLAIM *coverage.* In the present paper we addressed only a very limited subset of KLAIM. It would be interesting to cover a more significant subset of the language, at least μKLAIM, or KLAIM itself, or OPEN KLAIM. Moreover, the introduction of *rate expressions* would facilitate the specification of rates, also by means of variables. The possibility to exchange rates in communications and to express rates as functions of localities should be addressed as well. Another interesting extension would be the introduction of probabilistic choice and probabilistic parallel composition.

These are some of the topics we are currently investigating, on which we expect to obtain interesting results in the near future.

7 Acknowledgments

We want to thank Joost-Pieter Katoen for fruitful discussions on the issues raised by the research presented in this paper.

References

[1] C. Baier, B Haverkort, H. Hermanns, and J. Katoen. Automated performance and dependability evaluation using model checking. In *Computer Performance Evaluation*, pages 261–289. Springer-Verlag, 2002.

[2] C. Baier, J. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time markov chains. In J. Baeten and S. Mauw, editors, *Concur '99*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, 1999.

[3] L. Bettini, R. De Nicola, and M. Loreti. Formulae meet programs over the net: a framework for reliable network aware programming, 2003. (submitted for publication. Available at: `http://music.dsi.unifi.it`).

[4] L. Bettini, V. Non, R. De Nicola, G. Ferrari, D. Gorla, M. Loreti, E. Moggi, R. Pugliese, E. Tuosto, and B. Venneri. The Klaim Project: Theory and Practice. In C. Priami, editor, *Global Computing: Programming Environments, Languages, Security and Analysis of Systems*, volume 2874 of *Lecture Notes in Computer Science*, pages 88–150. Springer-Verlag, 2003.

[5] J. Bradley and N. Davies. Reliable Performance Modeling with Approximate Synchronisations. In J. Hillston and M. Silva, editors, *Proceedings of the 7th workshop on process algebras and performance modeling*, pages 99–118. Prensas Universitarias de Zaragoza, September 1999.

[6] P. Buchholz, J.-P Katoen, P. Kemper, and C. Tepper. Model-checking large structured Markov chains. *The Journal of Logic and Algebraic Programming. Elsevier Science*, 56(1-2):69–96, 2003.

[7] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999. ISBN 0-262-03270-8.

[8] C. Courcoubetis and M. Yannakakis. Verifying Temporal Properties of Finite STate Probabilistic Programs. In *29th Annual Symposium on Foundations of Computer Science*, pages 338–345. IEEE Computer Society Press, 1988.

[9] R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–329, 1998.

[10] A. Di Pierro, C. Hankin, and H. Wiklicky. Probabilistic KLAIM. In R. De Nicola, G. Ferrari, and G. Meredith, editors, *Coordination Models and Languages*, volume 2949 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[11] D. Gelernter. Generative Communication in Linda. *Communications of the ACM*, 7(1):80–112, 1985.

[12] S. Gilmore, J. Hillston, L. Kloul, and M. Ribaudo. PEPA nets: a structured performance modelling formalism. *Performance Evaluation - An International Journal. Elsevier*, 54:79–104, 2003.

[13] D. Gorla and R. Pugliese. A Semantic Theory for Global Computing Systems, 2004. (Submitted for publication. Available at http://www.dsi.uniroma1.it/˜gorla/papers/bis4k-full.pdf).

[14] S. Hart and M. Sharir. Probabilistic Temporal Logics for Finite and Bounded Models. In *29th Annual Symposium on Foundations of Computer Science*, pages 1–13. IEEE Computer Society Press, 1988.

[15] B. Haverkort. Markovian models for performance and dependability evaluation. In E. Brinksma, H. Hermanns, and J. Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *Lecture Notes in Computer Science*, pages 38–83. Springer-Verlag, 2001.

[16] O. Herescu and C. Palamidessi. Probabilistic Asynchronous π-Calculus. In J. Tiuryn, editor, *FoSSaCS 2000*, volume 1784 of *Lecture Notes in Computer Science*, pages 146–160. Springer-Verlag, 2000.

[17] H. Hermanns, J. Katoen, J. Meyer-Kayser, and M. Siegle. Towards Model Checking Stochastic Process Algebra. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *Integrated Formal Methods - IFM 2000*, volume 1945 of *Lecture Notes in Computer Science*, pages 420–439. Springer-Verlag, 2000.

[18] H. Hermanns, J Katoen, J. Meyer-Kayser, and M. Siegle. A tool for model-checking Markov chains. *International Journal on Software Tools for Technology Transfer*, 4(2):153–172, 2003.

[19] V. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.

[20] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In J.P. Katoen and P. Stevens, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *Lecture Notes in Computer Science*, pages 52–66. Springer-Verlag, 2002.

[21] D. Lehmann and S. Shelah. Reasoning with Time and Chance. *Information and Control*, 53:165–198, 1992.

[22] C. Priami. Stochastic π-Calculus. *The Computer Journal. Oxford University Press.*, 38(7):578–589, 1995.

[23] H. Younes and R. Simmons. Probabilistic verifi cation of discrete event systems using acceptance sampling. In E. Brinksma and K. Larsen, editors, *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 223–235. Springer-Verlag, 2002.