Formal modeling and quantitative analysis of 
KLAIM-based mobile systems

Rocco De Nicola
D.S.I. - Univ. di Firenze
v. Lombaro 6/17,
I50134 Firenze, Italy
denicola@dsi.unifi.it

Diego Latella
C.N.R.-I.S.T.I. - A. Faedo
v. Moruzzi 1,
I56124 Pisa, Italy
Diego.Latella@isti.cnr.it

Mieke Massink
C.N.R., I.S.T.I. - A. Faedo
v. Moruzzi 1,
I56124 Pisa, Italy
Mieke.Massink@isti.cnr.it

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. In this paper we propose STOCCLAIM, 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. We present a formal operational semantics of STOCCLAIM, which associates a labeled transition system to each STOCCLAIM network and a translation to Continuous Time Markov Chains for quantitative analysis. We also show how STOCCLAIM can be used by means of a simple example, i.e. the modeling of the spreading of a virus.

Categories and Subject Descriptors
D.2.4 [Software Engineering]: Software/Program Verification—Formal methods

General Terms
Languages, Performance, Verification

Keywords
Formal Modeling and Validation, Stochastic Behavior, Mobile Systems, Coordination Languages

1. INTRODUCTION
Components of modern widely distributed ubiquitous systems are characterized by highly dynamic behavior 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 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 [13]. 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 toward the extension of KLAIM with stochastic features. In particular, we focus on (a variant of) cKLAIM, the core subset of KLAIM, first introduced in [15] 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 STOCCLAIM, 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 STOCCLAIM network specification. The associated LTS defines in turn a stochastic process [21]—and in particular a Continuous Time Markov Chain (CTMC)—which can be used for checking stochastic properties of the behavior of the network.

CTMCs provide a modeling framework which has proved extremely useful for practical analysis of quantitative aspects of system behavior. Moreover, in recent years proper stochastic extensions of temporal logics have been proposed and efficient algorithms for checking the satisfiability of formulas of such logics on CTMCs (i.e. stochastic model checkers) have been implemented [20, 22, 24, 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 behavior.
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 \( nil \) 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 \( \text{out } l' @ l \) a process can write the datum \( l' \) in repository \( l \). With an input action \( \text{in } T @ l \) a process can withdraw a datum from repository \( l \). Processes can be written to/from withdrawn from a repository as well. The action \( \text{eval } Q @ l \) spawns process \( Q \) at repository \( l \) and action \( \text{newloc } u \) serves for creating a new node; thus providing a means for modeling dynamic network architectures. Action \( \text{newloc } u \) 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 of the definition of CTMCs.

The parameter which completely characterizes an exponentially distributed random variable is its rate \( \lambda \), which is a positive real number.\(^1\) Consequently, we equip each action \( a \) with a rate \( \lambda \) and call the resulting pair a stochastic action. The intended meaning of \( \langle a, \lambda \rangle \) is that the time taken for the complete execution of action \( a \) is a random variable distributed as \( \text{EXP}(\lambda) \).

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 an illustrative example—namely the spreading of a network virus—while some conclusions and an outline of future research are presented in Sect. 6. An extended version of this paper can be found in [10].

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

2. RELATED WORK

In our proposal, at a conceptual level, we follow essentially the same approach as Priami in [23] where he extends the \( \pi \)-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 \( \pi \)-Calculus processes is synchronous, while that of cKLAIM 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 behavior 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 behavioral 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 [11] 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 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 [23]. In the proposal of [11] 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
The syntax of StocKLaIM, given in Table 1, is exactly the same as that of KLaIM, except that processes have a richer parallel operator, and for the addition of an explicit choice composition operator. A typical problem with as many transitions as those of the components of the choice (and of possibly nested parallel compositions). A 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 \((a, 2\lambda), P\). 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 \textit{proved transition systems} as in [23] that permit distinguishing left and right components by labeling transitions starting from derivations labeled 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 \([+, \cdot]\) 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 behavior 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 behaviors are modeled via process definitions; it is assumed that each identifier \(A\) has a \textit{single defining equation} of the form \(A \triangleq P\) 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.

### 3. SYNTAX OF StocKLaIM

Let \(L\), ranged over by \(l, l', l_1, \ldots\), be a set of \textit{localities}, \(U\), ranged over by \(u, u', u_1, \ldots\), be a set of \textit{locations}, \(A\), ranged over by \(A, A', A_1, B, \ldots\), be a set of \textit{locality variables}, \(A\), ranged over by \(r, r', r_1, r_2, w, x, y, \ldots\), be a set of \textit{rate names}. We assume that the above sets are mutually disjoint. Moreover, we let \(l, l'\) range over \(L \cup U\).

The syntax of StocKLaIM, given in Table 1, is exactly the same as that of KLaIM, except that processes have a richer action \textit{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 \textit{out} (resp. \textit{in}) actions, in much the same way as data. A network node \(l :: \{l'\}\) intuitively means that value \(l'\) is stored, or located, in node, or locality, \(l\). Similarly, for process \(P\), \(l :: \{P\}\) 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 \textit{network parallel operator} \(\parallel\). Given network \(N\), the set of values located in locality \(l\) coincides with all those \(l'\) and \(P\) such that \(l :: \{l'\}\) or \(l :: \{P\}\) 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 \textit{rate name} \(r\). Rate names are mapped to rate values by means of binding functions, which are (partial) functions from \(R\) to \(R^+\). The main reason for using \textit{rate names} instead of just rates is related to the way we deal with the \textit{race condition} semantics for the non-deterministic choice operator \((P_1 + P_2)\) and interleaving one \((P_1 \parallel P_2)\). 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).
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, 15].

The transition relation is defined over (network) configurations, i.e., triples \((L, \beta, N)\)—henceforth written as \(L, \beta \vdash N\)—where \(L\) is a finite set of localities, \(\beta: \mathcal{R} \rightarrow \mathbb{R}^+\) is a mapping from rate names to rates, with \((\text{dom}(\beta))\)—the domain of \(\beta\)—also finite, and \(N\) a StocKLaIm network expression. Let \((\text{Loc}\ N)\) denote the set of all localities occurring free \(^3\) in \(N\) and \((\text{Rat}\ N)\) be the set of all rate names occurring in \(N\). We require \((\text{Loc}\ N) \subseteq L\) and \((\text{Rat}\ N) \subseteq (\text{dom}\ \beta)\). Finally, we require that all rate names occurring in \(N\) be distinct and that for expressions of the form \(\text{in} @ I'.P\), (i) there exists at most one free occurrence of \(A\) in \(P\) which is not the first argument of an \text{out} or \text{eval} operator, and (ii) there exists no defining equation for \(A\).

The Structural Congruence is the smallest congruence on configurations, containing the identity, 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 renaming. The law for rate renaming (REN) states that the rate names occurring in \(N\) can be replaced by means of a substitution \(\theta\). \(\theta\) is a function in \(\mathcal{R} \rightarrow \mathcal{R}\) such that (i) it preserves rate names uniqueness (i.e., it is injective) (ii) it is defined exactly on the rate names occurring in \(N\) (i.e., \((\text{dom}(\theta)) = (\text{Rat}(N))\), and (iii) the substitution does not interfere with the current binding (i.e., \((\text{dom}(\theta)) \cap (\text{rng}(\theta)) = \emptyset\)). 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 \(\theta\)).

The Reduction Relation \(\rightarrow\) is the smallest relation induced by the rules of Table 3. Let \(C\), ranged over by \(c, c_1, c_2, \ldots\) 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 \(\mathcal{R}\) elements, where we can use set inclusion for locality sets and binding-domain sets, and lexicographic order for network terms. For configuration \(c \in C\), let \(D \subseteq c\) be the smallest set such that (i) \(c \in D\), and (ii) if \(d \in D\), \(d', r \in \mathcal{R}\) and \((d', r, c') \in \rightarrow\) can be derived using the rules and laws of tables 3 and 2, then also \(c' \in D\).

The operational semantics of a network \(N_0\), with rate names defined by binding \(\beta_0\), associates a LTS, \(Ts(N_0, \beta_0) := \text{def} (C, \Lambda, \rightarrow, c_0)\) to \(N_0\), with \(\beta_0: C \rightarrow \mathbb{R}^+\) the set of states of the LTS, where the initial state of the LTS, \(c_0 \in C\), is the standard representative of \((\text{Loc}(N_0), \beta_0 \vdash N_0; \Lambda \subseteq \mathcal{R}\) is the set of labels (i.e., rate names) of the LTS and \(\rightarrow \subseteq C \times \Lambda \times C\) is its transition relation, as deduced from the Reduction Rules and the Congruence Laws. As usual, \(c \xrightarrow{r} c'\) stands for \((c, r, c') \in \rightarrow\); moreover, if \(c\) is the state \(L, \beta \vdash N, \text{ we let } \beta, \text{ denote the binding } \beta\) 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 an existing (possibly remote) locality. Notice that, if the destination locality does not exist then the rule cannot be applied. In order to preserve uniqueness of rate names, when executing action \((\text{out} \; Q @ I', r).P\) process \(Q'\) is stored instead of \(Q\); \(Q'\) is obtained from \(Q\) by means of function \(\text{RN}\), defined in Fig. 1, which renames all rate names in \(Q\) into fresh names; function \(\text{RN}\) returns also a new binding where the fresh names are bound to the same names as those the original ones were bound to. Notice that, in the definition of function \(\text{RN}\), rate name uniqueness is guaranteed by means of function \(\text{choose}\) and by a proper sequentialization of the application of \(\text{RN}\) to the components of (process) parallel and non-deterministic composition; such sequentialization is achieved by means of \(\beta^\theta\). The selection criterion of \(\text{choose}\) 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 \(\text{in} \; T \otimes I'\) is a blocking action that can be performed only if the required datum is present at the chosen locality \(I'\). Moreover, if the argument \((T)\) is a locality variable (\(u\)), resp. a process variable (\(\text{var}\)), 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 \(\text{new loc} \; u\) is a fresh name, i.e. a name not present in the set of all used names, \(L\); function \(\text{choose}\) is used to choose such a new element of \(\mathcal{C} \setminus L\), 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, non-deterministic choice and to take advantage of structural congruence. For the rest, the rules should be self-explanatory.

4.2 From LTSs to CTMCs

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, 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 rely on syntactical restrictions, like avoiding certain (combinations of) operators,
(OUTL) \[ L, \beta \vdash l :: (\text{out} \ t' @ l', r), \text{P} \ | \ t' :: P' \xrightarrow{\text{t}'} L, \beta \vdash l :: \text{P} \ | \ l' :: P' \ | \ t' :: \{ \text{t}'' \} \]

(OUTP) \[ L, \beta \vdash l :: (\text{out} \ Q @ l', r), \text{P} \ | \ l' :: P' \xrightarrow{\text{t}'} L, \beta \vdash l :: \text{P} \ | \ l' :: P' \ | \ l :: \{ Q' \} \]

where \( (Q', \beta) = \text{RN}(Q, \beta) \)

(INL) \[ L, \beta \vdash l :: (\text{in} \ t \ @ l', r), \text{P} \ | \ l' :: \{ t'' \} \xrightarrow{\text{t}'} L, \beta \vdash l :: \text{P} \ | \ l' :: \{ t'' \} \]

where \( \sigma = \{ \{ t''/u \}, \text{if} \ T = u \}

\quad \quad \text{if} \ T = t' \)

(IPN) \[ L, \beta \vdash l :: (\text{in} \ (A) @ l', r), \text{P} \ | \ l' :: \{ P' \} \xrightarrow{\text{t}'} L, \beta \vdash l :: \text{P} \ | \ l' :: \{ P' \} \]

where \( \ell = \text{choose} \ l \in L \setminus L \)

(NLC) \[ L, \beta \vdash l :: (\text{newloc} \ u, r), \text{P} \xrightarrow{\text{t}'} L \cup \{ \ell \}, \beta \vdash l :: \text{P} \ | \ u \ :: u \]

where \( \ell = \text{choose} \ l \in L \setminus L \)

(EVA) \[ L, \beta \vdash l :: (\text{eval} \ Q @ l', r), \text{P} \ | \ l' :: P' \xrightarrow{\text{t}'} L, \beta \vdash l :: \text{P} \ | \ l' :: P' \ | \ Q' \]

where \( \beta = \text{RN}(Q, \beta) \)

(PIN) \[ L, \beta \vdash l :: \text{P} \xrightarrow{\text{t}'} L', \beta' \xrightarrow{(A, \beta)} N \]

where \( A = \text{P} \) and \( (P', \beta') = \text{RN}(P, \beta) \)

(CHO) \[ L, \beta \vdash l :: P_1 \ | \ N \xrightarrow{\text{t}'} L', \beta \xrightarrow{(A, \beta)} N' \quad \text{where} \quad A = \text{P} \quad \text{and} \quad (P', \beta') = \text{RN}(P, \beta) \]

(PAR) \[ L, \beta \vdash N_1 \xrightarrow{\text{t}'} L', \beta \xrightarrow{(A, \beta)} N' \quad \text{where} \quad A = \text{P} \quad \text{and} \quad (P', \beta') = \text{RN}(P, \beta) \]

(STC) \[ L, \beta \vdash N \equiv L_1, \beta_1 \vdash N_1, \beta_1 = \beta_0^{-1} \]

\[ L_2, \beta_2 \vdash N_2 \equiv L_3, \beta_3 \vdash N_3, \]

\[ L_4, \beta_4 \vdash N_4 \equiv L_3, \beta_3 \vdash N' \]

\[ L, \beta \vdash N \xrightarrow{\text{t}'} L', \beta \xrightarrow{(A, \beta)} N' \quad \text{where} \quad A = \text{P} \quad \text{and} \quad (P', \beta') = \text{RN}(P, \beta) \]

Table 3: Reduction Rules of StocKlaim

\[
\text{RN}(\text{nil}, \beta) =: = \text{def} (\text{nil}, \beta) \\
\text{RN}((a, r).P, \beta) =: = \text{def} ((a, r').P', \beta') \quad \text{where} \quad r' = \text{choose} \ r_1 \in R \setminus \text{dom} \beta \\quad (P', \beta') = \text{RN}(P, \beta)[(r', r')]
\]

\[
\text{RN}(P \circ Q, \beta) =: = \text{def} (P' \circ Q', \beta', \{ \circ \}) \quad \text{where} \quad (P', \beta') = \text{RN}(P, \beta) \quad (Q', \beta') = \text{RN}(Q, \beta')
\]

\[
\text{RN}(A, \beta) =: = \text{def} (A, \beta), \text{if} \ A = \text{def} P
\]

Figure 1: Definition of function \( \text{RN} \)

and they have been studied extensively in the context of traditional process algebra (see e.g. [12]). 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 [21]; we suggest [17] for a gentle introduction). For the purposes of the present paper it suffices to recall that a CTMC \( M \) is a pair \((S, R)\) where \( S \) is a finite set of states and \( R : S \times S \rightarrow \mathbb{R}_{\geq 0} \) is the rate matrix. The rate matrix characterizes the transitions between the states of \( M \). If \( R(s, s') \neq 0 \) 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 \( 1 - e^{-R(s, s') \cdot t} \). If \( R(s, s') = 0 \) then no such a transition can take place\(^3\). 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 \( \beta \) and assuming \( T S(N, \beta) = \{ C, \Lambda, \rightarrow, c_0 \} \) finite, the CTMC \((S, R)\) associated to \( N \) with binding \( \beta \), denoted by \( CTMC(N, \beta) \), is defined as follows: the set \( S \) of states coincides with \( C \), and for all \( c, c' \in C \)

\[
R(c, c') =: = \text{def} \begin{cases} \sum_{r \in T_{c,c'}} \beta(r, r) & \text{if} \ T_{c,c'} \neq 0 \\ 0 & \text{if} \ T_{c,c'} = 0 \end{cases}
\]

where \( T_{c,c'} =: = \text{def} \{ r' \in c' \rightarrow c \} \quad \text{where} \quad E_{c,c'} \)

\[5. MODELING A VIRUS\]

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

5.1 Stochastic model-checking

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 [19] 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 [20]. For the example 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 process is running at a certain locality. We do this by using atomic propositions of the form \( A@l \) where \( A \) is a process identifier and \( l \) a

\[3\]The reader should be warned that the above intuitive interpretation of 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 \( s \).
locality. A state $s$ in the CTMC corresponding to a LTS will satisfy $A\beta L$, i.e., will be labeled by such an atomic proposition, if and only if the associated configuration $c$ in the LTS (belongs to the same congruence class which contains a configuration which) is of the form $I, \beta \vdash N$ and $I :: A$ 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 steady-state and the transient behavior of the mobile systems.

The meaning of atomic propositions ($\alpha$), negation ($\neg$) and disjunction ($\lor$) is standard; using these operators, other boolean operators such as conjunction ($\land$), implication ($\Rightarrow$), true (TRUE) and false (FALSE), and so forth, can be defined, as usual. The state-formula $\mathcal{S}_{\rho-p}(\Phi)$ asserts that the steady-state probability for the set of states satisfying $\Phi$, the $\Phi$-states, meets the bound $\forall p$. $\mathcal{P}_{\rho-p}(\varphi)$ asserts that the probability measure of the set of paths satisfying $\varphi$ meets the bound $\forall p$. The operator $\mathcal{P}_{\rho-p}(\cdot)$ 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 $\varphi$ and $\forall \varphi$ is valid if all paths satisfy $\varphi$. In CSL, the formula $\mathcal{P}_{\rho-1}(\varphi)$ holds if almost all paths satisfy $\varphi$. Moreover, clearly $\exists \varphi$ holds whenever $\forall \rho(\varphi)$ holds. Thus, qualitative as well as stochastic properties can be expressed in CSL.

In CTL, a path satisfies an until-formula $\mathcal{H}\varphi\psi$ if there is a state on the path where $\psi$ holds, and at every preceding state on the path, if any, $\varphi$ holds. The CSL counterpart, $\mathcal{H}\varphi\psi$ is satisfied by a path if $\psi$ holds at time $t \in I$ and at every preceding state on the path, if any, $\varphi$ holds. In CSL, temporal operators like $\varphi$, $\Box$ and their real-time variants $\varphi^t$ or $\Box^t$ can be derived, e.g., $\mathcal{P}_{\rho-1}(\varphi^t) = \mathcal{P}_{\rho-1}(\mathcal{H}\varphi\psi)$ and $\mathcal{P}_{\rho-1}(\Box^t \varphi) = \mathcal{P}_{\rho-1}(\varphi \Rightarrow \Box^t \varphi)$. The untimed next- and until-operators are obtained by $X \varphi = \varphi^t \land \varphi \land \Box^t \varphi = \varphi \land \Box^t \varphi$ for $I = [0, \infty)$.

Four different types of performance and dependability measures

\begin{align*}
\mathbf{V}_{ij} & \triangleq (\text{out }\mathbf{V}_{i,j-1} \land \mathbf{1}_{i,j-1} \land \mathbf{n}_{j} \land \text{nil}) + \\
& \text{/* alternative present only for } i > 1 */ \\
\mathbf{0}_{ij} & \triangleq (\text{in }\mathbf{C} \land \mathbf{1}_{i,j} \land \mathbf{u}_{j} \land \mathbf{x}_{j}) + \\
& \text{/* alternative present only for } i > 1 */
\end{align*}

Figure 2: Specification of an infected network

The specification schema of the virus and the operating system running at each node is given in Fig. 2, where a network is conventionally represented as an $n \times m$ matrix of localities $L_{ij}$. For the verification, we chose $n = m = 3$ with the following initial state: $L_{11} = \text{nil}$ and $L_{11} = \text{nil}$, while $L_{11} \neq L_{11}$ for $i = 1$ and $i \neq 1$. The resulting LTS is not shown for space reasons; it consists of 28 states and 50 transitions.

There are several issues that can be analyzed. First of all we can study the probability that the virus is running at node $L_{ij}$ within $t$ time-units after the infection of node $L_{11}$; for instance we can...
check if such a probability is slightly smaller than a given upper bound \( p \).
This question becomes more interesting when we define the rates associated to the detection (resp. lack of detection) of the virus in such a way that the operating systems of the localities on the diagonal from bottom-left to top-right—\( D_1 \), \( D_2 \), and \( D_3 \)—have a relatively high rate of detection and can be considered as a firewall to protect the nodes \( 1, 2, \) and \( 3 \).

We can now express the above property in CSL for locality \( 3 \) and \( p = 0.2 \) as follows:

\[
P_{\leq 0.2}(\neg (V_{33} \land 0 \leq 1) U^{<1} V_{33} \land 0 \leq 1)
\]

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 \( \neg (V_{33} \land 0 \leq 1) U^{<1} V_{33} \land 0 \leq 1 \) is satisfied. In the lowest curve of Fig. 3 the probability to reach, from the initial state, a state where the virus is running in locality \( 3 \) is presented for time values ranging from 1 to 10 with \( \beta_0 e_{ij} = \beta_0 s_{ij} = \beta_0 s_{ij} = \beta_0 u_{ij} = \beta_0 r_{ij} = 2 \) for \( 1 \leq i \leq 3 \), \( \beta_0 d_{31} = \beta_0 d_{22} = \beta_0 d_{33} = 10 \), and \( \beta_0 d_{ij} = 1 \) otherwise, \( \beta_0 u_{31} = \beta_0 u_{22} = \beta_0 u_{33} = 1 \), and \( \beta_0 u_{ij} = 10 \) otherwise. We performed similar analyzes for different values of the detection (resp. lack of detection) rates of the firewall. In particular for \( d_{31}, d_{22}, d_{13} \) and \( u_{31}, u_{22}, u_{13} \) ranging over \( 1, 2, \ldots, 10 \), with \( d_{1+1} + u_{1+1} \) constant for \( 1 \leq i \leq 3 \) (and equal to 11).

For the sake of readability, in Fig. 3 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 \( 3 \) to run the virus within a certain time interval is lower.

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 available 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.

\[
P_{\leq 0.2}(\diamond (V_{12} \land 0 \leq 1) \land V_{32} \land 0 \leq 1)
\]

For example, verification shows that the formula \( P_{\leq 0.2}(\diamond (V_{12} \land 0 \leq 1) \land V_{32} \land 0 \leq 1) \) 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.

\[
P_{\leq 0.2}(\diamond P_{\leq 0.2}(\diamond (\bigvee_{1 \leq i,j \leq 3} V_{ij} \land 0 \leq 1)))
\]

This highly desired property is satisfied in every state. The outermost path-formula in the last formula is satisfied only by one state, which is absorbing.

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 labeled transition system to each STocKLAIM network and showed how it can be transformed into a Continuous Time Markov Chain (CTMC). We worked out a small example modeling the spreading of a virus through a network. We analyzed some of the qualitative and quantitative aspects of this example such as the velocity of spreading of the 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. There is a need for equipping STocKLAIM with a logic for the integrated expression of functional, stochastic and mobility aspects of behavior. Moreover, proper tools for supporting system modeling and verification based on STocKLAIM and related logic should be developped. Finally, we plan also to cover more significant subsets of KLAIM.

7. ACKNOWLEDGMENTS

We want to thank Joost-Pieter Katoen for fruitful discussions on the issues raised by the research presented in this paper. We also thank the anonymous reviewers for their valuable comments.
8. REFERENCES


