ELSEVIER

Performance Evaluation 50 (2002) 189–218

PERFORMANCE EVALUATION
An International Journal

www.elsevier.com/locate/peva

Performance evaluation with temporal rewards

Jeroen P.M. Voeten[1]

Section of Information and Communication Systems, Faculty of Electrical Engineering/Eindhoven Embedded Systems Institute, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands

Abstract

Today many formalisms exist for specifying complex Markov chains. In contrast, formalisms for specifying rewards, enabling the analysis of long-run average performance properties, have remained quite primitive. Basically, they only support the analysis of relatively simple performance metrics that can be expressed as long-run averages of *atomic rewards*, i.e. rewards that are deductible directly from the individual states of the initial Markov chain specification. To deal with complex performance metrics that are dependent on the accumulation of atomic rewards over sequences of states, the initial specification has to be extended explicitly to provide the required state information.

To solve this problem, we introduce in this paper a new formalism of *temporal rewards* that allows complex quantitative properties to be expressed in terms of temporal reward formulas. Together, an initial (discrete-time) Markov chain and the temporal reward formulas *implicitly* define an extended Markov chain that allows the determination of the quantitative property by traditional techniques for computing long-run averages. A method to construct the extended chain is given and it is proved that this method leaves long-run averages invariant for atomic rewards. We further establish conditions that guarantee the preservation of ergodicity. The construction method can build the extended chain in an on-the-fly manner allowing for efficient simulation.

Keywords: Path-based reward variables; Reward functions; Temporal logic; Performance evaluation; Markov chains

1. Introduction

To manage complexity and to shorten design cycles, system-level methods for specification and design are becoming more and more important. System-level methods focus on the creation of executable system models, describing a system in the earliest phases of the design process. They allow quantitative (performance) and qualitative (correctness) system properties to be analysed before the system is actually being realised in terms of hardware and software components. The analysis results are used as input for taking structured and well-founded design decisions.

Today many formalisms exist for modelling complex systems. In this paper, we focus on formalisms that utilise discrete-time Markov chains to analyse performance properties. Examples of such formalisms

[1] This research is supported by PROGRESS, the embedded systems research program of the Dutch organisation for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW.
E-mail address: j.p.m.voeten@tue.nl (J.P.M. Voeten).

0166-5316/02/$ – see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0166-5316(02)00105-0

include, but are not limited to, synchronous probabilistic process calculi (see e.g. [17,32]), asynchronous probabilistic process calculi (see e.g. [19,34]), Petri nets (see e.g. [21]) as well as industrial-strength modelling languages such as Parallel Object-Oriented Specification Language (POOSL) [16,27]. Performance metrics in these formalisms are often defined as long-run average rewards, which can be determined by standard techniques for computing long-run averages (see for instance [31,34]).

In contrast to the expressiveness of available formalisms for specifying complex Markov chains, formalisms for specifying the rewards have remained quite primitive. Basically, they only support the analysis of relatively simple performance metrics that can be expressed as long-run averages of *atomic rewards*, i.e. rewards that are deductible directly from the individual states of the initial Markov-chain specification. Examples of atomic rewards are the time progress in a state, the value of a variable in a state or an indication (0 or 1) of whether or not an action has occurred in a state.

Atomic rewards allow the expression of relatively simple performance metrics, such as the long-run average number of actions performed per unit of time or the long-run average occupancy level of a buffer. Many performance problems encountered in practice however, are more complex in the sense that they can only be expressed as long-run averages of accumulated atomic rewards over sequences of states. Examples of such performance metrics are the probability of losing a sequence of consecutive cells in a switch, the long-run average inter-arrival time between messages on a channel, the long-run variance of time between two events, or the long-run variance in rate (the burstiness) of a traffic source. The latter three metrics are often called delay-type measures [9], since they measure the duration of activities over time.

In case the modelling language is expressive enough, it is possible to compute the required accumulated rewards in the model itself, thereby *explicitly* extending the initial Markov chain by making the accumulated rewards of the original chain available as atomic rewards in the extended chain. However, this approach has two major drawbacks:

- The model is polluted with information that is only necessary for performance analysis and that has nothing to do with the essential behaviour of the model itself.
- The model must be changed for each performance metric to be analysed.

To solve these problems, we extend in this paper the technique of temporal rewards that we first introduced in [35]. The extension includes

- an embedding of the formalism in the context of the industrial-strength modelling language POOSL;
- the establishment of conditions guaranteeing the preservation of ergodicity;
- the proofs of all theorems;
- a number of realistic examples.

The technique of temporal rewards allows one to express the qualitative property of interest in terms of a collection of *temporal reward* formulas. A temporal reward formula is able to express accumulated atomic rewards that are available in the initial Markov chain. Together, the initial Markov chain and the temporal reward formulas *implicitly* define an extended Markov chain. This extended chain allows the evaluation of the quantitative property by traditional techniques for computing long-run averages. The formalism of temporal rewards will be developed in the context of discrete-time Markov chains. The formalism will be explained in the context of the industrial-strength modelling language POOSL [14,27], but is generally applicable to any formalism utilising discrete-time Markov chains for performance evaluation. Further, the formalism can be easily adapted for continuous formalisms as well.

The plan of the remainder of this paper is as follows. In Section 2, the formal modelling language POOSL is explained and a modelling example of a communication protocol is given that will be used throughout the remainder of the paper. The semantic model of the language is a labelled transition system. It is shown how a labelled transition system can be interpreted as a discrete-time Markov chain together with a set of reward variables. These are used to compute some simple performance metrics of the protocol. The section also poses more complex performance metrics, that are difficult to compute without explicitly extending the initial model. In Section 3, we develop the formalism of temporal rewards and a method to construct an extended Markov chain that allows these temporal rewards to be interpreted. We further prove that the construction method leaves long-run averages invariant for atomic rewards and we establish conditions that guarantee the construction method to preserve ergodicity. A comparison with related research is also given in this section. In Section 4, the performance metrics as posed in Section 2 are worked out in terms of temporal rewards. Other, more involved, metrics are demonstrated as well. Finally, in Section 5, the conclusions and directions for future research are given.

2. The POOSL language

In [27,28], we introduced the methodology Software/Hardware Engineering (SHE). SHE is a framework for object-oriented specification and design of hardware/software systems. The framework incorporates activities found in many modern object-oriented methods such as UML. Such activities include the creation of object class diagrams, message-flow diagrams and interaction diagrams (message-sequence charts). Starting from such informal and often graphical models, SHE produces rigorous behaviour and architecture models expressed in the formal specification language POOSL [14,27].

The POOSL language is briefly explained in Section 2.1 by giving an example of a protocol stack. The semantic model of the language is a labelled transition system, which is explained in Section 2.2. In Section 2.3, it is shown how a labelled transition system can be interpreted as a discrete-time Markov chain with a collection of reward variables. These are used to compute some simple performance metrics of the protocol.

2.1. The language

POOSL is a formal modelling language for complex hardware/software systems. The language combines a data part with a process part. The data part is based upon the concepts of traditional sequential object-oriented programming languages such as Smalltalk [18]. The process part is based on a probabilistic extension of TCCS [23], the timed extension of the CCS process algebra [22]. A specification in POOSL consists of a collection of asynchronous concurrent *process objects* (also called processes) and *clusters* that communicate synchronously by passing messages over channels. A cluster is a hierarchical entity consisting of processes and other clusters. Processes can contain, operate on and exchange items of data called *data objects*. Each process object, cluster and data object is an instance of a corresponding process class, cluster class or data class, defining its behaviour.

The POOSL language has specifically been developed to be used as a system-level modelling language. Once POOSL models have been constructed, they can be executed and interesting design properties can be determined. The execution of POOSL models is defined by a formal semantics which allows one to create tools to validate and verify the design and estimate performance figures very early in the design

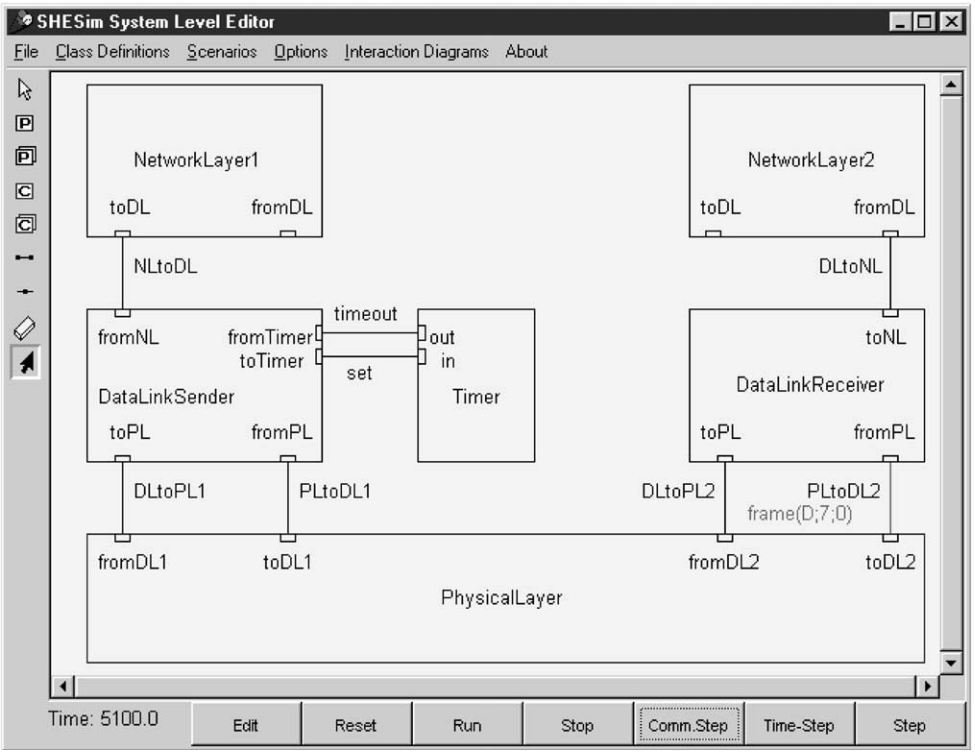

Fig. 1. Simulator window showing a protocol stack.

trajectory. The SHE methodology and the POOSL language are supported by the interactive model editing and simulation tool called SHESim [16]. The tool is used to incrementally specify and modify classes of data, processes and clusters in a graphical way. It further allows models to be simulated, several types of model viewers to be opened and message-sequence charts to be generated automatically. Currently the tool is being extended with capabilities for verification and performance analysis.

We will explain the POOSL language by means of a model of a protocol stack [30], which we will use throughout the remainder of this paper. The protocol stack is shown in the SHESim simulator window of Fig. 1. The stack consists of a *NetworkLayer*1 process, a *NetworkLayer*2 process, a *DataLinkSender* process, a *Timer* process, a *DataLinkReceiver* process and a *PhysicalLayer* cluster. These entities communicate by exchanging messages over channels. *NetworkLayer*1 and *NetworkLayer*2 exchange *Packets*. To this end, *NetworkLayer*1 can send messages of the form *packet*(*somePacket*) to the *DatalinkSender* by executing *toDL!packet*(*somePacket*) statements. *toDL* is a logical communication port of the *NetworkLayer*1 object and this port is connected to channel *NLtoDL*. The *DataLinkSender* can receive the *packet*(*somePacket*) messages by executing *fromNL?packet*(*somePacket*) statements. Here *somePacket* denotes a data object of data class *Packet*. Upon reception of a *Packet*, the *DatalinkSender* wraps this *Packet* (together with some control information) in a data object of class *Frame*. This *Frame* is then sent to the *PhysicalLayer* which transports it to the *Datalink Receiver*. Consequently, the *Packet* is retrieved from the *Frame* object, and is delivered to *NetworkLayer*2.

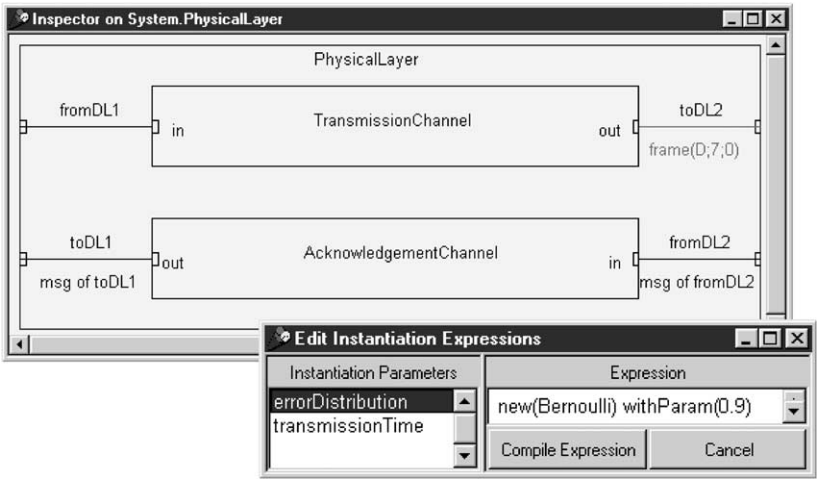

Fig. 2. Inspector on the PhysicalLayer cluster.

The *PhysicalLayer* is unreliable and can lose messages. To make sure that all *Packets* are delivered in the correct order, the *DatalinkSender* and *DatalinkReceiver* make use of a sliding-window protocol [30]. Therefore, the *DatalinkSender* can send a *Frame* several times and the *DatalinkReceiver* can also send an acknowledgement *Frame*. We will take a closer look at the specification of the *PhysicalLayer*. *PhysicalLayer* is a cluster consisting of a *TransmissionChannel* process and an *AcknowledgementChannel* process, see Fig. 2. These entities have two ports: *in* and *out*. From port *in*, they receive *frame*(*aFrame*) messages which will either get lost or get delivered at port *out* after some specified amount of time. Both the *TransmissionChannel* and the *AcknowledgementChannel* are instances of class *LossyChannel* defining their common behaviour. This behaviour is specified in Fig. 3. In this figure *transferFrames*()() is the name of a piece of behaviour (of the *LossyChannel* class) called a *method*. Clause $|f : Frame|$ declares a local variable f of data class *Frame*. The declaration of local variables is followed by the actual body of the method. Message-receive statement *in?frame*(*f*) indicates that the process wants to receive message *frame* with parameter f via port *in*. This statement is blocking; it is only executed if some other process executes a corresponding message-send statement of the form *p!frame*(*someFrame*), where p is any port that is connected via a channel to port *in*.

```
transferFrames()()
  | f : Frame |

  in?frame(f);
  if errorDistribution yieldsSuccess
     then delay(transmissionTime); out!frame(f);
          transferFrames()()
     else transferFrames()()
  fi.
```

Fig. 3. Definition of a *LossyChannel*.

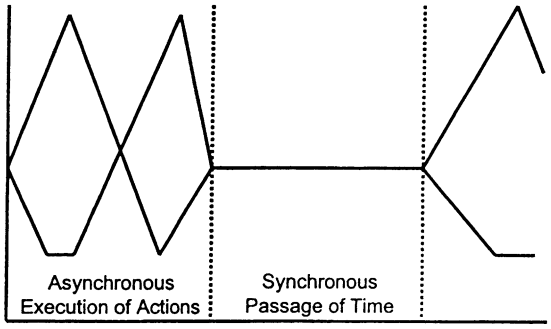

Fig. 4. Execution model (from [24]).

After the reception of message *frame*(f), a *LossyChannel* draws a sample from an *error-Distribution*. This is a variable which is instantiated to a data object of class *Bernoulli* specified by expression *new*(*Bernoulli*)*withParam*(0.9) (see Fig. 2). Drawing a sample is performed by sending message *yieldsSuccess* to this *Bernoulli* data object. This object returns the value *true* with probability 0.9 and the value *false* with probability 0.1. In the former case, the *LossyChannel* delays for *transmissionTime* units of time (*transmissionTime* is a variable which is assumed to be instantiated to value 3), sends the frame to port *out* and starts again by calling method *transferFrames*()() in a tail-recursive manner. In the latter case, the frame is lost and the *LossyChannel* starts receiving the next frame by calling method *transferFrames*()().

2.2. Formal semantics

The POOSL language is equipped with a mathematical semantics. The semantics of the non-real-time part of POOSL is given in [27]. The formalisation of the real-time extension is described in [14,15] and the probabilistic version is currently being developed (an initial version in the form of a simple calculus PRTCCS is given in [34]). The semantics is based on a two-phase execution model [24]. The state of a system can either change by asynchronously executing atomic (communication or data processing) actions (taking no time) or by letting time pass (synchronously) inactively, see Fig. 4. The semantics of a POOSL specification is given by a Plotkin-style structural operational semantics [26], defining a labelled transition system:

$$(\hat{\mathcal{S}}, S_s, \mathcal{A}, \mathcal{T}, \{\overset{a}{\rightarrow} \subseteq \hat{\mathcal{S}} \times \mathrm{Distr}(\hat{\mathcal{S}}) | a \in \mathcal{A}\}, \{\overset{t}{\rightarrow} \subseteq \hat{\mathcal{S}} \times \hat{\mathcal{S}} | t \in \mathcal{T}\}).$$

It consists of a (countable) set $\hat{\mathcal{S}}$ of states (POOSL models during execution), an initial state $S_s \in \hat{\mathcal{S}}$, a set of actions $\mathcal{A}$, a time domain $\mathcal{T}$ and two labelled transition relations.[2] $\mathrm{Distr}(\hat{\mathcal{S}})$ is the collection of probability distributions on $\hat{\mathcal{S}}$, i.e. the collection of functions $D : \hat{\mathcal{S}} \rightarrow [0, 1]$ for which $\sum_{S \in \hat{\mathcal{S}}} D(S) = 1$. Relation $S \overset{a}{\rightarrow} D$ holds if in state S action a can be performed after which the model will enter state S' with probability $D(S')$ for each S'. In case several (possibly different) actions can be performed in S, the choice of which action is actually being executed is taken non-deterministically.[3] After the non-deterministic choice has been made, a probabilistic choice determines the next state.

[2] We will assume $\mathcal{T}$ to be the set of positive real numbers.

[3] For performance analysis, non-determinism is first resolved by an external scheduler, see also Section 2.3.

Relation $S \xrightarrow{t} S'$ holds in state S if the model can delay for t units of time and after that transit to state S'. Time transitions follow the concepts as presented in [34]. The basic idea is that a process only has time transitions for maximal durations of time it wants to delay, where (for reasons of compositionality) it is implicitly understood that it is also willing to delay for shorter durations of time. Hence time transitions are deterministic; for each state S, there is at the most one t and S' such that $S \xrightarrow{t} S'$. Therefore, a time transition can be considered to always be taken with probability 1.

The semantic model of POOSL is closely related to the model of non-deterministic probabilistic labelled transition systems as introduced by Hansson and Jonsson [19]. Our model, however, follows the approach of Segala [29] and does not distinguish probabilistic states from non-deterministic ones. A non-deterministic choice is followed immediately by a probabilistic one; an intermediate state is not represented explicitly as is done in the work of Hansson and Jonsson [19]. Further, our model uses a dense time domain, whereas in [19] a discrete-time domain is used. The semantic model of POOSL also resembles the model of timed probabilistic systems [1] which in its turn is closely related to the model of Markov decision processes [12]. Finally, our model is often implicitly used as the execution model of many discrete-event simulators such as OPNET.

When the contents of the actual data frames is abstracted from, the transition system of a *LossyChannel* is given in Fig. 5. The ellipses denote the states of the transition system. The > symbol indicates the initial state. The (double) arrows labelled with actions as well as with probabilities denote action transitions. Here actions *in* and *out* abbreviate actions of executing statements *in?frame*(*f*) and *out!frame*(*f*), respectively (see Fig. 3). Action τ denotes an internal unobservable action. In Fig. 5, the action τ represents the action of drawing a sample from the *Bernoulli* distribution (see Fig. 3). The arrows labelled with numbers or with the symbol ∞ denote time transitions. The time transition labelled with 3 indicates that the *LossyChannel* wants to delay for 3 units of time after which is transits to the next state. The time transitions with labels ∞ indicate that a *LossyChannel* is willing to wait for an arbitrary amount of time before performing the *in* and *out* actions. The reason is that *in* and *out* are synchronisation actions; if the environment is not willing to participate in them at a certain instant in time, a *LossyChannel* is allowed to wait until the environment is ready to send respectively receive a frame. (It is not allowed to wait longer than that, which is enforced by the property of action urgency [24].) In this way, the transition system of a component model is always 'open' to the environment it is placed in. This allows for the compositional construction of a transition system of a composite model from the individual transition systems of its constituents. For instance, the transition system of the *PhysicalLayer* cluster is built from the transition systems of the *TransmissionChannel* process and the *AcknowledgementChannel* process, and the transition system of the complete protocol stack is built from the transition systems of the *NetworkLayer*1 process, the *NetworkLayer*2 process, the *DataLinkSender* process, the *Timer* process, the *DataLinkReceiver* process and the *PhysicalLayer* cluster (see Figs. 1 and 2).

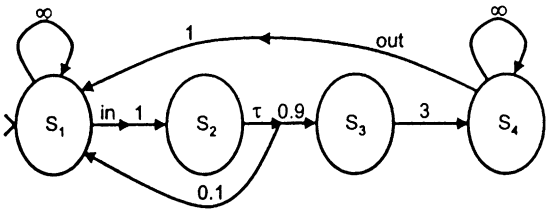

Fig. 5. Labelled transition system of a *LossyChannel*.

2.3. Rewards and performance evaluation

Suppose we would like to know the capacity of a *LossyChannel* process, i.e. the long-run average number of frames accepted per unit of time. Since the process can only accept or deliver a frame if the environment is willing to offer respectively receive it, we have to make an assumption about the behaviour of this environment. To study the behaviour of the channel in isolation, the environment is assumed to be always willing to participate in any action the process wants to perform. This means that the process will always make maximal progress; if it is able to perform an action, it is not allowed to delay. Maximal progress implies the removal of time transitions from states allowing both time and action transitions. For the transition system in Fig. 5 this implies the removal of the time transitions labelled with symbol ∞. After the removal of these time transitions, the remaining time transitions are interpreted in a slightly different way than before. Instead of indicating maximal durations of time the process wants to delay, time transitions now indicate the exact amount of time the process must delay.

Next we transform the labelled transition system into a Markov chain. To this end, first the non-determinism is resolved by an external scheduler [33] (also called adversary in [29] and rule or policy in the theory of Markov decision processes [12]). After resolving the non-determinism, performance metrics are given by a single performance figure (at least if the property of ergodicity is satisfied, see Section 3.7). Different schedulers can yield different figures. Therefore, in case the non-determinism is not resolved, performance metrics give rise to a collection of performance figures. Notice that in the transition system in Fig. 5 non-determinism is not present (if the ∞ transitions are removed). Also, in the sequel we will not consider schedulers and assume that all non-determinism has been resolved.

After resolving the non-determinism, we obtain a new labelled transition system, which is of the form:

$$(\hat{\mathcal{S}}, S_s, \mathcal{A}, \mathcal{T}, \{\overset{a,p}{\Rightarrow} \subseteq \hat{\mathcal{S}} \times \hat{\mathcal{S}} | a \in \mathcal{A}, p \in [0,1]\}, \{\overset{t}{\Rightarrow} \subseteq \hat{\mathcal{S}} \times \hat{\mathcal{S}} | t \in \mathcal{T}\}).$$

As in the original semantic model, $\hat{\mathcal{S}}$, S_s, $\mathcal{A}$ and $\mathcal{T}$ denote the collection of states, the initial state, the set of actions and the time domain, respectively. For states S, S', action a and probability p, relation $S \overset{a,p}{\Rightarrow} S'$ holds if in state S, action a can be performed with probability p after which state S' is entered. The action relation satisfies the property that for each a, p and p', $S \overset{a,p}{\Rightarrow} S'$ and $S \overset{a,p'}{\Rightarrow} S'$ implies $p = p'$. Hence for each action a and state S', there is at most one a-transition leading to S'. Relation $\overset{a,p}{\Rightarrow}$ is determined straightforwardly from relation $\overset{a}{\rightarrow}$ (see Section 2.2) and the scheduler. Relation $S \overset{t}{\Rightarrow} S'$ holds if in state S a delay of t units of time must be performed, after which a transition to state S' is made. Due to the assumption of maximal progress, $S \overset{t}{\Rightarrow} S'$ holds if $S \overset{t}{\rightarrow} S'$ holds and if no action transitions can be performed in state S. Because time transitions are deterministic, each time transition is considered to be taken with probability 1. The resulting transition system is stochastic, i.e. for each state the sum of probabilities of the outgoing transitions equals 1. Notice that in each state either action transitions or a time transition can be performed, but not both.

Next, the resulting labelled transition system is transformed into a Markov chain by shifting the (action and time) information of the transition labels into the states. Although this transformation may seem to be redundant, it will enable us to retrieve from the states information about the *actual* occurrences of actions and the passage of time (from the transitions of the labelled transition system one can only deduce about a state that an action or time transition *can* be taken). This makes it possible to encode action and time information as atomic rewards, which is necessary when developing the temporal reward formalism. The technique of encoding transition information in the states is often used implicitly by

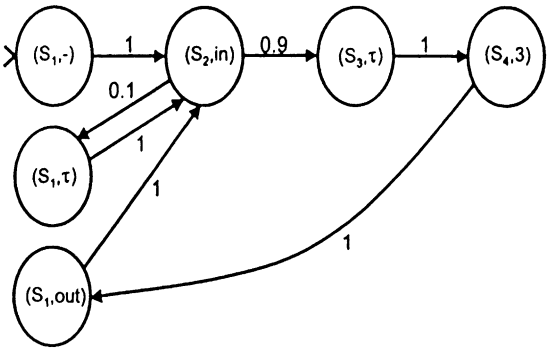

Fig. 6. Markov chain of a *LossyChannel*.

state-space construction methods for continuous-time Markov chains (see e.g. [25]). In our case, we make the transformation explicit. The transformation yields a Markov chain with state space $\mathcal{S} = \hat{\mathcal{S}} \times \mathcal{A} \cup \mathcal{T} \cup \{-\}$.[4] The interpretation of a state of the form (S, a) is that state S in the labelled transition system was entered by a transition labelled with action a. The interpretation of (S, t) is that S in the labelled transition system was entered by a time transition labelled with t and the interpretation of $(S, -)$ is that S in the labelled transition system was entered without performing any action or time transition. The initial state of the Markov chain is defined by $(S_s, -)$, where S_s denotes the initial state of the labelled transition system. For $\gamma \in \mathcal{A} \cup \mathcal{T} \cup \{-\}$, the stochastic probability matrix $\mathcal{M} : \mathcal{S} \to \mathcal{S}$ is defined as follows:

$$\mathcal{M}_{(S,\gamma),(S',\gamma')} = \begin{cases} p & \text{if } \gamma = a \text{ and } S \overset{a,p}{\Rightarrow} S', \\ 1 & \text{if } \gamma = t \text{ and } S \overset{t}{\Rightarrow} S', \\ 0 & \text{otherwise.} \end{cases}$$

Here $\mathcal{M}_{(S,\gamma),(S',\gamma')}$ denotes the probability that the Markov chain transits from state (S, γ) to state (S', γ'). In case the labelled transition system is finite, the number of states of the Markov chain equals 1 plus the number of transitions in the labelled transition system with different label/target-state combinations. For each state in the Markov chain, the number of outgoing transitions is equal to the number of outgoing transitions of the corresponding state in the transition system. After the transformation, we obtain the Markov chain as shown in Fig. 6.

The states of the chain consist of two components. The first component refers to the corresponding state in the labelled transition system and the second component refers to the action that was performed or the amount of time that elapsed when this state was entered. The initial state $(S_1, -)$ of the chain corresponds to the initial state S_1 of the labelled transition system. Before the initial state of the labelled transition system was entered, no action was performed and the time did not elapse. This is encoded in the corresponding state of the Markov chain by symbol $-$. States (S_1, τ) and (S_1, out) of the chain also correspond to the initial state S_1 of the labelled transition system, but symbols τ and *out* encode that this state was entered by first performing actions τ and *out* respectively. Similarly, symbol 3 in state $(S_4, 3)$ of the Markov chain encodes that the corresponding state S_4 of the labelled transition system was entered

[4] $\mathcal{S}$ is an uncountable object, but we assume that the actual state space is defined by the states that are reachable from the initial state with positive probability.

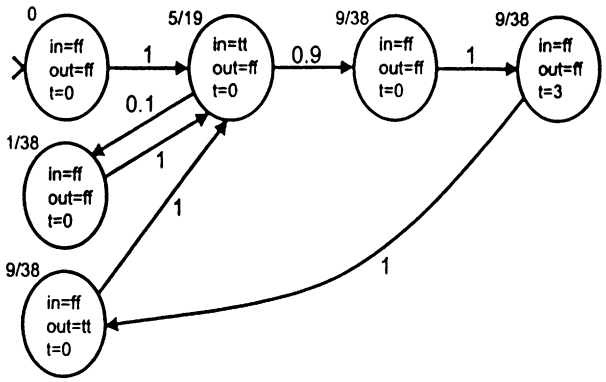

Fig. 7. Markov chain and rewards of a *LossyChannel*.

by first letting the time pass for 3 units of time. Notice that, in contrast to the labelled transition system of Fig. 5, actions now occur in the states and also the time passes in the states (and not in the transitions).

For performance evaluation, the information contained in the states is used to define a number of rewards. A reward is a function on the state space and the idea is that each time a certain state is visited a 'reward' as specified by the reward function is obtained [31,34]. Usually reward functions are real-valued, but in this paper we will also consider Boolean-valued reward functions. We will use the symbols *in*, *out* and *t* as the names of three rewards. These symbols are interpreted as functions on the state space of the Markov chain and these functions are denoted by $\overline{in}$, $\overline{out}$ and $\bar{t}$, respectively. Functions $\overline{in}$ and $\overline{out}$ are Boolean-valued functions indicating whether actions *in*, respectively *out*, occur in a state or not. Function $\bar{t}$ is a real-valued function indicating the progress of time in a state. For $i = 1$–4, these functions are defined by

- $\overline{in}((S_i, \gamma)) = \begin{cases} tt & \text{if } \gamma = in, \\ ff & \text{otherwise.} \end{cases}$
- $\overline{out}((S_i, \gamma)) = \begin{cases} tt & \text{if } \gamma = out, \\ ff & \text{otherwise.} \end{cases}$
- $\bar{t}((S_i, \gamma)) = \begin{cases} \gamma & \text{if } \gamma \in \mathbb{R}, \\ 0 & \text{otherwise.} \end{cases}$

Symbols *tt* and *ff* denote the Boolean values *true* and *false*, respectively. Fig. 7 visualises the Markov chain where the values of the rewards are indicated in the states. For instance, the intended meaning of clause $in = ff$ in the initial state is that $\overline{in}((S_1, -)) = ff$. Remark that, in contrast to the former models, no fundamental distinction is made anymore between the occurrence of actions and the passage of time; all information is uniformly represented as rewards.

Now that we have defined a Markov chain and a collection of reward functions, we are able to compute a number of simple performance metrics such as

(i) The long-run average number of *in* actions performed per discrete-time epoch of the Markov chain.
(ii) The long-run average number of *in* actions performed per unit of model time.

If $\{X_i | i \geq 1\}$ denotes the Markov chain defined by the graph in Fig. 7, then performance metric (i) is given by the limiting behaviour of random variable

$$\frac{1}{n}\sum_{i=1}^{n} \mathit{if}\ \overline{\mathit{in}}(X_i)\ \mathit{then}\ 1\ \mathit{else}\ 0\ \mathit{fi}.$$

In case the chain is ergodic (see Section 3.7), this random variable converges (by the strong law of large numbers) strongly (with probability 1) to a constant c and (by the ergodic theorem) this constant is given by

$$c = \sum_{(S_i,\gamma)} \mathit{if}\ \overline{\mathit{in}}((S_i,\gamma))\ \mathit{then}\ 1\ \mathit{else}\ 0\ \mathit{fi}\ \cdot \pi_{(S_i,\gamma)}.$$

Here $\pi_{(S_i,\gamma)}$ denotes the equilibrium probability of state (S_i,γ) (these probabilities are indicated at the upper-left positions of the states in Fig. 7). Similarly, performance metric (ii) is given by the limiting behaviour of random variable

$$\frac{\sum_{i=1}^{n} \mathit{if}\ \overline{\mathit{in}}(X_i)\ \mathit{then}\ 1\ \mathit{else}\ 0\ \mathit{fi}}{\sum_{i=1}^{n} \bar{t}(X_i)}.$$

This variable converges strongly to constant

$$d = \frac{\sum_{(S_i,\gamma)} \mathit{if}\ \overline{\mathit{in}}((S_i,\gamma))\ \mathit{then}\ 1\ \mathit{else}\ 0\ \mathit{fi} \cdot \pi_{(S_i,\gamma)}}{\sum_{(S_i,\gamma)} \bar{t}((S_i,\gamma)) \cdot \pi_{(S_i,\gamma)}}.$$

The reader may want to check that $c = 5/19$ and $d = 10/27$. Hence the capacity of the *LossyChannel* is 10/27 frames per unit of time.

In the example above, only rewards concerning the occurrence of actions or the passage of time are used. A POOSL model can give rise to other rewards as well. These rewards include the value of a variable in a certain state and the value of a parameter passed between two entities. Developing a specification language for retrieving such rewards is subject of current research.

3. Temporal rewards

In this section, the formalism of temporal rewards is introduced. In Section 3.1, the motivation for the formalism is given. The formalism is introduced in Section 3.2. The interpretation of temporal rewards over extended chains is defined in Section 3.3 and a method to construct an extended chain is given in Section 3.4. Section 3.5 explains how to construct the extended chain in an incremental way. In Section 3.6, it is proved that the construction method preserves long-run averages of atomic rewards. In Section 3.7, we establish conditions guaranteeing the construction method to preserve ergodicity. Finally, a comparison with related research is made in Section 3.8.

3.1. Motivation

Performance metrics (i) and (ii) given in the previous section are simple in the sense that they can be expressed as long-run averages of atomic rewards, whose values can be determined directly from the

states of the initial Markov chain. Often, however, performance metrics are more involved because they can only be expressed as long-run averages of accumulated atomic rewards over a sequence of states where the order in which this sequence is traversed can be important.

As an example concerning the *LossyChannel* again, consider the following performance metrics:

(iii) The long-run average time between two *in* actions.
(iv) The long-run variance of time between two *in* actions.

Both these performance metrics can be expressed as long-run averages of time durations (or the squares thereof) between two *in* actions. Such a time duration, however, depends on the sequence of states that is traversed from a state in which action *in* occurs and a state in which action *in* occurs again. For instance if action *in* occurs (see Fig. 7), the next instance of *in* happens either 0 or 3 units of time later, dependent on whether the left-middle state is visited in-between or not. (Notice that measure (iii) could be determined indirectly by computing the reciprocal of measure (ii). This is however not possible for performance measure (iv).)

The values of performance metrics (iii) and (iv) could be obtained by *explicitly* extending the Markov chain of Fig. 7 in such a way that the time durations between two successive *in* actions are defined as atomic rewards. This can be carried out by modifying the behaviour specification of the *LossyChannel* (see Fig. 3). One could for instance add a variable recording the time between two successive executions of statement *in?frame*(*f*). The value of this variable would then be defined as an atomic reward in the underlying Markov chain.

A major disadvantage of this approach is that it requires the model to be polluted with an additional variable, thereby hampering the readability and clarity of the model. Another disadvantage is that the model must be modified again if one is interested in different performance measures such as the throughput of the channel. The temporal reward formalism enables the specification of performance measures (iii) and (iv) without having to modify the initial model. Metrics (iii) and (iv) are elaborated in Section 4.

3.2. Formalism

Consider a discrete-time Markov chain $\{X_i | i \geq 1\}$ with countable state space $\mathcal{S}$, initial state S_s and probability matrix $\mathcal{M} : \mathcal{S} \times \mathcal{S} \to [0, 1]$. The definition of such a chain will be given in the form of the ensemble $(\mathcal{S}, S_s, \mathcal{M})$. For $S, T \in \mathcal{S}$, $\mathcal{M}_{S,T}$ is the probability that the chain transits from state S to state T.

We let $\mathcal{B}$ denote the set of atomic Boolean-valued rewards and let $b, \ldots$ range over $\mathcal{B}$. We further let $\mathcal{R}$ denote the set of atomic real-valued rewards, ranged over by $r, \ldots$. We assume that each Boolean-valued or real-valued reward is interpreted over a chain defined by $(\mathcal{S}, S_s, \mathcal{M})$. In particular, we assume the existence of an interpretation function $\bar{\cdot} : \mathcal{B} \cup \mathcal{R} \to \mathcal{S} \hookrightarrow \mathbb{B} \cup \mathbb{R}$. For a Boolean-valued reward b and state S, $\bar{b}(S)$ is either undefined, denoted by $\bar{b}(S) = \perp$, or element of the Boolean domain $\mathbb{B}$. In case $\bar{b}(S) = tt$, we will say that state S satisfies b. For real-valued reward r, $\bar{r}(S)$ is either undefined, denoted by $\bar{r}(S) = \perp$ or element of the domain $\mathbb{R}$ of reals. The symbol $\hookrightarrow$ indicates that $\bar{b}$ and $\bar{r}$ are partial functions.

We will assume that symbols such as *true*, *false*, 1 and $\sqrt{2}$ are elements of $\mathcal{B}$ and $\mathcal{R}$, respectively, and are interpreted in the standard way, i.e. $\overline{true}(S) = tt$, $\overline{false}(S) = ff$, $\bar{1}(S) = 1$ and $\overline{\sqrt{2}}(S) = \sqrt{2}$. We further assume that symbol $\perp$ denotes both a Boolean-valued and a real-valued reward and is interpreted as $\bar{\perp}(S) = \perp$ for each state S.

Based on the set $\mathcal{B}$ of atomic Boolean-valued rewards, we define the collection $\mathcal{B}^t$ of temporal Boolean-valued rewards. The interpretation of these temporal rewards will depend not only on the current state of the Markov chain, but also on states visited in the past. Collection $\mathcal{B}^t$, ranged over by $\mathbf{b}, \ldots$, is inductively defined by

$$\mathbf{b} ::= b|\neg\mathbf{b}|\mathbf{b} \vee \mathbf{c}|\mathbf{r} < \mathbf{s}| \uparrow\sigma|\mathit{if}\ \mathbf{b}\ \mathit{then}\ \mathbf{c}\ \mathit{else}\ \mathbf{d}\ \mathit{fi}|\Theta\mathbf{b}|B.$$

Except for the $\uparrow$ and *if then else fi* operators, the interpretation of all operators is strict, i.e. if one of the constituent rewards is undefined, the composite reward is undefined as well. The intuitive interpretation of the operators is as follows. The Boolean operators $\neg$ and $\vee$ have their usual meaning. Reward $\mathbf{r} < \mathbf{s}$ holds if reward $\mathbf{r}$ is smaller than reward $\mathbf{s}$. Here $\mathbf{r}$ and $\mathbf{s}$ are temporal real-valued rewards and will be explained shortly. $\uparrow\sigma$ indicates whether reward σ is defined. Reward σ is either Boolean-valued or real-valued. Reward *if* $\mathbf{b}$ *then* $\mathbf{c}$ *else* $\mathbf{d}$ *fi* equals $\mathbf{c}$ if $\mathbf{b}$ holds, equals $\mathbf{d}$ if $\mathbf{b}$ does not hold, and is undefined if $\mathbf{b}$ is undefined. Θ denotes the previous-state operator. The interpretation is that $\Theta\mathbf{b}$ holds in the current state if $\mathbf{b}$ holds in the previous state (temporal rewards will be interpreted over an extended chain satisfying the property that the values of $\mathbf{b}$ in different previous states are equal). Finally, reward B denotes a Boolean-valued reward constant. These constants are denoted by symbols $B, \ldots$. It is assumed that each constant B is defined by an equation of the form $B \stackrel{\text{def}}{=} \mathbf{b}$, such that if B is (indirectly) defined in terms of B itself, this latter occurrence of B is guarded by the previous-state operator Θ.

Note that other Boolean operators can be defined in terms of the operators mentioned above. For example,

- $\mathbf{b} \wedge \mathbf{c}$ is defined as $\neg(\neg\mathbf{b} \vee \neg\mathbf{c})$;
- $\downarrow\sigma$ is defined as $\neg \uparrow\sigma$;
- $\mathbf{s} \geq \mathbf{t}$ is defined as $\neg\mathbf{s} < \mathbf{t}$.

Next we define the collection $\mathcal{R}^t$ of temporal real-valued rewards. We shall let $\mathbf{r}, \ldots$ range over $\mathcal{R}^t$ and we further let $\sigma, \ldots$ range over $\mathcal{B}^t \cup \mathcal{R}^t$. Collection $\mathcal{R}^t$ is inductively defined by

$$\mathbf{r} ::= r|\mathbf{r} + \mathbf{s}|\mathbf{r} * \mathbf{s}|\mathit{if}\ \mathbf{b}\ \mathit{then}\ \mathbf{r}\ \mathit{else}\ \mathbf{s}\ \mathit{fi}|\Theta\,\mathbf{r}|R.$$

Except for the *if then else fi* operator, all operators are strict. Operators $+$ and $*$ have their usual meaning. Reward *if* $\mathbf{b}$ *then* $\mathbf{r}$ *else* $\mathbf{s}$ *fi* equals $\mathbf{r}$ if $\mathbf{b}$ holds, equals $\mathbf{s}$ if $\mathbf{b}$ does not hold, and is undefined if $\mathbf{b}$ is undefined. The value of reward $\Theta\mathbf{r}$ in the current state equals the value of $\mathbf{r}$ in the previous state (temporal rewards will be interpreted over an extended chain satisfying the property that the values of $\mathbf{r}$ in different previous states are equal). Finally, reward R is a real-valued reward constant. These constants are denoted by symbols $R, \ldots$. It is assumed that each constant R is defined by an equation of the form $R \stackrel{\text{def}}{=} \mathbf{r}$, such that if R is (indirectly) defined in terms of R itself, this latter occurrence of R is guarded by the previous-state operator Θ.

Again, other operators can be defined in terms of the operators defined above. For instance

- $\mathbf{r} - \mathbf{s}$ is defined as $\mathbf{r} + (-1 * \mathbf{s})$;
- $\mathbf{r}\max\mathbf{s}$ is defined as *if* $\mathbf{r} > \mathbf{s}$ *then* $\mathbf{r}$ *else* $\mathbf{s}$ *fi*;
- $\mathbf{r}^2$ is defined as $\mathbf{r} * \mathbf{r}$.

Further for $n \geq 1$ and temporal reward σ, we will sometimes write $\Theta^n\sigma$ to denote the temporal reward consisting of n previous-state operators followed by σ.

3.3. Interpretation

Temporal rewards cannot be interpreted directly over the initial chain defined by $(\mathcal{S}, S_s, \mathcal{M})$. This is caused by the previous-state operator Θ. For temporal reward $\sigma \in \mathcal{B}^t \cup \mathcal{R}^t$, the value of expression $\Theta\sigma$ in a state S should depend on the value of σ in a state S' preceding S. The problem is that if S has more than one preceding state, the values of σ in these states may differ. To solve this problem, we will interpret temporal rewards over an extended chain that does not suffer this problem. The technique we use to construct the extended chain resembles the technique applied in [6] for encoding past information concerning the past operators in pCTL*.

The extended chain is constructed from $(\mathcal{S}, S_s, \mathcal{M})$ and a finite collection $\mathcal{C} \subseteq \mathcal{B}^t \cup \mathcal{R}^t$ of temporal rewards and is denoted by $(\mathcal{S}, S_s, \mathcal{M})^{\mathcal{C}} = (\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$. $\mathcal{C}$ is the collection of rewards required to analyse the quantitative properties of interest. Collection $\mathcal{C}$ is assumed to be closed in the sense that

- If $\neg\mathbf{b} \in \mathcal{C}$ then $\mathbf{b} \in \mathcal{C}$.
- If $\mathbf{b} \vee \mathbf{c} \in \mathcal{C}$ then $\mathbf{b} \in \mathcal{C}$ and $\mathbf{c} \in \mathcal{C}$.
- If $\mathbf{r} < \mathbf{s} \in \mathcal{C}$ then $\mathbf{r} \in \mathcal{C}$ and $\mathbf{s} \in \mathcal{C}$.
- If $\uparrow\sigma \in \mathcal{C}$ then $\sigma \in \mathcal{C}$.
- If *if* $\mathbf{b}$ *then* $\mathbf{c}$ *else* $\mathbf{d}$ *fi* $\in \mathcal{C}$ then $\mathbf{b} \in \mathcal{C}$, $\mathbf{c} \in \mathcal{C}$ and $\mathbf{d} \in \mathcal{C}$.
- If $\Theta\mathbf{b} \in \mathcal{C}$ then $\mathbf{b} \in \mathcal{C}$.
- If $B \in \mathcal{C}$ and $B \stackrel{\text{def}}{=} \mathbf{b}$ then $\mathbf{b} \in \mathcal{C}$.
- If $\mathbf{r} + \mathbf{s} \in \mathcal{C}$ or $\mathbf{r} * \mathbf{s} \in \mathcal{C}$ then $\mathbf{r} \in \mathcal{C}$ and $\mathbf{s} \in \mathcal{C}$.
- If *if* $\mathbf{b}$ *then* $\mathbf{r}$ *else* $\mathbf{s}$ *fi* $\in \mathcal{C}$ then $\mathbf{b} \in \mathcal{C}$, $\mathbf{r} \in \mathcal{C}$ and $\mathbf{s} \in \mathcal{C}$.
- If $\Theta\mathbf{r} \in \mathcal{C}$ then $\mathbf{r} \in \mathcal{C}$.
- If $R \in \mathcal{C}$ and $R \stackrel{\text{def}}{=} \mathbf{r}$ then $\mathbf{r} \in \mathcal{C}$.

Now state space $\mathcal{S}^{\mathcal{C}}$ is defined as the Cartesian product $\mathcal{S} \times (\mathbb{B} \cup \mathbb{R})^{\mathcal{C}_{\Theta\mathbf{b}} \cup \mathcal{C}_{\Theta\mathbf{r}}}$, where $\mathcal{C}_{\Theta\mathbf{b}}$ and $\mathcal{C}_{\Theta\mathbf{r}}$ are defined by $\mathcal{C}_{\Theta\mathbf{b}} = \{\sigma \in \mathcal{C} | \sigma \equiv \Theta\mathbf{b} \text{ for some } \mathbf{b} \in \mathcal{C}\}$ and $\mathcal{C}_{\Theta\mathbf{r}} = \{\sigma \in \mathcal{C} | \sigma \equiv \Theta\mathbf{r} \text{ for some } \mathbf{r} \in \mathcal{C}\}$.[5] Here symbol $\equiv$ denotes syntactic identity. Further $(\mathbb{B} \cup \mathbb{R})^{\mathcal{C}_{\Theta\mathbf{b}} \cup \mathcal{C}_{\Theta\mathbf{r}}}$ denotes the collection of all partial functions from $\mathcal{C}_{\Theta\mathbf{b}} \cup \mathcal{C}_{\Theta\mathbf{r}}$ to $\mathbb{B} \cup \mathbb{R}$ that map Boolean-valued rewards to Booleans and real-valued rewards to reals. We will denote these functions with symbols $f, g, \ldots$. For a state $(S, f) \in \mathcal{S}^{\mathcal{C}}$, the S part represents the original state over which the non-temporal rewards are interpreted and the f part stores the information about the temporal rewards of the form $\Theta\sigma$. Based on this information, we are able to extend the reward interpretation function $\bar{\cdot}$ to domain $\mathcal{B} \cup \mathcal{R} \cup \mathcal{C}$. For each $\sigma \in \mathcal{C}$, $\bar{\sigma}$ is a partial function from $\mathcal{S}^{\mathcal{C}}$ to $\mathbb{B} \cup \mathbb{R}$. For $\sigma \in \mathcal{C}$ and $(S, f) \in \mathcal{S}^{\mathcal{C}}$, $\bar{\sigma}((S, f))$ is defined inductively by the following rules:

(i) If $\sigma \equiv b$ then $\bar{\sigma}((S, f)) = \bar{b}(S)$.
(ii) If $\sigma \equiv \neg\mathbf{b}$ then $\bar{\sigma}((S, f)) = \neg\bar{\mathbf{b}}((S, f))$.
(iii) If $\sigma \equiv \mathbf{b} \vee \mathbf{c}$ then $\bar{\sigma}((S, f)) = \bar{\mathbf{b}}((S, f)) \vee \bar{\mathbf{c}}((S, f))$.
(iv) If $\sigma \equiv \mathbf{r} < \mathbf{s}$ then $\bar{\sigma}((S, f)) = \bar{\mathbf{r}}((S, f)) < \bar{\mathbf{s}}((S, f))$.
(v) If $\sigma \equiv \uparrow\sigma'$ then $\bar{\sigma}((S, f)) = \begin{cases} ff & \text{if } \overline{\sigma'}((S, f)) = \bot, \\ tt & \text{otherwise.} \end{cases}$

[5] Notice that in general $\mathcal{C}_{\Theta\mathbf{r}}$ is an uncountably infinite object.

(vi) If $\sigma \equiv \mathit{if}\ \mathbf{b}\ \mathit{then}\ \mathbf{c}\ \mathit{else}\ \mathbf{d}\ \mathit{fi}$ then $\bar{\sigma}((S, f)) = \begin{cases} \bar{\mathbf{c}}((S, f)) & \text{if } \bar{\mathbf{b}}((S, f)) = tt, \\ \bar{\mathbf{d}}((S, f)) & \text{if } \bar{\mathbf{b}}((S, f)) = ff, \\ \bot & \text{if } \bar{\mathbf{b}}((S, f)) = \bot. \end{cases}$

(vii) If $\sigma \equiv \Theta\mathbf{b}$ then $\bar{\sigma}((S, f)) = f(\Theta\mathbf{b})$.

(viii) If $\sigma \equiv B$ and $B \stackrel{\text{def}}{=} \mathbf{b}$ then $\bar{\sigma}((S, f)) = \bar{\mathbf{b}}((S, f))$.

(ix) If $\sigma \equiv r$ then $\bar{\sigma}((S, f)) = \bar{r}(S)$.

(x) If $\sigma \equiv \mathbf{r} + \mathbf{s}$ then $\bar{\sigma}((S, f)) = \bar{\mathbf{r}}((S, f)) + \bar{\mathbf{s}}((S, f))$.

(xi) If $\sigma \equiv \mathbf{r} * \mathbf{s}$ then $\bar{\sigma}((S, f)) = \bar{\mathbf{r}}((S, f)) * \bar{\mathbf{s}}((S, f))$.

(xii) If $\sigma \equiv \mathit{if}\ \mathbf{b}\ \mathit{then}\ \mathbf{r}\ \mathit{else}\ \mathbf{s}\ \mathit{fi}$ then $\bar{\sigma}((S, f)) = \begin{cases} \bar{\mathbf{r}}((S, f)) & \text{if } \bar{\mathbf{b}}((S, f)) = tt, \\ \bar{\mathbf{s}}((S, f)) & \text{if } \bar{\mathbf{b}}((S, f)) = ff, \\ \bot & \text{if } \bar{\mathbf{b}}((S, f)) = \bot. \end{cases}$

(xiii) If $\sigma \equiv \Theta\mathbf{r}$ then $\bar{\sigma}((S, f)) = f(\Theta\mathbf{r})$.

(xiv) If $\sigma \equiv R$ and $R \stackrel{\text{def}}{=} \mathbf{r}$ then $\bar{\sigma}((S, f)) = \bar{\mathbf{r}}((S, f))$.

The operators $\neg$, $\vee$, $<$, $+$ and $*$, applied to the reward interpretations in state (S, f) are interpreted strictly. Rules (i) and (ix) state that the value of an atomic reward in an extended state equals the value in the corresponding state of the initial chain. Rules (ii)–(vi) and (x)–(xii) express the value of a composite reward in terms of the values of its constituents. Rules (vii) and (xiii) state that the value of a temporal reward of the form $\Theta\sigma$ is stored in the second component of an extended state. Finally, rules (viii) and (xiv) define the value of a reward constant as the value of its defining reward. Notice that an extended state contains precisely the information necessary to determine $\bar{\sigma}(S)$ for each $\sigma \in \mathcal{C}$.

Now that we have defined state space $\mathcal{S}^{\mathcal{C}}$, we can give a definition of probability matrix $\mathcal{M}^{\mathcal{C}} : \mathcal{S}^{\mathcal{C}} \times \mathcal{S}^{\mathcal{C}} \to [0, 1]$. For states $(S, f), (S', f') \in \mathcal{S}^{\mathcal{C}}$, $\mathcal{M}^{\mathcal{C}}_{(S,f),(S',f')}$ is defined by

$$\mathcal{M}^{\mathcal{C}}_{(S,f),(S',f')} = \begin{cases} \mathcal{M}_{S,S'} & \text{if } \overline{\Theta\sigma}((S', f')) = \bar{\sigma}((S, f)) \text{ for each } \Theta\sigma \in \mathcal{C}, \\ 0 & \text{otherwise.} \end{cases}$$

Hence a transition with positive probability from state (S, f) to state (S', f') can be made if and only if a transition with the same probability from state S to state S' is available in the initial chain and if the values of each temporal reward of the form $\Theta\sigma$ in the next state (S', f') is consistent with the reward σ in the previous state (S, f). In case $\mathcal{M}_{S,S'} = 0$, $\mathcal{M}^{\mathcal{C}}_{(S,f),(S',f')} = 0$ for each f and f'. In this case, the previous and next states do not have to be (and cannot be required to be) consistent.

From the definition of $\mathcal{M}^{\mathcal{C}}$ it follows immediately that for each $(S, f) \in \mathcal{S}^{\mathcal{C}}$ and $S' \in \mathcal{S}$ with $\mathcal{M}_{S,S'} > 0$, there exists a unique function f' (defined by $f'(\Theta\sigma) = \bar{\sigma}((S, f))$ for each $\Theta\sigma \in \mathcal{C}$) such that $\mathcal{M}^{\mathcal{C}}_{(S,f),(S',f')} > 0$. For this function f', $\mathcal{M}^{\mathcal{C}}_{(S,f),(S',f')} = \mathcal{M}_{S,S'}$ and for each function $g \neq f'$, $\mathcal{M}^{\mathcal{C}}_{(S,f),(S',g)} = 0$. Using these facts, it is easily shown that $\mathcal{M}^{\mathcal{C}}$ defines a proper probability matrix.

From the definition of $\mathcal{M}^{\mathcal{C}}$, we also have that for all $(S, f), (S', f') \in \mathcal{S}^{\mathcal{C}}$ with $\mathcal{M}'_{(S,f),(S',f')} > 0$, $\overline{\Theta\sigma}((S', f')) = \bar{\sigma}((S, f))$ for each $\Theta\sigma \in \mathcal{C}$. Hence the value of an expression $\Theta\sigma$ in a state (S', f') is determined uniquely by the value of expression σ evaluated in some preceding state. If (S, f) has more than one preceding state, the values σ in these states are equal. This shows that the interpretation of our previous-state operator is sound.

Finally, we will define the initial state $S^{\mathcal{C}}_s$ of $(\mathcal{S}^{\mathcal{C}}, S^{\mathcal{C}}_s, \mathcal{M}^{\mathcal{C}})$. This state is defined by $S^{\mathcal{C}}_s = (S_s, f)$ such that for each $\Theta\sigma \in \mathcal{C}_{\Theta\mathbf{b}} \cup \mathcal{C}_{\Theta\mathbf{r}}$, $f(\Theta\sigma) = \bot$. So in the initial state, previous-state rewards are undefined.

The reader may have noticed that the state space of the Markov chain defined by $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ will in general not be countable. However, it is not difficult to prove that (since the original state space $\mathcal{S}$ is countable) the collection of states that are reachable from the initial state $S_s^{\mathcal{C}}$ is countable. It is therefore possible to reduce the chain by only considering the reachable states, thereby obtaining a discrete-time Markov chain with a countable state space. In the remainder of this text, we will assume that $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ is already of this reduced form.

Even if the original Markov chain has a finite-state space, the extended chain may be infinite. For instance, consider the real-valued 'counting' reward R defined by

$$R \stackrel{\text{def}}{=} \textit{if} \downarrow\Theta R \textit{ then } 0 \textit{ else } \Theta R + 1 \textit{ fi}.$$

Then starting with a Markov chain with a single state and a single transition, one obtains an extended chain with an infinite number of states, each assigning a different natural number to reward R. Here we state without proof that starting from an initial chain with a finite-state space, an extended chain with an infinite-state space can only be constructed if collection $\mathcal{C}$ contains recursively defined real-valued reward constants.

3.4. Construction method

If the extended Markov chain has a finite-state space, it is easily constructed using the following search algorithm:

1. Define collection *States To Visit* $= \{S_s^{\mathcal{C}}\}$ and *States Visited* $= \emptyset$.
2. If *States To Visit* $= \emptyset$, we are done. Otherwise remove an element (S, f) from *States To Visit* and add it to *States Visited*.
3. For each state T of the initial chain for which $\mathcal{M}_{S,T} > 0$, determine function g such that $g(\Theta\sigma) = \bar{\sigma}((S, f))$ for each $\Theta\sigma \in \mathcal{C}$. Notice that this function can be computed directly from rules (i) to (xiv) given in Section 3.3. Add a transition labelled with probability $\mathcal{M}_{S,T}$ from state (S, f) to state (T, g) in the extended chain. If $(T, g) \notin$ *States To Visit* $\cup$ *States Visited* then add (T, g) to *States To Visit*.
4. Continue with step 2.

Notice that the algorithm can be modified to generate a single trace of the chain. This algorithm consists of evaluating for each transition step, a number of simple update-functions corresponding to the previous-state temporal reward formulas in $\mathcal{C}$. This efficient algorithm computes the temporal rewards in an on-the-fly manner (without having to construct the complete extended chain) and can thus be applied for simulation purposes in case the extended Markov chain is very large or even infinite. In those cases statistical techniques (such as Markov chain Monte Carlo [13]) utilising central limit theorems for Markov chains can be applied to estimate the performance measures.

3.5. Incremental construction

Sometimes it is convenient to build the extended chain in an incremental fashion. This can be done by defining an increasing sequence $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \cdots \subseteq \mathcal{C}_n = \mathcal{C}$ of closed subsets of $\mathcal{C}$. Then starting from an initial chain $(\mathcal{S}, S_s, \mathcal{M})$ we first build the extended chain $(\mathcal{S}, S_s, \mathcal{M})^{\mathcal{C}_1}$. This extended chain is the initial chain for the next step of the construction by considering all rewards in $\mathcal{C}_1$ as atomic. We then obtain extended chain $(\mathcal{S}, S_s, \mathcal{M})^{\mathcal{C}_1\mathcal{C}_2}$. This process is repeated for all the other sets as well resulting in

chain $(\mathcal{S}, S_s, \mathcal{M})^{\mathcal{C}_1\mathcal{C}_2\cdots\mathcal{C}_n}$. By induction on n and by using the extended chain definition, it is easy to prove that this chain is the same as $(\mathcal{S}, S_s, \mathcal{M})^{\mathcal{C}}$, except for the naming of states. Corresponding states further assign the same values to all atomic rewards and to all rewards in $\mathcal{C}$. Applications of this incremental construction technique are demonstrated in Sections 3.7 and 4.2.

3.6. Preservation of long-run averages

In this subsection, we will prove that the construction method preserves long-run averages for atomic rewards. Let $(\mathcal{S}, S_s, \mathcal{M})$ define Markov chain $\{X_i | i \geq 1\}$. We assume that the probability space of this chain is given by triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the collection of all infinite-state sequences starting with S_s, $\mathcal{F}$ is the σ-algebra generated by the collection of cylindrical sets and $\mathbb{P}$ is the corresponding probability measure (see [7] for more details). Let $n \geq 1$. A finite-state sequence $\underline{S}_n = (S_1, S_2, \ldots, S_n)$ or infinite-state sequence $\underline{S} = (S_1, S_2, \cdots)$ is called a trace if for each $i : 1 \leq i < n$, respectively, each $i : i \geq 1$, $\mathcal{M}_{S_i,S_{i+1}} > 0$. We will denote the collection of all finite traces of length n with S_s as first state by $\mathit{Traces}_n(\mathcal{S}, S_s, \mathcal{M})$ and all infinite traces with S_s as first state by $\mathit{Traces}_\omega(\mathcal{S}, S_s, \mathcal{M})$. For any infinite trace $\underline{S}$, we let $\underline{S}_n$ denote the finite prefix of length n. For any finite trace $\underline{S}_n$, we define $\underline{S}_n^\omega$ to be the thin cylinder of infinite traces having $\underline{S}_n$ as prefix. The probability of occurrence of finite trace $\underline{S}_n$ and infinite trace $\underline{S}$ is denoted by $\mathbb{P}(\underline{S}_n)$ and $\mathbb{P}(\underline{S})$, respectively. These probabilities are given by $\mathbb{P}(\underline{S}_n) = \mathbb{P}(\underline{S}_n^\omega) = \prod_{i=1}^{n-1} \mathcal{M}_{S_i,S_{i+1}}$ and $\mathbb{P}(\underline{S}) = \mathbb{P}(\bigcap_{n=1}^{\infty} \underline{S}_n^\omega) = \prod_{i=1}^{\infty} \mathcal{M}_{S_i,S_{i+1}}$. Notice that $\mathbb{P}(\underline{S}_n) = \mathbb{P}(X_m = S_1, X_{m+1} = S_2, \ldots, X_{m+n-1} = S_n | X_m = S_1)$ and $\mathbb{P}(\underline{S}) = \mathbb{P}(X_m = S_1, X_{m+1} = S_2, \ldots | X_m = S_1)$ for each m such that $\mathbb{P}(X_m = S_1) > 0$.

Lemma 1. *Let $\underline{S}_n$ denote a finite trace of $(\mathcal{S}, S_s, \mathcal{M})$ and let (S_1, f_1) be a state of $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$. Then there exists a unique sequence $(f_1, f_2, \ldots, f_n)$ of functions such that $((S_1, f_1), (S_2, f_2), \ldots, (S_n, f_n))$ is a finite trace of $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$. Further $\mathbb{P}(\underline{S}_n) = \mathbb{P}((S_1, f_1), (S_2, f_2), \ldots, (S_n, f_n))$. Vice versa, if $((S_1, f_1), (S_2, f_2), \ldots, (S_n, f_n))$ is a finite trace of $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$, then $\underline{S}_n$ is a finite trace of $(\mathcal{S}, S_s, \mathcal{M})$ and $\mathbb{P}((S_1, f_1), (S_2, f_2), \ldots, (S_n, f_n)) = \mathbb{P}(\underline{S}_n)$. The same holds for infinite traces.*

Proof. The proof follows from the definition of probability matrix $\mathcal{M}^{\mathcal{C}}$. The crux of the proof is that since each transition in a trace has a positive probability, each function f_n ($n \geq 2$) is determined uniquely from state sequence $(S_1, f_1), (S_2, f_2), \ldots, (S_{n-1}, f_{n-1})$ and state S_n. □

As a consequence of this lemma there exists a unique trace in $\mathit{Traces}_n(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ (respectively in $\mathit{Traces}_\omega(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$) for each trace in $\mathit{Traces}_n(\mathcal{S}, S_s, \mathcal{M})$ (respectively in $\mathit{Traces}_\omega(\mathcal{S}, S_s, \mathcal{M})$) and vice versa. Further, the probabilities of occurrence of these traces are the same. For all traces $\underline{S}_n \in \mathit{Traces}_n(\mathcal{S}, S_s, \mathcal{M})$ and $\underline{S} \in \mathit{Traces}_\omega(\mathcal{S}, S_s, \mathcal{M})$, we will denote the corresponding traces in $\mathit{Traces}_n(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ and $\mathit{Traces}_\omega(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ by $\underline{S}_n^{\mathcal{C}}$ and $\underline{S}^{\mathcal{C}}$, respectively. Notice that $(\underline{S}^{\mathcal{C}})_n = (\underline{S}_n)^{\mathcal{C}}$.

Theorem 1. *Let $\{X_i | i \geq 1\}$ and $\{X_i^{\mathcal{C}} | i \geq 1\}$ be Markov chains defined by $(\mathcal{S}, S_s, \mathcal{M})$ and $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$, respectively. Further, let $r \in \mathcal{R}$ be an atomic real-valued reward and let $c \in \mathbb{R}$ be a constant. Then $1/n \sum_{i=1}^{n} \bar{r}(X_i) \overset{\text{a.s.}}{\to} c$ if and only if $1/n \sum_{i=1}^{n} \bar{r}(X_i^{\mathcal{C}}) \overset{\text{a.s.}}{\to} c$.*

Proof. We will only prove the 'only if' part of Theorem 1; the 'if' part of the proof proceeds in a similar way.

Assume $1/n \sum_{i=1}^{n} \bar{r}(X_i) \overset{\text{a.s.}}{\to} c$.

Define set E by $E = \{\underline{S} \in \mathit{Traces}_\omega(\mathcal{S}, S_s, \mathcal{M}) | \lim_{n\to\infty} 1/n \sum_{i=1}^{n} \bar{r}(S_i) = c\}$.

Then by definition of almost sure convergence, E is an event (i.e. is measurable) and $\mathbb{P}(E) = 1$. Now define set $E^{\mathcal{C}}$ by $E^{\mathcal{C}} = \{\underline{S} \in \mathit{Traces}_\omega(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}}) | \lim_{n\to\infty} 1/n \sum_{i=1}^{n} \bar{r}(S_i) = c\}$.

Notice that since r is an atomic reward, depending only on the state-part S of some extended state (S, f), we have that $\underline{S} \in E$ iff $\underline{S}^C \in E^{\mathcal{C}}(*)$.

We have to prove that $E^{\mathcal{C}}$ is an event with $\mathbb{P}(E^{\mathcal{C}}) = 1$. To this end define for each $n \geq 1$ sets E_n and $E_n^{\mathcal{C}}$ by $E_n = \bigcup_{\underline{S}\in E} \underline{S}_n^{\omega}$ and $E_n^{\mathcal{C}} = \bigcup_{\underline{S}\in E^C} \underline{S}_n^{\omega}$, respectively.

Now E_n and $E_n^{\mathcal{C}}$ are cylindrical sets of rank n and are therefore events. Further, for each $n \geq 1$, we have $\mathbb{P}(E_n^{\mathcal{C}}) = \{\text{by countable additivity}\} \sum_{\{\underline{S}_n | \underline{S}\in E^C\}} \mathbb{P}(\underline{S}_n^{\omega}) = \{\text{by definition of } \mathbb{P}(\underline{S}_n)\} \sum_{\{\underline{S}_n | \underline{S}\in E^C\}} \mathbb{P}(\underline{S}_n) = \{\text{by}(*)\} \sum_{\{\underline{S}_n | \underline{S}\in E\}} \mathbb{P}(\underline{S}_n^{\mathcal{C}}) = \{\text{by Lemma1}\} \sum_{\{\underline{S}_n | \underline{S}\in E\}} \mathbb{P}(\underline{S}_n) = \{\text{by countable additivity}\}\mathbb{P}(E_n)$.

Further $E_1 \supseteq E_2 \supseteq \cdots$ and $E_1^{\mathcal{C}} \supseteq E_2^{\mathcal{C}} \supseteq \cdots$ are decreasing sequences of events with limits E and $E^{\mathcal{C}}$, respectively.

Hence $E^{\mathcal{C}}$ is an event and $\mathbb{P}(E^{\mathcal{C}}) = \mathbb{P}(\bigcap_{i=1}^{\infty} E_i^{\mathcal{C}}) = \lim_{i\to\infty} \mathbb{P}(E_i^{\mathcal{C}}) = \lim_{i\to\infty} \mathbb{P}(E_i) = \mathbb{P}(\bigcap_{i=1}^{\infty} E_i) = \mathbb{P}(E) = 1$. □

Theorem 1 states that for each atomic real-valued reward, the long-run average rewards of the original chain and the extended one are equal. It is not difficult to check that this result also holds for temporal real-valued rewards whose definition does not (directly or indirectly) contain the previous-state operator (at least if we extend the reward interpretation function $\bar{\cdot}$ of the original chain to deal with these rewards).

In conclusion, we have that the extended chain $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ can be used to

- compute all the long-run average (atomic) rewards that can be computed for the original chain $(\mathcal{S}, S_s, \mathcal{M})$ (by Theorem 1);
- compute all the long-run average values for the temporal real-valued rewards defined in $\mathcal{C}$ by definition.

Here we would like to note that this is only possible if the extended chain is ergodic. It is easy to show that the property of ergodicity is not preserved by the construction method in general. In the next subsection, we will derive conditions that guarantee the preservation of ergodicity.

Notice that in case the extended chain is not ergodic, long-run averages may still exist (if they exist for atomic rewards in the original chain they also exist in the extended chain by Theorem 1). But since the ergodic theorem is not applicable then, these long-run averages cannot be expressed in terms of the equilibrium probabilities of the chain.

For example, the Markov chains depicted in Fig. 8 both have a long-run average r-reward of 1 (see also [34]). For the left-hand ergodic chain, this value can be expressed in terms of the equilibrium probabilities

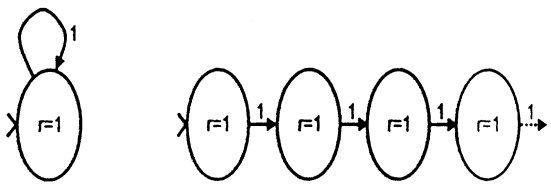

Fig. 8. Markov chains with equal long-run averages.

of this chain. The right-hand chain, on the other hand, is transient and for this chain equilibrium probabilities do not exist.

3.7. Preservation of ergodicity

The construction method for temporal rewards does not preserve the property of ergodicity in general. Due to the expressive power of the formalism, a procedure for deciding from an initial chain and reward collection whether the chain to construct will be ergodic, is certainly difficult, probably even impossible, to establish. It would therefore be desirable if we could find conditions on the initial chain and reward collection that guarantee the preservation of ergodicity. Such conditions will be developed in this section. The conditions have turned out to be suitable in all practical applications we have encountered thus far.

The property of ergodicity considered in this paper is taken from [8,31]. Let $(\mathcal{S}, S_s, \mathcal{M})$ define a Markov chain. We define this chain to be ergodic if and only if it has some positive state $S_r \in \mathcal{S}$ that is ultimately reached from any state $S \in \mathcal{S}$ with probability 1 (a state is called positive if it is positive recurrent).[6] This implies that the chain has no two disjoint closed sets of states. Further the (non-empty) collection C of recurrent states is irreducible, each state in C is positive and each state in $\mathcal{S}\backslash C$ is transient. It can be shown that each ergodic chain has a unique equilibrium distribution $\{\pi_S | S \in \mathcal{S}\}$. It further satisfies the following important ergodic theorem.

Ergodic Theorem. *Let $(\mathcal{S}, S_s, \mathcal{M})$ be ergodic and let $r \in \mathcal{R}$ be an atomic real-valued reward. Then*

$$\frac{1}{n}\sum_{i=1}^{n} \bar{r}(X_i) \xrightarrow{\text{a.s.}} \sum_{S\in\mathcal{S}} \bar{r}(S)\cdot\pi_S \text{ provided that } \sum_{S\in\mathcal{S}} \bar{r}(S)\cdot\pi_S \text{ converges absolutely.}$$

Here the condition of absolute convergence is required if reward r takes both positive and negative values. In case $r(S) \geq 0$ for all $S \in \mathcal{S}$ or $r(S) \leq 0$ for all $S \in \mathcal{S}$, the condition can be omitted if we make use of the extended real number system and allow limits to take infinite values.

Starting with an ergodic chain defined by $(\mathcal{S}, S_s, \mathcal{M})$ our construction method does not in general yield another ergodic chain. This is due to the use of recursively defined reward constants, which will be shown in this subsection. For example, consider the following reward R counting the number of epochs since the last occurrence of event e (where e is a Boolean-valued atomic reward):

$$R \stackrel{\text{def}}{=} \textit{if } e \textit{ then } 0 \textit{ else } \Theta R + 1 \textit{ fi}.$$

In each state where e occurs R is set to 0. In any other state R is increased by 1 (if we assume that e is defined in every state). The extended chain is built starting from the closure of reward collection $\{R\}$. The only reward of the form $\Theta\sigma$ in this closure is ΘR. Hence all extended states are of the form (S, f), where $S \in \mathcal{S}$ and where f is a function from $\{\Theta R\}$ to $\mathbb{R}$. Now it is not hard to find out that in case the initial chain only has transient states satisfying e, the extended chain will be transient. The reason for this is that from the moment the extended chain has reached a state (S, f) where S is positive, reward R will keep on increasing and is never 'reset' to 0 anymore.

[6] In literature the term ergodic chain often refers to an irreducible, positive recurrent Markov chain. In this paper, we allow the chain to have transient states as well, but we only consider chains having one initial state.

The intuition behind guaranteeing the extended chain to be ergodic is as follows. We assume the existence of a positive state S in the initial chain determining the value of R. This means that the value of R in any extended state (S, f) is independent on the past information stored in f. For instance any state satisfying e determines R. If we define V to be the collection of all extended states of the form (S, f), we thus have that for any two extended states $(S, f_1), (S, f_2) \in V$, R will assume the same value. Now consider any successor state (T, g) of $(S, f) \in V$. Then, since g maps ΘT to the value of T in (S, f), (T, g) is a successor state of any state in V. Further the transitions from any of the states in V to (T, g) happen with the same probabilities. Now since S is reachable from any state in the initial chain with probability 1, set V is reachable from any state in the extended chain with probability 1. Also since S is positive, the expected first passage time of V is finite starting from any positive state. Because all states in V have a common successor state (T, g), this state must be positive and reachable from any other state with probability 1. Hence the extended Markov chain is ergodic.

We will now formalise and generalise the above intuitions. Let $(\mathcal{S}, S_s, \mathcal{M})$ define a Markov chain, let $S \in \mathcal{S}$ and let σ be a temporal reward. We say that S determines σ (or equivalently σ is determined by S) if there exists a constant c such that for all $\mathcal{C}$ containing σ, $\bar{\sigma}((S, f)) = c$ for all $f \in (\mathbb{B} \cup \mathbb{R})^{\mathcal{C}_{\Theta \mathbf{b}} \cup \mathcal{C}_{\Theta \mathbf{r}}}$. Hence S determines σ if it contains all the information necessary to determine the value of σ. For example, atomic rewards are always determined by any state and reward *if b then r else Θr fi* (where r is an atomic reward) is determined by any state satisfying atomic reward b. In case σ is determined by S, its value will be denoted by $\bar{\sigma}(S)$. It is easy to prove that if S determines σ any extended state (S, f) determines σ as well.

Theorem 2. *Let $(\mathcal{S}, S_s, \mathcal{M})$ define an ergodic chain, let $\mathcal{C}$ be a closed finite collection of temporal rewards and let $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ define the constructed extended chain. Determine n as the smallest $k \geq 1$ such that $\Theta^{k+1}\sigma \notin \mathcal{C}$ for all $\sigma \in \mathcal{C}$. Let $V_1, \ldots, V_n$ be sets of temporal rewards such that $V_i = \{\sigma \in \mathcal{C} | \sigma' \equiv \Theta^i \sigma$ for some $\sigma' \in \mathcal{C}$ and $\sigma \not\equiv \Theta\sigma''$ for all $\sigma'' \in \mathcal{C}\}$. Then the extended chain is ergodic if there exists a finite trace $\underline{S}_n$ of positive states in $\mathcal{S}$ such that S_i determines all rewards in V_{n-i+1} for all $i \in \{1, \ldots, n\}$.*

Before we prove Theorem 2, we give some definitions and another lemma. Let U be a set of finite traces. Then U will be called proper if all the traces in U have the same initial state and if no trace in U is a proper prefix of another trace in U. If U is a proper set of finite traces, $\mathbb{P}(U)$ and $\mathcal{E}(U)$ are defined by

$$\mathbb{P}(U) = \sum_{\underline{S}_n \in U} \mathbb{P}(\underline{S}_n), \qquad \mathcal{E}(U) = \sum_{\underline{S}_n \in U} (n-1)\mathbb{P}(\underline{S}_n).$$

In case $\mathbb{P}(U) = 1$ expression $\mathcal{E}(U)$ denotes the expected length of the traces in U, where the length of trace $\underline{S}_n$ is defined by $n-1$. Let U be a proper collection of finite traces with the same final state and let V be a proper collection of finite traces such that the final states in U are the same as the initial states in V if both U and V are non-empty. Then $U \cdot V$ is defined by

$$U \cdot V = \{(S_1, \ldots, S_n, T_2, \ldots, T_k) | \underline{S}_n \in U \text{ and } \underline{T}_k \in V\}.$$

Notice that $U \cdot V$ is proper, $\mathbb{P}(U \cdot V) = \mathbb{P}(U)\mathbb{P}(V)$ and $\mathcal{E}(U \cdot V) = \mathbb{P}(U)\mathcal{E}(V) + \mathbb{P}(V)\mathcal{E}(U)$.

Lemma 2. *Let $(\mathcal{S}, S_s, \mathcal{M})$ define an ergodic chain and let $\underline{S}_n$ be a finite trace consisting of positive states. Then starting from any state, $\underline{S}_n$ is eventually traversed with probability* 1. *Further from any positive state in the chain the expected first traversal time of $\underline{S}_n$ is finite.*

Proof. Let $W_S^{\underline{S}_n}$ denote the collection of finite traces of the form $(S, S'_1, \ldots, S'_m)$ such that $\underline{S}'_m$ ends with sub-trace $\underline{S}_n$ and has no sub-trace $\underline{S}_n$ elsewhere. Further let $W_{S,T}^{\underline{S}_n}$ be defined as $\{\underline{T}_m \in W_S^{\underline{S}_n} | T_2 \neq T\}$. Notice that $W_S^{\underline{S}_n}$ and $W_{S,T}^{\underline{S}_n}$ are proper sets. We have to prove that $\mathbb{P}(W_S^{\underline{S}_n}) = 1$ for each S and $\mathcal{E}(W_S^{\underline{S}_n}) < \infty$ for each positive S. The proof is by induction on n. For $n = 1$, the result follows immediately from the fact that $\underline{S}_1$ consists of a single positive state. For the induction step define collection $X_S^{\underline{S}_{n+1}}$ by

$$X_S^{\underline{S}_{n+1}} = \bigcup_{i=0}^{\infty} W_S^{\underline{S}_n} \cdot (W_{S_n,S_{n+1}}^{\underline{S}_n})^i \cdot \{(S_n, S_{n+1})\}.$$

Here $(W_{S_n,S_{n+1}}^{\underline{S}_n})^i = \{(S_n)\}$ if $i = 0$ and $(W_{S_n,S_{n+1}}^{\underline{S}_n})^i = W_{S_n,S_{n+1}}^{\underline{S}_n} \cdot (W_{S_n,S_{n+1}}^{\underline{S}_n})^{i-1}$ otherwise. Now for each $i \geq 0$, we have

$$\mathbb{P}(W_S^{\underline{S}_n} \cdot (W_{S_n,S_{n+1}}^{\underline{S}_n})^i \cdot \{(S_n, S_{n+1})\}) = \mathbb{P}(W_S^{\underline{S}_n})\mathbb{P}(W_{S_n,S_{n+1}}^{\underline{S}_n})^i \mathcal{M}_{S_n,S_{n+1}}.$$

By induction and since $\mathbb{P}(W_{S_n,S_{n+1}}^{\underline{S}_n}) = 1 - \mathcal{M}_{S_n,S_{n+1}}$ this expression equals $\mathcal{M}_{S_n,S_{n+1}}(1-\mathcal{M}_{S_n,S_{n+1}})^i$. Hence $\mathbb{P}(X_S^{\underline{S}_{n+1}}) = \sum_{i=0}^{\infty} \mathcal{M}_{S_n,S_{n+1}}(1 - \mathcal{M}_{S_n,S_{n+1}})^i = 1$. In a similar way, we can derive

$$\begin{aligned}&\mathcal{E}(W_S^{\underline{S}_n} \cdot (W_{S_n,S_{n+1}}^{\underline{S}_n})^i \cdot \{(S_n, S_{n+1})\}) \\ &\quad = \mathcal{M}_{S_n,S_{n+1}}(\mathcal{E}(W_S^{\underline{S}_n})\mathbb{P}(W_{S_n,S_{n+1}}^{\underline{S}_n})^i + \mathrm{i}\mathcal{E}(W_{S_n,S_{n+1}}^{\underline{S}_n})\mathbb{P}(W_{S_n,S_{n+1}}^{\underline{S}_n})^{i-1} + \mathbb{P}(W_{S_n,S_{n+1}}^{\underline{S}_n})^i).\end{aligned}$$

From this, it follows that

$$\mathcal{E}(X_S^{\underline{S}_{n+1}}) = 1 + \mathcal{E}(W_S^{\underline{S}_n}) + \frac{\mathcal{E}(W_{S_n,S_{n+1}}^{\underline{S}_n})}{(\mathcal{M}_{S_n,S_{n+1}})^2}.$$

Now this expression is finite since $\mathcal{E}(W_S^{\underline{S}_n})$ is finite by induction. By induction we also have that $\mathcal{E}(W_{S_n}^{\underline{S}_n})$ is finite and from this it follows immediately that $\mathcal{E}(W_{S_n,S_{n+1}}^{\underline{S}_n})$ is also finite. Notice further that $\mathcal{M}_{S_n,S_{n+1}} > 0$.

We conclude by remarking that each trace or prefix thereof in $X_S^{\underline{S}_{n+1}}$ is also a trace in $W_S^{\underline{S}_{n+1}}$. From this, it follows that $\mathbb{P}(W_S^{\underline{S}_{n+1}}) \geq \mathbb{P}(X_S^{\underline{S}_{n+1}})$ and $\mathcal{E}(W_S^{\underline{S}_{n+1}}) \leq \mathcal{E}(X_S^{\underline{S}_{n+1}})$. Hence $\mathbb{P}(W_S^{\underline{S}_{n+1}}) = 1$ and $\mathcal{E}(W_S^{\underline{S}_{n+1}}) < \infty$. This concludes the proof of Lemma 2. □

We are now able to give a proof of Theorem 2.

Proof of Theorem 2. Define W as the collection of finite traces of $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ of the form $((S_1, f_1), \ldots, (S_n, f_n))$. Using Lemma 1, it is not hard to show that $W \neq \emptyset$. Assume that $((S_1, g_1), \ldots, (S_n, g_n)) \in W$. Pick a state (T, g) such that $((S_1, g_1), \ldots, (S_n, g_n), (T, g))$ is a trace (notice that such a state must exist). We will show that for all $((S_1, h_1), \ldots, (S_n, h_n)) \in W$, $((S_1, h_1), \ldots, (S_n, h_n), (T, g))$ is a trace of $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$. Assume $((S_1, h_1), \ldots, (S_n, h_n)) \in W$ and let $\Theta\sigma' \in \mathcal{C}$. Then there is some $k \geq 1$ and $\sigma \not\equiv \Theta\sigma''$ (for all $\sigma'' \in \mathcal{C}$) such that $\Theta\sigma' = \Theta^k\sigma$. Then $\sigma \in V_k$ and σ is determined by S_{n-k+1}. By definition $\bar{\sigma}((S_{n-k+1}, g_{n-k+1})) = \bar{\sigma}((S_{n-k+1}, h_{n-k+1}))$. But then it follows that $\overline{\Theta^{k-1}\sigma}((S_n, g_n)) = \overline{\Theta^{k-1}\sigma}((S_n, h_n))$. We further have that $\overline{\Theta^k\sigma}((T, g)) = \overline{\Theta^{k-1}\sigma}((S_n, g_n))$. But then we also have $\overline{\Theta^k\sigma}((T, g)) = \overline{\Theta^{k-1}\sigma}((S_n, h_n))$ and thus $\overline{\Theta\sigma'}((T, g)) = \overline{\sigma'}((S_n, h_n))$. This holds for all $\Theta\sigma' \in \mathcal{C}$. Further $\mathcal{M}_{S_n,T} > 0$. Hence $((S_1, h_1), \ldots, (S_n, h_n), (T, g))$ is a trace of $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$.

By Lemma 2, trace $\underline{S}_n$ is eventually traversed with probability 1 starting from any state in $\mathcal{S}$. Further, starting from any positive state, the expected first traversal time of $\underline{S}_n$ is finite. By applying Lemma 1, it is then easy to prove that the collection of the traces in W is reachable from any state in $\mathcal{S}^{\mathcal{C}}$ with probability 1. Further, starting from any extended state (S, f) where S is a positive state in $\mathcal{S}$, the expected time to traverse some trace in W is finite. Since the traces of W are of the form $((S_1, f_1), \ldots, (S_n, f_n))$, where each S_i is positive in $\mathcal{S}$, the expected time to traverse any trace in W, when such a trace was just traversed, is also finite (and independent of the trace just traversed). After the traversal of any trace in W, the extended chain can always transit to state (T, g) with probability $\mathcal{M}_{S_n,T} > 0$ (by the discussion above). By using an argument similar as that of the proof of Lemma 2, it can be proved that (T, g) must be a positive state, reachable from any state in $\mathcal{S}^{\mathcal{C}}$ with probability 1. Hence the extended Markov chain is ergodic. This ends the proof of Theorem 2. □

Consider reward R defined above again. The only reward of the form $\Theta\sigma$ in the closure of $\{R\}$ is ΘR. When applying Theorem 2, we get $n = 1$ and $V_1 = \{R\}$. Hence the extended Markov chain is ergodic if there exists a positive state in the initial chain that determines R. Remark that R is determined by any state satisfying e. Consequently, the extended chain is ergodic if the initial chain has a positive state that satisfies e.

As another example consider reward S defined by

$$S \stackrel{\text{def}}{=} \textit{if } e \textit{ then } 0 \textit{ else if } \downarrow \Theta\Theta S \textit{ then } 0 \textit{ else } \Theta\Theta S + 1 \textit{ fi fi}.$$

Just as in the case of reward R, one might expect the extended chain to be ergodic if e is satisfied by some positive state in the initial ergodic chain. However, it is easy to construct a counterexample showing that this is not true in general. Notice that (by Theorem 2), the extended chain will always be ergodic if two successive positive states in the initial ergodic chain satisfy e.

Recursively defined real-valued reward constants are not the only possible cause for constructing a non-ergodic chain. For example, the Boolean-valued reward constant B defined by

$$B \stackrel{\text{def}}{=} \textit{if } \uparrow b \textit{ then } b \textit{ else } \Theta B \textit{ fi}$$

can result in a chain with two disjoint closed sets of states if both b and $\neg b$ are satisfied by some transient states and if b is undefined in all recurrent states. By Theorem 2, the extended chain will be ergodic if b is defined in some positive state.

Sometimes Theorem 2 is not applicable directly. Consider for instance rewards R and T defined by

$$R \stackrel{\text{def}}{=} \textit{if } e \textit{ then } 0 \textit{ else } \Theta R + 1 \textit{ fi}, \qquad T \stackrel{\text{def}}{=} \textit{if } f \textit{ then } 0 \textit{ else } \Theta T + 1 \textit{ fi}.$$

To apply Theorem 2, there must be a positive state in the initial chain determining both R and T. In case such a state does not exist, we can try to use the incremental construction technique given in Section 3.5. We first build an extended chain using reward R only, and extend this chain by using T in a second step. Now suppose that the initial chain has a positive state determining R and also has a positive state determining T. Then the extended chain resulting from the first step is ergodic. Further it is easy to show that since S is determined by some positive state in the initial chain, it is also determined by some positive state in the extended chain. Hence the extended chain resulting from the second step is also ergodic.

The previous idea can be generalised to the following useful result. Let $(\mathcal{S}, S_s, \mathcal{M})$ define the initial chain and let $\mathcal{C}$ denote a finite closed collection of temporal rewards. If $\mathcal{C}$ can be split into n (not necessarily disjoint) closed collections of rewards, each satisfying the conditions of Theorem 2, then $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ is ergodic. By applying the incremental construction technique of Section 3.5, it is further straightforward to show that the property of ergodicity is always preserved by the construction method if no recursively defined reward constants are used. In Section 4.2, the incremental construction technique together with Theorem 2 is applied to prove the preservation of ergodicity for a collection of complex temporal rewards.

3.8. Related research

The idea to develop expressive techniques for specifying performance properties is not new. Especially in the area of continuous-time Markov chains, a number of specification techniques have been developed. Our formalism of temporal rewards is developed in the context of discrete-time Markov chains. However, it is not difficult to make the technique suitable for continuous formalisms (see the comparison made in this section with the technique of path-based automata). For this reason, we include related research from the continuous-time setting as well.

In the context of the stochastic process algebra PEPA, reward specification formalisms based on modal logic have been developed [9–11]. In these formalisms, states to be assigned a particular reward are specified by means of a modal logic formula. Each state satisfying this formula is assigned a reward as specified by a simple arithmetic expression. In this way, states to be assigned a reward can be specified conveniently by characterising these states in a behavioural way. However, the modal logic approaches cannot deal with the accumulation of rewards over sequences of states. Each state of the Markov chain underlying a PEPA model is assigned a single reward value and no extended chain is built. The reason is that the assignment of a reward to a state depends on the possible future behaviours from that state only; it cannot depend on the behaviour performed in the past. As a consequence ‘counting’ rewards are difficult to deal with [10]. Of course, reward assignments in our formalism cannot depend on future behaviours but only on behaviours performed in the past.

In [5], a method to associate rewards with terms of the stochastic process algebra EMPA is introduced. Instead of using a separate formalism for specifying rewards, the language of the process algebra itself is extended. The rewards are described within the actions performed by the algebraic terms. Although this method is less powerful in general than the logic-based approach of [9], it is somewhat simpler and allows for the easy specification of several frequently occurring performance measures. Since rewards have to be defined within the algebraic models themselves, these models are somewhat polluted, be it in a minor way. Just as in the case of the modal logic approaches, the method does not allow rewards to be accumulated over sequences of states.

In [25], a technique of path-based reward variables is introduced for continuous-time Markov chains that allows rewards to be accumulated over sequences of states and transitions. Path-based rewards are specified by means of deterministic finite-state path automata. A path automaton accepts inputs from the initial Markov model. Such a model consists of states and events. Events occur at specified rates and the occurrence of an event triggers a state transition. A path automaton accepts input traces (called paths) of state-event pairs of the model. Based on the path automaton, a reward structure is defined that assigns (impulse and rate) rewards dependent on the state of the chain and a possible event occurring in this state, as well as on the state of the automaton. In this way, different rate rewards can be assigned to the same state of the chain and different impulse rewards to the same event. A state-space construction procedure is

presented for automatically generating an extended state space that supports multiple path-based reward variables. Each path automaton has a collection of final states that are interpreted by this construction procedure as indicating that if such a final state is encountered, the current state of the chain should not be explored any further.

The construction method for temporal rewards can easily be adapted to deal with continuous models. To this end, one only has to switch from transition probabilities to transition rates. Both the atomic and temporal rewards are then automatically defined on the states of the continuous-time Markov chain. Due to the arithmetic operators and recursive relations, the temporal reward formalism is then somewhat more flexible and expressive than the formalism of path automata. In particular, each path automaton can be encoded in terms of temporal rewards in the following way:

- Starting from a continuous model with states and events as described above, one first shifts the event information inside the states (in a similar fashion as described in Section 2.3[7]). Event and state information can then be encoded as atomic rewards.
- The states (including final states) and the behaviour of a path automaton can be encoded by a recursively defined temporal real-valued reward constant.
- The reward structure can be encoded by appropriate real-valued (non-temporal) rewards.

Encoding temporal rewards in terms of path automata, on the other hand, is not easy in general. For instance, a 'counting' reward counting the number of occurrences of a certain event before some other event happens, requires a path automaton with infinitely many states. Such an automaton cannot be specified without proper linguistic facilities. Also, due to the lack of arithmetic support, accumulations (such as sums and products) of real-valued rewards are cumbersome, if not impossible, to specify with path automata.

Currently interesting work is carried out in the context of the temporal logic continuous stochastic logic (CSL). CSL was introduced in [2] and the verification via model checking is described in [3]. An extension of the logic to real-valued rewards is given in [4] and in [20] it is described how to move from state-based to transition-based measures. CSL allows the specification and verification of state-based and path-based dependability measures as well as of steady-state dependability measures in a compact and flexible way. CSL differs from temporal rewards in the sense that in CSL one specifies a quantitative property and verifies whether the model satisfies this property (which means that the property has to hold for all the initial states of the model). A temporal reward, on the other hand, does not specify a property to be satisfied by the model, but extends the original model and assigns accumulated rewards to the (extended) states. Based on the extended model, complex properties can be specified and analysed with relative ease, but the specification of the properties themselves must be done in another way, for instance by defining the limit of a sequence of random variables as is done in this paper or by applying a temporal logic.

An approach that actually combines the definition of an extended model with a logic to specify behavioural properties is presented in [1]. This work introduces a framework for specifying and verifying long-run average properties of systems in which both non-deterministic and probabilistic choices are present. The framework uses a discrete model, based on a timed extension of Markov decision processes, which is similar to the semantic model of POOSL. To enable the definition of long-run average

[7] In fact a similar shift of information from transitions to states is performed by the state-space construction method for path automata.

properties, the concept of experiment is introduced. Experiments are deterministic finite-state automata used to describe behaviour patterns of interest. Experiments associate with each occurrence of these patterns a real-valued outcome. The synchronous product of an initial model and an experiment determines an extended model encoding these outcomes as (atomic) rewards. The long-run average properties themselves are specified in extended versions of CTL and CTL* and model checking techniques are used to verify whether the model satisfies these properties.

Experiments are closely related to path automata [25] as well as to temporal rewards. Just as path automata can be encoded as temporal rewards, experiments can be encoded as temporal rewards as well. On the other hand, encoding temporal rewards in terms of experiments is not possible in general.

4. Examples

In this section, some examples of temporal rewards will be given. In Section 4.1, we will work out the performance metrics (iii) and (iv) for the initial Markov chain of Fig. 7. Section 4.2 gives a more involved example of the specification of the long-run average transmission rate and the long-run variance in transmission rate on a communication channel.

4.1. Average and variance of inter-arrival time

Metrics (iii) and (iv) were defined in Section 3.1 as:

(iii) The long-run average time between two *in* actions.
(iv) The long-run variance of time between two *in* actions.

We assume that atomic rewards *in*, *out* and t are as defined in Section 2.3. Boolean-valued rewards *in* and *out* indicate whether action *in* respectively *out* is performed in a state. Real-valued reward t denotes the amount of time progress in a state.

Based on these atomic rewards, we define a temporal reward T denoting the time since the last occurrence of action *in*. The real-valued reward constant T is defined by the following equation:

$$T \stackrel{\text{def}}{=} \mathit{if}\ \mathit{in}\ \mathit{then}\ 0\ \mathit{else}\ \Theta T + t\ \mathit{fi}.$$

The idea is that in a state satisfying *in*, T is set to 0. In any other state, T is increased with the amount of time progress in that state. If T is undefined, an action *in* did not yet occur in the past. Note that an increase with t leaves T undefined in this case. Note further that in any state where action *in* occurs, the time since the previous *in* action is given by ΘT, where ΘT is undefined if no such action did occur yet.

If $\{X_i^{\mathcal{C}} | i \geq 1\}$ denotes the extended Markov chain, we can describe performance metric (iii) as the limiting behaviour of random variable

$$\frac{\sum_{i=1}^{n} \overline{\mathit{if}\ \mathit{in} \wedge \uparrow \Theta T\ \mathit{then}\ \Theta T\ \mathit{else}\ 0\ \mathit{fi}}(X_i^{\mathcal{C}})}{\sum_{i=1}^{n} \overline{\mathit{if}\ \mathit{in} \wedge \uparrow \Theta T\ \mathit{then}\ 1\ \mathit{else}\ 0\ \mathit{fi}}(X_i^{\mathcal{C}})}.$$

Here the numerator denotes the sum of inter-arrival times in the first n epochs of the Markov chain. The denominator denotes the number of such inter-arrival times.

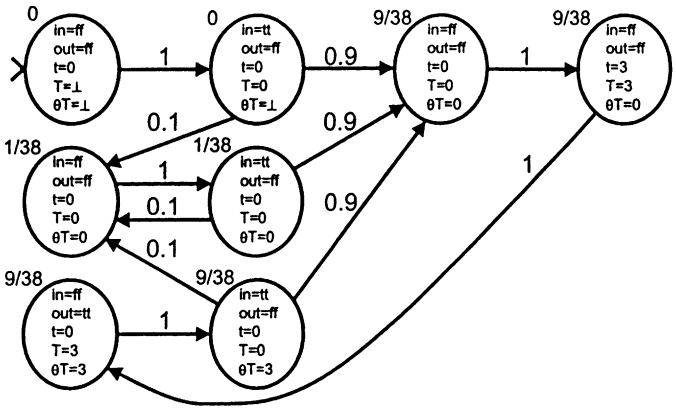

Fig. 9. Extended Markov chain definition.

In the construction of the extended Markov chain definition, we start with the closure $\mathcal{C}$ of reward collection

$$\{\mathit{if}\, \mathit{in} \wedge \uparrow \Theta T\, \mathit{then}\, \Theta T\, \mathit{else}\, 0\, \mathit{fi}, \mathit{if}\, \mathit{in} \wedge \uparrow \Theta T\, \mathit{then}\, 1\, \mathit{else}\, 0\, \mathit{fi}\}.$$

The only reward of the form $\Theta\sigma$ in $\mathcal{C}$ is ΘT. Therefore a state in the extended chain is of the form (S, f), where S is a state of the initial Markov chain (Fig. 7) and where f is a partial function from $\{\Theta T\}$ to $\mathbb{R}$. By Theorem 2, the extended chain is ergodic if action *in* is performed in some positive state in the Markov chain of Fig. 7. Notice that this is indeed the case.

It is not difficult to verify that the extended Markov chain definition $(\mathcal{S}^{\mathcal{C}}, S_s^{\mathcal{C}}, \mathcal{M}^{\mathcal{C}})$ is as shown in Fig. 9. For reasons of clarity, next to the atomic rewards and the temporal reward ΘT, the figure also shows the values of reward T.

By applying the ergodic theorem for Markov chains, the random variable defined above converges strongly to the constant given by

$$\frac{\sum_{S_i^{\mathcal{C}}} \overline{\mathit{if}\, \mathit{in} \wedge \uparrow \Theta T\, \mathit{then}\, \Theta T\, \mathit{else}\, 0\, \mathit{fi}}(S_i^{\mathcal{C}}) \cdot \pi_{S_i^{\mathcal{C}}}}{\sum_{S_i^{\mathcal{C}}} \overline{\mathit{if}\, \mathit{in} \wedge \uparrow \Theta T\, \mathit{then}\, 1\, \mathit{else}\, 0\, \mathit{fi}}(S_i^{\mathcal{C}}) \cdot \pi_{S_i^{\mathcal{C}}}}.$$

It is now easy to calculate that performance metric (iii) has value 27/10 (the equilibrium probabilities are indicated at the upper-left positions of the states in Fig. 9).

Having constructed the extended Markov chain for metric (iii), the value of metric (iv) is also easy to compute. This metric is defined by the limiting behaviour of random variable

$$\frac{\sum_{i=1}^{n} \overline{\mathit{if}\, \mathit{in} \wedge \uparrow \Theta T\, \mathit{then}\, (\Theta T - 27/10)^2\, \mathit{else}\, 0\, \mathit{fi}}(X_i^{\mathcal{C}})}{\sum_{i=1}^{n} \overline{\mathit{if}\, \mathit{in} \wedge \uparrow \Theta T\, \mathit{then}\, 1\, \mathit{else}\, 0\, \mathit{fi}}(X_i^{\mathcal{C}})}.$$

Since the only reward of the form $\Theta\sigma$ is ΘT, the extended chain in Fig. 9 can be used to calculate the value of this metric as well. It is easy to check that this value equals 81/100.

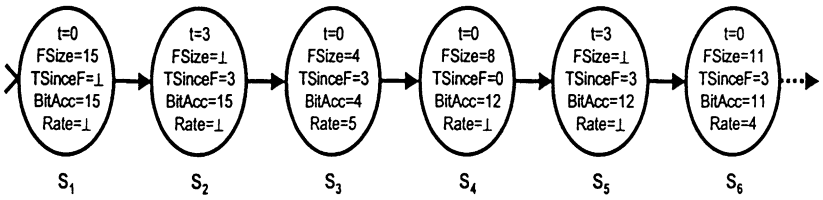

Fig. 10. Execution trace of the protocol.

4.2. Average and variance of transmission rate

The temporal reward formalism developed in this paper is very expressive and can be used to specify many complex performance metrics. As an example, we will consider the frame traffic entering a *LossyChannel* again, but now the environment of this channel is taken into account. We will specify the long-run average transmission rate as well as the long-run variance in transmission rate in bits per unit of time on channel *DLtoPL*1 of the protocol stack in Fig. 1. The long-run variance in rate is a measure for the burstiness of the traffic on the channel. Since the *DataLinkSender* and the *PhysicalLayer* (the *TransmissionChannel*) exchange frames, the size of these frames, and not the frames themselves are required. This size could be retrieved by sending the message *bitSize* to an object of data class *Frame*. A real-valued reward constant *FSize*, giving the size in bits of a frame exchanged on channel *DLtoPL*1, could[8] be defined by

$$FSize \stackrel{\text{def}}{=} DLtoPL1.frame(f).f\ bitSize.$$

In case no frame is exchanged in a state reward *FSize* is undefined; otherwise the value of *FSize* gives the number of bits of the frame. In Fig. 10, a possible execution trace of the protocol is shown. To specify the rate metrics, rewards t (the time progress) and *FSize* need to be used to define some reward *Rate* representing a bit-rate. The basic idea is to define *Rate* in those states where a frame is passed as the number of bits of a previous frame, divided by the inter-arrival time of the current frame and the previous one. For instance, in states S_1 and S_2 of Fig. 10, the value of *Rate* is undefined (in state S_1 the first frame is passed and in state S_2 no frame is passed). In state S_3, the second frame is passed and the transmission rate of the previous frame is 15 bits in 3 units of time. Therefore $Rate = 5$ in this state. In state S_4, we encounter a problem however, since in this state the third frame is passed, but the time since the second frame is 0 units of time. (Notice that this situation can actually happen since a *LossyChannel* is able to accept a new frame immediately after a previous frame is lost, see Fig. 3.) This problem can easily be solved by accumulating the bit sizes in states S_3 and S_4 and assign a rate of $12/3 = 4$ bits per time unit to state S_6.

Reward *Rate* can be defined in terms of two other rewards: *TSinceF* and *BitAcc*. *TSinceF* gives the time since the previous frame and in reward *BitAcc* the bit sizes are accumulated until a state is encountered in which a frame is exchanged and in which the time since the previous frame is positive.

Although these rewards can be defined directly, we will introduce some convenient derived temporal operators first. For rewards $\mathbf{b}$, $\mathbf{r_1}$ and $\mathbf{r_2}$ these derived operators are defined by

[8] The development of a specification language for temporal rewards that is consistent with POOSL and follows its object-oriented and modular character is not completed yet. Here we show what some language elements might look like.

- $C(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2})$ is defined as *if* $\mathbf{b}$ *then* $\mathbf{r_1}$ *else* $\mathbf{r_2}$ *fi*;
- $C^t(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2})$ is defined as $C(\uparrow \mathbf{b}, C(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2}), \mathbf{r_1})$;
- $C^f(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2})$ is defined as $C(\uparrow \mathbf{b}, C(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2}), \mathbf{r_2})$.

Here $C(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2})$ is just an abbreviation of the conditional reward *if* $\mathbf{b}$ *then* $\mathbf{r_1}$ *else* $\mathbf{r_2}$ *fi*. $C^t(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2})$ and $C^f(\mathbf{b}, \mathbf{r_1}, \mathbf{r_2})$ are also conditional rewards, but they take the values $\mathbf{r_1}$ respectively $\mathbf{r_2}$ if $\mathbf{b}$ is undefined.

With these derived operators, rewards *TSinceF*, *BitAcc* and *Rate* can be defined by

$$TSinceF \stackrel{\text{def}}{=} C^f(\Theta \uparrow FSize, t, \Theta TSinceF + t),$$

$$BitAcc \stackrel{\text{def}}{=} C(\uparrow FSize, C^t(TSinceF > 0, FSize, \Theta BitAcc + FSize), \Theta BitAcc),$$

$$Rate \stackrel{\text{def}}{=} C^f\left(TSinceF > 0, \frac{\Theta BitAcc}{TSinceF}, \bot\right).$$

Notice that $\uparrow FSize$ is satisfied precisely when a frame arrival has taken place. The values of the rewards are indicated in the execution trace of Fig. 10. If $\{X_i^{\mathcal{C}} | i \geq 1\}$ denotes the extended chain, the long-run average transmission rate is given by the limiting behaviour of random variable

$$\frac{\sum_{i=1}^{n} \overline{C(\uparrow Rate, TSinceF * Rate, 0)}(X_i^{\mathcal{C}})}{\sum_{i=1}^{n} \overline{C(\uparrow Rate, TSinceF, 0)}(X_i^{\mathcal{C}})}.$$

If this variable converges to constant r, the long-run variance is defined by the limiting behaviour of variable

$$\frac{\sum_{i=1}^{n} \overline{C(\uparrow Rate, TSinceF * (Rate - r)^2, 0)}(X_i^{\mathcal{C}})}{\sum_{i=1}^{n} \overline{C(\uparrow Rate, TSinceF, 0)}(X_i^{\mathcal{C}})}.$$

To verify the ergodicity property of the extended Markov chain, we will use the incremental construction technique of Section 3.5. We will show that the extended chain is ergodic if *FSize* is defined in some positive state of the initial Markov chain and if for some positive state $t > 0$ (i.e. the time progresses in some positive state). First we extend the initial ergodic chain with the closure of $\{\Theta \uparrow FSize\}$. Since $\uparrow FSize$ is determined by any state in the initial chain, the extended chain will be ergodic. Further, $\uparrow FSize$ holds in some positive state, $\Theta \uparrow FSize$ holds in some positive state and $t > 0$ holds in some positive state of the extended chain. As a second extension, we use the closure of $\{TSinceF\}$. Since $\Theta \uparrow FSize$ is considered to be atomic, the only reward of the form $\Theta\sigma$ is $\Theta TSinceF$. Since $\Theta \uparrow FSize$ holds in some positive state, *TSinceF* is determined by some positive state. The second extension therefore leads to an ergodic chain. For this chain, we also have that $\uparrow FSize$ holds in some positive state, $\Theta \uparrow FSize$ holds in some positive state and $t > 0$ holds in some positive state. It is further easy to verify that there must be a positive state satisfying both $\uparrow FSize$ and $TSinceF > 0$. As a third extension, we use the closure of $\{BitAcc\}$. This extension also leads to an ergodic chain since *BitAcc* is determined by a positive state satisfying both $\uparrow FSize$ and $TSinceF > 0$. Finally as a fourth extension the rewards in the numerators and denominators of the performance metrics are used. The closure of these rewards does not contain rewards of the form $\Theta\sigma$ and therefore does not differ from the chain resulting from the third extension.

As a consequence, under the (realistic) conditions mentioned above, the average and variance of transmission rate can be safely deduced from the extended Markov chain. To compute or estimate these metrics, tools are necessary to generate (a trace of) the (extended) Markov chain underlying the model of the protocol in Fig. 1. The development of these tools is subject of current activities.

5. Conclusions and future research

In this paper, we introduced a new approach for expressing and calculating complex performance metrics for discrete-time Markov chains. The technique allows a quantitative property of interest to be expressed in the form of *temporal reward* formulas. Such a formula is able to express accumulated atomic rewards over sequences of states of an initial Markov chain. From this initial Markov chain and the temporal reward formulas, an extended Markov chain is constructed that allows the quantitative property to be evaluated through standard techniques for computing long-run averages. We proved that the construction method leaves long-run averages invariant for atomic (non-temporal) rewards and we established conditions guaranteeing the preservation of ergodicity.

The emphasis in this paper is on the computation of performance figures. In practice, however, Markov chains are often too large to allow for an analytical exhaustive analysis. In those cases, statistical techniques (such as Markov chain Monte Carlo [13]) utilising central limit theorems for Markov chains are indispensable. Since our construction method builds the extended chain in an on-the-fly manner (without having to construct the complete extended chain), it can be applied for efficient simulation purposes in case the extended Markov chain is very large or even infinite. We are currently working on the topic of simulation in the context of the industrial-strength modelling language POOSL, that is successfully being applied in a number of industrial case studies. We are also developing a specification language for temporal rewards that is consistent with POOSL and follows its object-oriented and modular character. Further, the modelling and analysis tool SHESim is being extended with performance analysis and verification capabilities.

Acknowledgements

I would like to thank the people involved in the PROGRESS project ‘Modelling and performance analysis of telecommunication systems’ for all the fruitful interactions. I thank Gjalt de Jong and Alex Niemegeers for giving me the opportunity to study industrial-sized systems within Alcatel Antwerp and for the interesting discussions on system design and performance modelling. I am further grateful to Holger Hermanns for studying earlier drafts of this paper and for giving a lot of useful feedback.

References

[1] L. de Alfaro, How to specify and verify long-run average behaviour of probabilistic systems?, in: Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science, 1998, pp. 454–465.
[2] A. Aziz, K. Sanwal, V. Singhal, R. Brayton, Verifying continuous time Markov chains, in: Proceedings of the CAV’96, LNCS 1102, 1996, pp. 270–276.
[3] C. Baier, J.P. Katoen, H. Hermanns, Approximate symbolic model checking of continuous-time Markov chains, in: Proceedings of the CONCUR’99, LNCS 1664, 1996, pp. 146–162.
[4] C. Baier, B. Haverkort, H. Hermanns, J.P. Katoen, On the logical characterisation of performability properties, in: Proceedings of the ICALP’00, LNCS 1853, 2000, pp. 780–792.
[5] M. Bernado, An algebra-based method to associate rewards with EMPA terms, in: Proceedings of the ICALP’97, LNCS 1256, 1997, pp. 358–368.
[6] A. Bianco, L. de Alfaro, Model checking of probabilistic and nondeterministic systems, in: Proceedings of the Foundations of Software Technology and Theoretical Computer Science, 1995, pp. 499–513.
[7] P. Billingsley, Probability and Measure, Wiley, New York, 1995.
[8] K.L. Chung, Markov Chains with Stationary Transition Probabilities, Springer, Berlin, 1967.

[9] G. Clark, Formalising the specification of rewards with PEPA, in: Proceedings of the Fourth Annual Workshop on Process Algebra and Performance Modelling, 1996, pp. 139–160.
[10] G. Clark, Techniques for the construction and analysis of algebraic performance models, Ph.D. Thesis, University of Edinburgh, 2000.
[11] G. Clark, S. Gilmore, J. Hillston, M. Ribaudo, Exploiting modal logic to express performance measures, in: Proceedings of the 11th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools, 2000, pp. 211–227.
[12] C. Derman, Finite State Markovian Decision Processes, Academic Press, New York, 1970.
[13] D. Gamerman, Markov Chain Monte Carlo, Chapman & Hall, London, 1997.
[14] M.C.W. Geilen, Real-time concepts for software/hardware engineering. Master's Thesis, Eindhoven University of Technology, 1996.
[15] M. Geilen, J.P.M. Voeten, Real-time concepts for a formal specification language for software/hardware systems, in: Proceedings of the ProRISC/IEEE'97, 1997, pp. 185–192.
[16] M.C.W. Geilen, J.P.M. Voeten, Object-oriented modelling and specification using SHE, in: Proceedings of the First International Symposium on Visual Formal Methods VFM'99, 1999, pp. 16–24.
[17] A. Giacalone, C.-C. Jou, S.A. Smolka, Algebraic reasoning for probabilistic concurrent systems, in: Proceedings of the Working Conference on Programming Concepts and Methods, 1990, pp. 443–458.
[18] A. Goldberg, D. Robson, Smalltalk-80: The Language, Addison-Wesley, Reading, MA, 1989.
[19] H. Hansson, B. Jonsson, A calculus for communicating systems with time and probabilities, in: Proceedings of the International Workshop on Timed Petri Nets, 1985, pp. 181–190.
[20] H. Hermanns, J.P. Katoen, J. Meyer-Kayser, M. Siegle, Towards model checking stochastic process algebra, in: Proceedings of the IFM 2000, LNCS 1945, 2000, pp. 420–439.
[21] M.A. Hilliday, M.K. Vernon, A generalized timed Petri net model for performance analysis, in: Proceedings of the International Workshop on Timed Petri Nets, 1995, pp. 181–190.
[22] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[23] F. Moller, C. Tofts, A temporal calculus of communicating systems, in: Proceedings of the CONCUR'90, 1990, pp. 401–415.
[24] X. Nicollin, J. Sifakis, An overview and synthesis on timed process algebras, in: Proceedings of the CAV'91 Third International Workshop Computer Aided Verification, 1992, pp. 376–398.
[25] W.D. Obal, W.H. Sanders, State-space support for path-based reward variables, in: Proceedings of the Computer Performance and Dependability Symposium, 1998, pp. 228–237.
[26] G.D. Plotkin, A structural approach to operational semantics, Technical Report DAIMI FN-19, University of Aarhus, Denmark, 1981.
[27] P.H.A. van der Putten, J.P.M. Voeten, Specification of reactive hardware/software systems, Ph.D. Thesis, Eindhoven University of Technology, 1997.
[28] P.H.A. van der Putten, J.P.M. Voeten, M.P.J. Stevens, Object-oriented co-design for hardware/software systems, in: Proceedings of the Euromicro'95, 1995, pp. 718–726.
[29] R. Segala, Modeling and verification of randomized distributed real-time systems, Ph.D. Thesis, Massachusetts Institute of Technology, 1995.
[30] A. Tanenbaum, Computer Networks, third ed., Prentice-Hall, Englewood Cliffs, NJ, 1996.
[31] H.C. Tijms, Stochastic Models—An Algorithmic Approach, Wiley, Chichester, UK, 1998.
[32] C. Tofts, Processes with probabilities, priority and time, Form. Aspects Comput. 6 (1994) 536–564.
[33] M. Vardi, Automatic verification of probabilistic concurrent finite-state programs, in: Proceedings of the FOCS'85, 1985, pp. 327–338.
[34] J.P.M. Voeten, M.C.W. Geilen, L. van Bokhoven, P.H.A. van der Putten, M.P.J. Stevens, A probabilistic real-time calculus for performance evaluation, in: Proceedings of the 11th European Simulation Symposium, 1999, pp. 608–617.
[35] J.P.M. Voeten, Temporal rewards for performance evaluation, in: Proceedings of the PAPM 2000 (ICALP Workshops 2000), 2000, pp. 511–522.

Jeroen P.M. Voeten received his master's degree in Mathematics and Computing Science in 1991 and his Ph.D. in Electrical Engineering in 1997 from the Eindhoven University of Technology, the Netherlands. Since 1997 he is working in this institution as an Assistant Professor in the section of Information and Communication Systems. Current research interests include formal techniques for the design, specification, analysis and implementation of communicating hardware/software systems.