Extending the Logic IM-SPDL with Impulse and State Rewards

Boudewijn R. Haverkort, Matthias Kuntz

University of Twente,
Faculty for Electrical Engineering, Mathematics and Computer Science

Abstract. This paper presents the logic SDRL (Stochastic Dynamic Reward Logic), an extension of the stochastic logic IM-SPDL, which supports the specication of complex performance and dependability requirements. SDRL extends IM-SPDL with the possibility to express impulse- and state reward measures. The logic is interpreted over extended action-based Markov reward model (EMRM), i.e. transition systems containing both immediate and Markovian transitions, where additionally the states and transitions can be enriched with rewards. We define ne the syntax and semantics of the new logic and show that SDRL provides powerful means to specify path-based properties with timing and reward-based restrictions. In general, paths can be characterised by regular expressions, also called programs, where the executability of a program may depend on the validity of test formulae. For the model checking of SDRL time- and reward-bounded path formulae, a deterministic program automaton is constructed from the requirement. Afterwards the product transition system between this automaton and the EMRM is built and subsequently transformed into a continuous time Markov reward model (MRM) on which numerical analysis is performed.

1 Introduction

Distributed hard- and software systems have become part of our daily life and it becomes more and more important to assert that they are working correctly and that they meet high performance and dependability requirements (performability, cf. [19, 20]). In order to carry out performance and dependability analysis, it is necessary to have both a model and a number of measures of interest, such as utilisation, mean number of jobs, mean time to failure, etc.

In the realm of functional verification, temporal logics such as CTL [8] provide powerful means to specify complex requirements that a system has to satisfy. In the recent years big efforts have been made to provide similar means for the specification of system properties in the area of performance analysis. One result of these efforts is the logic CSL (continuous stochastic logic) [1, 4].

Very recently, the relatively new but established technique of stochastic model checking has been extended to the verification of performability properties. This extension required a new semantic model (Markov Reward models (MRM)), new logics (continuous stochastic reward logic (CSRL) and new model checking algorithms [9, 5].

In [17] we presented an extension of the logic CSL, IM-SPDL, which allows us to reason on a very abstract manner about satisfying paths of a model that is under verification. Paths can be specified via regular expressions of actions and so-called tests. IM-SPDL can be used for specifying requirements for models that contain both immediate and Markovian transitions.

Here, we will combine the logic GCSRL [11], with IM-SPDL, i.e. we extend IM-SPDL with the possibility of expressing reward-based measures.

In recent years, many efforts have been made to devise temporal logics for the specification of system properties in the area of performance analysis, where the underlying model is a labelled Markov chain. One result of these efforts is the logic CSL (continuous stochastic logic) [4], introduced by [1] and extended in [6] with an operator to reason about steady-state behaviour. CSL allows the specification of certain types of performability measures (cf. [3]), but the specification of these measures is completely state-oriented, i.e. based on atomic propositions. In [12] an action-based variant of CSL, called aCSL, was proposed, which is not based on atomic propositions but on sequences of named actions and therefore more suitable for action-oriented formalisms such as process algebras. A first combination of the state-oriented and action-oriented approach was the logic aCSL+ [21], where regular expressions of actions are used to characterise satisfying paths. In [15] we presented the first ideas of a stochastic extension of the logic PDL (SPDL) which also combines state-oriented and action-oriented features, and where paths can be specified via regular expressions of actions and so-called tests. The logic asCSL [2], inspired by the path-based reward variables of [22], follows a similar motivation. However, we emphasise the fact that the model to be checked by all logics mentioned in this paragraph is a labelled CTMC which is not allowed to contain immediate transitions. For stochastic logics *without* rewards, there are two logics, that allows us to reason about systems with both timed and untimed behaviour. In [7] an extension of the logic CSL is described that allows to specify and verify CSL properties over Markov chains with both timed and untimed transitions. In [17] an extension of the logic SPDL [14], IM-SPDL, having the same aim was described. For stochastic *reward* logics, such an extension has not been proposed so far.

We will now introduce a running example, that is used throughout this report.

Example 1 (Running example: Fault-tolerant packet collector). Throughout this paper, we use the example of a fault-tolerant packet collector. Fig. 1 shows the GSPN-style specification of this simple system which has the following repeating behaviour: n data packets arrive independently, are stored, and then all n data packets are jointly processed. Arrivals can either be error-free (upper transition ARR, rate λ) or erroneous (lower transition ARR, rate μ). Rather unusual for GSPNs, there are two timed transitions bearing the same name, ARR, which expresses the fact that these transitions are not distinguishable by an observer. If a data packet contains an error, this error can be correctable (immediate transition c) with a certain probability p, or non-correctable (immediate transition nc) with the complementary probability. In case of a correctable error, the error is corrected (transition CO) and more data packets can be received. If the er-

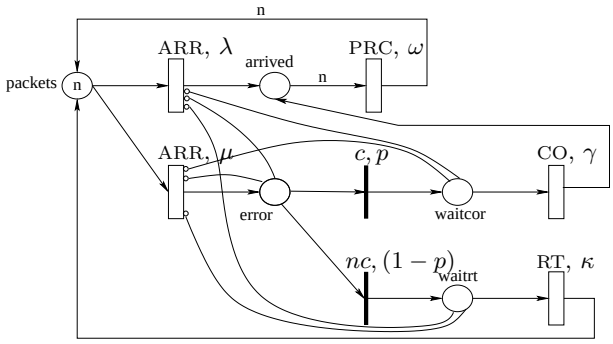

Fig. 1. GSPN-style model of a fault-tolerant packet collector

ror is non-correctable, the data packet has to be retransmitted (transition RT). During the processing of an erroneous packet, no new packet can arrive, which is modelled by the inhibitor arcs from places *error*, *waitcor*, and *waitrt* to the ARR transitions of the model.

We add to the following transitions and places have rewards greater than zero:

- c, p obtains an impulse reward of 2 and $nc, 1-p$ has an impulse reward of zero
- The place "*arrived*" is enriched with a state reward of 5 and the place "*waitcorr*" will have a state reward of 2.

□

2 The Semantic Model

The model of the logic SDRL is an extended Markov reward model (EMRM). An EMRM has two types of transitions and states, immediate and Markovian transitions and vanishing and tangible states. Immediate transitions are untimed transitions, whereas Markovian transitions are associated with an exponentially distributed delay. Tangible states possess only Markovian outgoing transitions, whereas vanishing states have at least one outgoing immediate transition. Furthermore the tangible state EMRMs can be enriched with state rewards, and both types of transitions can be annotated with impulse rewards.

Definition 1 (Extended Markov Reward Model). *An extended Markov reward model (EMRM) is a nine-tuple* $\mathcal{M} := (s, S, L, \mathsf{AP}, \mathsf{Act}, \rho_Z, \rho_J, R_I, R_M)$*, where:*

- $s \in S$ *is the unique initial state of* $\mathcal{M}$*.*
- S *is a finite set of states.*
- $L : S \mapsto 2^{\mathsf{AP}}$ *is the state labelling function that associates with every state* $s \in S$ *the set of atomic propositions which hold in that state.*
- AP *is the set of atomic propositions.*
- Act *is a finite set of actions, that is composed of two disjoint subsets:*

- Act_I *is a finite set of immediate action labels, i.e. actions, that are associated with immediate transitions.*
- Act_M *is a finite set of Markovian action labels, i.e. actions, that are associated with Markovian transitions.*

– $\rho_Z : S \mapsto \mathbb{R}_{>0}$*, is the state reward function, that associates with every state a reward rate,*

– $\rho_J : (R_I \cup R_M) \mapsto \mathbb{R}_{>0}$ *is the impulse reward function, that relates to every transition in* $\mathcal{M}$ *an impulse reward,*

– $R_I : S \times \mathsf{Act}_I \times \mathbb{P} \times S$ *is the immediate transition relation, where* $\mathbb{P} = (0, 1]$*. If* $(s, a, p, j, s') \in R_I$*, we write* $s \xrightarrow{a,p,j} s'$*.* Act_I *is a finite set of immediate action labels, i.e. actions, that are associated with immediate transitions,* $p \in \mathbb{P}$ *is a probability, and* j *is the impulse reward attached to that transition. The probabilities associated with the immediate transitions leaving a particular state must sum up to 1 (provided that the state has at least one emanating immediate transition).*

– $R_M : S \times \mathsf{Act}_M \times \mathbb{R} \times S$ *is the Markovian transition relation.* Act_M *is a finite set of Markovian action labels, i.e. actions, that are associated with Markovian transitions. We require that* $\mathsf{Act}_I \cap \mathsf{Act}_M = \emptyset$*. If* $(s, a, \lambda, j, s') \in R_M$*, we write* $s \xrightarrow{a,\lambda,j} s'$*, with* λ *the transition rate and* j *the impulse reward of that transition.*

Example 2 (EMRM of packet collector). In Fig. 2, the EMRM $\mathcal{M}$ for the packet collector GSPN from Fig. 1 is shown, where we assume that the number n of data packets that are to be processed is equal to four.

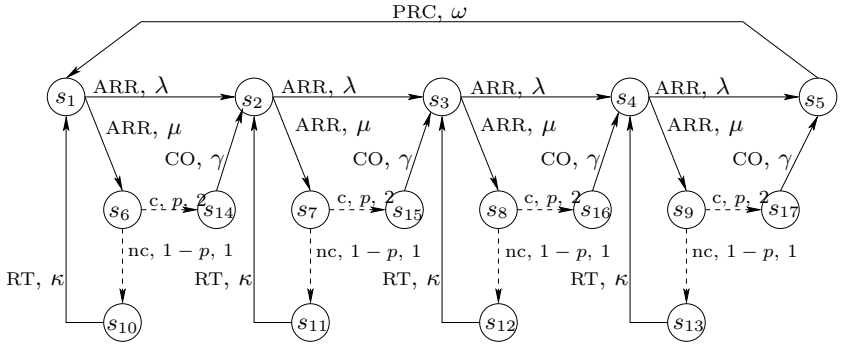

Fig. 2. EMRM of the GSPN model for $n = 4$

The system has the following state labels:

$$L(s_5) = \{\mathsf{full}\}, \quad L(s_6) = ... = L(s_9) = \{\mathsf{error}\},$$
$$L(s_{10}) = ... = L(s_{13}) = \{\mathsf{waitrt}\}, \quad L(s_{14}) = ... = L(s_{17}) = \{\mathsf{waitcor}\}$$

The sets of immediate and Markovian actions are given as follows:

$$\mathsf{Act}_I := \{nc, c\}, \quad \mathsf{Act}_M := \{ARR, RT, CO, PRC\}$$

For example, transitions $s_6 \overset{nc,1-p,1}{\dashrightarrow} s_{10}$ and $s_6 \overset{c,p,2}{\dashrightarrow} s_{14}$ are immediate, whereas $s_1 \xrightarrow{ARR,\lambda} s_2$ and $s_{14} \xrightarrow{CO,\gamma} s_2$ are Markovian transitions.

From the high-level description we know, that c, p and $nc, 1-p$ transitions are attached an impulse reward of 2 respectively 1. In the low-level model, we find the actual transitions with those rewards in Fig. 2. The state rewards of the high-level model correspond to the following low-level state rewards:

- The state reward of place "*arrived*" is attached to state s_5.
- The state reward of place "*waitcorr*" is attached to states s_{14} to s_{17}.

□

Since an EMRM may have two types of transitions, there are two types of states, vanishing and tangible states.

Definition 2. ***(Vanishing and tangible states)*** *A state of an EMRM is called vanishing (instable) if it possesses at least one outgoing immediate transition. Otherwise the state is called tangible (stable).*

A vanishing state is left as soon as it is entered, i.e. its sojourn time is zero. A tangible state has at least one outgoing Markovian transition (unless it is absorbing), therefore its sojourn time is governed by an exponential distribution whose rate parameter λ equals the sum of all the rates of the Markovian transitions emanating from that state. In the context of compositional modelling formalisms, such as stochastic process algebras, a further refinement of the notions of tangible/vanishing states is possible [23]. However, as our model checking approach is independent of any high-level modelling formalism, as long as the model to be checked is a single EMRM which is considered in isolation, it suffices to distinguish between vanishing and tangible states.

Example 3. In Fig. 2 states s_6, s_7, s_8, and s_9 are vanishing, the remaining states are tangible. □

For the semantics of the logic SDRL, the following notion of a path is of great importance:

Definition 3. ***(Paths in $\mathcal{M}$)*** *An infinite path σ of an EMRM $\mathcal{M}$ is a sequence of transitions of the form $s_0 \xrightarrow{a_0,t_0,j_0} s_1 \xrightarrow{a_1,t_1,j_1} s_2 \ldots$ where:*

- *$t_i = \tau(\sigma, i) \in \mathbb{R}_{\geq 0}$ is the real-valued sojourn time in s_i before passing to s_{i+1}.*
- *if $a_i \in \mathsf{Act}_M$, then $\exists \lambda : (s_i, a_i, \lambda, j, s_{i+1}) \in R_M$ and $t_i > 0$ is the sojourn time in state s_i (i.e. t_i is the value drawn from an exponential distribution).*
- *if $a_i \in \mathsf{Act}_I$, then $\exists p : (s_i, a_i, p, j, s_{i+1}) \in R_I$ and $t_i = 0$ is the sojourn time in state s_i.*
- *$\sigma[i]$ is the $(i+1)$st state on path σ.*
- *$\sigma@t$ is the state at time point t.*
- *$a[i]$ is the $(i+1)$st action on path σ.*

A finite path σ is a finite sequence of transitions of the form: $s_0 \xrightarrow{a_0,t_0,j_0} s_1 \xrightarrow{a_1,t_1,j_1} s_2 \dots s_{n-1} \xrightarrow{a_{n-1},t_{n-1},j_{n-1}} s_n$, where s_n is an absorbing state. For a finite path, $\tau(\sigma,i)$ is defined for $i < n$ as for infinite paths, and for $i = n$ we define $\tau(\sigma,i) = \infty$. The set $\mathsf{PATH}^{\mathcal{M}}(s) := \{\sigma \big| \sigma[0] = s\}$ is the set of all finite or infinite paths with initial state s.

3 Syntax of the Logic SDRL

The logic SDRL is a stochastic extension of PDL [10], a multi-modal program logic. Beside the standard ingredients such as propositional logic and the modal $\diamond$-operator ("possibly"), PDL enriches the $\diamond$-operator with so-called regular programs which are basically regular expressions of actions and tests (cf. Def. 5 below). If Φ and Ψ are PDL formulae and ρ is a program, then $\Phi \vee \Psi$, $\neg\Phi$ and $\langle\rho\rangle\Psi$ are formulae. $\langle\rho\rangle\Psi$ means that it is possible to execute program ρ, thereby ending up in a state that satisfies Ψ.

With respect to PDL we have added the following operators to obtain SDRL:

- A path operator that extends the original PDL $\langle . \rangle$-operator by specifying time and state and impulse reward bounds that must be met, if the formula is to be satisfied.
- a probabilistic path quantifier $\mathcal{P}_{\bowtie p}$ to reason about the transient probabilistic behaviour of a system,
- a steady-state operator $\mathcal{S}_{\bowtie p}$ to reason about the behaviour of the system once stationarity of the underlying Markov chain is reached.

3.1 Syntax of SDRL

The formulae of SDRL are formally defined as follows:

Definition 4. ***(Syntax of SDRL)*** *Let $p \in [0,1]$ be a probability and $q \in \mathsf{AP}$ an atomic proposition and $\bowtie \in \{\leq, <, \geq, >\}$ a comparison operator.*
The state formulae Φ of SPDL are defined as follows:

$$\Phi := q \big| \Phi \vee \Phi \big| \neg\Phi \big| \mathcal{S}_{\bowtie p}(\Phi) \big| \mathcal{P}_{\bowtie p}(\phi) \big| (\Phi)$$

Path formulae ϕ are defined by:

$$\phi := \Phi[\rho]^I_{J,Z}\Phi$$

where I is the closed time interval $[t,t']$ of the real axis. $J = [j,j']$ is the real impulse reward interval, with $j \in \mathbb{R}_{\geq 0}$ and $j' \in \mathbb{R}_{>0} \cup \{\infty\}$, and $Z = [y,y']$ is the real state reward interval, with $y \in \mathbb{R}_{\geq 0}$ and $y' \in \mathbb{R}_{>0} \cup \{\infty\}$. The symbol ρ represents a program as defined by Def. 5.

Definition 5. ***(Programs)*** *Let $\mathsf{Act} = \mathsf{Act}_I \cup \mathsf{Act}_M$ be a set of actions, which are also called atomic programs, and TEST be a set of SDRL state formulae. A program ρ is defined by the following grammar:*

$$\rho := \epsilon \big| \Phi?; a \big| \rho; \rho \big| \rho \cup \rho \big| \rho^* \big| \Phi?; \rho \big| (\rho)$$

where $\epsilon \notin \mathsf{Act}$ is the empty program, $a \in \mathsf{Act}$ and $\Phi \in \mathsf{TEST}$.

The operators ; (sequential composition), $\cup$ (choice), and $*$ (Kleene star) have their usual meaning. The operator $\Phi?;\rho$ (resp. $\Phi?;a$) is the so-called test operator (also called guard operator). Its informal semantics is as follows: Test whether Φ holds in the current state of the model. If this is the case, then execute program ρ, otherwise ρ is not executable. Def. 5 requires that every atomic program is preceded by a test formula Φ, but this can be the trivial test (i.e. $\Phi = \mathsf{true}$). From standard automata theory it is known that regular expressions coincide with regular languages, i.e. sets of words that are generated according to the rules of regular expressions. Programs as defined in Def. 5 can be seen as regular expressions over the alphabet $\Sigma = \mathsf{TEST} \times (\mathsf{Act} \cup \epsilon)$. Words that are generated from programs in SDRL will be referred to as *program instances*. The set of these program instances is called, as before, a language.

The length of a program instance r, denoted by $\left|r\right|$, is the number of elements from Σ occuring in it. For $0 \leq i < \left|r\right|$, $r[i]$ is the $(i+1)$st element of r. $TF(r[i])$ denotes the test formula part of $r[i]$, and $Act(r[i])$ denotes the action part of $r[i]$.

Example 4 (Programs and program instances). Let $\mathsf{Act} = \mathsf{Act}_I \cup \mathsf{Act}_M$ as in Ex. 2 be the set of atomic programs, and $\mathsf{TEST} = \{\mathsf{error}, \mathsf{full}, ..., \neg\mathsf{error}, \neg\mathsf{full}, ...\}$ the set of test formulae. Using the grammar from Def. 5, possible programs are[1]

$\rho_1 = ARR; (\neg\mathsf{error}?; ARR)^*; c; CO; ARR^*$ and
$\rho_2 = (\neg\mathsf{full}?; ARR); c; CO; (\mathsf{full}?; \epsilon)$.

Some program instances of ρ_1 are:

$q = ARR; c; CO; ARR; ARR$,
$r = ARR; (\neg\mathsf{error}?; ARR); c; CO$ and
$s = ARR; (\neg\mathsf{error}?; ARR); (\neg\mathsf{error}?; ARR); c; CO; ARR$.

For r it holds that $\left|r\right| = 4$, $Act(r[1]) = ARR$ and $TF(r[1]) = \neg\mathsf{error}$. □

4 Semantics of the Logic SDRL

At first, we will describe the semantics of SDRL in an informal style

4.1 Informal SDRL Semantics

The meaning of SDRL formulae can informally be described as follows.

1. All states that are labelled with atomic proposition q satisfy q.
2. The semantics of negation ($\neg\Phi$), and disjunction ($\Phi \vee \Psi$) is defined the usual way [18].
3. $\mathcal{S}_{\bowtie p}(\Phi)$ asserts that the steady-state probability of the set of Φ-states, i.e. the probability to reside in a Φ-state, once the system has reached stationarity, satisfies the bounds as imposed by $\bowtie p$.

[1] For better readability we often omit the trivial test formula, i.e. we write a instead of $(true?; a)$.

4. $\mathcal{P}_{\bowtie p}(\phi)$ asserts that the (transient) probability measure of the paths that satisfy ϕ is within the bounds as given by $\bowtie p$.
5. $\Phi[\rho]^I_{J,Z}\Psi$ means that a state that satisfies Ψ is reached within at least t but at most t' time units, the state resp. impulse rewards gained on σ must lie within J resp. Z, and that all preceding states must satisfy Φ. Additionally, the action sequence of the path to the Ψ state must correspond to the action sequence of a word from the language $\mathcal{L}_\rho$ (the language induced by program ρ) and all test formulae that are part of program ρ must be satisfied by the corresponding states on the path.

4.2 Formal Semantics of GCSRL

In the sequel, we will define the formal semantics of both SDRL state and path formulae in a very detailed way.

Definition 6. ***(State probabilities)*** *The probability to be in state s' at time point t, provided that the system is in state s at time 0, is given by*

$$\pi^{\mathcal{M}}(s, s', t) = Pr(\sigma \in \mathsf{PATH}^{\mathcal{M}}(s) \big| \sigma@t = s')$$

The definition for steady-state probabilities is similar, taking into account that steady-state means 'on the long run':

$$\pi^{\mathcal{M}}(s, s') = lim_{t\to\infty}\pi^{\mathcal{M}}(s, s', t)$$

These definitions can be extended to sets of states: For $S' \subseteq S$:

$$\pi^{\mathcal{M}}(s, S', t) := \sum_{s'\in S'} \pi^{\mathcal{M}}(s, s', t) \quad \textit{and} \quad \pi^{\mathcal{M}}(s, S') := \sum_{s'\in S'} \pi^{\mathcal{M}}(s, s').$$

As usual, we define the semantics over the binary satisfiability relation $\models$ [18].

Definition 7 (Semantics of SDRL State Formulae). *Let $q \in \mathsf{AP}$ be an atomic proposition, $p \in [0, 1]$ be a probability and $\bowtie \in \{<, \leq, >, \geq\}$ be a comparison operator, then the semantics of state formulae is defined as follows:*

$$\begin{aligned}
\mathcal{M}, s \models q &\iff q \in L(s)\\
\mathcal{M}, s \models \neg\Phi &\iff \mathcal{M}, s \not\models \Phi\\
\mathcal{M}, s \models (\Phi \vee \Psi) &\iff \mathcal{M}, s \models \Phi \textit{ or } \mathcal{M}, s \models \Psi\\
\mathcal{M}, s \models \mathcal{S}_{\bowtie p}(\Phi) &\iff \pi^{\mathcal{M}}(s, Sat(\Phi)) \bowtie p\\
\mathcal{M}, s \models \mathcal{P}_{\bowtie p}(\phi) &\iff Prob^{\mathcal{M}}(s, \phi) \bowtie p
\end{aligned}$$

$Sat(\Phi)$ is the set of states that satisfy Φ, and $Prob^{\mathcal{M}}(s, \phi)$ is the probability measure of all paths $\sigma \in \mathsf{PATH}(s)$ that satisfy ϕ:

$$Prob^{\mathcal{M}}(s, \phi) := Pr(\sigma \in \mathsf{PATH}^{\mathcal{M}}(s) \big| \mathcal{M}, \sigma \models \phi)$$

For the semantics of path formulae we have to relate the instances of the program ρ to words on paths in the EMRM $\mathcal{M}$.

Definition 8. ***(Words on paths)*** *The word $\mathcal{W}^k$ of length $k \geq 0$ on a path $\sigma \in \mathsf{PATH}^{\mathcal{M}}$ is defined as follows:*

$$\begin{aligned} & \mathcal{W}^0(\sigma) := \epsilon \\ & \mathcal{W}^k(\sigma) := \mathcal{W}^{k-1}(\sigma) \circ a[k-1] \\ \textit{where} \quad & a[k-1] \in \mathsf{Act}_M \wedge \sigma[k-1] \xrightarrow{a[k-1], t_{k-1}} \sigma[k] \quad \textit{or} \\ & a[k-1] \in \mathsf{Act}_I \wedge \sigma[k-1] \xrightarrow{a[k-1], 0} \sigma[k]. \end{aligned}$$

For $i = 0, 1, \ldots, k-1$, $\mathcal{W}^k(\sigma)[i]$ denotes the $i+1$st action on path σ.

Example 5. Consider a path $\sigma := s_1 \xrightarrow{ARR, t_1} s_6 \xrightarrow{c,0} s_{14} \xrightarrow{CO, t_2} s_2 \xrightarrow{ARR, t_3} \ldots$ of the EMRM from Fig. 2. The word of length 2 induced by σ is (ARR, c), the word of length 4 is (ARR, c, CO, ARR) and $\mathcal{W}^4(\sigma)[2] = CO$. □

Definition 9 (Semantics of SDRL Path Formulae). *Let σ be a path in an EMRM $\mathcal{M}$, let I, J, and Z be intervals as defined in definition 1. The semantics of path formulae is defined as follows:*

$$\begin{aligned} \mathcal{M}, \sigma \models \Phi[\rho]^{[t,t']}\Psi \iff & \exists k \big(\mathcal{M}, \sigma[k] \models \Psi \wedge \forall 0 \leq i < k (\mathcal{M}, \sigma[i] \models \Phi) \\ & \wedge \quad \textit{time\_reward\_restriction} \\ & \wedge \quad \textit{program\_matching} \big) \end{aligned}$$

The formula time_reward_restriction is defined as follows:

$$\begin{aligned} & \textit{time\_reward\_restriction} := \\ & (t \leq \sum_{l=0}^{k-1} t_l \leq t' \wedge \mathcal{JR}_k \in J \wedge \mathcal{SR}_k \in Z) \vee \\ & (\sum_{l=0}^{k-1} t_l < t \wedge ((\sum_{l=0}^{k-1} t_l + \delta) \geq t \wedge \mathcal{JR}_k \in J \wedge (\mathcal{SR}_k + \delta \cdot \rho_Z(\sigma[k])) \in Z) \wedge \\ & (\mathcal{M}, \sigma[k] \models \Phi) \wedge ((\delta = t - \sum_{l=0}^{k-1} t_l) \wedge \delta \leq \tau(\sigma, k))) \\ & \textit{where } \delta := t - \sum_{l=0}^{k-1} . \end{aligned}$$

The formula program_matching is defined as follows:

$$\begin{aligned} & \textit{program\_matching} := \\ & \big(\exists r \in \mathcal{L}(\rho) \wedge \big|r\big| = k \wedge Act(r[k-1]) \neq \epsilon \wedge \\ & \forall 0 \leq i \leq k-1 (Act(r[i]) = \mathcal{W}^{(k)}(\sigma)[i] \wedge \mathcal{M}, \sigma[i] \models TF(r[i]))\big) \vee \\ & \big(\exists r \in \mathcal{L}(\rho) \wedge \big|r\big| = k+1 \wedge Act(r[k]) = \epsilon \wedge \sigma[k] \models TF(r[k]) \wedge \\ & \forall 0 \leq i \leq k-1 (Act(r[i]) = \mathcal{W}^{(k)}(\sigma)[i] \wedge \mathcal{M}, \sigma[i] \models TF(r[i]))\big) \end{aligned}$$

This formula expresses that the word induced on path σ must be matched by the corresponding action parts of a program instance r and that the tests appearing in the program must be satisfied by the appropriate states on the path. There are two possibilities, as indicated in the formula: (1) If the last element of r is of the form $\Phi?;a$, where $a \neq \epsilon$, the corresponding state must satisfy the test formula and the last transition on the path must have a label identical to the action part of $r[k-1]$. (2) If the last element of r is of the form $\Phi?;\epsilon$, i.e. has an empty action part, then it only has to be checked whether the corresponding state on the path satisfies the test formula.

Example 6 (SDRL formulae). With respect to the EMRM $\mathcal{M}$ of Fig. 2 we specify four example requirements:

- Is the probability to receive four data packets with at most one packet containing a non-correctable error within 5 time units greater than 0.9? If, additionally a state reward between 15 and 25 was accrued?

$$\Phi_1 := \mathcal{P}_{>0.9}(\neg \mathsf{full}\ [ARR^*; nc; RT; ARR^* \cup ARR^*]^{[0,5]}_{[0,0],[15,25]}\ \mathsf{full})$$

- Is the probability to reach a state in which the buffer is full with a single arrival greater than zero?

$$\Phi_2 := \mathcal{P}_{>0}(\neg \mathsf{full}\ [ARR]^{[0,\infty]}_{[0,0],\infty}\ \mathsf{full})$$

  Requirement Φ_2 characterises state s_4.
- Is the probability that the buffer is full after at most 7.3 time units greater than 75 percent, if the following side conditions must be met:
  - The only packet that contains an error is the fourth packet. This error must be correctable.
  - An impulse reward of exactly 2 is gained and at most 45 state rewards are accrued.

$$\Phi_3 := \mathcal{P}_{>0.75}(\mathsf{true}\ [ARR^*; (\Phi_2?; ARR); c; CO]^{[0,7.3]}_{[2,2],[0,45]}\ \mathsf{true})$$

- In steady-state, is the probability that the system is currently processing either a correctable or a non-correctable error, less than 3%?

$$\Phi_4 := \mathcal{S}_{<0.03}(\mathsf{waitcor} \vee \mathsf{waitrt})$$

□

5 Model Checking SDRL

In this section, we describe the model checking algorithm for the logic SDRL. Central for this are the notions of program automata and product transition systems which we introduce in the sequel. Due to restricted space we will only describe the general idea of how to model check SDRL path formulae, full details can be found in [16].

5.1 General Idea

The overall model checking algorithm for SDRL is similar to that of CTL in the sense that we start by checking elementary subformulae and then proceed to the checking of more and more complex subformulae until the overall formula has been checked. Model checking propositional logic subformulae works as for CTL. Steady-state subformulae are checked in three steps as follows:

1. The EMRM $\mathcal{M}$ is transformed into a state-labelled CTMC $\mathcal{M}'$, by eliminating the vanishing states, as described, by the algorithm in section 5.2.
2. On $\mathcal{M}'$, model checking the steady-state operator works as for CSL [4]. Step 2 yields the verification results for the tangible states only.
3. During step 1, for each vanishing state the probability to reach a certain tangible state as the next tangible state is recorded. These probabilities are now combined with the results of step 2 in order to obtain the verification results for the vanishing states.

The basic model checking procedure for SDRL path formulae with leading $\mathcal{P}_{\bowtie p}$ operator is more involved: We assume that we want to check whether state s of a given EMRM $\mathcal{M}$ satisfies the formula $\mathcal{P}_{\bowtie p}(\phi)$, where $\phi = \Phi[\rho]^{[t,t']}\Psi$. The basic idea is to reduce the SDRL model checking problem $\mathcal{M}, s \models \mathcal{P}_{\bowtie p}(\phi)$ to the CSRL model checking problem of deciding whether $\mathcal{M}^*, s^* \models \mathcal{P}_{\bowtie p}(\mathsf{F}^I_{J,Z}\mathsf{succ})$ for a CTMC $\mathcal{M}^*$ (to be constructed) and a state s^* of $\mathcal{M}^*$. A path satisfies the CSRL path formula $\mathsf{F}^I_{J,Z}\mathsf{succ}$, if within the time interval I a state is reached that satisfies the new atomic proposition succ, and where the accrued rewards lie within the bounds as imposed by J and Z respectively. We take the following steps:

1. From the program ρ we derive a deterministic program automaton $\mathcal{A}_\rho$, which is a variant of deterministic finite automata.
2. Using the given EMRM $\mathcal{M}$ and the program automaton $\mathcal{A}_\rho$, we construct a product EMRM (PEMRM) $\mathcal{M}^\times$. The state space of $\mathcal{M}^\times$ is the product between $\mathcal{M}$ and $\mathcal{A}_\rho$, i.e. states are of the form (s_i, z_i), where s_i is a state of $\mathcal{M}$ and z_i a state of $\mathcal{A}_\rho$. In addition, $\mathcal{M}^\times$ contains an absorbing error state with the new state label fail. The transitions in $\mathcal{M}^\times$ are labelled with rates in case of Markovian transitions and with probabilities in case of immediate transitions. The purpose of building this PEMRM is to check whether $\phi = \Phi[\rho]^{[t,t']}\Psi$ is functionally satisfiable in $\mathcal{M}$ or not.
3. In order to compute the probability measure of the paths satisfying ϕ we proceed as follows:
   (a) All states (s_i, z_i) of $\mathcal{M}^\times$ for which s_i is a Ψ-state and z_i is an accepting state are replaced by a single absorbing goal state, with the special state label succ (for "success"). All transitions leading to a state (s_j, z_j) of the kind just described are redirected to this succ-state.
   (b) The PEMRM $\mathcal{M}^\times$ is transformed into a CTMC $\mathcal{M}^*$ by eliminating vanishing states as in section 5.2.
4. On $\mathcal{M}^*$ we can compute the probability measure of all paths satisfying the CSRL formula $\mathcal{P}_{\bowtie p}(\mathsf{F}^I_{J,Z}\mathsf{succ})$, which is equivalent to the probability measure of the paths satisfying the original formula $\mathcal{P}_{\bowtie p}(\phi)$ in the original model $\mathcal{M}$.

5.2 Elimination of Vanishing States

Following [23], the procedure of transforming an EMRM $\mathcal{M}$ into an MRM $\mathcal{M}'$ can roughly be described as follows.

1. Make $\neg\Phi$ and Ψ states in $\mathcal{M}$ absorbing: $\mathcal{M}[\neg\Phi \vee \Psi]$ [4].
2. Compute $\mathcal{M}'$ from $\mathcal{M}[\neg\Phi \vee \Psi]$ [23]:
   (a) While S_{Van} is not empty
      i. choose a state s_v from S_{Van}
      ii. incoming transitions to s_v have to be redirected to its successor:
      - $s \xrightarrow{a,\lambda,j_1} s_v \wedge s_v \dashrightarrow s' \Rightarrow s \xrightarrow{a,p\cdot\lambda,j_1+j_2} s'$
      - $s \dashrightarrow s_v \wedge s_v \dashrightarrow s' \Rightarrow s \dashrightarrow s'$ (labels: a,p_1,j_1 and b,p_2,j_2; $a,p_1\cdot p_2,j_1+j_2$)

Generally, the algorithm computes the transitive hull over the untimed transitions.

5.3 Program Automata

According to Sec. 5.1 we have to derive an automaton from a given program ρ. This is done by the following steps:

- At first, we construct from ρ a non-deterministic program automaton (NPA) $\#_\rho$. The definition of NPA is identical to that of non-deterministic finite automata as known from standard automata theory, albeit with special input alphabet Σ as introduced above in Sec. 3.1.
- Secondly, we turn $\#_\rho$ into a deterministic program automaton (DPA) $\mathcal{A}_\rho$. DPAs are formally defined in Def. 10. From this definition, it follows that the determinisation of an NPA is quite different from making a non-deterministic finite automaton deterministic. We will exemplify and justify our approach in example 7.

Definition 10 (Deterministic program automaton DPA). *A DPA $\mathcal{A}$ is a quintuple $(Z_\mathcal{A}, \Sigma_\mathcal{A}, z^{Start}, E_\mathcal{A}, \delta_\mathcal{A})$ where*

- *$Z_\mathcal{A}$ is a finite set of states,*
- *$\Sigma_\mathcal{A} = \mathsf{TEST} \times (\mathsf{Act} \cup \epsilon)$ is the input alphabet,*
- *$z_\mathcal{A}^{Start} \in Z_\mathcal{A}$ is the initial state,*
- *$E_\mathcal{A} \subseteq Z_\mathcal{A}$ is the set of accepting states and*
- *$\delta_\mathcal{A} : Z_\mathcal{A} \times \Sigma_\mathcal{A} \to Z_\mathcal{A}$ is the state transition function which has to satisfy the following condition: If a state z possesses more than one outgoing transition then, either the action parts of the labellings of all outgoing transitions must be pairwise different, or if there are two or more transitions whose action parts are identical, then the test formula parts of them must not be true at the same time.*

Our model checking approach relies on the following theorem:

Theorem 1. *For every NPA, an equivalent DPA can be constructed.*

Although Theorem 1 seems quite obvious, it should be noted that its proof is not the same as the equivalence proof of deterministic and non-deterministic finite automata from standard automata theory, since the input symbols have both a test part and an action part, and during determinisation the semantics of the test part must be taken into account. Instead of a formal proof of Theorem 1, we consider the following illustrative example:

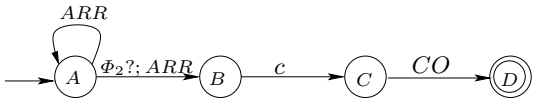

Fig. 3. Non-deterministic program automaton $\#_\rho$ for the program of Φ_3

Example 7 (NPA and DPA). Fig. 3 shows a non-deterministic program automaton $\#_\rho$ for the program $\rho = ARR^*; (\Phi_2?; ARR); c; CO$ (taken from Ex. 6, requirement Φ_3). The automaton is non-deterministic since the arcs emanating from state A, labelled with ARR (which is equivalent to $(true?; ARR)$) and $(\Phi_2?; ARR)$, have identical action label and the test parts are not disjoint. We cannot directly use such a non-deterministic automaton for our model checking algorithm, as the product construction explained in Sec. 5.4 could modify the stochastic behaviour of $\mathcal{M}$ and thus lead to wrong numerical results. Therefore we first construct a deterministic program automaton $\mathcal{A}_\rho$, which is shown in Fig 4. In $\mathcal{A}_\rho$, no two transitions are activated at the same time. This determinisation guarantees that the product automaton will preserve the branching structure and therefore the stochastic behaviour of $\mathcal{M}$. □

5.4 Product EMRM Construction and Analysis

The central part of model checking probabilistic path formulae is the construction of the PEMRM of the model $\mathcal{M}$ and the DPA $\mathcal{A}_\rho$ for the program ρ of the path formula that is to be checked.

Definition 11. ***(Product Extended Markov Reward Model (PEMRM))*** *Let the EMRM $\mathcal{M} = (s, S, \mathsf{AP}, \mathsf{Act}, L, \rho_Z, \rho_J, R_I, R_M)$ and the DPA $A_\rho = (Z_\rho, \Sigma_\rho, z_\rho^{start}, E_\rho, \delta_\rho)$ given. A PEMRM $\mathcal{M}^\times = (s^\times, \mathsf{AP}, L^\times, S^\times, \rho_Z^\times, \rho_J^\times, R_I^\times, R_M^\times)$ is defined as follows:*

- *state space:* $S^\times := \{(s_i, z_\rho^j) \big| s_i \in S \wedge z_\rho^j \in Z_\rho\} \cup \{\mathsf{FAIL}, \mathsf{SUCC}\}$
- *initial state:* $s_{Start}^\times := (s_i, z_\rho^{Start})$
- *accepting states:* $S_{Acc}^\times := \{(s_i, z_\rho^j) \in S^\times \big| s_i \in Sat(\Psi) \wedge z_\rho^j \in E_\rho\}$
- *labelling:*
  1. $\forall (s_i, z_\rho^j) \in S^\times \backslash S_{Acc}^\times (L^\times(s_i, z_\rho^j) = L(s_i))$
  2. $\forall (s_i, z_\rho^j) \in S_{Acc}^\times (L^\times(s_i, z_\rho^j) = \{\mathsf{succ}\})$
  3. $L^\times(\mathsf{FAIL}) = \{\mathsf{FAIL}\}$
  4. $L^\times(\mathsf{SUCC}) = \{\mathsf{succ}\}$
- $\rho_Z^\times, = \rho_Z$ *and* $\rho_J^\times = \rho_J$*, i.e.* $\rho_Z^\times((s, z)) = \rho_Z(s)$ *and* $\rho_J^\times(((s, z), a, \lambda, (s', z'))) = \rho_J((s, a, \lambda, s'))$

– *Immediate transition relation:* $R_I^\times \subseteq S^\times \times p \times S^\times$ *where* p *is a probability.*
– *Markovian transition relation:* $R_M^\times \subseteq S^\times \times \mathbb{R}_{>0} \times S^\times$

The relation $R_I^\times$ *is defined in definition 13. The relation* $R_M^\times$ *is as defined in definition 12.*

succ is a state formula that characterises exactly those states whose automaton part is an accepting state and whose Markov chain part is a state in which the formula Ψ *of the path formula* $\Phi[\rho]^{[t,t']}\Psi$ *holds.*

Definition 12. *For* $A, B \in 2^{S^\times \times \mathbb{R}^+ \times S^\times}$ *with* $B = \emptyset$ *or* $|B| = 1$, $A \uplus B$ *is defined as follows:*

– $B = \emptyset$*:* $A \uplus B = A$
– $|B| = 1 \wedge B = \{(s, \lambda, s')\}$*:*

$$A \uplus B = \begin{cases} A \cup B & iff\ \nexists \gamma \in \mathbb{R}_{>0}((s,\gamma,s') \in A) \\ (A \backslash \{(s,\gamma,s')\} \cup \{(s, \gamma+\lambda, s')\} & otherwise \end{cases}$$

$R_M^\times$ *is successively defined as follows:*

1. *In the PEMRM no accepting state has been reached. The original state* s *in* $\mathcal{M}$ *satisfies* Φ*.* $\mathcal{M}$ *offers transitions with labelling* a*, so does* A_ρ*.*

$$R_M^\times \uplus \{(s, z_\rho), \lambda, j, (s', z'_\rho)\big| \begin{array}{l} s \xrightarrow{a,\lambda,j} s' \wedge z_\rho \xrightarrow{a} z'_\rho \wedge \\ s \in Sat(\Phi) \wedge \\ (s, z_\rho) \notin S_{Acc}^\times\} \end{array}$$

2. *In the PEMRM no accepting state has been reached. The original state* s *in* $\mathcal{M}$ *satisfies* Φ*.* A_ρ *offers a test transition with test* $\Xi?$*; and atomic program* a*.* $\mathcal{M}$ *offers transitions with labelling* a *and satisfies the test formula of the corresponding transition in the* DPA*.*

$$R_M^\times \uplus \{(s, z_\rho), \lambda, j, (s', z'_\rho)\big| \begin{array}{l} s \xrightarrow{a,\lambda,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \wedge \\ s \in Sat(\Phi) \wedge \\ (s, z_\rho) \notin S_{Acc}^\times \wedge \\ s \in Sat(\Xi)\} \end{array}$$

3. *In the PEMRM no accepting state has been reached. The original state* s *in* $\mathcal{M}$ *satisfies* Φ*.* A_ρ *offers in* z *a transition with labellings from* Act_ρ*.* $\mathcal{M}$ *satisfies in* s *the test formula of the corresponding* z*-transition. The target state of* $\mathcal{M}$ *satisfies* Ψ *and the target state of the* z*-transition is an accepting state.*

$$R_M^\times \uplus \{(s, z_\rho), \lambda, j, SUCC\big| \begin{array}{l} s \xrightarrow{a,\lambda,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \wedge \\ s \in Sat(\Phi) \wedge \\ s \in Sat(\Xi) \wedge \\ (s, z_\rho) \notin S_{Acc}^\times \wedge \\ (s', z'_\rho) \in S_{Acc}^\times\} \end{array}$$

4. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. A_ρ offers in z a transition with labellings from Act_ρ the target state of this transition offers a transition with a labelling from TEST, say Θ. $\mathcal{M}$ satisfies in s the test formula of the corresponding z-transition. The target state of $\mathcal{M}$ satisfies Ψ and Θ.*

$$R_M^\times \uplus \{(s, z_\rho), \lambda, j, SUCC \mid \begin{array}{l} s \xrightarrow{a,\lambda,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \xrightarrow{\Theta?} z''_\rho \wedge \\ s \in Sat(\Phi) \wedge \\ s \in Sat(\Xi) \wedge \\ (s, z_\rho) \notin S_{Acc}^\times \wedge \\ s' \in Sat(\Theta) \wedge \\ (s', z''_\rho) \in S_{Acc}^\times \} \end{array}$$

5. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. $\mathcal{M}$ offers in s a transition labelled with a. A_ρ does not offer a transition bearing such a labelling.*

$$R_M^\times \uplus \{(s, z_\rho), \lambda, j, \mathit{FAIL} \mid \begin{array}{l} s \xrightarrow{a,\lambda,j} s' \wedge \\ s \in Sat(\Phi) \wedge \\ (s, z_\rho) \notin S_{Acc}^\times \wedge \\ (z_\rho \xrightarrow{a} z'_\rho) \notin \delta_\rho \} \end{array}$$

6. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ does not satisfy Φ. The source state of $\mathcal{M}^\times$ is not accepting.*

$$R_M^\times \uplus \{(s, z_\rho), \lambda, j, \mathit{FAIL} \mid \begin{array}{l} s \notin Sat(\Phi) \wedge \\ (s, z_\rho) \notin S_{Acc}^\times \} \end{array}$$

7. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. Both $\mathcal{M}$ and A_ρ offer in s resp. z a transition labelled with a, where in A_ρ a is preceeded by a test. $\mathcal{M}$ does not satisfy the test formula in its current state s.*

$$R_M^\times \uplus \{(s, z_\rho), \lambda, j, \mathit{FAIL} \mid \begin{array}{l} s \xrightarrow{a,\lambda,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \wedge \\ s \in Sat(\Phi) \wedge \\ (s, z_\rho) \notin S_{Acc}^\times \wedge \\ s \notin Sat(\Xi) \} \end{array}$$

8. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. Both $\mathcal{M}$ and A_ρ offer in s resp. z a transition labelled with a, where in A_ρ a is preceeded by a test. The target state z'_ρ is not accepting and offers a transiton with labelling from TEST to an accepting state z''_ρ. The*

target state s' of $\mathcal{M}$ does not satisfy Θ.

$$R_M^{\times} \uplus \{(s, z_\rho), \lambda, j, \mathit{FAIL}\big| \ (s \xrightarrow{a,\lambda,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \xrightarrow{\Theta?} z''_\rho) \wedge (s, z_\rho) \notin S_{Acc}^{\times} \wedge s' \notin Sat(\Theta) \wedge z''_\rho \in E_\rho\}$$

Definition 13. *$R_I^{\times}$ is successively defined as follows:*

1. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. $\mathcal{M}$ offers transitions with labelling a, so does A_ρ.*

$$R_I^{\times} \uplus \{(s, z_\rho), p, j, (s', z'_\rho)\big| \ s \overset{a,p,j}{\dashrightarrow} s' \wedge z_\rho \xrightarrow{a} z'_\rho \wedge s \in Sat(\Phi) \wedge (s, z_\rho) \notin S_{Acc}^{\times}\}$$

2. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. A_ρ offers a test transition with test $\Xi?$; and atomic program a. $\mathcal{M}$ offers transitions with labelling a and satisfies the test formula of the corresponding transition in the DPA.*

$$R_I^{\times} \uplus \{(s, z_\rho), p, j, (s', z'_\rho)\big| \ s \overset{a,p,j}{\dashrightarrow} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \wedge s \in Sat(\Phi) \wedge (s, z_\rho) \notin S_{Acc}^{\times} \wedge s \in Sat(\Xi)\}$$

3. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. A_ρ offers in z a transition with labellings from Act_ρ the target state of this transition offers a transition with a labelling from TEST, say Θ. $\mathcal{M}$ satisfies in s the test formula of the corresponding z-transition. The target state of $\mathcal{M}$ satisfies Ψ and Θ.*

$$R_I^{\times} \uplus \{(s, z_\rho), p, j, \mathit{SUCC}\big| \ s \overset{a,p,j}{\dashrightarrow} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \xrightarrow{\Theta?} z''_\rho \wedge s \in Sat(\Phi) \wedge s \in Sat(\Xi) \wedge (s, z_\rho) \notin S_{Acc}^{\times} \wedge s' \in Sat(\Theta) \wedge (s', z''_\rho) \in S_{Acc}^{\times}\}$$

4. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. A_ρ offers in z a transition with labellings from Act_ρ to z'. $\mathcal{M}$ satisfies in s the test formula of the corresponding z-transition. The target*

state of $\mathcal{M}$ satisfies Ψ and z' of A_ρ is an accepting state.

$$R_I^\times \uplus \{(s, z_\rho), p, j, SUCC \mid\ s \xrightarrow{a,p,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \wedge s \in Sat(\Phi) \wedge s \in Sat(\Xi) \wedge (s, z_\rho) \notin S_{Acc}^\times \wedge (s', z'_\rho) \in S_{Acc}^\times\}$$

5. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. $\mathcal{M}$ offers in s a transition labelled with a. A_ρ does not offer a transition bearing such a labelling.*

$$R_I^\times \uplus \{(s, z_\rho), p, j, \mathsf{FAIL} \mid\ s \xrightarrow{a,p,j} s' \wedge s \in Sat(\Phi) \wedge (s, z_\rho) \notin S_{Acc}^\times \wedge (z_\rho \xrightarrow{a} z'_\rho) \notin \delta_\rho\}$$

6. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ does not satisfy Φ. The source state of $\mathcal{M}^\times$ is not accepting.*

$$R_I^\times \uplus \{(s, z_\rho), p, j, \mathsf{FAIL} \mid\ s \notin Sat(\Phi) \wedge (s, z_\rho) \notin S_{Acc}^\times\}$$

7. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. Both $\mathcal{M}$ and A_ρ offer in s resp. z a transition labelled with a, where in A_ρ a is preceeded by a test. $\mathcal{M}$ does not satisfy the test formula in its current state s.*

$$R_I^\times \uplus \{(s, z_\rho), p, j, \mathsf{FAIL} \mid\ s \xrightarrow{a,p,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \wedge s \in Sat(\Phi) \wedge (s, z_\rho) \notin S_{Acc}^\times \wedge s \notin Sat(\Xi)\}$$

8. *In the PEMRM no accepting state has been reached. The original state s in $\mathcal{M}$ satisfies Φ. Both $\mathcal{M}$ and A_ρ offer in s resp. z a transition labelled with a, where in A_ρ a is preceeded by a test. The target state z'_ρ is not accepting and offers a transiton with labelling from TEST to an accepting state z''_ρ. The target state s' of $\mathcal{M}$ does not satisfy Θ.*

$$R_I^\times \uplus \{(s, z_\rho), p, j, \mathsf{FAIL} \mid\ (s \xrightarrow{a,p,j} s' \wedge z_\rho \xrightarrow{\Xi?;a} z'_\rho \xrightarrow{\Theta?} z''_\rho) \wedge (s, z_\rho) \notin S_{Acc}^\times \wedge s' \notin Sat(\Theta) \wedge z''_\rho \in E_\rho\}$$

Example 8 (Constructing the PEMRM). Let the EMRM $\mathcal{M}$ from Fig. 2 and the DPA $\mathcal{A}_\rho$, shown in Fig. 4, be given. We now explain by example, how their PEMRM $\mathcal{M}^\times$, shown in Fig. 5, is constructed:

- The combinations of the transitions $s_1 \xrightarrow{ARR,\lambda} s_2$ in $\mathcal{M}$ and $A \xrightarrow{\neg\Phi_2?;ARR} A$ in $\mathcal{A}_\rho$ leads to the transition $(s_1, A) \xrightarrow{\lambda} (s_2, A)$ in $\mathcal{M}^\times$.
- In $\mathcal{M}$, transition $s_1 \xrightarrow{ARR,\mu} s_6$ is also possible, therefore $\mathcal{M}^\times$ also has the transition $(s_1, A) \xrightarrow{\mu} (s_6, A)$.
- Transition $(s_9, AB) \overset{p,2}{\dashrightarrow} (s_{17}, C)$ in $\mathcal{M}^\times$ stems from the transition $s_9 \overset{c,p,2}{\dashrightarrow} s_{17}$ in $\mathcal{M}$ and $AB \xrightarrow{c} C$ in $\mathcal{A}_\rho$.
- Transition $(s_6, A) \overset{1,3}{\dashrightarrow} \mathsf{fail}$ is composed of the transitions $s_6 \overset{c,p,2}{\dashrightarrow} s_{14}$ and $s_6 \overset{nc,1-p,1}{\dashrightarrow} s_{10}$ in $\mathcal{M}$, since in state A neither a c nor an nc transition is possible. Therefore in $\mathcal{M}^\times$ we obtain the transitions $(s_6, A) \overset{p,2}{\dashrightarrow} \mathsf{fail}$ and $(s_6, A) \overset{1-p,1}{\dashrightarrow} \mathsf{fail}$ which can be replaced by a single immediate transition that has probability one.
- In state C there is a transition $C \xrightarrow{CO} D$, where D is an accepting state, and in $\mathcal{M}$ there is a transition $s_{17} \xrightarrow{CO,\gamma} s_5$. In $\mathcal{M}^\times$ this leads to transition $(s_{17}, C) \xrightarrow{\gamma} \mathsf{succ}$, which stems from the fact that the automaton goal state D is accepting and that the goal state s_5 of the EMRM satisfies $\Psi = \mathsf{true}$, i.e. state (s_5, D) satisfies the conditions of Sec. 5.1, item 3(a). □

Example 9 (Elimination of vanishing states). Let the PESLTS $\mathcal{M}^\times$ from Fig. 5 be given. The vanishing states (s_6, A), (s_7, A), (s_8, A) and (s_9, AB) are eliminated, thereby redirecting their incoming arcs to the respective successor states[2] and weighing them with the corresponding probabilities. This leads to the labelled CTMC $\mathcal{M}^*$ shown in Fig. 6. □

6 A Bisimulation for SDRL

In this section we define a bisimulation relation for EMRMs and prove that the satisfiability of SDRL formulae is preserved under this notion of bisimulation. Having such a result model checking SDRL can be performed on EMRMs which are minimal with respect to bisimulation.

[2] In general, sequences and even cycles of immediate transitions are possible, which situation can be handled by several published elimination algorithms.

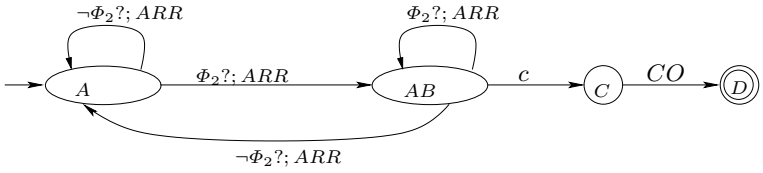

Fig. 4. Deterministic program automaton $\mathcal{A}_\rho$ for the program of Φ_3

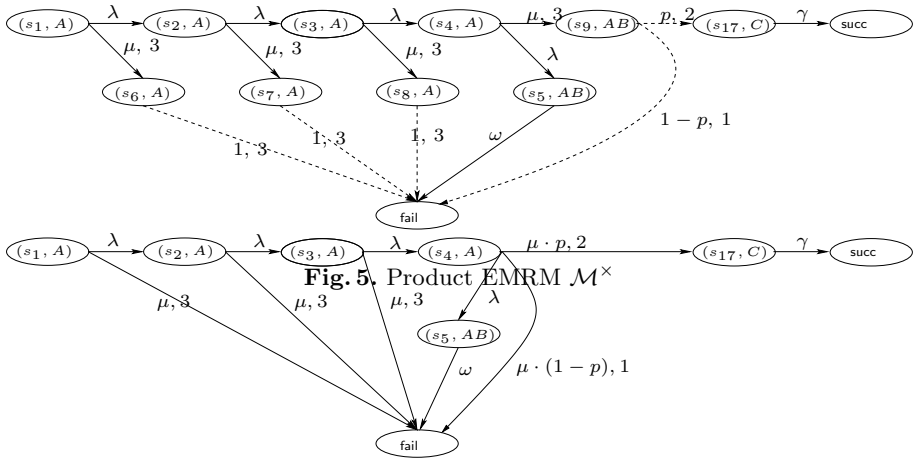

Fig. 5. Product EMRM $\mathcal{M}^{\times}$

Fig. 6. $\mathcal{M}^*$: Result of the elimination of vanishing states

6.1 A Bisimulation Relation for EMRMs

Definition 14 (Probabilistic Markov AP Reward (APMAPR) Bisimulation). *An equivalence relation $\mathcal{B}$ on the set S of states of an EMRM $\mathcal{M}$ is a probabilstic- Markov-AP-Reward bisimulation (APMAPR-bisimulation), if $(s_1, s_2) \in \mathcal{B}$ implies that for all equivalence classes $\mathcal{C}$ of $\mathcal{B}$ it holds that:*

1. $L(s_1) = L(s_2)$
2. $\rho_Z(s_1) = \rho_Z(s_2)$
3. $\forall a \in \mathsf{Act}_M(\gamma_{rate}(a, s_1) = \gamma_{rate}(a, s_2))$
4. $\forall a \in \mathsf{Act}_I(\gamma_{prob}(a, s_1) = \gamma_{prob}(a, s_2))$
5. $\forall a \in \mathsf{Act}(\gamma_{imp}(a, s_1) = \gamma_{imp}(a, s_2))$

Two states s_1 and s_2 are APMAPR-bisimilar $s_1 \sim_{APMAPR} s_2$ if they are contained in a APMAPR-bisimulation.

Definition 15 (Cumulative Rate). *The cumulative rate from a state s of an EMRM $\mathcal{M}$ to a set of states $\mathcal{C}$ of $\mathcal{M}$ is defined by function $\gamma_{rate} : \mathsf{Act}_M \times S \times 2^S \mapsto \mathbb{R}$:*

$$\gamma_{rate}(a, s, \mathcal{C}) := \sum_{\lambda \in E(a,s,C)} \lambda,$$

where

$$E(a, s, \mathcal{C}) := \{\!|\lambda \big| s \xrightarrow{a,\lambda} s' \wedge s' \in \mathcal{C}|\!\}$$

$\{\!|, |\!\}$ *denote multi-set brackets.*

Definition 16 (Cumulative Probability). *The cumulative probability to come from a state s of EMRM $\mathcal{M}$ to a set of states $\mathcal{C}$ of $\mathcal{M}$ is defined by the function $\gamma_{prob} : \mathsf{Act}_I \times S \times 2^S \mapsto \mathbb{P}$:*

$$\gamma_{prob}(a, s, \mathcal{C}) := \sum_{p \in E_p(a,s,C)} p,$$

where

$$E_p(a, s, \mathcal{C}) := \{\!|p| s \xrightarrow{a,p} s' \wedge s' \in \mathcal{C}|\!\}$$

$\{\!|, |\!\}$ *denote multi-set brackets.*

Definition 17 (Cumulative Impulse Reward). *The cumulative impulse reward of a state s of EMRM $\mathcal{M}$ that is accumulated when taking transitions to a set of states $\mathcal{C}$ of $\mathcal{M}$ is defined by the function $\gamma_{imp} : \mathsf{Act} \times S \times 2^S \mapsto \mathbb{R}$:*

$$\gamma_{imp}(a, s, \mathcal{C}) := \sum_{j \in E_j(a,s,C)} j,$$

where

$$E_j(a, s, \mathcal{C}) := \{\!|j| (s \xrightarrow{a,p,j} s' \vee s \xrightarrow{a,\lambda,j} s') \wedge s' \in \mathcal{C}|\!\}$$

$\{\!|, |\!\}$ *denote multi-set brackets.*

Theorem 2 (APMAPR-Bisimulation Preserves SDRL Satisfiability). *Let $\mathcal{B}$ be a APMAPR-bisimulation, and s a state of EMRM $\mathcal{M}$, it holds that:*

1. *For all SDRL state formulae Φ: $\mathcal{M}, s \models \Phi \Leftrightarrow \mathcal{M}/\mathcal{B}, [s]_{\mathcal{B}} \models \Phi$*
2. *For all SDRL path formulae ϕ: $Prob^{\mathcal{M}}(s, \phi) = Prob^{\mathcal{M}/\mathcal{B}}([s]_{\mathcal{B}}, \phi)$.*

Especially it holds, bisimilar states satisfy the same set of SDRL formulae.

The proof of theorem 2 is by induction on the length and structure of the SDRL formulae under consideration. The proof is straight-forward and will therefore be omitted.

7 Conclusion

In this paper we have presented an extension of the logic IM-SPDL to SDRL such that we can also reason about reward-based properties of systems that have both timed and untimed behaviour.

We have defined in some detail the semantics and model checking algorithms for this logic.

We have also introduced a notion of bisimulation on EMRMs, such that model checking of SDRL formulae can also be carried out on systems which are minimal with respect to this special bisimulation relation. If we consider the high numerical complexity of GCSRL model checking, to which we could reduce SDRL model checking such a result can yield in a reduction of model checking times, as reported for CSL model checking in [13].

In the next future we plan to implement model checking and bisimulation algorithms for SDRL for Markovian approximation [9] in a symbolic, i.e. BDD-based way both serially and on parallel machines.

References

1. A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous time Markov chains. In R. Alur and T.A. Henzinger, editors, *Computer-Aided Verification*, volume LNCS 1102, pages 146–162. Springer, 1996.
2. Ch. Baier, L. Cloth, B. Haverkort, M. Kuntz, and M. Siegle. Model checking action- and state-labelled Markov chains. In *Proc. of Int. Conf. on Dependable Systems and Networks (DSN), Performance and Dependability Symposium*, pages 701–710. IEEE Press, 2004.
3. Ch. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. On the logical characterisation of performability properties. In *ICALP*, pages 780–792. Springer, LNCS 1853, 2000.
4. Ch. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Trans. Software Eng.*, 29(7):1–18, July 2003.
5. Ch. Baier, B.R. Haverkort, J.-P. Katoen, and H. Hermanns. Model Checking Continuous-Time Markov Chains by Transient Analysis. In E.A. Emerson and A.P. Sistla, editors, *Computer Aided Verification*, volume LNCS 1855, pages 358–372. Springer, 2000.
6. Ch. Baier, J.-P. Katoen, and H. Hermanns. Approximate Symbolic Model Checking of Continuous-Time Markov Chains. In J.C.M. Baeten and S. Mauw, editors, *Conurrency Theory*, volume LNCS 1664, pages 146–162. Springer, 1999.
7. Davide Cerotti, Susanna Donatelli, Andras Horvath, and Jeremy Sproston. CSL Model Checking for Generalized Stochastic Petri Nets. In *QEST '06: Proceedings of the 3rd international conference on the Quantitative Evaluation of Systems*, pages 199–210, Washington, DC, USA, 2006. IEEE Computer Society.
8. E.M. Clarke, E.A. Emerson, and A. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. In *10th ACM Annual Symp. on Principles of Programming Languages*, pages 117–126, 1983.
9. L. Cloth. *Model Checking Algorithms for Markov Reward Models*. PhD thesis, University of Twente, Enschede, Netherlands, 2006.
10. M. Fischer and R. Ladner. Propositional dynamic logic of regular programs. *J. Comput. System Sci.*, 18:194–211, 1979.
11. B.R. Haverkort and M. Kuntz. GCSRL - A Logic for Stochastic Reward Models with Timed and Untimed Behaviour. In *Proc. 8th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS-8)*. to appear, 2007.
12. H. Hermanns, J.P. Katoen, J. Meyer-Kayser, and M. Siegle. Towards model checking stochastic process algebra. In *Integrated Formal Methods*, volume LNCS 1945, pages 420–439. Springer, 2000.
13. J.-P. Katoen, T. Kemna, I. Zapreev, and D. Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In *Proc. 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, LNCS 4424, pages 76–92. Springer, 2007.
14. M. Kuntz. *Symbolic Semantics and Verification of Stochastic Process Algebras*. PhD thesis, Universität Erlangen-Nürnberg, Institut für Informatik 7, 2006.
15. M. Kuntz and M. Siegle. A stochastic extension of the logic PDL. In *Proc. Sixth Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS-6)*, pages 58–61, Monticello, Illinois, 2003.

16. M. Kuntz and M. Siegle. A Stochastic Extension of the Logic PDL. Technical Report 03/03, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2003.
17. M. Kuntz and M. Siegle. Symbolic Model Checking of Stochastic Systems: Theory and Implementation. In *13th International SPIN Workshop*, pages 89–107. Springer, LNCS 3925, 2006.
18. M. Manzano. *Model Theory*. Oxford University Press, 1999.
19. J.F. Meyer. On Evaluating the Performability of Degradable Computing Systems. *IEEE Transactions on Computer Systems*, C-29(8):720–731, August 1980.
20. J.F. Meyer. Performability: A retrospective and some pointers to the future. *Performance Evaluation*, 14(3-4):139–156, February 1992.
21. J. Meyer-Kayser. Verifikation stochastischer, prozessalgebraischer Modelle mit aCSL+. Technical Report IB 01/03, Universität Erlangen-Nürnberg, Institut für Informatik 7, 2003.
22. W.D. Obal and W.H. Sanders. State-space support for path-based reward variables. *Performance Evaluation*, 35:233–251, 1999.
23. M. Siegle. *Behaviour analysis of communication systems: Compositional modelling, compact representation and analysis of performability properties*. Shaker Verlag, Aachen, 2002.