Transformation of LARES performability models to continuous-time Markov reward models

Alexander Gouberman, Martin Riedl, Markus Siegle
Institut für Technische Informatik
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, 85579 Neubiberg, Germany
firstname.lastname@unibw.de

This paper describes the newest version of LARES, a language that allows modellers to specify the behaviour of systems which are dependable, fault-tolerant and reconfigurable. In particular, the language now provides means to express state and transition rewards, as well as different types of related performability measures of interest. The transformation of the highly structured (i.e. modular and hierarchic) user-level model to a flat continuous-time Markov reward model is discussed, including semantical issues such as guarded behaviour, forwarding of information between levels of the model hierarchy, and different forms of synchronisation between submodels. As the reward extension is fully orthogonal to the transformation of the behavioural aspects, the standard LARES reachability defined by the execution semantics and the calculation of reward measures are completely decoupled.

1. INTRODUCTION

The LARES formalism serves as a description language for dependable, fault-tolerant and reconfigurable systems. It has first been described in (Gouberman et al. 2009). LARES provides means to define individual abstract behaviour (Behavior definition) based on states (State statement) and transitions (Transitions statement) and self-contained modules (Module definition) comprising behaviour or submodule instantiations and a specification of their interaction. Furthermore, each definition may include named statements which can be addressed by the environment: e.g. to select a provided initial configuration of an instance (Initial statement), to refer to a boolean variable that asserts about the (composed) state of the addressed instance (Condition statement), or to trigger a specific reaction if a composed state satisfies a certain condition (guards statement). The reaction is specified in terms of a reactive expression and allows for different forms of synchronisations. The triggered reactions may successively be refined in the addressed subcomponents (forward statements). The ability to give abstract definitions of behaviours or modules, which allow subinstantiations, enable the modeller to specify clearly structured (i.e. modular and hierarchic) models. For hierarchical LARES models, these features can be used to capture highly combinatoric interaction behaviours by just a few simple statements. The transformation semantics of LARES to a stochastic process algebra (SPA) has been partially sketched in (Riedl and Siegle 2012).

The contribution of this paper is threefold: Firstly, we present an extension of the LARES formalism for specifying state and transition rewards. Secondly, an alternative semantics is defined which maps via some form of reachability analysis to labelled transition systems (LTS). Thirdly, it is discussed how reward measures can be calculated based on the obtained reachability graph. The LTS semantics presented in this paper is defined in such a way that it complies with the semantics given by the stochastic process algebra (which again is defined by means of LTS).

Among earlier approaches to the modelling of rewards in a Markovian context we mention Stochastic Reward Nets (Muppala et al. 1994), Stochastic Activity Networks (Qureshi et al. 1996) and the models accepted by the model checker MRMC (Katoen et al. 2011). In comparison with these, LARES models are at a higher level of abstraction, which offers much more structure and flexibility and thus eases modelling. LARES and its reward extension are intended to be used as engineering formalisms on a similar level as, for instance, AltaRica (Point 2000) or the SLIM language (Bozzano et al. 2009).

The paper is organised as follows. Subsequent to the introduction, Section 2 briefly outlines continuous-time Markov reward models (CTMRMs) and introduces the

reward extension LARES.re with a running example. In Section 3 we present the LTS semantics to construct the reachable state space. Section 4 shows the CTMRM semantics of LARES.re and discusses model properties and compositionality aspects. Finally, Section 5 concludes the paper.

2. REWARD EXTENSION

2.1. Continuous Time Markov Reward Models

A *continuous-time Markov reward model* (CTMRM) is a structure (S, Q, i, r) with finite state space S, generator matrix $Q : S \times S \to \mathbb{R}$, impulse reward $i : S \times S \to \mathbb{R}$ and rate reward $r : S \to \mathbb{R}$.

Let $E(s) := -Q(s,s) \geq 0$ be the exit rate in s. Consider the case $E(s) > 0$. Then if in a state s the sojourn time up to transition is $\tau_s \geq 0$ and the next visited state is s' then the reward accumulated in s is given by $i(s,s') + \tau_s r(s)$. Therefore the expected accumulated reward in s is given by $Rew(s) := \sum_{s' \in S} P(s,s') i(s,s') + \frac{1}{E(s)} r(s)$. In case $E(s) = 0$ the state s is absorbing, thus no impulse reward for transition is gained and $Rew(s)$ can be 0 or $\pm\infty$ depending on whether $r(s) = 0$ or not.
Note that a CTMRM can be transformed into a CTMRM $(S, Q, \overline{r})$ without impulse rewards. The reason is, that any impulse reward i can be continuized into a rate reward and merged with r:

$$\overline{r}(s) := \sum_{s' \neq s} i(s,s') Q(s,s') + r(s). \tag{1}$$

Let R denote the rate matrix, i.e. the off-diagonal entries of Q. Then Eq. (1) can be written in vector notation as

$$\overline{r} = diag(R\ i^T) + r, \tag{2}$$

where $diag$ is the diagonal of a matrix.
We will stick to the CTMRM structure with impulse rewards, since it is more natural to distinguish time-based rewards from time-independent rewards, especially in the context of modular model specification.
In order to analyze a CTMRM one has to choose a reward measure, which maps a CTMRM model into a value function $V : S \to \mathbb{R}$. In the following we shortly outline the most important reward measures (Guo and Hernandez-Lerma 2009).
(i) The *total reward measure* $V_\infty(s)$ is the expectation over all paths from s of all rewards accumulated along these paths. If this accumulation process converges, then V_∞ is a solution to

$$QV_\infty = -\overline{r}. \tag{3}$$

By setting the value $V(s) = 0$ for all recurrent states s, V_∞ is the unique solution to Eq. (3).
(ii) The *α-discounted reward measure* V^α with $\alpha > 0$ is the accumulation of rewards along all paths where rewards gained in the future are continuously discounted with rate α. It holds that the accumulation process converges for all $\alpha > 0$ and V^α is the unique solution to

$$(Q - \alpha I)V^\alpha = -\overline{r}. \tag{4}$$

(iii) The *average reward* measures the accumulated reward over all paths averaged over time. Let $P(t) = e^{Qt}$ be the probability distribution matrix at time point $t > 0$, i.e. $P(t)(s,s')$ is the probability to be in state s' at time point t, when starting in s at time point 0. Let $P^* := \lim_{t\to\infty} P(t)$ be the limiting distribution matrix of the CTMC (S, Q), i.e. the row corresponding to the s-th state is the stationary distribution of the CTMC when starting in s. (In case the CTMC is ergodic, P^* has identical rows.) Then

$$g = P^* \overline{r}. \tag{5}$$

2.2. Parallel Composition of CTMRMs

As a motivation for the LARES reward extension in Section 2.3, we shortly present here the parallel composition of CTMRM models. Let $(S_i, Q_i, r_i), i = 1, 2$ denote two CTMRMs representing the independent components of a system. Typically the CTMRM (S, Q, r) with $S = S_1 \times S_2$ representing the composed system is constructed by parallel composition of the component CTMRMs by

$$Q := Q_1 \oplus Q_2, \quad r := r_1 \otimes \mathbf{1}_{|S_2|} + \mathbf{1}_{|S_1|} \otimes r_2, \tag{6}$$

where $\oplus$ is the Kronecker sum, $\otimes$ the Kronecker product and $\mathbf{1}_n \in \mathbb{R}^n$ a vector of 1's (Buchholz 1994; Markovski et al. 2009). Although the generator Q of a CTMC with independent components can be constructed by the Kronecker sum, such a parallel composition of rewards restricts the possible definitions for typical reward structures arising in the field of performance evaluation. As an example, consider a system consisting of two identical components which can be active (state 1) or fail into the failed state 2 with rate λ. Thus the generator of the components is given by

$$Q_1 = Q_2 = \begin{pmatrix} -\lambda & \lambda \\ 0 & 0 \end{pmatrix}$$

The MTTF of a component can be computed by applying the total reward measure on the rate reward $r_i = \begin{pmatrix} 1 & 0 \end{pmatrix}^T$. Equivalently, the MTTF of the whole parallel system can be computed by defining $r = \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}^T$. However, the parallel composition as in Eq. (6) returns $r = r_1 \otimes \mathbf{1}_2 + \mathbf{1}_2 \otimes r_2 = \begin{pmatrix} 2 & 1 & 1 & 0 \end{pmatrix}^T$. Even worse, it is not possible to define a rate reward $r_i = \begin{pmatrix} a_i & b_i \end{pmatrix}^T$ for the components, s.t. the parallel composition yields the desired MTTF reward measure for the whole system. Therefore, even if the system consists of independent components for the system behavior, a desired performance measure can make them in a certain way "dependent". We show in the next

```
1  Behavior B(mu) {
     State active, failed, inRep
     Transitions from active
       if ⟨true⟩ → failed, delay exponential 0.1
     Transitions from failed
6      if ⟨rep⟩ → inRep, weight 1.0
     Transitions from inRep
       if ⟨true⟩ → active, delay exponential mu
   }

11 Module Component : B(mu=2.0) {
     Initial init = B.active
     Condition failed = !B.active
     forward ⟨rep⟩ to B.⟨rep⟩

16   StateReward energy = 2.5*[B.active]
     TransitionReward rst = 5*[B.inRep → B.active]
   }

   Module Container {
21   Instance SC1 of Component
     Instance SC2 of Component
     Initial init = SC1.init, SC2.init
     Condition failed = SC1.failed | SC2.failed
     forward ⟨rep⟩ to maxsync{SC1.⟨rep⟩,SC2.⟨rep⟩}
26
     StateReward cEnergy = SC1.energy + SC2.energy
     StateReward energy = 0.9*cEnergy + 0.5
     TransitionReward rst = 6*[failed → !failed]
   }
31
   System main {
     Instance C1 of Component
     Instance C2 of Container
     Initial init = C1.init, C2.init
36   Condition failed = C1.failed & C2.failed
     C1.failed guards {
       C1.⟨rep⟩
       C2.⟨rep⟩ if C2.failed
     }
41
     StateReward energy = C1.energy + C2.energy
     RewardMeasure M1 = energy discounted 0.01

     TransitionReward rst = C1.rst + C2.rst +
46     10.0*[failed → !failed]
     RewardMeasure M2 = (energy, rst) average
   }
```

Figure 1: *Example: LARES.re model*

section, how this problem can be addressed with LARES. Roughly speaking, the modeller can specify, how to combine reward structures in order to return the desired performance measure.

2.3. LARES.re (Reward Extension)

In order to be able to define a reward structure in LARES we extend the set of statements inside a Module definition: a rate reward can be modelled by a StateReward statement, whereas an impulse reward is specified by a TransitionReward statement. Furthermore a RewardMeasure statement assigns to a state reward or a transition reward (or a tuple consisting of both) one of the reward measure analysis types total, discounted or average.

2.3.1. Running Example

Figure 1 provides the specification of the running example used throughout this paper. It shows a system `main` (line 32) which consists of two instances `C1` and `C2` representing the components of the system. The container component `C2` contains two subcomponents `SC1` and `SC2` (line 20). Each of these basic components `C1`, `SC1` and `SC2` inherits from a behavior `B` (line 11) which describes that from an `active` state the component fails with rate `0.1`, can then be turned via the `<rep>` guard label immediately into the `inRep` state denoting an ongoing repair process and finally get active with rate `mu` (lines 1..9). The repair rate `mu` is set to the value `2.0` for every basic component. In contrast to the basic components which fail if their behavior is not active (defined by the condition `failed`, line 13) the container component `C2` is a series circuit which fails if some of its subcomponents fail (line 24). The whole system is a parallel circuit of `C1` and `C2` (line 36).

The guards statement (lines 37-40) is responsible for the repair process, which is triggered only if the condition `C1.failed` is satisfied. The reactive part `C1.<rep>` triggers the forward label `<rep>` in the `Component` module, which in turn triggers the `<rep>` guard label in the behavior `B`. In case, if `C2` has also already failed, the repair is processed in form of a choice: `C2.<rep>` triggers in parallel to `C1.<rep>` the forward label `<rep>` in the `Container` module which is responsible for the repair of both of its subcomponents `SC1` and `SC2` in a maximal synchronisation way: all subcomponents which are failed are assumed to be put to repair synchronously, i.e. all of them move to the `inRep` state immediately in one transition. With the Initial statements the initial state for each instantiated behavior is set to `active`.

We assume that every component consumes energy and we want to measure the global energy consumption of the system. A basic component continuously consumes `2.5` units of energy per time unit if it is active (line 16). If the repair process of a component is successful, an impulse reward of `5.0` energy units for the restart is consumed (line 17). The `Container` module describes in `cEnergy` the energy of its components consumed over time, and the total `energy` which decreases the energy consumption of its components by 10% and adds its own consumption of `0.5`. Furthermore when restarting the `Container` there is only energy cost for the container itself but not for its subcomponents (line 29). Finally, in the `System` definition we define the total `energy` of its components (line 42).

The reward measure `M1` specifies all the ingredients which are necessary in order to compute the discounted consumption of the continuous energy for the whole system with a discount rate of `0.01`. This discount rate can be interpreted as a rate for an exponential distribution which describes the random time length (horizon) in which the rewards are accumulated. Therefore the expected horizon length for reward accumulation is 100 time units. Note that since the system can get repaired, the total energy consumption over the whole infinite horizon without discounting

```
System main {
  Instance C1 of Component
  Instance C2 of Container
  Initial init = C1.init, C2.init
  Condition failed = C1.failed & C2.failed
  failed guards maxsync{C1.⟨stop⟩, C2.⟨stop⟩}
  !failed guards {
    C1.⟨rep⟩ if C1.failed
    C2.⟨rep⟩ if C2.failed
  }

  StateReward energy = C1.energy + C2.energy
  StateReward energyUF = energy * [!failed]
  RewardMeasure M3 = energyUF total
}

Behavior B(mu) {
  ...
  Transitions from inRep
    if ⟨stop⟩    → failed, weight 1.0
}

Module Component : B(mu=2.0) {
  ...
  forward ⟨stop⟩ to B.⟨stop⟩
}

Module Container {
  ...
  forward ⟨stop⟩ to
    maxsync{SC1.⟨stop⟩,SC2.⟨stop⟩}
}
```

Figure 2: *Example: alternative model for specification of total reward measure. The dots indicate the same statements as in the corresponding definitions in the original model in Figure 1.*

would lead to a value function which diverges to ∞. Therefore, Figure 2 shows an alternative model in which repair is only processed as long as the system has not yet failed (line 7). All ongoing repair processes are stopped synchronously by applying the `maxsync` operator and forwarding the <stop> labels. In this way, one can assert absorbing states reachable from the initial state, which allows to measure the total energy consumption up to system failure. The `energyUF` reward restricts the `energy` reward to non-failed states by multiplying it with the indicator `[!failed]`, s.t. an absorbing state is not rewarded and thus the total reward measure is finite.
Back to Figure 1, the reward measure `M2` specifies the average reward measure for the complete energy consumption, which consists of the continuous components energy and some restart energy for the components and the system.

2.3.2. Syntax description for LARES.re

A StateReward statement consists of a state reward expression, which is an arithmetic expression and extends arithmetic atoms (number value or reference to some parameter) by state reward atoms. A state reward atom can be an indicator over conditions (e.g. `[!failed]`) or a reference to another state reward statement (within the LARES scope). Similarly, a TransitionReward statement differs from a StateReward statement only in the type of the indicator, which needs both a precondition and a postcondition (e.g. `[failed -> !failed]`). In this way, the LARES modularity and hierarchy concepts nicely integrate the reward extension.
As usual, an indicator is evaluated to either 1 or 0. Therefore, in a reward expression, we do not allow to divide through other reward expressions, since the indicators can be evaluated to 0.
For the complete transformational semantics of LARES.re (described in Section 4) we need to introduce the behavioral semantics first.

3. THE LTS SEMANTICS OF LARES

A $\text{LARES}_{\text{FLAT}}$ model is obtained by a number of transformations applied to standard LARES to construct the instance tree (resolve parameters, Condition, forward and guards statements, cf. (Riedl and Siegle 2012)) and perform a flattening process from LARES to obtain a planar representation of a system. A $\text{LARES}_{\text{FLAT}}$ model

$$(B, G, M) \in \mathcal{P}(\mathcal{B}) \times multiset(\mathcal{G}) \times \mathcal{P}(\mathcal{M}) \qquad (7)$$

comprises a set of instantiated behaviours B, a multiset[1] of guards statements G denoting the interaction between the instantiated behaviours and a set of probability measure statements M. A behaviour instance $b \in B$ can be denoted as a tuple (S, T_U, T_G, s^0), where S is the set of states, T_G is the multiset of guarded transitions (its underlying set is given by $\mathcal{T}_G$), T_U is the multiset of unguarded transitions (with the underlying set $\mathcal{T}_U$) and s^0 denotes the initial state. The universal set of unguarded transitions is a cartesian product between source and target states and the possible distributions $\mathcal{D}$:

$$\mathcal{T}_U = \mathcal{S} \times \mathcal{S} \times \mathcal{D}$$

where a distribution $d \in \mathcal{D}$ is either an exponential or a discrete distribution comprising a weight. The universal set of guarded transitions is a cartesian product between source and target states and the possible distributions $\mathcal{D}$ but comprises in addition to the unguarded transitions a *guard label* $l \in \mathcal{L}$:

$$\mathcal{T}_G = \mathcal{S} \times \mathcal{S} \times \mathcal{D} \times \mathcal{L}$$

Let the set of behaviour instances B contain $n \in \mathbb{N}$ distinguishable elements $b_1, b_2, \ldots, b_n \in B$. Each behaviour instance b_i (with $i \in I^B$, where $I^B := \{1, \ldots, n\}$ denotes the index set of the behaviours) is represented by a tuple $(S_i, T_{Ui}, T_{Gi}, s_i^0) = b_i$. For two behaviour instances b_i and b_j, where $i \neq j$, the elements are disjoint, i.e. $S_i \cap S_j = \emptyset$, $T_{Ui} \cap T_{Uj} = \emptyset$ and $T_{Gi} \cap T_{Gj} = \emptyset$. We define the potential state space $S := S_1 \times \ldots \times S_n$ and denote a composed state $s \in S$ as the tuple $(s_1, s_2, \ldots, s_n) = s$. For each system

[1] in this paper, multisets are employed whenever there is a choice between identical interaction patterns, their multiplicity has to be taken into account

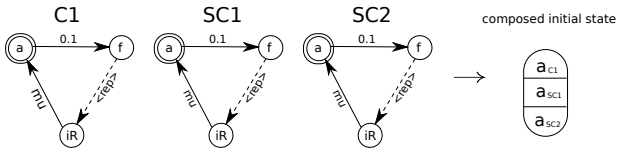

Figure 3: *Example: Composed Initial State*

of instantiated behaviours the initial composed state $s^0 \in S$ is given by $s^0 := (s_1^0, ..., s_n^0)$. The goal is to define the semantics of how the system behaves to construct the reachability graph in terms of an LTS.

3.1. Semantics: Unguarded Transitions

If one of the instantiated behaviours b_i can perform an unguarded immediate transition $\overset{w}{\dashrightarrow}$ from state s_i into the state s_i' then also the composed state s comprising s_i changes corresponding the given Structural Operational Semantics (SOS) rule as defined in Plotkin (2004). The SOS rule for an unguarded immediate transition is

$$\frac{s_i \overset{w}{\dashrightarrow} s_i'}{s \overset{w}{\dashrightarrow} s'} \qquad s' = (\{s_i'\}, s) \tag{8}$$

where s' is obtained from s by substituting the ith element of the tuple s by s_i', denoted by $s' = (\{s_i'\}, s)$.

A similar rule describes the case for s_i comprising an unguarded Markovian transition $\overset{\lambda}{\rightarrow}$ into the state s_i':

$$\frac{s_i \overset{\lambda}{\rightarrow} s_i'}{s \overset{\lambda}{\rightarrow} s'} \qquad s' = (\{s_i'\}, s) \tag{9}$$

For illustration, the example model given in Figure 1 is revisited, for which the composed initial state of the example model (cf. Figure 3) is derived. For brevity, the identifiers of the states are renamed, i.e. `a`, `f` and `iR` instead of `active`, `failed` and `inRep`. As one can see, only a Markovian transition can take place from each of the initial states of the behaviour instances. In order to determine the next reachable composed state, the matching SOS rule (9) is applied (cf. Figure 4) for each behaviour instance and each unguarded transition leaving the associated initial state to construct the first three reachable composed states.

3.2. Semantics: Guarded Transitions (informally)

It remains to define the semantics for the guarded transitions controlled by guards statements. As depicted by Figure 3, transitions with the guard label $\langle$`rep`$\rangle$ may take place from the failed states `f` if triggered by the environment. The events to do so are generated by the guards statement specified in Figure 1 in lines 37..40, which denotes how the behaviour instances interact: as long as `C1` is not failed, the SOS rules for the unguarded transitions can be safely used to further construct the state space (cf. Figure 4). If `C1` and `C2` are failed,

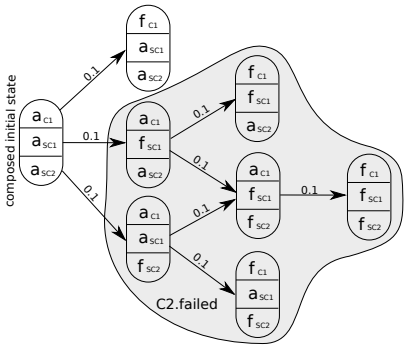

Figure 4: *Example: Recurring Application of Rule (9)*

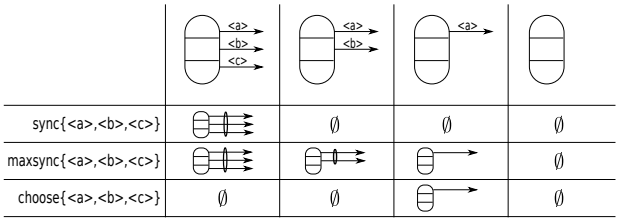

Figure 5: *Reactive Operator Semantics*

the choices `C1`.$\langle$`rep`$\rangle$ and `C2`.$\langle$`rep`$\rangle$ as reactions may be triggered, whereas only the reaction `C1`.$\langle$`rep`$\rangle$ is available if `C2` has not failed.

Up to now, nothing has been said about the internals of the container component `C2`. It is considered to be failed if either subcomponent `SC1` or `SC2` has failed. Whenever the container instance is triggered from the environment via $\langle$`rep`$\rangle$, a repair reaction within its behaviour instances is initiated. Due to the delay introduced by the environment, waiting for `C1` to fail before a repair event is generated, it might be the case that both subcomponents `SC1` or `SC2` have already failed. Using the maxsync operator to denote the repair reaction, both subcomponents are repaired at once if needed or else just a single one. The synchronisation semantics of the operators available to define reactive expressions is denoted in Figure 5. It depicts the cooperation among the behaviour instances from the viewpoint of the current composed state. The operands refer to transition guard labels of the behaviour instances. The content of the table depicts for each operator and the currently available addressed guard labels if the transitions into the next composed state can be performed simultaneously. As an example from the figure, the choose operator with the operands `a`, `b` and `c` lead to a composed transition, since a minterm $a\bar{b}\bar{c}$ in the disjunctive normal form of the choose operator $a\bar{b}\bar{c} \vee \bar{a}b\bar{c} \vee \bar{a}\bar{b}c$ is fulfilled.

Due to the transformation and flattening process to obtain a $\text{LARES}_{\text{FLAT}}$ model from a user level LARES

specification, the guards statement is resolved to

```
not C1.B.a guards {
  C1.⟨rep⟩
  maxsync{
    C2.SC1.⟨rep⟩, C2.SC1.⟨rep⟩
  } if (not C2.SC1.B.a) or (not C2.SC2.B.a)
}
```

As an example, in Figure 4 the state $(\mathtt{f}_{\mathtt{C1}}, \mathtt{f}_{\mathtt{SC1}}, \mathtt{f}_{\mathtt{SC2}})$ could be reached. As one might see, the generative part of the guards statement is thereby satisfied:

$$(\mathtt{f}_{\mathtt{C1}}, \mathtt{f}_{\mathtt{SC1}}, \mathtt{f}_{\mathtt{SC2}}) \vDash \mathtt{not\ C1.B.a}$$

As a consequence the reactive part has to be considered. Two reactive expressions have been defined, i.e. $\mathtt{C1}.\langle\mathtt{rep}\rangle$ and the conditional reactive comprising the maxsync operator. Since $(\mathtt{f}_{\mathtt{C1}}, \mathtt{f}_{\mathtt{SC1}}, \mathtt{f}_{\mathtt{SC2}})$ satisfies also the by-condition $(\mathtt{not\ C2.SC1.B.a})$ or $(\mathtt{not\ C2.SC2.B.a})$, a choice between both reactions $\mathtt{C1}.\langle\mathtt{rep}\rangle$ and $\mathtt{maxsync}\{\mathtt{C2.SC1}.\langle\mathtt{rep}\rangle, \mathtt{C2.SC1}.\langle\mathtt{rep}\rangle\}$ is possible.

This case corresponds to the second row first column of Figure 5, i.e. all guard labels referred to by the operands of the maxsync operator are available in the current state $(\mathtt{f}_{\mathtt{C1}}, \mathtt{f}_{\mathtt{SC1}}, \mathtt{f}_{\mathtt{SC2}})$. As a result a composed successor state is $(\mathtt{f}_{\mathtt{C1}}, \mathtt{iR}_{\mathtt{SC1}}, \mathtt{iR}_{\mathtt{SC2}})$ as the addressed transitions with guard label $\langle\mathtt{rep}\rangle$ of SC1 and SC2 are performed synchronously.

Based on the application of the yet informally described semantic rule, the whole state space can be generated as indicated in Figure 6. The labelling of transitions originated from the guards statement is not further detailed. Actually, the labelling relies on the enumeration of guards statement inside a specific instance and the enumeration of arising combinations thereof. In Figure 6 however, these labels are denoted as $\langle\mathtt{rep}\rangle$ for the sake of convenience.

3.3. Semantics: Guarded Behavior (formally)

Let a guards statement g be represented as a tuple $(c_0, [(c_1, r'_1), ..., (c_l, r'_l)])$ of a generative condition expression c_0 and a multiset of conditional reactives comprising l objects represented by the tuples (c_k, r'_k) with $k \leq l$.

From the viewpoint of universal algebra, the algebraic structure of a reactive expression r'_k is given by the tuple $\mathfrak{R} = (A, \text{maxsync}, \text{choose}, \text{sync}, \top, \bot)$, where A is a set of atomic assertions on transitions comprising the corresponding guard label and maxsync, choose and sync represent a number of n-ary operators, all mapping into the boolean domain $\{\top, \bot\}$. Following the semantics illustrated in Figure 5, the reactive part is translated into its boolean algebra equivalent r_k with the structure $\mathfrak{B} = (A, \wedge, \vee, \neg, \top, \bot)$, that instead of maxsync, choose and sync comprises two binary operators $\wedge$ and $\vee$ and the unary operator $\neg$ beyond A, $\top$ and $\bot$.

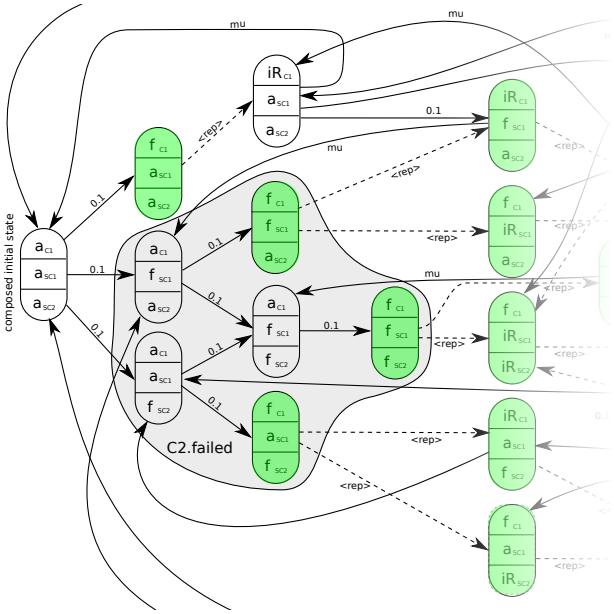

Figure 6: *Generated State Space*

A new multiset of conditional reactive tuples can be constructed, each comprising $c_0 \wedge c_k$ and r_k:

$$[(c_0 \wedge c_1, r_1), ..., (c_0 \wedge c_l, r_l)]$$

Let the set c_k^{mt} comprise all minterms arising from $c_0 \wedge c_k$ and r_k^{mt} comprise all canonical minterms originating from r_k. All combinatorial cases can be considered when a composed state satisfies the specified condition/reactive expressions of a guards statement.

3.3.1. Reactive Minterm Synchronisation Semantics

Let $\mathtt{C1}.\langle\mathtt{rep}\rangle \wedge \mathtt{SC1}.\langle\mathtt{rep1}\rangle \wedge \mathtt{SC1}.\langle\mathtt{rep2}\rangle$ be a reactive minterm $mt_r^g \in r_k^{mt}$ arising from the expression $\mathtt{sync}\{C1.\langle\mathtt{rep}\rangle, SC1.\langle\mathtt{rep1}\rangle, SC1.\langle\mathtt{rep2}\rangle\}$ of a guards statement g (as an isolated example, in order to consider a special case not captured by the example given in Figure 1). The reactive minterm is considered if the composed state $(\mathtt{f}_{\mathtt{C1}}, \mathtt{f}_{\mathtt{SC1}}, \mathtt{a}_{\mathtt{SC2}})$ satisfies a generative minterm $mt_c^g \in c_k^{mt}$. The reactive minterm is satisfied if C1 in the composed state exhibits the repair behaviour $\langle\mathtt{rep}\rangle$ and SC1 exhibits the repair behaviours $\langle\mathtt{rep1}\rangle$ and $\langle\mathtt{rep2}\rangle$.

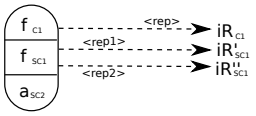

Figure 7: *Addressed Guarded Transitions*

As SC1 allows a choice between two kinds of repairs denoted by $\langle\mathtt{rep1}\rangle$ and $\langle\mathtt{rep2}\rangle$ (cf. Figure 7), the semantics in such a situation is not obvious. A synchronous execution of the transitions labelled by $\langle\mathtt{rep1}\rangle$ and $\langle\mathtt{rep2}\rangle$ contradicts the intuition of a choice. If both transitions end up in different states (as depicted by Figure 7), no unambiguous composed state can

be reached. From a number of different kinds of semantics we decided to evaluate the availability of the required guard labels by the reactive minterm, such that the exhibited behaviour satisfies the minterm, and preserves the choice such that the possible synchronous combinations are allowed to perform synchronously. As a result, a choice within a guards statement behaves consistent to a choice within a sequential behaviour and thus preserves compositionality (as a substitution of subcomponents may not lead to invalid models contrary to other semantics). In the following, the chosen semantics will be detailed formally and argued how to deal with the presence of choices.

For each minterm mt_r in r_k^{mt} and each behaviour instance b_i ($i \in I^B$), the sets $T_i^{mt_r}$ are constructed (using the multiset builder notation) comprising the guarded transitions available in a current component state s_i (denoted by $T_{Gi}|_{s_i}$) and referred to by the minterm literals $l \in mt_r$:

$$T_i^{mt_r} = [(s_i, s_i', d, l) \in^m T_{Gi}|_{s_i} : l \in mt_r]$$

As T_{Gi} is a multiset, the multiplicity m of elements is preserved in the construction.

Let $I^{mt_r} \subseteq I^B$ denote the index set of behaviour instances b_i addressed by mt_r. We define the set of sets of transitions as $T^{mt_r} := \{ T_i^{mt_r} \mid i \in I^{mt_r} \}$. Each $T_i^{mt_r} \in T^{mt_r}$ contains the enabled local choices of a behaviour instance corresponding to a minterm mt_r. Due to the chosen semantics, only one choice can be performed resulting in a number of simultaneous transition combinations declared by

$$T_{sync}^{mt_r} = \prod_{T_i^{mt_r} \in T^{mt_r}} T_i^{mt_r}$$

representing a multiset of synchronising transition tuples, where each tuple solely consists of transitions of distinct behaviour instances.

To illustrate, again Figure 7 is considered. Let mt_r be the reactive minterm $\texttt{C1}.\langle\texttt{rep}\rangle \wedge \texttt{SC1}.\langle\texttt{rep1}\rangle \wedge \texttt{SC1}.\langle\texttt{rep2}\rangle$, then

$$T^{mt_r} = \left\{ \begin{array}{l} [\texttt{f}_{\texttt{C1}} \dashrightarrow \texttt{iR}_{\texttt{C1}}], \\ [\texttt{f}_{\texttt{SC1}} \dashrightarrow \texttt{iR}'_{\texttt{SC1}}, \texttt{f}_{\texttt{SC1}} \dashrightarrow \texttt{iR}''_{\texttt{SC1}}] \end{array} \right\}$$

and the thereof arising synchronising transition tuples

$$T_{sync}^{mt_r} = \left[\begin{array}{l} (\texttt{f}_{\texttt{C1}} \dashrightarrow \texttt{iR}_{\texttt{C1}}, \texttt{f}_{\texttt{SC1}} \dashrightarrow \texttt{iR}'_{\texttt{SC1}}), \\ (\texttt{f}_{\texttt{C1}} \dashrightarrow \texttt{iR}_{\texttt{C1}}, \texttt{f}_{\texttt{SC1}} \dashrightarrow \texttt{iR}''_{\texttt{SC1}}) \end{array} \right]$$

As mentioned, a tuple in $T_{sync}^{mt_r}$ consists of a number of transitions, each from a different behaviour instance. Let $t_{sync}^{mt_r}$ denote the tuple's set of elements. A composed transition can be derived that moves those component states s_i into their successor states s_i', where $i \in I^{mt_r}$. The notation to determine the composed successor state is given by $s' = (\{s_i' \mid i \in I^{mt_r}\}, s)$, as Figure 8 depicts for the current example.

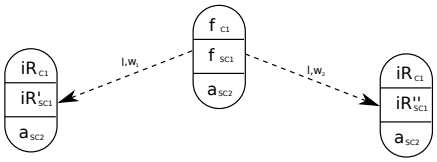

Figure 8: *Reachable Composed States*

3.3.2. SOS for Conditional Reactives

The semantics described so far can be completed and used by SOS rules. At current, LARES provides two kinds of transitions, i.e. timed (exponential distribution) and immediate (discrete distribution). By definition, it is not allowed to synchronise different types of distributions. Due to this distinction two SOS rules for immediate transition synchronisation and for Markovian transition synchronisation arise.

Both rules share common features. We assume that a rule applies for each $t_{sync}^{mt_r}$ constructed from the generative/reactive minterm combinations originated from the guards statements $g \in G$:

- The **Precondition** requires a combination of a generative and reactive minterms to be satisfied in a composed state $s = (s_1, ..., s_n)$:

  $$s \vDash mt_c^g \iff \forall v \in mt_c^g \left\{ \begin{array}{ll} \nexists i \in I^B : v \stackrel{\wedge}{=} s_i & \text{if } v \text{ negated} \\ \exists i \in I^B : v \stackrel{\wedge}{=} s_i & \text{else.} \end{array} \right.$$

  $$s \vDash mt_r^g \iff \forall v \in mt_r^g \left\{ \begin{array}{ll} \nexists i \in I^B : \chi(s_i, v, T_{Gi}) & \text{if } v \text{ neg.} \\ \exists i \in I^B : \chi(s_i, v, T_{Gi}) & \text{else,} \end{array} \right.$$
  where $\chi : (s_i, v, T) \mapsto \exists (s_i, s_i', d, v) \in T$

  Further, each of the composed transitions $t \in t_{sync}^{mt_r}$ must have the same distribution type.

- The **Side-conditions** determines the rate r rsp. weight w and the composed successor state s' for a composed transition tuple $t_{sync}^{mt_r}$. Furthermore, the label l of the composed transition is constructed using a function enc.

- The **Consequence** combines the current composed state s and the reachable state s' by a composed transition labelled by l (depending on the instance namespace, the enumeration of the guards statement and the minterm combination) with the specific distribution.

In the following, the rule considering immediate distribution type synchronisations is given, relocating the side-conditions into the subsequent explanation for clarity:

$$\frac{s \vDash mt_c^g \wedge s \vDash mt_r^g \wedge \forall t \in t_{sync}^{mt_r} : t \stackrel{\wedge}{=} \dashrightarrow}{s \overset{l,w}{\dashrightarrow} s'} \tag{10}$$

It is assumed that the rule is applicable for each guards statement and for all originating generative/reactive minterm combinations that are satisfied. Due to choices in a single instantiated behaviour, a number of composed transition tuples might arise. For each composed transition tuple $t^{mt_r}_{sync}$, where all transitions have the same distribution type, the rule determines the composed transition. The target state is given by

$$s' = (\left\{ s'_i \mid s_i \overset{l_i,w_i}{\dashrightarrow} s'_i \in t^{mt_r}_{sync} \right\}, s)$$

The associated weight of the composed transition is

$$w = \prod \left\{ w_i \mid s_i \overset{l_i,w_i}{\dashrightarrow} s'_i \in t^{mt_r}_{sync} \right\}$$

and the label $l = enc(mt^g_c, mt^g_r)$.

Similary to the SOS rule (10) to construct a composed immediate transition, a rule is defined to capture the Markovian case:

$$\frac{\begin{array}{c} s \vDash mt^g_c \wedge s \vDash mt^g_r \wedge \\ \forall t \in t^{mt_r}_{sync} : t \overset{\triangle}{=} \longrightarrow \end{array}}{s \xrightarrow{l,r} s'} \quad \begin{array}{c} l = enc(mt^g_c, mt^g_r) \\ r = \prod \left\{ r_i \mid s_i \xrightarrow{l_i,r_i} s'_i \in t^{mt_r}_{sync} \right\} \\ s' = (\left\{ s'_i \mid s_i \xrightarrow{l_i,r_i} s'_i \in t^{mt_r}_{sync} \right\}, s) \end{array} \tag{11}$$

3.4. Performing Reachability Analysis

Taking the initial composed state s^0, the recurring application of the SOS rules (8), (9), (10) and (11) explores all reachable states as partially done in Figure 6 for the example model. If the maximum progress assumption is applied to assure that immediate transitions will always take place before a Markovian transition, the rules can be prioritised, i.e. if due the the application of the immediate rules (8) and (10) an immediate composed transition is obtained from a composed state, the rules to address Markovian transitions can be neglected. The generated reachability graph is represented by a transition system $\mathcal{E} = (S, L, \dashrightarrow, \longrightarrow, s^0)$ with reachable composed state space S, a set of labels L, weighted immediate transitions $\dashrightarrow \subseteq multiset(S \times S \times \mathbb{R}^+ \times L)$, Markovian transitions $\longrightarrow \subseteq multiset(S \times S \times \mathbb{R}^+ \times L)$ and initial state s^0.

4. ANALYSIS OF LARES REWARD MODELS

In Section 2.3 we introduced LARES.re as an extension to LARES and described its syntax and informal semantics. We now define formal semantics for LARES.re which allows to analyse reward measures. For that purpose we first introduce a planar representation of LARES.re models by extending the LARES$_{\text{FLAT}}$ formalism presented in Section 3. The semantics is then defined through model transformation into a CTMRM by performing a reachability analysis as outlined in Section 3.4.

4.1. Planar Representation of LARES.re

The planar representation for LARES.re models builds up on the LARES$_{\text{FLAT}}$ formalism as defined in Eq. (7). The set of measures $\mathcal{M}$ comprises reward measures $\mathcal{M}^{RE}$ in addition to the probability measures $\mathcal{M}^{PR}$ of LARES, i.e. $\mathcal{M} := \mathcal{M}^{PR} \cup \mathcal{M}^{RE}$. A reward measure $M \in \mathcal{M}^{RE}$ is a structure $(SE, TE, type)$ where SE is a state reward expression, TE is a transition reward expression and $type$ is the specified reward analysis type, i.e. total, average or discounted(α) (cf. Section 2.3). The difference between user-level LARES.re and its planar representation is that parameters and hierarchy are resolved, s.t. all reward expressions consist of atoms which are either number values or indicators over conditions which directly point to states of instantiated behaviors (by their namespace). Note that if in the textual specification of the RewardMeasure statement one of the state reward or the transition reward expressions is not referenced it is set to the value `0`.
As an example the reward measure `M1` defined in Figure 1 by `C1.energy + C2.energy discounted 0.01` is resolved to an element $(SE, TE, type) \in \mathcal{M}^{RE}$, where

$$\begin{aligned} SE = \quad & (2.5 * [\texttt{C1.B.active}]) + \\ & (0.9 * (2.5 * [\texttt{C2.SC1.B.active}] + \\ & \qquad 2.5 * [\texttt{C2.SC2.B.active}]) + 0.5), \\ TE = \quad & \texttt{0} \text{ and } type = \texttt{discounted 0.01}. \end{aligned} \tag{12}$$

4.2. Reachability: Reward Expression Evaluation

The reachability analysis performed on a LARES$_{\text{FLAT}}$ model as described in Section 3.4 provides a transition system $\mathcal{E} = (S, L, \dashrightarrow, \longrightarrow, s^0)$. Denote by $\{s \to s'\}$ the set of all immediate and Markovian transitions from s to s'. For the reachability analysis of the planar representation of LARES.re, we extend $\mathcal{E}$ for a fixed reward measure $(SE, TE, type) \in \mathcal{M}^{RE}$ to a structure $(\mathcal{E}, \mathcal{R})$, where $\mathcal{R} = (sr, tr, type)$ with $sr : S \to \mathbb{R}$ and $tr : S \times S \to \mathbb{R}$. The functions sr and tr are defined as follows: For $s \in S$ evaluate SE to a real number by evaluating state indicators $[A]$ with a condition expression A by

$$eval([A], s) := \begin{cases} 1, & s \vDash A \\ 0, & \text{else.} \end{cases}$$

For every state s' reachable from s, i.e. $\{s \to s'\} \neq \emptyset$ evaluate TE to a real number by taking an arbitrary transition from $\{s \to s'\}$ and evaluating transition indicators $[A \to B]$ by

$$eval([A \to B], (s, s')) := \begin{cases} 1, & s \vDash A \wedge s' \vDash B \\ 0, & \text{else.} \end{cases}$$

Note that $eval([A \to B], (s, s'))$ does not depend on the chosen transition in $\{s \to s'\}$ but only on its source and target states, s.t. tr is well-defined on $S \times S$. It holds

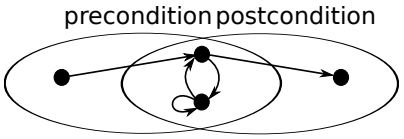

Figure 9: *Overlapping pre- and postcondition of a transition indicator*

that $eval([A \to B], (s, s')) = eval([A], s) \cdot eval([B], s')$, i.e. a transition indicator describes only a state tuple (s, s') satisfying the pre- and postconditions and does not provide any information, whether there is some transition from s to s' in the transition system.

Remark 1. In a transition indicator `[A -> B]` the pre- and postconditions can overlap (Figure 9). This means that as long as both conditions are satisfied along a path, the reward for transition is gained. If such a reward accumulation is not intended by the modeller it can be overcome by making the pre- and postcondition disjoint, e.g. by defining `[A -> !A & B]` for leaving states A into B or `[!B & A -> B]` for entering B for the first time from A. We will discuss this issue in more detail in the context of compositionality in Section 4.7.

4.3. Normalization

The state reward resp. transition reward expressions in the planar representation can be transformed (e.g. for efficiency reasons) into a normal form, which is a sum of multiplications and each multiplication is a weighted indicator, i.e. consists of a number value and an indicator. As an example, the state reward expression from Eq. (12) is normalized to

```
2.5 * [C1.B.active] + 2.25 * [C2.SC1.B.active] +
2.25 * [C2.SC2.B.active] + 0.45
```

In general, the normalization of a state reward expression can be computed by performing the following rules: Let $SE_{(i)}$, $AE_{(i)}$ and $CE_{(i)}$ denote some state reward, arithmetic resp. condition expression.

(1) Resolve all divisions and modulo expressions, e.g.

$$SE/AE \longrightarrow SE \ * \ eval(1/AE)$$

(2) Apply distributivity, e.g.

$$SE_1 * (AE_1 - SE_2) \longrightarrow SE_1 * AE_1 - SE_1 * SE_2$$

As a result, a multiplication consists of factors which are either numbers or indicators.

(3) In each multiplication, multiply all number factors together and merge all indicators by conjuncting their conditions, e.g.

$$[CE_1] * 2 * [CE_2] * (-3) \longrightarrow (-6) * [CE_1 \wedge CE_2]$$

For a transition reward expression, rules (1) and (2) are equivalent and in rule (3) two indicators $[CE_1 \to CE_2]$ and $[CE_3 \to CE_4]$ in a multiplication are merged to

$$[CE_1 \wedge CE_3 \to CE_2 \wedge CE_4].$$

Thus the normalization of reward expressions partially resolves arithmetic expressions and modifies state reward expressions. We will use the normalization in Section 4.7.

4.4. Transformation to CTMRM

In Section 4.2 the semantics of a planar LARES.re representation was defined by performing a reachability transformation into a structure $(\mathcal{E}, \mathcal{R})$, where $\mathcal{E} = (S, L, \dashrightarrow, \longrightarrow, s^0)$ is the transition system and $\mathcal{R} = (sr, tr, type)$ a reward structure on $\mathcal{E}$. In order to complete the semantics of LARES.re, we transform $(\mathcal{E}, \mathcal{R})$ into a CTMRM upon which the specified reward measure can be finally evaluated to a real valued function as described in Section 2.1.
In a CTMRM all states have only Markovian outgoing transitions, while in $\mathcal{E}$ there are also immediate transitions available. In order to get rid of these immediate transitions, we eliminate them by applying the "maximum progress assumption", which states that an immediate transition takes no time and fires before any Markovian transition can fire. In analogy with generalized stochastic Petri nets (Ajmone Marsan et al. 1984) we thus define the notion of vanishing states and discuss how these states can be eliminated.

Definition 1. A state $s \in S$ is *vanishing* if there exists $s' \in S \setminus \{s\}$ s.t. $(s, a, w, s') \in \dashrightarrow$ for some $a \in L$ and $w \in \mathbb{R}^+$. Otherwise s is called *tangible*.

Our goal is to eliminate all vanishing states, s.t. the resulting CTMRM consists only of tangible states. However, as we will see, not every rewarded transition system $(\mathcal{E}, \mathcal{R})$ can be transformed by elimination into a meaningful CTMRM. For this reason we restrict the set of all LARES.re models to valid models, which fulfill the following assumptions:

(1) the initial state s^0 is not vanishing

(2) there are no timeless traps, i.e. for all $s \in S$ there exists a finite path from s which leads into a tangible state t

(3) the value function for the reward measure is finite.

We will mention in Section 4.6, how some of these assumptions can be relaxed.

Elimination:
During the elimination process, all Markovian transitions from vanishing states are neglected due to the maximum progress assumption. Figure 10 shows by example the elimination of a vanishing state t. All incoming Markovian and immediate transitions into t are redirected to those target states t' which are reached by an immediate outgoing transition from t. Since t is vanishing, the sojourn time in t is 0 and thus

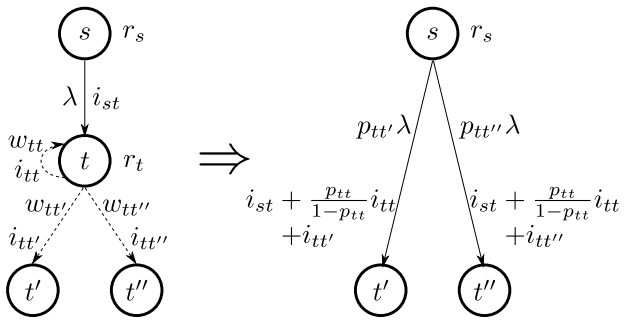

Figure 10: *Elimination of vanishing states in a reward model. An incoming Markovian transition with rate $\lambda > 0$ is split into Markovian transitions by the probability distribution $p_{tt'} = w_{tt'} w^{-1}$, $p_{tt''} = w_{tt''} w^{-1}$, where $w = w_{tt'} + w_{tt''}$ is the total outgoing weight. (A self-loop does no affect the transition probabilities $p_{tt'}$ and $p_{tt''}$.) The impulse rewards are accumulated by expectation.*

the rate reward r_t does not contribute to the reward accumulation process. However, impulse rewards are accumulated independent of the sojourn time. The transformed impulse rewards are given by expectation of the accumulation of impulse rewards along paths consisting of immediate transitions. As an example, the outgoing impulse reward $i_{tt'}$ and the self-loop impulse reward i_{tt} (weighted by the expected number of cycles) are summed up into the impulse reward i_{st} for the incoming transition.

Due to the multiset nature of transitions in the transition system $\mathcal{E}$, there can be several Markovian transitions from a source to the same target state which are merged together, i.e. their rates are summed up (race condition of exponential distributions).

The elimination process stops when there are no more vanishing states in the model. Note that during elimination, cycles of immediate transitions can end up in self-loops. A state s with only one immediate transition, which is a self-loop is not regarded as vanishing due to Definition 1. Assumption (2) guarantees that when the elimination process stops, there are no vanishing states and no immediate transitions left. This is due to the absence of timeless traps, s.t. each cycle consisting of immediate transitions can be left and thus will not shrink to an immediate self-loop at the end of the elimination process.

Therefore, the result of the elimination finally provides a CTMRM model $(\hat{S}, Q, r, i)$ consisting of tangible states $\hat{S} \subseteq S$, generator matrix $Q \in \mathbb{R}^{n \times n}$ given by the remaining Markovian transitions, a rate reward function $r : \hat{S} \to \mathbb{R}$ s.t. $r(s) = sr(s)$ and an expected accumulated impulse reward function $i : \hat{S} \times \hat{S} \to \mathbb{R}$ as a described in the elimination process.

4.5. Analysis of Running Example

In Section 2.3 we introduced the running example (Figure 1, referred by $\mathcal{L}^{orig}$) and its modification (Figure 2, referred by $\mathcal{L}^{alt}$), for which we now present the evaluation of the specified reward measures `M1`, `M2` and `M3`. Both models consist of 3 instances of Behavior `B`, s.t. the composed potential state space consists of 27 states.

In the transition system for $\mathcal{L}^{orig}$ all 27 states are reachable, from which 13 states in the eliminated CTMRM are tangible. Table 1 shows the value function V^α for the discounted reward measure `M1` with discount rate $\alpha = 0.01$ (Eq. (4)). Since the CTMRM for $\mathcal{L}^{orig}$ is ergodic (has only one recurrent class) the stationary distribution is independent on the initial state, s.t. P^* has constant rows (Eq. (5)). Therefore, the value function $g = P^* \overline{r}$ for average reward measure `M2` is constant on S with value $g = 6.729$.

In the transition system for $\mathcal{L}^{alt}$ only 18 states are reachable and the eliminated CTMRM consists of 8 tangible states. Table 2 shows the value function V_∞ for the total reward measure `M3` as specified in the alternative model (computed by Eq. (3) with value 0 for the absorbing states).

In the following sections we discuss several aspects regarding LARES reward extension as a specification language for CTMRMs. Section 4.6 deals with restrictions on models, whereas Section 4.7 mainly discusses compositionality issues of LARES.re. In Section 4.8 we shortly outline how the reward extension can be used in order to specify target functions for optimization of Markov Decision Processes.

4.6. Relaxation of Model Restrictions

In this section, we briefly outline what might happen, if the LARES.re model assumptions made in Section 4.4 are relaxed. We assume that our model is not trivial, in the sense that it shall have at least one tangible state.

Assumption (1): A vanishing inital state $s^0 \in S$ can also be eliminated, which provides an initial distribution over tangible states (reached from s^0 by only immediate transitions) instead of a unique initial tangible state. The value function $V : \hat{S} \to \mathbb{R}$ returns the value $V(s)$ over tangible states $s \in \hat{S}$. As long as there are no transition rewards from the vanishing state s^0 to all tangible states, this elimination has no effect on the reward measures. However, if there is an accumulation of transition rewards along paths from s^0 to tangible states, an initial reward value $R_{init} : \hat{S} \to \mathbb{R}$ has to be added to the total and discounted reward measures, i.e. $\hat{V}_\infty = R_{init} + V_\infty$ and $\hat{V}^\alpha = R_{init} + V^\alpha$.

Assumption (2): We do not want to analyze models with timeless traps, since the elimination procedure as described in Section 4.4 can not eliminate states s with only outgoing immediate self-loops. Such a state can not be assumed as vanishing, because it is absorbing. For this case several natural but different semantics can be applied and we do not want to distinguish between them in the scope of this paper.

Assumption (3): Since the state space is finite, the only reward measure (from the introduced measures)

(a,a,a)	(a,f,a)	(a,a,f)	(iR,a,a)	(a,iR,a)	(a,a,iR)	(a,f,f)	(a,f,iR)	(a,iR,f)	(iR,iR,a)	(iR,a,iR)	(a,iR,iR)	(iR,iR,iR)
525.71	514.60	514.60	525.50	525.13	525.13	503.49	514.02	514.02	524.67	524.67	524.55	523.84

Table 1: *Evaluation of reward measure* `M1` *for the CTMRM of the original LARES.re model* $\mathcal{L}^{orig}$ *(Figure 1). The composed states are encoded in the order* (`C1, SC1, SC2`)*, e.g.* (a,a,a) *means* (a_{C1}, a_{SC1}, a_{SC2}).

(a,a,a)	(iR,a,a)	(a,iR,a)	(a,a,iR)	(f,f,a)	(f,a,f)	(a,iR,iR)	(f,f,f)
437.13	399.66	418.36	418.36	0.00	0.00	408.89	0.00

Table 2: *Evaluation of reward measure* `M3` *for the CTMRM of the alternative LARES.re model* $\mathcal{L}^{alt}$ *(Figure 2)*

which can lead to a non-convergent value function is the total reward measure. A sufficient condition for its finiteness is, that all reachable recurrent states in the CTMRM do not contribute rewards. Furthermore, this assumption is not completely independent of assumption (2). One possibility would be to use as target models the recently introduced discontinuous CTMRM formalism (Markovski et al. 2009), which allows to deal with absorbing states comprising immediate self-loops (resp. timeless traps). While residing in such an absorbing state, time is moving on. However, if such an immediate self-loop has a non-zero impulse reward defined, an infinite reward is accumulated even for the discounted and average reward measures.

4.7. Compositionality

As already mentioned in Section 2.2 the rate rewards in CTMRM models are typically composed by summing up the rate rewards of the components (cf. Eq. (6)). This type of composition can of course be specified in LARES.re by manually defining a state reward as a sum of state rewards over its components (as it is done with the state reward `energy` in the running example in Figure 1). We say informally, that two LARES models $\mathcal{L}_1$ and $\mathcal{L}_2$ are parallely composed to a LARES model $\mathcal{L}$, if all Condition, guards and forward statements in $\mathcal{L}$ belonging to $\mathcal{L}_1$ are independent of the statements belonging to $\mathcal{L}_2$ and vice versa. This means, that if all Instance statements in $\mathcal{L}$ belonging to $\mathcal{L}_1$ or $\mathcal{L}_2$ are removed, the result is still a valid LARES model.
Note that if there is such a dependency between instances it can influence the value function of a reward measure, even if some of these instances do not contribute to the definition of the reward measure M. Even more important to note is, that if instances are indeed independent (i.e. the LARES model is parallely composed as described above) the reward measure which comprises a transition indicator can also be influenced as shown in Figure 11. Here in the original model with state space $\{s, s', s''\}$ two transitions $s \to s'$ and $s' \to s''$ are rewarded, while after composition with a further instance with state space $\{t, t'\}$ the number of rewarded transitions gets enlarged. For this reason, we have to restrict the parallel composition to a subset of models, as proposed in Definition 2.

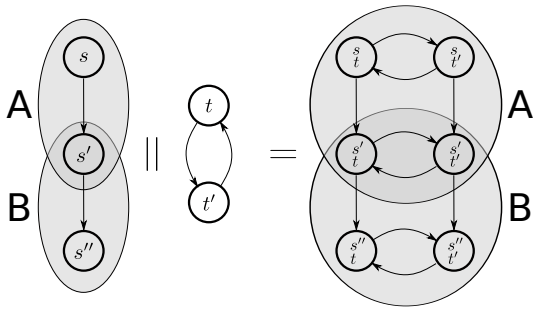

Figure 11: *Composition of two instances with overlapping pre- and postcondition of a transition indicator* `[A -> B]`

Definition 2 (*Parallel composition of LARES.re models*)**.** We call a LARES.re model *parallely composable* if for all reward measures $(se, te, type) \in M$ in its planar representation $\mathcal{L} = (B, G, M)$ holds that for all transition indicators $[A \to B]$ in te the pre- and postconditions are disjoint, i.e. $A \cap B = \emptyset$.
A LARES.re model $\mathcal{L} = (B, G, M)$ is the *parallel composition* of two LARES.re models $\mathcal{L}_k = (B_k, G_k, M_k)$ if $\mathcal{L}_1$ and $\mathcal{L}_2$ are parallely composable, $B = B_1 \cup B_2$, $G = G_1 \cup G_2$ and for all $(se, te, type) \in M$ holds that se and te are additively separable (after normalization, cf. Section 4.3), i.e. $se = se_1 + se_2$ and $te = te_1 + te_2$, s.t. se_k and te_k are expressions on $\mathcal{L}_k$.

The following theorem now states, that a parallel composition of user-level parallely composable LARES.re models is reflected as a parallel composition of the corresponding CTMRMs on the state-space level.

Theorem 1. Let $\mathcal{L} = (B_1 \cup B_2, G_1 \cup G_2, M)$ be a parallel composition of $\mathcal{L}_1$ and $\mathcal{L}_2$ and consider the additive separation of a reward measure $rm = (se_1 + se_2, te_1 + te_2, type) \in M$. Let $\mathcal{C}$ denote the CTMRM induced by evaluating rm on $\mathcal{L}$ and $\mathcal{C}_k$ the CTMRMs induced by evaluating $(se_k, te_k, type)$ on $\mathcal{L}_k$. Further let $\mathcal{C}^c$ and $\mathcal{C}_k^c$ denote the continuized CTMRMs. Then $\mathcal{C}^c = \mathcal{C}_1^c \mid\mid \mathcal{C}_2^c$.

Proof. The reachability on the union of guards statements G_1 and G_2 is performed in form of a choice, thus representing the parallel composition of the induced Markov chains. We focus in this proof on the sufficiency of the disjointness of

the pre- and postconditions in transition indicators. For simplicity let herefore $se = u_1 \cdot [U_1] + u_2 \cdot [U_2]$ and $te = v_1 \cdot [V_1 \to W_1] + v_2 \cdot [V_2 \to W_2]$ with $u_k, v_k \in \mathbb{R}$ and U_k, V_k, W_k condition expressions on $\mathcal{L}_k$, s.t. $se_k = u_k \cdot [U_k]$ and $te_k = v_k \cdot [V_k \to W_k]$. Let S_k be the composed state space of $\mathcal{L}_k$ and $S = S_1 \times S_2$ the composed state space of $\mathcal{L}$. For $s_k, s'_k \in S_k$ and denote by $\mathbf{1}^{(k)}_{U_k} \in \mathbb{R}^{|S_k|}$ the column vector and $\mathbf{1}^{(k)}_{V_k \times W_k} \in \mathbb{R}^{|S_k| \times |S_k|}$ the matrix representing the indicators $[U_k]$ resp. $[V_k \to W_k]$, i.e. $\mathbf{1}^{(k)}_{U_k}(s_k) = eval([U_k], s_k)$ and $\mathbf{1}^{(k)}_{V_k \times W_k}(s_k, s'_k) = eval([V_k \to W_k], (s_k, s'_k))$. Thus it holds $r_k = u_k \mathbf{1}^{(k)}_{U_k}$ for the rate reward and $i_k = v_k \mathbf{1}^{(k)}_{V_k \times W_k}$ for the impulse reward on $\mathcal{C}_k$. Note that $\mathbf{1}^{(k)}_{V_k \times W_k} = \mathbf{1}^{(k)}_{V_k} (\mathbf{1}^{(k)}_{W_k})^T$. In analogy let $\mathbf{1}_U \in \mathbb{R}^{|S|}$ and $\mathbf{1}_{V \times W} = \mathbf{1}_V (\mathbf{1}_W)^T \in \mathbb{R}^{|S| \times |S|}$ denote the indicator representations over conditions U, V, W on the composed state space $S = S_1 \times S_2$ of model $\mathcal{L}$. Then $r = u_1 \mathbf{1}_{U_1} + r_2 \mathbf{1}_{U_2}$ and $i = v_1 \mathbf{1}_{V_1 \times W_1} + v_2 \mathbf{1}_{V_2 \times W_2}$ are the rate reward resp. the impulse reward in $\mathcal{C}$.
Now let R_k denote the rate matrix (off-diagonal entries of Q_k). By Eq. (2) it holds for the continuized rate rewards $\overline{r}_k = diag(R_k i_k^T) + r_k$ and $\overline{r} = diag(Ri^T) + r$. The rate matrix of $\mathcal{C}$ is given by $R = R_1 \oplus R_2$ since $Q = Q_1 \oplus Q_2$. We have to show Eq. (6), i.e.

$$\overline{r} \stackrel{!}{=} \overline{r}_1 \otimes \mathbf{1}^{(2)}_\top + \mathbf{1}^{(1)}_\top \otimes \overline{r}_2, \tag{13}$$

where $\top$ is the condition expression representing 'true', s.t. $\mathbf{1}^{(k)}_\top$ is constantly 1 on S_k.
An indicator on the composed state space splits into indicators of its components by

$$\mathbf{1}_{U_1} = \mathbf{1}^{(1)}_{U_1} \otimes \mathbf{1}^{(2)}_\top \quad \text{and} \quad \mathbf{1}_{U_2} = \mathbf{1}^{(1)}_\top \otimes \mathbf{1}^{(2)}_{U_2}.$$

Therefore r is composed out of r_k as expected:

$$\begin{aligned} r &= u_1 \mathbf{1}_{U_1} + u_2 \mathbf{1}_{U_2} = u_1 (\mathbf{1}^{(1)}_{U_1} \otimes \mathbf{1}^{(2)}_\top) + u_2 (\mathbf{1}^{(1)}_\top \otimes \mathbf{1}^{(2)}_{U_2}) \\ &= r_1 \otimes \mathbf{1}^{(2)}_\top + \mathbf{1}^{(1)}_\top \otimes r_2 \end{aligned}$$

Now it remains to show an according splitting of the continuized impulse reward i. First note that

$$\begin{aligned} \mathbf{1}_{V_1 \times W_1} &= \mathbf{1}_{V_1} (\mathbf{1}_{W_1})^T = \left(\mathbf{1}^{(1)}_{V_1} \otimes \mathbf{1}^{(2)}_\top\right) \left(\mathbf{1}^{(1)}_{W_1} \otimes \mathbf{1}^{(2)}_\top\right)^T \\ &= \left(\mathbf{1}^{(1)}_{V_1} (\mathbf{1}^{(1)}_{W_1})^T\right) \otimes \left(\mathbf{1}^{(2)}_\top (\mathbf{1}^{(2)}_\top)^T\right) = \mathbf{1}^{(1)}_{V_1 \times W_1} \otimes \mathbf{1}^{(2)}_{\top \times \top} \end{aligned}$$

and equivalently $\mathbf{1}_{V_2 \times W_2} = \mathbf{1}^{(1)}_{\top \times \top} \otimes \mathbf{1}^{(2)}_{V_2 \times W_2}$. Therefore

$$\begin{aligned} i &= \left(v_1 \mathbf{1}^{(1)}_{V_1 \times W_1}\right) \otimes \mathbf{1}^{(2)}_{\top \times \top} + \mathbf{1}^{(1)}_{\top \times \top} \otimes \left(v_2 \mathbf{1}^{(2)}_{V_2 \times W_2}\right) \\ &= i_1 \otimes \mathbf{1}^{(2)}_{\top \times \top} + \mathbf{1}^{(1)}_{\top \times \top} \otimes i_2. \end{aligned}$$

It follows

$$\begin{aligned} Ri^T = (R_1 i_1^T) \otimes \mathbf{1}^{(2)}_{\top \times \top} + \mathbf{1}^{(1)}_{\top \times \top} \otimes (R_2 i_2^T) + \\ (R_1 \mathbf{1}^{(1)}_{\top \times \top}) \otimes i_2^T + i_1^T \otimes (R_2 \mathbf{1}^{(2)}_{\top \times \top}). \end{aligned}$$

In order to deduce the splitting of i we have to simplify $diag(Ri^T)$. Note that $diag(A \otimes B) = diag(A) \otimes diag(B)$. Now since $\mathcal{L}_k$ are parallelly composable by assumption, the pre- and postconditions of all transition indicators are disjoint, i.e. $V_k \cap W_k = \emptyset$. Therefore $diag(i_k) = v_k \cdot diag(\mathbf{1}^{(k)}_{V_k \times W_k}) = 0$. Combing all together Eq. (13) follows, since

$$\begin{aligned} \overline{r} &= diag(Ri^T) + r \\ &= \left(diag(R_1 i_1^T) \otimes \mathbf{1}^{(2)}_\top + \mathbf{1}^{(1)}_\top \otimes diag(R_2 i_2^T)\right) + \\ &\quad \left(r_1 \otimes \mathbf{1}^{(2)}_\top + \mathbf{1}^{(1)}_\top \otimes r_2\right) \\ &= \overline{r}_1 \otimes \mathbf{1}^{(2)}_\top + \mathbf{1}^{(1)}_\top \otimes \overline{r}_2. \end{aligned}$$

□

As a corollary, we deduce that for parallely composable LARES.re models, the composition problem as shown in Figure 11 can not occur.

Corollary 2. Let $\mathcal{L}_1 = (B_1, G_1, M_1)$ be a valid LARES.re model with composed state space S_1 and value function $V_1 : S_1 \to \mathbb{R}$. Consider a further set of Behavior instances $B_2 = \{b_k \mid k = n+1, \ldots, N\}$ (s.t. $B_1 \cap B_2 = \emptyset$) and interactions G_2 independent on G_1 (i.e. all condition expressions and reactive expressions in G_2 consider only instances B_2). If $\mathcal{L}_1$ is enhanced with B_2 and G_2 to a valid LARES.re model $\mathcal{L} := (B_1 \cup B_2, G_1 \cup G_2, M_1)$ this has no influence on the value function: If S_2 is the composed state space of the instances B_2 s.t. $S_1 \times S_2$ is the composed state space of $\mathcal{L}$ with value function $V : S_1 \times S_2 \to \mathbb{R}$ then $V(s,t) = V_1(s)$ for all $s \in S_1$ and $t \in S_2$.

Proof. The proof is obvious, since the reward measure M_1 in $\mathcal{L}_1$ is left unchanged: applying Theorem 1 yields the reward expressions $se_2 = 0$ and $te_2 = 0$ on $\mathcal{L}_2$. Therefore $\overline{r}_2 = 0$ and $\overline{r} = \overline{r}_1 \otimes \mathbf{1}^{(2)}_\top$ is not modified on states S_1. From $Q = Q_1 \oplus Q_2$ an analogous splitting holds for the time-dependent transition probabilitiy matrix $P(t) = e^{Qt}$ and its limiting distribution matrix $P^* = \lim_{t \to \infty} P(t)$. Together with Eqs. (3), (4) and (5) it follows that the value functions V_∞, V^α and g are not modified on states S_1. □

4.8. Outlook: Combination with LARES.de

In (Gouberman et al. 2013) the LARES Decision Extension (LARES.de) has been introduced which allows to model Markov Decision Processes (MDP) in a modular and hierarchical way. The MDP control actions can be created locally on the level of a Module as conditional reactives inside guards and forward statements. In this way it is possible to define both controllable and concurrent reactions. In order to provide optimization criteria for the MDP, the reward measures as defined in LARES.re can be used as target functions. An optimal policy of an MDP can be

computed by solving the non-linear Bellman equations which maximize the value function (Guo and Hernandez-Lerma 2009).

5. CONCLUSION

We have presented LARES.re as an extension to standard LARES, which allows to specify performance measures for dependable, fault-tolerant and configurable systems. Beside the user-level language we have defined a planar representation for LARES.re models. The behavioral semantics for standard LARES was defined by a model transformation into a state-based LTS formalism, which finally was transformed into a Markov chain. We have also shown the transformation of reward measures into the CTMRM formalism, which finally allows to compute the value functions corresponding to the measure. Furthermore, we discussed several important modelling issues, like synchronisation semantics, model restrictions and compositionality of LARES.re models. As future work, it is planned to employ LARES.re in industrial real-world case studies.

Acknowledgments.
We would like to thank Deutsche Forschungsgemeinschaft (DFG) who supported this work under grants SI 710/7-1 and for partial support by DFG/NWO Bilateral Research Programme ROCKS.

6. REFERENCES

M. Ajmone Marsan, G. Balbo, and G. Conte. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

M. Bozzano, A. Cimatti, M. Roveri, J.-P. Katoen, V. Nguyen, and T. Noll. Codesign of dependable systems: a component-based modeling language. In *Proc. of the 7th IEEE/ACM int. conf. on Formal Methods and Models for Codesign*, MEMOCODE'09, pages 121–130, 2009.

P. Buchholz. Markovian process algebra: Composition and equivalence. *Proc. 2nd Workshop on Process Algebras and Performance Modelling*, 27(4):11–30, 1994.

A. Gouberman, M. Riedl, J. Schuster, M. Siegle, and M. Walter. LARES - A Novel Approach for Describing System Reconfigurability in Dependability Models of Fault-Tolerant Systems. In *ESREL '09: Proceedings of the European Safety and Reliability Conference*, pages 153–160. Taylor & Francis Ltd., 2009.

A. Gouberman, M. Riedl, and M. Siegle. A Modular and Hierarchical Modelling Approach for Stochastic Control. In *MIC '13: Proceedings of the 32nd IASTED International Conference on Modelling, Identification and Control*. ACTA Press, 2013.

X. Guo and O. Hernandez-Lerma. *Continuous-Time Markov Decision Processes - Theory and Applications*. Springer, 2009.

J.-P. Katoen, I. Zapreev, E. Hahn, H. Hermanns, and D. Jansen. The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation*, 68(2):90 – 104, 2011.

J. Markovski, A. Sokolova, N. Trčka, and E. Vink. Compositionality for Markov reward chains with fast and silent transitions. *Peformance Evaluation*, 66(8):435–452, 2009.

J. Muppala, G. Ciardo, and K. Trivedi. Stochastic reward nets for reliability prediction. *Communications in Reliability, Maintainability and Serviceability*, 1(2):9–20, July 1994.

G. D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.

G. Point. *AltaRica: Contribution à l'unification des méthodes formelles et de la sûreté de fonctionnement*. Thèse de doctorat, Université Bordeaux I, 2000.

M. Qureshi, W. Sanders, A. van Moorsel, E. Y, and R. German. Algorithms for the generation of state-level representation of stochastic activity networks with general reward structures. *IEEE Trans. Software Eng.*, 22(9):603–614, 1996.

M. Riedl and M. Siegle. A LAnguage for REconfigurable dependable Systems: Semantics & Dependability Model Transformation. In *Proc. 6th International Workshop on Verification and Evaluation of Computer and Communication Systems (VECOS'12)*, eWiC, pages 78–89. British Computer Society, Aug. 2012.