Enhancing Models Correctness through Formal Verification: a Case Study from the Railway Domain

Davide Basile[1], Felicita Di Giandomenico[1] and Stefania Gnesi[1]
[1] *Istituto di Scienza e Tecnologia dell'Informazione "A. Faedo",Consiglio Nazionale delle Ricerche, ISTI-CNR, Pisa, Italy*
{*davide.basile,felicita.digiandomenico,stefania.gnesi*}*@isti.cnr.it*

Keywords: Verification, Cyber-physical System, Communication-centric Applications

Abstract: Model-based approaches are widely used for analysing systems belonging to a variety of domains, including the transportation sector. A critical issue with models is their validation, in order to justifiably put reliance on the analysis results they provide (including non functional indicators such as reliability, performance and energy consumption). Typically, cross-validation is performed, e.g. through exercising modelling by different formalisms/tools or through forms of experimental analysis. In this paper, we address validation of a case study from the railway domain via formal techniques, specifically with automata-based models. Validation of interaction aspects of Stochastic Activity Networks models of rail road switch heaters, developed for the purpose of evaluating energy consumption and reliability indicators, is performed through a tool based on contract automata, a recently introduced formalism for verifying properties of communication-based applications.

1 Introduction

Stochastic model-based analysis is a widely adopted methodology for evaluating measures of interest related to dependability and efficiency aspects, such as performance, energy consumed and probability of failures. This analysis approach is useful for expressing the stochastic nature of physical phenomena involved in Cyber-Physical Systems (CPS) (Lee, 2008), where digital control units interact with continuous phenomena describing the surrounding environment.

Formalisms as (extensions of) Petri Nets (Balbo, 2007; Sanders and Meyer, 2000; David and Alla, 2001) and (Non) Markov based models (Sanders and Meyer, 2000; Bause and Kritzinger, 1996) are used for modelling and evaluating CPS, where reward structures (Reibman et al., 1989) are defined in order to evaluate measures of interest (e.g. reliability, performance, energy consumption) at the variation of relevant parameters, either analytically or through simulation. However, in general these measures are assessed without performing any validation of the model through which they are obtained, especially when error prone communication-based applications are involved. These errors may compromise the trustworthiness of the results obtained through analysis, which means a loss of time and moneys for industries. Hence, models validation is paramount for safety critical applications.

In this paper, we propose a case study from the railway domain. In particular, we verify the soundness of interaction aspects of a rail road switch heating system, originally modelled through *Stochastic Activity Networks* (Sanders and Meyer, 2000) (SAN) for evaluating the energy consumption and the probability of failure of these devices (Basile et al., 2016a; Basile et al., 2016e). Rail road switch heaters are essential components for the correct functioning of railway stations, in absence of which possible disasters can take place (i.e. derailments, trains collision). In particularly cold regions, ice and snow can prevent the switches from working properly, hence heaters are used for guaranteeing the correct functioning of the rail road switch system. In the analysed system a central control unit is in charge of managing policies of energy consumption while satisfying reliability constraints, by communicating with the network of switches to manage the energy supply.

Representation of the interactions between the system of heaters and the central control unit constitutes a critical part of the developed SAN models. It is therefore beneficial to provide evidences of the correct modelling, which is the goal of this paper. We will adopt the *contract automata* (CA) formalism (Basile et al., 2016b; Basile et al., 2016d), that has been originally introduced for composing and verifying services. The interactions of the analysed system will be modelled with CA for guaranteeing their soundness. Contract automata are endowed with a specific tool, called *Contract Automata Tool* (CAT) (Basile et al., 2016c), that implements the theoretical results of CA, using techniques of control theory and linear programming. We automatically verify the correctness of the interactions through CAT. It is possible then to combine the quantitative assessment of stochastic measures of interest with the qualitative verification of interactions correctness, so improving the correctness of the obtained results.

Structure of the paper. A brief review of related work is discussed in Section 2; while Section 3 introduces the formalisms used in the paper, which are SAN and CA. The proposed methodology is described in Section 4, and in Section 5 we recall the formalization of the proposed case study as SAN models and we verify the correctness of the interactions through CAT. Finally, conclusions and future directions are in Section 6.

2 Related work

In the literature, several approaches for the verification and validation of stochastic models have been proposed, as for example testing, fault injection, model checking. In particular, model checking (Clarke et al., 1999) is a widely-used and powerful approach for the verification of finite state systems, based on an exhaustive exploration of the state-space. Generating the whole state space is in general inefficient, since the number of states grows exponentially in the number of components of a system. Several techniques have been developed to overcome this issue, among them we mention *modular model checking* (Kupferman and Vardi, 1998), *partial order reduction* (Clarke et al.,), *symbolic model checking* (McMillan, 1993). Tools like SPIN (Holzmann, 2003) are widely-adopted for modelling and verifying finite state systems, which implement the above techniques. However, the continuous dynamics nature of CPS is not always captured by finite state systems, and models as Timed automata (Alur and Dill, 1994), Hybrid Petri Nets (David and Alla, 2001), Stochastic Activity Networks (Sanders and Meyer, 2000) have been proposed for modelling CPS, where the evolution of the continuous variables can be uniform or described by ordinary differential equations. Several tools have been proposed for their modelling, evaluation and verification, as for example UPPAAL (Larsen et al., 1997), Kronos (Yovine, 1997), Möbius (Clark et al., 2001). When the continuous time behaviours of CPS are subject to complex and stochastic dynamics, the model checking problem is undecidable (Henzinger and Ho, 1995), and generally an approximation to more tractable models, as for example Timed automata, is performed. *Statistical Model Checking* (SMC) (Legay et al., 2010) uses results from statistics on top of simulations of a system to decide whether a given property is satisfied with some degree of confidence. UPPAAL-SMC (David et al., 2015) has been proposed as a tool that implements the above techniques.

We will provide a validation of interaction aspects of the analysed case study by formalising the interactions into a finite state formal model, which allows to apply formal verification techniques.

3 Background

In this section, we briefly introduce the Stochastic Activity Networks and contract automata formalisms.

3.1 Stochastic Activity Networks

The Stochastic Activity Networks (Sanders and Meyer, 2000) formalism is widely used for performance, dependability and performability evaluation of complex systems, given its high expressiveness and the powerful tools for modelling and evaluating them (Clark et al., 2001). The SAN formalism is a variant of Stochastic Petri Nets (Bause and Kritzinger, 1996), and has similarities with Generalised Stochastic Petri Nets (Balbo, 2007). A SAN is composed of the following primitives: *places, activities, input gates* and *output gates*. Places and activities have the same interpretation as places and transitions of Petri Nets. Input gates control the enabling conditions of an activity and define the change of marking when an activity completes. Output gates define the change of marking upon completion of the activity. Each enabled activity may complete. Activities are of two types: *instantaneous* and *timed*. Instantaneous activities complete once the enabling conditions are satisfied. Timed activities take an amount of time to complete following a temporal stochastic distribution function. An enabled activity is aborted, i.e. it cannot complete, when the SAN moves into a new marking in which the enabling conditions of the activity no longer hold. Cases are associated to activities, and are used to represent probabilistic uncertainty about the action taken upon completion of the activity. When an activity completes, the following steps are executed: (*i*) one of the cases of the activity is chosen according to its marking-depending probability; (*ii*) the function of each input gate of the activity is executed; (iii) the function of each output gate linked to the case selected at first step is executed. The primitives of the SAN models are defined using C++ code. SAN models are defined and solved by using the multi-formalism multi-solvers tool Möbius (Clark et al., 2001). Möbius is a tool that supports various formalisms such as SAN, PEPA, Fault Tree, and different analytical and simulative solvers. Möbius can be used for studying the reliability, availability, and performability of systems. It follows a modular modelling approach, where atomic models are building blocks that can be composed with proper operators *Rep* and *Join* to generate a composed model.

3.2 Contract Automata

Contract automata have been introduced in (Basile et al., 2016d; Basile et al., 2016b) for modelling and verifying contract-based services applications. A contract automaton (see Definition 1 below) represents the behaviour of a set of *principals* capable of performing some *actions*. More precisely, the tuples of actions are restricted to be offers/requests (a principal performs an offer/request and the other principals stay idle), or matches (two principals handshake their request/offer while the others stay idle). Consequently, transitions of CA will be labelled with tuples of elements in the set $\mathbb{L} \overset{\text{def}}{=} \mathbb{R} \cup \mathbb{O} \cup \{\square\}$ where: requests of principals will be built out of $\mathbb{R}$ while their *offers* will be built out of $\mathbb{O}$, $\mathbb{R} \cap \mathbb{O} = \emptyset$, and $\square \notin \mathbb{R} \cup \mathbb{O}$ is a distinguished label to represent components that stay idle. We let $a, b, c, \ldots$ range over $\mathbb{L}$ and fix an involution $\bar{\cdot} : \mathbb{L} \to \mathbb{L}$ such that $\overline{\mathbb{R}} \subseteq \mathbb{O}, \overline{\mathbb{O}} \subseteq \mathbb{R}, \forall a \in \mathbb{R} \cup \mathbb{O} : \ \overline{\overline{a}} = a$ and $\overline{\square} = \square$. As usual, offer actions will be topped by bar (i.e. $\overline{a}$). Service composition is naturally described in terms of product automata. The matching between offers and requests has to guarantee *agreement* properties that amount to *safe* communications. Intuitively, an automaton admits *strong agreement* if it has at least one trace made only by match transitions; and it is *strongly safe* if all the traces are in strong agreement. Basically, strong agreement guarantees that the composition of services has a sound execution, while strong safety guarantees that *all* executions of the composition are sound.

We borrow the following definition from (Basile et al., 2016b), where the *rank* is the number of principals inside the contract automaton, and $\vec{q}$ stands for a vector where $\vec{q}_{(i)}$ is the i-th element.

Definition 1 (Contract automata)**.** *Assume as given a finite set of states* $\mathcal{Q} = \{q_1, q_2, \ldots\}$. *Then a* contract automaton $\mathcal{A}$ *of rank n is a tuple* $\langle Q, \vec{q}_0, A^r, A^o, T, F \rangle$, *where*

- $Q = Q_1 \times \ldots \times Q_n \subseteq \mathcal{Q}^n$
- $\vec{q}_0 \in Q$ *is the initial state*
- $A^r \subseteq \mathbb{R}, A^o \subseteq \mathbb{O}$ *are finite sets (of requests and offers, respectively)*
- $F \subseteq Q$ *is the set of final states*
- $T \subseteq Q \times A \times Q$ *is the set of transitions, where* $A \subseteq (A^r \cup A^o \cup \{\square\})^n$ *and if* $(\vec{q}, \vec{a}, \vec{q'}) \in T$ *then both the following conditions hold:*
  - $\vec{a}$ *is either a request or an offer or a match*
  - *if* $\vec{a}_{(i)} = \square$ *then it must be* $\vec{q}_{(i)} = \vec{q'}_{(i)}$

A principal *is a contract automaton of rank 1 such that* $A^r \cap co(A^o) = \emptyset$.

A principal is not allowed to make a request on actions that it offers. Contract automata are endowed with two (associative/non associative) operators of composition that interleave or match the transitions of their operands. Synchronisations are forced to happen when two contract automata are ready on their respective request/offer action.

A tool called *Contract Automata Tool* (CAT) (Basile et al., 2016c) has been implemented for supporting the modelling and verification of contract automata. It provides functionalities for generating and composing different models, for synthesising the central orchestrator, and for checking if the composition of automata is correct under different properties, which amount to synchronous or asynchronous interactions, for a closed or an open-ended system. Moreover, in case one of such properties is not satisfied, it is possible to point out which principals in the composition are responsible of violating the analysed property.

4 Methodology

In this section we briefly describe the proposed approach to model and validate CPS. In the following sections this approach will be concretely applied to a representative case study.

Generally, in energy-saving CPS (Lee, 2008) the supervision of the cyber-control is in charge of strategies to supply energy to components of the physical system, necessary to keep them effective and reliable in the service they accomplish. Our interest is in assessing measures that are representative of the energy consumption, to be combined with other dependability-related properties dictated by the critical domain the CPS is employed in. In the case of critical systems, it is paramount to further guarantee the soundness of interactions among components and the trusthworthiness of the related measures. Indeed, in case of misbehaving interactions,

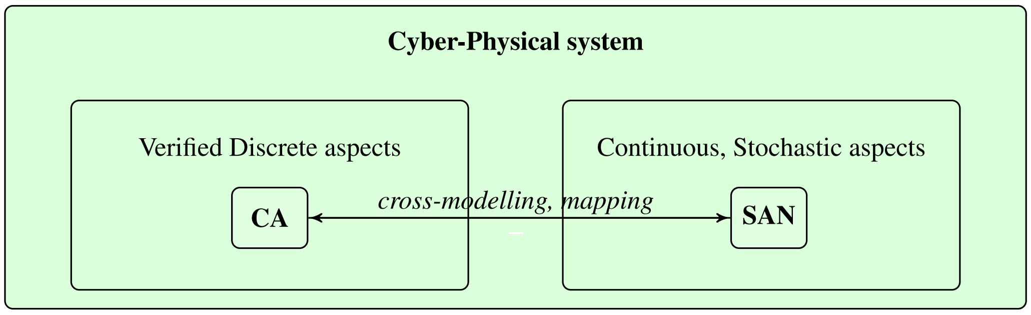


Figure 1: The proposed approach for modelling CPS based on CA and SAN models

wrong measures could be assessed. Unexpected evaluation results could be due to a non-optimal policy of energy consumption but also to a wrong implementation of the adopted policy. Generally, identifying these issues is not an easy task, and an optimal policy could be discarded due to a wrong implementation of it. In order to avoid this unpleasant situation, we aim at formally proving the soundness of the interactions. As mentioned in Section 2, the formal verification of stochastic hybrid systems is in general undecidable. To overcome this difficulty, we propose to separately model through different formalisms the cyber and the physical modules of the analyzed system.

The diagram depicted in Figure 1 illustrates our methodology. In particular, the discrete aspects concerning the interactions among components will be modelled and verified through CA, while SAN models will be also adopted for modelling the stochastic hybrid behaviours of the analysed system and for assessing the measures of interest. In the future, we plan to formally relate these two formalisations. In the following case study, the CA models of interactions will be extracted from the underlying SAN models that were previously designed (Basile et al., 2016a; Basile et al., 2016e) for evaluating energy consumption and reliability measures of the analysed system.

5 Case study: Rail road switch heating system

The considered case study is a rail road switch heating system. A rail road switch is a mechanism enabling trains to be guided from one track to another. It works with a pair of linked tapering rails, known as points. These points can be moved laterally into different positions, in order to direct a train into the straight path or the diverging path. Such switches are therefore critical components in the railway domain, and an error in the communications protocol of the system may have potentially catastrophic consequences.

During winter, snow and ice can prevent the switches to work properly, hence heaters are used so that the temperature of the rail road switches can be kept above freezing. Different policies may be adopted to power the heaters (by electricity), as for example to heat a selection of switches for a given amount of time or to heat all the switches together.

5.1 Stochastic model

We briefly recall the SAN models of the system of (remotely controlled) rail road switch heaters, which have been used for evaluating energy and reliability indicators in (Basile et al., 2016a). An on-off policy is considered for heating the switches, with parametric thresholds representing the temperatures triggering the activation/deactivation of the heating. The management of the heaters is automatic, and is remotely controlled by a central computational unit. Generally, in a railway station there are tracks which are less important than others, for example the side tracks. In case of extremely cold conditions, the total amount of energy available could not be sufficient to heat the overall system, hence it is important to assign priorities to the heaters for identifying those that must be primarily heated and those that may be heated later on.

We identify two main logical components describing the system: the *heater* and the *central coordinator*. The network of heaters is realised by replicating the heater component, and the activation/deactivation of each heater is controlled by the central coordinator. The policy employed to activate/deactivate the heating is based on two threshold temperatures: the *warning threshold* (T_{wa}) represents the lower temperature that the track should not trespass. If the temperature is lower than T_{wa}, then the risk of ice or snow can lead to a failure of the rail road switch and therefore the heating system needs to be activated; the *working threshold* (T_{wo}) is the working temperature of the heating system. Once this temperature is reached, the

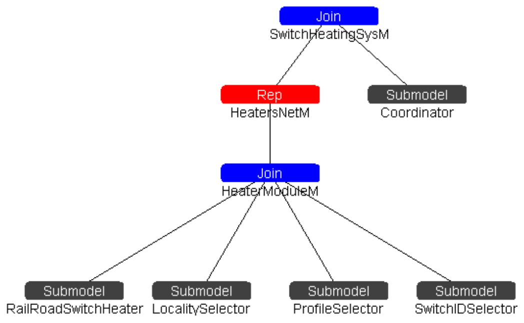


Figure 2: The composed model.

heating system can be safely turned off in order to avoid an excessive waste of energy.

The coordinator collects the requests of activation from the pending heaters, and it manages the energy supply according to a prioritized order. Indeed, the first heater which asks to be turned on will be the first to be activated. We assign priorities to switches based on their criticality on the track; the purpose of considering priorities is to guarantee higher reliability to those switches that are vital for the correct functioning of the overall station. If there is no energy available, each request will be enqueued in the queue of pending heaters.

The overall model is obtained by the composition of the atomic models, using the *Join* and *Rep* operators of the Möbius tool, as shown in Figure 2. Basically, with the *Join* operator different models are linked by sharing some places, called *shared places*, through which they interact. The *Rep* operator generates several instances of the same model, which can be uniquely identified using a tailored SAN model ($SwitchIDSelector$ in our case).

The atomic model $Coordinator$ is the central coordinator. The submodel $HeatherModuleM$ is used for modelling an instance of a single heater module, obtained by the composition, using the join operator, of the four atomic SAN models $ProfileSelector$, $LocalitySelector$, $SwitchIDSelector$ and $RailRoadSwitchHeater$, which shares different parameters concerning a single rail road switch heater. The submodel $HeatersNetM$, obtained by replicating $numRep$ times the model $HeatherModuleM$, represents the network of heaters, where the parameter $numRep$ identifies the number of devices composing the network. Finally, the model $SwitchHeatingSysM$, obtained using the join operator, represents the overall system. Indeed all the submodels share the same coordinator.

In Figure 3a the SAN model representing the rail road switch heater is depicted. We identify three logical components inside this SAN model: the *init sub-net*, the *clock sub-net* and the *heater sub-net*. The heater sub-net implements the protocol of communication between a single heater and the central coordinator, while the remaining modules are used for initialising and evaluating other parameters concerning the energy consumption and the probability of failure of the modelled heater. The heater sub-net represents the status of the rail road switch heater. The heater can be activated (one token in the place on), waiting for being activated (one token in the place $ready$), turned off (one token in the place off), or failed (one token in the place $failure$). The heater sub-net interacts with the Coordinator SAN model through places shared among all the replicas of the heater model and the Coordinator model implementing the logic described below. For example, if a heater `H` is on state ready, in order to be turned on, the input gate $i1_{ready2on}$ checks if the marking of the shared place $notifyIn$ is equal to the marking of the place $SwitchID$ of `H`(i.e. its unique identifier). This means that the coordinator has notified `H` to be turned on.

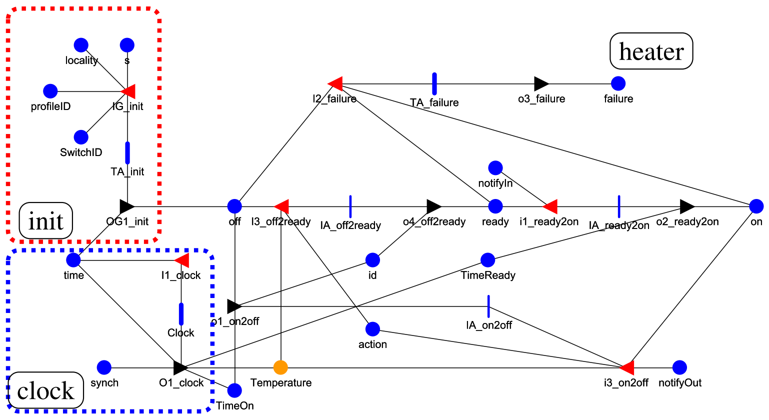
We emphasise the cyber-physical nature of the case study by briefly describing its physical aspects. The increment and decrement of the temperature of the rail road track, respectively when the heater is turned on or off, is modelled by a differential equation representing the balance of energy (Basile et al., 2016a), where assuming that the values of the temperature of the surrounding area T_e and the previous internal temperature T are known, the updated internal temperature T after time t is:

$$mc\frac{\partial T}{\partial t} = -uA(T - T_e) + \dot{Q},$$

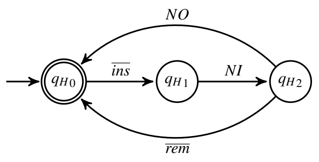
where u is the coefficient of convective exchange; c, the heat capacity of iron; A, the surface area exposed to the external temperature; m, the mass of the iron bar; $\dot{Q}$, the power used when the heater is turned on, if the heater is turned off this value will be zero.

5.2 Formal Verification of Interactions Correctness

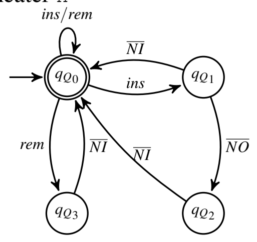
We now formally verify the interactions between the network of heating switches and the central coordinator, to improve the trusthworthiness of the results obtained through the proposed SAN models in (Basile et al., 2016a). For this purpose, we model the interactions of the SAN models with contract automata. We remark that we are not providing a generic mapping from SAN models to CA models, which is matter of future investigations. Indeed, here we focus on verifying the interactions of specific SAN models that have been previously implemented. An over approx-



(a) The SAN model RailRoadSwitchHeater, logically divided into three sub-nets: the init, clock and heater sub-nets.



(b) the CA model of the rail road switch heater H



(c) the CA model of the coordinator Q

Figure 3: The SAN model and the contract automata models.

imation of all the possible behaviours of the system will be verified against the property of strong agreement of the product automaton, which only allows for synchronous interactions (i.e. traces only made by match actions), as those are the behaviour showed by the modelled system. In Figure 3b the contract automaton representing a rail road switch heater `H` is displayed, while the contract automaton of the coordinator `Q` is in Figure 3c. We now describe all the possible interactions between the central control unit and the network of switches implemented in the SAN models and modelled with CA:

- `H`: in the initial state q_{H_0} the heater is switched off and the internal temperature is above the warning threshold. Once the internal temperature goes below the warning threshold, the heater issues a request to be activated to the coordinator with the action $\overline{ins}$.

  In state q_{H_1} the internal temperature is below the warning threshold and the heater is waiting a notification from the central coordinator to be turned on. When the message NI (i.e. notify in) is received, the heater is turned on, represented by the state q_{H_2}. From this state two transitions are allowed:

  - $\overline{rem}$ (i.e. remove), the heater reaches an internal temperature above the working threshold, and communicates to the central coordinator the termination of the heating phase and switches to state q_{H_0};
  - NO (i.e. notify out) a second heater `H'` with higher priority asks to be turned on. The energy delivered to `H` is turned off and `H` is switched to state q_{H_0}, even though it has not yet reached an internal temperature above T_{wo} (however the temperature could be above T_{wa}: if it is not the case there will be an instantaneous transition from q_{H_0} to q_{H_1} as previously described).

  The target state of both transitions is q_{H_0}, which is also the final state of the heater.

- `Q`: in the initial (and final) state q_{Q_0} the central coordinator is waiting for a message from one of the heaters in the network. Two messages can be received:

  - ins: a heater asks to be activated. This request can be rejected in case there is no available energy and the priority is not higher than those activated heaters, which is modelled by the inner loop (q_{Q_0}, ins, q_{Q_0}). In this case a notification of activation will be issued as soon as there is energy available (see below).

    Otherwise, the request is accepted and the target state is q_{Q_1}. In state q_{Q_1} two transitions are allowed. In case there is enough available energy, the heater is activated with the message $\overline{NI}$. Otherwise, if there is no available energy but `H` has a priority higher than one of the activated heaters `H'`, firstly a message $\overline{NO}$ is issued to `H'`, which will be consequently turned off, and then the activation is notified to `H` with the message $\overline{NI}$;
  - rem: a heater `H` notifies the deactivation. If there are no heaters `H'` activated or waiting for being activated then no action is performed, modelled with the inner loop (q_{Q_0}, rem, q_{Q_0}). Otherwise, after receiving the message rem,

one of the pending heater $\mathtt{H}'$ is activated by issuing the message $\overline{NI}$ to $\mathtt{H}'$.

The places $off, ready, on$ of the SAN model in Figure 3a are modelled as states $q_{H_0}, q_{H_1}, q_{H_2}$ of the corresponding automaton in Figure 3b. The modelling of states of the SAN model *Coordinator* (not displayed here) with the contract automaton $\mathtt{Q}$ in Figure 3c is similar. The interactions of $\mathtt{H}$ and $\mathtt{Q}$ are modelled in the SAN models through the places $notifyIn, notifyOut, action$ and id that are shared among the different replicas of the SAN models for the heater and the unique SAN model of the coordinator. More precisely, through the places $notifyIn, notifyOut$ the coordinator sends messages to one of the heaters in the network, which is uniquely identified by the number of tokens into these places. These interactions are modelled as transitions of the contract automata labeled by the actions $\overline{NI}$ and $\overline{NO}$. Similarly, the places *action* and *id* are used by the heaters to communicate to the coordinator the action $\overline{ins}$ or $\overline{rem}$ of the contract automata (respectively one or two tokens in the place *action*) and the identity of the sender (place *id*).

The contract automata models over approximate the real behaviour of the system. For example, from states q_{Q_0} of $\mathtt{Q}$ different transitions can be chosen non-deterministically. Moreover, since here we want to verify the progress of interactions, we do not consider possible failures of heaters. By making the product automaton of *numRep* instances of the heater model $\mathtt{H}$ with the coordinator $\mathtt{Q}$, it is possible to analyse the behaviour of the overall system. For displaying purposes, here we only discuss the verification of a network with two heaters and the coordinator, which is composed of 29 transitions and 16 states.

We remark that this approach scales to more complex systems. We have successfully analysed a model with $\approx$ 5000 transitions, which can be accessed at `https://github.com/davidebasile/CAT/tree/master/JaMata/HeatersNet`. The corresponding liable transitions (see below) have been checked in few seconds.Note that generally it is not decidable to verify the correctness of a network with an arbitrary number of replicas of a SAN model (through *Rep* operator), because the different replicas can be uniquely identified. Indeed, a reduction to the halting problem for Turing machines is provided in (Apt and Kozen, 1986).

We check that no interactions between the central coordinator and the heaters lead to a deadlock. For this purpose, CAT checks the presence of *strongly liable* transitions in the composed automaton, which represent the bad behaviour of the system (i.e. interactions responsible of leading the composition to a deadlock). CAT detects different strongly liable transitions, displayed in Figure 4. In the first row, we have the transition $((q_{H_0}, q_{H_1}, q_{Q_0}), (\overline{ins}, \square, ins), (q_{H_1}, q_{H_1}, q_{Q_0}))$, where $\mathtt{H}_1$ interacts with $\mathtt{Q}$ by synchronising on the action $\overline{ins}$, and $\mathtt{Q}$ non deterministically decides to take its inner loop, while $\mathtt{H}_1$ stays idle. In the target state, both heaters are on the state q_{H_1}. In this configuration $\mathtt{Q}$ cannot send the message $\overline{NI}$ any more and the system gets stuck with both heaters waiting their turn for being activated. Note that in the source state of the transition $(q_{H_0}, q_{H_1}, q_{Q_0})$, the component $\mathtt{H}_2$ is in state q_{H_1}, hence there are no heaters activated. Indeed, this transition represents a *false positive* that is a behaviour that will never be possible in the real system; because if there are no activated heaters, a request of activation will never be refused. The second transition, i.e. $((q_{H_1}, q_{H_0}, q_{Q_0}), (\square, \overline{ins}, ins), (q_{H_1}, q_{H_1}, q_{Q_0}))$, is the symmetric case for the heater $\mathtt{H}_2$. All the other liable transitions represent all the possible combinations of interactions where none of the two heaters is waiting for being activated on state q_{H_1}, one heater terminates the heating phase by sending the message $\overline{rem}$ and the coordinator $\mathtt{Q}$ moves non deterministically to state q_{Q_2}, however no heater is waiting for being activated and the message $\overline{NI}$ cannot be delivered. Once again, all these transitions are *false positive*. Indeed if no heater is in state q_{H_1}, $\mathtt{Q}$ will never send a $\overline{NI}$ message, because there are no heaters waiting for being activated, and the inner loop of $\mathtt{Q}$ will always be taken.

$$((q_{H_0}, q_{H_1}, q_{Q_0}), (\overline{ins}, \square, ins), (q_{H_1}, q_{H_1}, q_{Q_0}))$$
$$((q_{H_1}, q_{H_0}, q_{Q_0}), (\square, \overline{ins}, ins), (q_{H_1}, q_{H_1}, q_{Q_0}))$$
$$((q_{H_2}, q_{H_2}, q_{Q_0}), (\overline{rem}, \square, rem), (q_{H_0}, q_{H_2}, q_{Q_3}))$$
$$((q_{H_2}, q_{H_2}, q_{Q_0}), (\square, \overline{rem}, rem), (q_{H_2}, q_{H_0}, q_{Q_3}))$$
$$((q_{H_2}, q_{H_0}, q_{Q_0}), (\overline{rem}, \square, rem), (q_{H_0}, q_{H_0}, q_{Q_3}))$$
$$((q_{H_0}, q_{H_2}, q_{Q_0}), (\square, \overline{rem}, rem), (q_{H_0}, q_{H_0}, q_{Q_3}))$$

Figure 4: the strongly liable transitions of $\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}$.

These false positives are automatically removed by CAT, by computing the so-called most permissive controller. In particular, the "bad" states $\vec{q}$ that are removed are those satisfying the conditions: $\forall i \in 0 \ldots numRep - 1. \vec{q}_{(i)} = q_{H_1} \land \vec{q}_{(numRep+1)} = q_Q0$ or $\forall i \in 0 \ldots numRep - 1. \vec{q}_{(i)} \neq q_{H_1} \land \vec{q}_{(numRep+1)} = q_Q3$.

No strongly liable transitions are found in the refined automaton, and we conclude that the interactions of the system are *sound*: no deadlock will ever occur.

6 Conclusions

We have verified the correctness of the interactions of a stochastic model of a cyber-physical system. The selected case study is a system of rail road switch heaters that have been modelled in (Basile et al., 2016a) through SAN models in order to evaluate indicators of the reliability and the energy consumption. Starting from the SAN models, the interaction patterns have been modelled with contract automata, so allowing the automatic verification of the progress of interactions through the Contract Automata Tool (Basile et al., 2016c).

Based on the obtained results, we are planning to generalise the adopted technique to formally relate SAN models and CA models and proving the correctness of the mapping. Moreover, we would like to extend the comparisons with: (i) stochastic hybrid automata that have been used in (Basile et al., 2017a) to analyse rail road switch heating systems, and (ii) an extension of contract autamata formalism to express necessary and mandatory requirements (Basile et al., 2017b), useful for modelling critical requests that must be fulfilled.

REFERENCES

Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235.

Apt, K. R. and Kozen, D. (1986). Limits for automatic verification of finite-state concurrent systems. *Inf. Process. Lett.*, 22(6).

Balbo, G. (2007). Introduction to generalized stochastic petri nets. In Bernardo, M. and Hillston, J., editors, *Formal Methods for Performance Evaluation*, volume 4486 of *LNCS*. Springer.

Basile, D., Chiaradonna, S., Giandomenico, F. D., and Gnesi, S. (2016a). A stochastic model-based approach to analyse reliable energy-saving rail road switch heating systems. *Journal of Rail Transport Planning & Management*, 6(2):163 – 181.

Basile, D., Degano, P., and Ferrari, G. L. (2016b). Automata for specifying and orchestrating service contracts. *CoRR*, abs/1607.08363.

Basile, D., Degano, P., Ferrari, G.-L., and Tuosto, E. (2016c). *Playing with Our CAT and Communication-Centric Applications*, pages 62–73. Springer International Publishing, Cham.

Basile, D., Degano, P., Ferrari, G. L., and Tuosto, E. (2016d). Relating two automata-based models of orchestration and choreography. *Journal of Logical and Algebraic Methods in Programming*, 85(3).

Basile, D., Di Giandomenico, F., and Gnesi, S. (2016e). Tuning energy consumption strategies in the railway domain: a model-based approach. In *7TH International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, ISOLA 2016*.

Basile, D., Di Giandomenico, F., and Gnesi, S. (2017a). Statistical model checking of an energy-saving cyber-physical system in the railway domain. In *The 32nd ACM Symposium on Applied Computing, SAC 2017*. (to appear).

Basile, D., Di Giandomenico, F., Gnesi, S., Degano, P., and Ferrari, G.-L. (2017b). Specifying variability in service contracts. In *Proceedings of the 11th International Workshop on Variability Modelling of Software-intensive Systems (Vamos), February 1 - 3*. (to appear).

Bause, F. and Kritzinger, P. S. (1996). Stochastic petri nets: An introduction to the theory. *SIGMETRICS Perform. Eval. Rev.*, 26(2).

Clark, G., Courtney, T., Daly, D., Deavours, D., Derisavi, S., Doyle, J. M., Sanders, W. H., and Webster, P. (2001). The möbius modeling tool. In *Proceedings of the 9th International Workshop on Petri Nets and Performance Models*, pages 241–250.

Clarke, E., Grumberg, O., Minea, M., and Peled, D. State space reduction using partial order techniques. *International Journal on Software Tools for Technology Transfer*, 2(3).

Clarke, Jr., E. M., Grumberg, O., and Peled, D. A. (1999). *Model Checking*. MIT Press, Cambridge, MA, USA.

David, A., Larsen, K. G., Legay, A., Mikuǎionis, M., and Poulsen, D. B. (2015). Uppaal smc tutorial. *Int. J. Softw. Tools Technol. Transf.*, 17.

David, R. and Alla, H. (2001). On hybrid petri nets. *Discrete Event Dynamic Systems*, 11(1-2):9–40.

Henzinger, T. A. and Ho, P. (1995). Algorithmic analysis of nonlinear hybrid systems. In *Computer Aided Verification, 7th International Conference, Proceedings*.

Holzmann, G. (2003). *Spin Model Checker, the: Primer and Reference Manual*. Addison-Wesley Professional, first edition.

Kupferman, O. and Vardi, M. Y. (1998). *COMPOS'97*, chapter Modular Model Checking.

Larsen, K. G., Pettersson, P., and Yi, W. (1997). Uppaal in a nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing*, ISORC '08. IEEE Computer Society.

Legay, A., Delahaye, B., and Bensalem, S. (2010). *RV 2010. Proceedings*, chapter Statistical Model Checking: An Overview. Springer.

McMillan, K. L. (1993). *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell, MA, USA.

Reibman, A., Smith, R., and Trivedi, K. (1989). Markov and markov reward model transient analysis: An overview of numerical approaches. *European Journal of Operational Research*, 40(2).

Sanders, W. H. and Meyer, J. F. (2000). Stochastic activity networks: Formal definitions and concepts. In *Lectures on Formal Methods and Performance Analysis*.

Yovine, S. (1997). Kronos: A verification tool for real-time systems. (kronos user's manual release 2.2). *International Journal on Software Tools for Technology Transfer*, 1:123–133.