A Refinement Approach to Analyse Critical Cyber-Physical Systems

Davide Basile[2 1], Felicita Di Giandomenico[1], and Stefania Gnesi[1]

[1] I.S.T.I "A.Faedo" CNR Pisa, Italy
[2] Dept. of Information Engineering University of Florence, Italy

Abstract. Cyber-Physical Systems (CPS) are characterised by digital components controlling physical equipment, and CPS are typically influenced by the surrounding environment conditions. Due to the stochastic continuous nature of the involved physical phenomena, for quantitative evaluation of non-functional properties (e.g. dependability, performance) stochastic hybrid model-based approaches are mainly used. In case of critical applications, it is also important to verify specific qualitative aspects (e.g. safety). Generally, stochastic hybrid approaches are not suitable to account for the co-existence of both qualitative and quantitative aspects. In this paper we address this issue by proposing a refinement approach for analysing stochastic hybrid systems starting from a verified discrete representation of their logic. Different formalisms are used and formally related. It is then possible to combine the quantitative assessment of stochastic continuous properties with the qualitative verification of logic soundness, thus improving the trustworthiness of the analysis results.

1 Introduction

Recently, Cyber-Physical Systems (CPS) [18] have been given attention from the research community and are characterised by digital components (e.g. transducers) interacting with continuous phenomena describing the surrounding physical environment. These systems can be thought as communication-based applications (CBA), where different cyber entities (e.g. sensors, actuators) communicate to realise the overall behaviour. CBA are generally error-prone and verifying them is not an easy task [7].
Dependability and efficiency aspects of CPS can be analysed through a stochastic model-based approach, because of the stochastic nature of the involved physical phenomena. When critical applications are considered, it is definitely not sufficient to concentrate the verification efforts on quantitative properties only, but the validation of qualitative properties such as safety aspects is paramount.
However, when stochastic continuous behaviours are introduced in the models, the verification of qualitative safety properties becomes undecidable [16].
Previously, in [8,5] we have analysed a cyber-physical system from the railway domain with a tailored approach, not reusable in other cyber-physical systems. In this paper we propose a general approach for validating CPS models based on modelling the system starting from its logical aspects (e.g. components interactions), to be refined and decorated to include dependability aspects. A key insight of this approach is to analyse fault-tolerant systems acting when dependability/performance aspects are threatened, to restore them to safe values. These

aspects are related to stochastic hybrid quantities. However, the logic can be efficiently modelled and verified separately, by assuming that the conditions detecting threats hold. If it is not the case then the system can be assumed to operate safely. The verified logical aspects are automatically synthesised and embedded into the overall model. Then the cyber-physical model is (1) equipped with guarantees on the soundness of the implemented logic, and (2) provides a verified basis for detailing the stochastic continuous aspects of interest (e.g. related to performance, dependability) and analyse them.
Through the combination of quantitative assessment of stochastic continuous properties with the qualitative verification of interactions soundness, we aim at improving the trustworthiness of the obtained analysis results. To illustrate its application and the potential benefits, the proposed methodology will be applied to an industrial case study from the railway domain [5].
Structure of the paper. In Section 2 a motivating example is introduced that will be used throughout the paper. The proposed methodology is described in Section 3 and a brief description of contract automata and stochastic activity networks is in Section 3.1. The case study is modelled through contract automata in Section 4. The main results of the paper (i.e. the mapping from contract automata to activity networks) are in Section 5. The extension to include stochastic continuous aspects is detailed in Section 6, while related work and conclusions are in Section 7. All proofs and additional details can be found in [4].

2 Motivating example

We start by introducing the case study, a rail road switch heating system [5], that will be used in the paper to explain our methodology. A rail road switch is a mechanism enabling trains to be guided from one track to another. Heaters are used so that the temperature of the rail road switches can be kept above freezing, to avoid possible disasters.
We will consider a dynamic power management policy for heating the switches, with parametric thresholds representing the temperatures triggering the activation/deactivation of the heating. In particular, the policy employed is based on two threshold temperatures: the *warning threshold* (T_{wa}) represents the lower temperature that the track should not trespass. If the temperature T 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 (i.e. $T > T_{wo}$), the heating system can be safely turned off in order to avoid an excessive waste of energy. This is an example of a system reaction to threats, to guarantee required reliability while improving energy consumption. Indeed if $T_{wa} < T < T_{wo}$ then, respectively, the switch will not freeze and will not waste energy: in this case there is no threat to the reliability and energy consumption of the analysed system.
The control part of our system mainly consists of two components realising the logic described above: the *heater* and the *central coordinator*. A network of heaters is realised by composing the heater components, and their activation/deactivation is controlled by the central coordinator. The coordinator collects the requests of activation from the pending heaters, and it manages the energy supply according to a prioritised order. In particular, for each priority class, the

first heater that asks to be turned on will be the first to be activated, according to a FIFO order. If there is no energy available, each request will be enqueued in the queue of pending heaters. The continuous aspects are related to the temperature of the rail road track. The induction heating will be modelled with a differential equation modelling the balance of energy. The stochastic aspects of the case study concern modelling the environment temperature (weather conditions) and the probabilistic time-to-failure when the freezing threshold is reached.

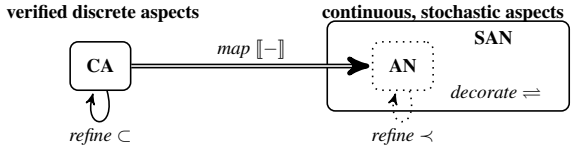


Fig. 1: The proposed framework for CPS based on CA, AN and SAN models

3 Modelling Cyber-physical Systems

Our approach to model cyber-physical systems relies on the combined usage of the following formalisms and tools. *Contract automata* (CA) [6] are a recent formalism for modelling and verifying CBA, implemented in the *Contract Automata Tool* (CAT) [7]. An original facet of CA is the adoption of techniques from Control Theory to synthesise a controller enforcing safety properties (thus motivating our choice over this formalism). Stochastic Activity Networks (SAN) [20] are widely used for analysing CPS. They generalise Stochastic Petri Nets [10] and are equipped with a powerful tool, called Möbius [11], useful for evaluating quantitative properties under a given degree of confidence. However, despite their popularity, the formal verification of qualitative properties of SAN models has received low attention [1].

In Figure 1 our modelling framework is depicted, which can be divided into three phases. In the first phase, the system logic will be modelled through the CA formalism and the verified control part will be synthesised. In the second phase, Activity Networks (AN) models are automatically generated through a mapping from CA models. Finally, in the third phase the SAN models will be obtained by properly extending the generated AN models to introduce the stochastic continuous physical aspects of the modelled system (e.g. differential equations and stochastic phenomena describing the surrounding physical environment), while identifying the point of interactions between logical and physical sub-modules. We remark that the CA are supported by mechanisms (e.g. controller synthesis) not available for AN and SAN, and this motivates the mapping from CA models to AN models. In all the phases it is possible to refine the abstract models to more concrete representations. The correspondence among the different models is guaranteed by formal results. In the following we will apply this modelling framework to our case study. The detailed formalization of the above phases is performed with the support of the case study introduced in Section 2. This makes the steps concrete, while exposing the developed theory at the basis of our approach. Before doing this we shortly present the adopted formalisms.

3.1 Background

Contract Automata and (Stochastic) Activity Networks are recalled below.

Contract Automata A contract automaton (see Definition 1) is a finite state automaton representing the behaviour of a set of *principals* performing some *actions*. States of CA are vectors of states of principals, where $\vec{q}$ stands for a vector and $\vec{q}_{(i)}$ is the i-th element. The transitions of CA are labelled with *actions*, that are vectors of elements in the set $\mathbb{L} = \mathbb{R} \cup \mathbb{O} \cup \{\bullet\}$ where $\mathbb{R} \cap \mathbb{O} = \emptyset$, and $\bullet \notin \mathbb{R} \cup \mathbb{O}$ is a distinguished label to represent components that stay idle. Actions are as followed: *offers* (belonging to the set $\mathbb{O}$ and depicted as overlined labels on arcs, e.g. $\overline{ins}$), *requests* (belonging to the set $\mathbb{R}$ and depicted as non-overlined labels on arcs, e.g. *ins*), or *match* actions (i.e. handshake between a request and an offer). The goal of each principal is to reach an accepting (*final*) state where all its requests and offers are matched. We borrow the following definition from [6], where the *rank* is the number of principals inside the contract automaton.

Definition 1. *Given a finite set of states* $Q = \{q_1, q_2, \ldots\}$*, a* contract automaton $\mathcal{A}$ *of rank n is a tuple* $\langle Q, \vec{q_0}, A^r, A^o, T, F \rangle$*, where* $Q = Q_1 \times \ldots \times Q_n \subseteq Q^n$ *is the set of states,* $\vec{q_0} \in Q$ *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* $A \subseteq (A^r \cup A^o \cup \{\bullet\})^n$ *and if* $(\vec{q}, \vec{a}, \vec{q'}) \in T$ *then both the following conditions hold: (1)* $\vec{a}$ *is either a request or an offer or a match; (2) if* $\vec{a}_{(i)} = \bullet$ *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 step $(w, \vec{q}) \xrightarrow{\vec{a}} (w', \vec{q'})$ occurs if and only if $w = \vec{a}w', w' \in A^*$ and $(\vec{q}, \vec{a}, \vec{q'}) \in T$. Let $\rightarrow^*$ be the reflexive, transitive closure of the transition relation $\rightarrow$. The language of $\mathcal{A}$ is denoted as $\mathscr{L}(\mathcal{A}) = \{w \mid (w, \vec{q_0}) \xrightarrow{w}{}^* (\varepsilon, \vec{q}), \vec{q} \in F\}$. A step is denoted as $\vec{q} \xrightarrow{\vec{a}}$ when w, w' and $\vec{q'}$ are irrelevant and $(w, \vec{q}) \rightarrow (w', \vec{q'})$ when $\vec{a}$ is irrelevant.
We now describe informally the composition operators of CA. The product automaton basically interleaves or matches the transitions of principals. Synchronisations are forced to happen when two contract automata are ready on their respective request/offer action and in a composed CA, states and actions are, respectively, vectors of states and actions of principals (i.e. the formalism is compositional). Moreover, a contract 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.
An original facet of CA is the possibility of synthesising a controller; that is a non-empty sub-portion of a CA $\mathcal{A}$ that is strongly safe and convergent, i.e. from each reachable state it is possible to reach a final state. The *most permissive strong controller* (mpc) $\mathcal{KS}_{\mathcal{A}}$ of $\mathcal{A}$ is such that all controllers $\mathcal{KS}'_{\mathcal{A}}$ are included in the mpc, and it is unique up to language equivalence. Techniques for synthesising the mpc have been introduced in [6] and implemented in [7]. The *Contract Automata Tool* (CAT) [7] has been implemented for supporting the modelling and verification of CA. It provides functionalities for generating and composing different CA, and for checking if their composition is correct under different properties, for example strong agreement and strong safety.

Stochastic Activity Networks. Stochastic activity networks (SAN) [20] models are widely used for performance, dependability and performability evaluation of complex systems. The SAN formalism is a variant of Stochastic Petri Nets [10], and has similarities with Generalised Stochastic Petri Nets [3].

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 predicate of an activity. 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 with activities, and are used to represent probabilistic uncertainty about the action taken upon completion of the activity. Moreover, each input or output gate is connected to a single activity and to a unique place.

A stochastic activity network is formally defined as a tuple $\langle AN, C, F, G\rangle$ where AN is the underlying activity network, C and F are functions assigning probabilistic distribution to cases of activity and time (for timed activities), respectively, and G is the predicate of reactivation. In Section 5 we will provide a mapping from contract automata to activity networks (AN), hence their formalisation is introduced. Let P be the set of all places of the network and $S \subseteq P$. The marking of S is formally defined as $\mu : S \to \mathbb{N}$. Moreover, $M_S = \{\mu | \mu \ : \ S \to \mathbb{N}, S \subseteq P\}$ is the set of possible markings of S. In the following input gates are defined as triples $(G,\ e,\ f)$ where $G \subseteq P$ is the set of input places, $e : M_G \to \{0,1\}$ is the enabling predicate and $f : M_G \to M_G$ is the input function. Output gates are defined as pairs $(G,\ f)$ where $G \subseteq P$ is the set of output places and $f : M_G \to M_G$ is the output function.

Definition 2. *An activity network (AN) is defined as* $\mathcal{N} = \langle P,\ A,\ I,\ O,\ \gamma,\ \tau,\ \iota,\ o\rangle$ *where: P is a finite set of places, A is a finite set of activities, I is a finite set of input gates, O is a finite set of output gates, $\gamma : A \to \mathbb{N}^+$ defines the number of cases for each activity, and $\tau : A \to \{$Timed, Instantaneous$\}$ specifies the type of each activity. Finally, the function $\iota : I \to A$ maps input gates to activities, and the function $o : O \to \{(a,c) \mid a \in A \wedge c \in \{n \mid n \in \mathbb{N}^+, n \leq \gamma(a)\}\}$ maps output gates to pairs of activity and corresponding cases.*

Moreover, let $IP(a)$ and $OP(a)$ be the input and output gates of an activity a, respectively. A step $\mu \xrightarrow{a,c} \mu'$ denotes the completion of the activity a (enabled in μ) and selected case c that yields μ'. We will denote $\mu \xrightarrow{a} \mu'$ when the activity has a single case and $\mu \xrightarrow{w} {}^* \mu'$ for the reflexive transitive closure of $\to$, where $w \in A^*$. Moreover, we assume the existence of an initial marking μ_0 and a set of final markings $F = \{\mu_1, \ldots, \mu_n\}$. We introduce the notion of *convergent* AN, that is an AN always capable of reaching a final (successful) state.

Definition 3. *An activity network $\mathcal{N}$ is* convergent *iff* $\forall \mu.\mu_0 \to^* \mu, \exists \mu_f \in F.\mu \to^* \mu_f$, *and is* deadlock-free *iff* $\forall \mu.\mu_0 \to^* \mu, \exists \mu'.\mu \to \mu'$.

Note that convergence implies the absence of both deadlocks and livelocks.

4 Modelling the System Logic through CA

The control part of our example is described below. In particular, following the general guidelines of our approach we firstly abstract the stochastic continuous behaviour related to the temperatures (assuming that the related conditions on temperatures are eventually satisfied) to efficiently verify the soundness of interactions (used to enforce reliability and energy saving when threatened). In particular, priorities are initially abstracted in the given CA, and will be enforced later on by the refinement operation of the controller.
Stochastic continuous aspects related to temperature can be introduced later on through model refinement to allow the evaluation of quantitative properties (e.g. performance, reliability), while preserving the soundness of interactions.
In Figure 2 the contract automaton representing a rail road switch heater H is displayed, while the contract automaton of the central control unit Q is in Figure 3.

Heater. In the initial state q_{H_0} the heater H is switched off and the internal temperature T is above T_{wa}. Once T goes below T_{wa}, H asks to be activated to the central control unit Q with the offer $\overline{ins}$ (i.e. insert). In state q_{H_1}, T is below T_{wa} and H is waiting for a notification from the Q to be turned on. When the message NI (i.e. notify in) is received, H is turned on, represented by the state q_{H_2}.

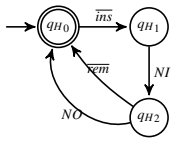


Fig. 2: H

From q_{H_2} two transitions are allowed: (1) $\overline{rem}$ (i.e. remove), H has $T > T_{wo}$, and communicates to Q the termination of the heating phase and switches to state q_{H_0}; (2) 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 a T above T_{wo} (however T 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 H.

Central control unit. In the initial (and final) state q_{Q_0} the central control unit Q is waiting for a message from one of the heaters. Two messages can be received: (1) *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).

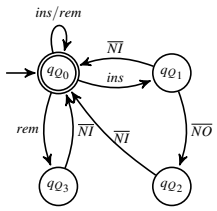


Fig. 3: Q

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 $\overline{NI}$. From state q_{Q_0} the second possible message is: (2) *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 $\mathtt{H}'$.

Composition. The composition of the network of switches with the control unit is now introduced. The CA models over approximate the real behaviour of the system. For example, from state q_{Q_0} of $\mathtt{Q}$ different transitions can be chosen non-deterministically. Indeed, priorities and energy available are not modelled in the CA, but they will be enforced by synthesising a controller such that all and only the behaviour satisfying these constraints will be obtained. Let *nH* be the number of heaters in the network. By composing (through the CA operators of composition) *nH* instances of the heater model $\mathtt{H}$ with the central control unit $\mathtt{Q}$, it is possible to analyse the behaviour of the overall system against the property of *strong agreement* of the product automaton (i.e. composed system).
For displaying purposes, in our working example we will consider a network composed of two heaters and the central control unit. We remark that the case study scales to a higher number of heaters. Through CAT it is possible to compute automatically the mpc of the composed automaton $\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}$ (subscript are used to identify an instance of $\mathtt{H}$). In Figure 4 $\mathcal{KS}_{\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}}$ is displayed.
CAT provides further information on the interactions between the central control unit and the heaters that could lead to a deadlock/livelock (i.e. those blocked by the controller). For this purpose, the *strongly liable* transitions in the composed automaton are checked, that are scenarios in which (1) no heater is activated (hence there is energy available) but $\mathtt{Q}$ refuses the activation, or (2) no heater is waiting for being activated (i.e. their temperatures are above the warning threshold), but $\mathtt{Q}$ issues a notification of activation.
We note that the liable transitions are due to non-deterministic behaviour of $\mathtt{Q}$, because we are not explicitly modelling the available energy, the priorities and the queue of pending heaters. The synthesis of $\mathcal{KS}_{\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}}$ automatically removes these unwanted behaviours. Indeed, the conditions "*if no heater is active then accept a request of activation*" and "*if no heater is waiting for being activated then do not notify any activation*" are inferred automatically, without explicitly modelling the energy and the queue of pending heaters, which would increase the state-space of the system. However, $\mathcal{KS}_{\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}}$ still admits behaviours not expected in the system. For example, a notification of deactivation (i.e. $\overline{NO}$) should be emitted only if one of the heaters has a priority higher than the other, but the controller admits traces where the message $\overline{NO}$ is delivered to both heaters.

Refinement. As mentioned earlier, the enforcement of FIFO priorities and energy available constraints in the CA of the example is discussed. For this purpose, there is the need to refine the given composition of CA to a more concrete one. Indeed, the behaviour of a CA $\mathcal{A}$ (or its mpc) over-approximates that of the analysed application. We introduce here a notion of refinement of an mpc $\mathcal{KS}_{\mathcal{A}}$ to remove unwanted behaviours. From the supervisory control theory, any controller $\mathcal{KS}'_{\mathcal{A}}$ of $\mathcal{A}$ is a refinement (i.e. a sub-automaton) of the mpc $\mathcal{KS}_{\mathcal{A}}$ of $\mathcal{A}$. In the following a set of "bad" states $Bad(Q)$ to be removed in the refinement of the mpc is identified.

Lemma 1. *Let $\mathcal{KS}_{\mathcal{A}}$ be the mpc of $\mathcal{A}$ and $Bad(Q) \subseteq (Q_{\mathcal{KS}_{\mathcal{A}}} \setminus F_{\mathcal{KS}_{\mathcal{A}}})$. Moreover, let $\mathcal{KS}_{spr}$ be a CA obtained from $\mathcal{KS}_{\mathcal{A}}$ by removing all states in $Bad(Q)$ and their incident transitions. The controller $\mathcal{KS}'_{\mathcal{A}} = \mathcal{KS}_{\mathcal{KS}_{spr}}$ is a refinement of $\mathcal{KS}_{\mathcal{A}}$.*

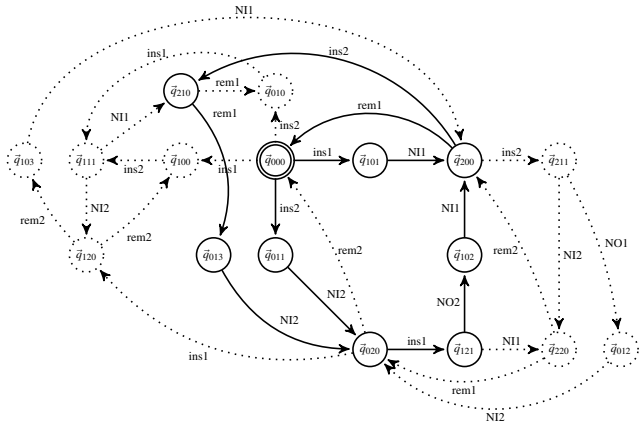


Fig. 4: The mpc $\mathcal{KS}_{\mathtt{H}\otimes\mathtt{H}\otimes\mathtt{Q}}$ (solid/dotted lines) and the controller $\mathcal{KS}' = \mathcal{KS}_{\mathcal{KS}_{spr}}$ (solid lines). For enhancing readability the labels are renamed as: $ins1 = (\overline{ins}, \bullet, ins)$, $ins2 = (\bullet, \overline{ins}, ins)$, $rem1 = (\overline{rem}, \bullet, rem)$, $rem2 = (\bullet, \overline{rem}, rem)$, $NI1 = (NI, \bullet, \overline{NI})$, $NI2 = (\bullet, NI, \overline{NI})$, $NO1 = (NO, \bullet, \overline{NO})$, $NO2 = (\bullet, NO, \overline{NO})$.

The refinement of the mpc of the example to a more concrete one is now discussed. The bad states representing behaviours to remove will be firstly declared as predicates ϕ_i and removed from the controller. Note that such bad states can be identified as those not satisfying one or more temporal logic formulae (e.g. CTL). In particular we will use four predicates to identify states $\vec{q} \in Q$ such that: $\phi_1(Q)$ – there are more active heaters than available energy; $\phi_2(Q)$ – the maximum number of active heaters has been reached, but a request of activation of a high/low priority heater has been wrongly accepted; $\phi_3(Q)$ – the maximum number of active heaters has been reached, but a request of activation of a high priority heater has been wrongly rejected, because there is an active low priority heater; $\phi_4(Q)$ – a request of activation of a heater has been wrongly rejected, because there is energy available. The set of bad states to be removed will be defined as the union of these four predicates, detailed below.

We assume that two priority classes are present in our system and each heater is uniquely identified by an index. Let $\mathbb{N}^{<n}$ be the set of positive natural numbers equal or smaller than n. Let $P1$ and $P2$ be respectively the (index) sets of high-priority and low priority heaters, such that $P1 \cap P2 = \emptyset$ and $P1 \cup P2 = \mathbb{N}^{<nH}$, and let $S = \mathcal{P}(\mathbb{N}^{<nH})$ be the power set of indexes. Moreover, $en \in \mathbb{N}^{<nH}$ is the maximum number of active heaters, i.e. the energy available to the system.

For example by assuming that $\mathtt{H}_1$ has a priority higher than $\mathtt{H}_2$ and only one heater can be active in a unit of time, we have $P1 = \{1\}, P2 = \{2\}, en = 1$. Let $\circ \in \{>, <, =\}$, and let $\phi_\circ(Q) = \{\vec{q} \in Q \,|\, \exists P \in S, |P| \circ en, \forall i \in P, j \in \mathbb{N}^{<nH} \setminus P : \vec{q}_{(i)} = q_{H2} \wedge \vec{q}_{(j)} \neq q_{H2}\}$ be the predicate stating that the maximum number of active heaters has been, respectively, exceeded ($\circ$ equal $>$), not reached ($\circ$ equal $<$) and reached ($\circ$ equal $=$). The "bad" states of the mpc are $Bad(Q) = \{\vec{q} \in Q \,|\, \vec{q} \in \bigcup_{i \in \mathbb{N}^{<4}} \phi_i(Q)\}$, where $\phi_1(Q) = \{\vec{q} \in \phi_>(Q)\}$, $\phi_2(Q) = \{\vec{q} \in \phi_=(Q) \,|\, (\vec{q}_{(nH)} = q_{Q1}) \wedge ((\bigvee_{i \in P2} \vec{q}_{(i)} = q_{H1}) \vee (P \subseteq P1 \wedge (\bigvee_{i \in P1} \vec{q}_{(i)} = q_{H1})))\}$, $(\vec{q}_{(nH)} = q_{Q1}$ is the

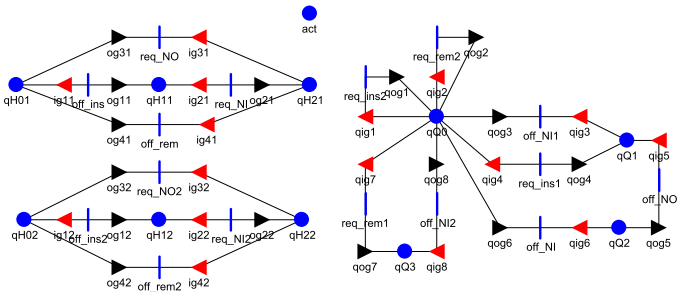


Fig. 5: The AN $[\![\mathcal{KS}_{\mathtt{H}\otimes\mathtt{H}\otimes\mathtt{Q}}]\!]$.

state of the coordinator that has accepted a request). Here a low priority heater cannot trigger the deactivation of another low priority heater, while for high priority heaters it is required $P \subseteq P1$ (all active heaters have high priority). Finally, $\phi_3(Q) = \{\vec{q} \in \phi_=(Q) \mid (\vec{q}_{(nH)} = q_{Q0}) \wedge (\bigvee_{i\in P2} \vec{q}_{(i)} = q_{H2}) \wedge (\bigvee_{i\in P1} \vec{q}_{(i)} = q_{H1})\}$ ($\vec{q}_{(nH)} = q_{Q0}$ states that the request of activation of the pending heater (q_{H1}) has been rejected), and $\phi_4(Q) = \{\vec{q} \in \phi_<(Q) \mid (\bigvee_{i\in \mathbb{N}^{<nH}\setminus P} \vec{q}_{(i)} = q_{H1}) \wedge (\vec{q}_{(nH)} = q_{Q0})\}$. Let us assume that in our example $\mathtt{H}_1$ has a priority higher than $\mathtt{H}_2$ and only one heater can be active in a unit of time. We know that state $\vec{q}_{220} \in \phi_1(Q)$ (see Figure 4), because both heaters are activated, $\vec{q}_{211} \in \phi_2(Q)$ since $\mathtt{Q}$ has accepted the request of activation of $\mathtt{H}_2$ but no energy is available and its priority is low, $\vec{q}_{120} \in \phi_3(Q)$ because $\mathtt{Q}$ has rejected the request of activation of $\mathtt{H}_1$ (high priority), and $\vec{q}_{100}, \vec{q}_{010} \in \phi_4(Q)$ because $\mathtt{Q}$ has refused the request of activation of a heater but there is energy available. This removal operation leads to a spurious controller $\mathcal{KS}_{spr}$. By reapplying the synthesis step a controller $\mathcal{KS}' = \mathcal{KS}_{\mathcal{KS}_{spr}}$ is computed (see Figure 4) where all dangling states and transitions are removed, e.g. $\vec{q}_{012}, \vec{q}_{103}$ and $\vec{q}_{111}$. Now the non-determinism has been removed from $\mathcal{KS}'$, and we have synthesised all and only sound interactions implementing the logic of the system (i.e. the policy of energy saving). Following Lemma 1, $\mathcal{KS}_{spr}$ is the automaton obtained from $\mathcal{KS}_{\mathtt{H}_1\otimes\mathtt{H}_2\otimes\mathtt{Q}}$ by removing these states, and $\mathcal{KS}_{\mathcal{KS}_{spr}} = \mathcal{KS}'$ is the corresponding controller in Figure 4.

5 Mapping from CA to AN

In this section we present the second phase of our approach, which is the core part of this paper: the formal mapping from CA to AN, together with a notion of refinement of AN. The mapping will be tailored to satisfy the property of strong agreement of CA.

We start by introducing some notation useful for defining the mapping. Let $id : \mathbb{O} \to \mathbb{N}^+$ be an injective function assigning a unique id to each offer. The set identifying all possible states $\vec{q}$ of $\mathcal{A}$ in which principal i in state $q = \vec{q}_{(i)}$ is ready to fire the offer $\overline{a}$ is denoted as: $S\uparrow_{q,i,\overline{a}} = \{\vec{q} \mid (\vec{q},\vec{a},\vec{q}') \in T, \vec{q}_{(i)} = q, \vec{a}_{(i)} = \overline{a}\}$. Conversely, the set of states $\vec{q}_1$ in which principal i in state $q = \vec{q}_{1(i)}$ is ready to perform the request a and principal j has performed the corresponding offer (i.e. $\vec{q}_{1(j)} = \vec{q}'_{(j)}$) is denoted as: $S\downarrow_{q,i,a} = \{\vec{q}_1 \mid (\vec{q},\vec{a},\vec{q}') \in T, \vec{q}_{(i)} = q, \vec{a}_{(i)} =$

$a, \exists j.\vec{a}_{(j)} = \overline{a}, \forall z \neq j.\vec{q}_{(z)} = \vec{q}1_{(z)}, \vec{q}1_{(j)} = \vec{q}'_{(j)}\}$. According to the strong agreement property, a request will be fired only after the corresponding offer has been made. Moreover, the marking function is extended to vectors of states/places as: $\mu(\vec{q}) = 1$ if $\forall i.\mu(\vec{q}_{(i)}) = 1, \mu(\vec{q}) = 0$ otherwise. Note that there could be other places in the AN ignored by $\mu(\vec{q})$.

The mapping from CA to AN is informally described now, to help intuition. The states of principals are in one-to-one correspondence with places in the AN, plus an additional place *act* for storing the offer performed in the intermediate step. Activities are in one-to-one correspondence with transitions of principals. All activities are instantaneous and have only one case. The firing of activities will be in correspondence with match transitions fired in the composed CA: given a match transition t where a principal i performs an offer transition t_i and a principal j performs a request transition t_j then the order of firing of activities of the corresponding AN will be: first a_{t_i} (offer) and then a_{t_j} (request). According to the semantics of AN, for each action a_t firstly the function of the corresponding input gate $IG(a_t)$ will be executed, and then the function of the corresponding output gate $OG(a_t)$.

For each input gate, its guard ensures that the corresponding principal does not evolve autonomously but its behaviour adheres with the synthesised controller. In particular, for an activity in correspondence with an offer transitions t_o the guard checks that the place *act* is empty (i.e. there are no other pending offers waiting to be received), and that the overall marking of the network corresponds to a state of the CA where the (offer) activity can be fired (i.e. $\mu(\vec{q})$ such that $\vec{q} \in S\uparrow_{q,i,\overline{a}}$).

The marking changes (function f of, respectively, input and output gate) by setting to zero the marking of source place q of the principal i who is firing the offer and by adding one token in the target place and $id(o)$ tokens to place *act*, to record that the offer o has been fired. For an activity in correspondence with a request transition t_r, the guard checks if the marking of *act* codifies the corresponding offer (i.e. the offer has been fired), and if the overall marking of the network corresponds to a state of the automaton where the (request) activity can be fired (i.e. in $S\downarrow_{q,i,a}$). The marking changes by setting to zero the markings of the source place *act* (the offer has been received) and by adding a token in the target place.

The conditions on the gates allow to define the set of places of the AN as the union of states of principals. An alternative solution could be to consider as set of places the states of the product of principals (i.e. $Q_1 \times \ldots \times Q_n$), and avoiding the guards on input gates. However, the latter solution would generate a bigger state-space. In the following, $\Pi^i(\mathcal{A})$ is the projection operator of CA, extracting from a CA $\mathcal{A}$ of rank $n > 1$ the i-th principal with $1 \leq i \leq n$. The mapping is formally defined below.

Definition 4 (Mapping). *Let $\mathcal{A} = \langle Q, \vec{q}_0, A^r, A^o, T, F\rangle$ be a CA of rank n such that $\forall i \in \mathbb{N}^{<n}$. $\Pi^i(\mathcal{A}) = \langle Q_i, \vec{q}_{0(i)}, A^r_i, A^o_i, T_i, F_i\rangle$. The mapping function $[\![-]\!]$: CA $\rightarrow$ AN is defined as $[\![\mathcal{A}]\!] = \mathcal{N}$ where $\mathcal{N} = \langle P, A, I, O, \gamma, \tau, \iota, o\rangle$ and:*

- $P = \bigcup_{i\in\mathbb{N}^{<n}} Q_i \cup \{act\}$, $A = \{a_t \mid t \in T_i, i \in \mathbb{N}^{<n}\}$,
- $I = \{IG(a_t) \mid a_t \in A\}$, *where* $\forall a_t \in A$ *s.t.* $t = (q, a, q') \in T_i, i \in \mathbb{N}^{<n}$

$$\text{if } a \in \mathbb{O} \text{ then } \begin{cases} IG(a_t) = (\bigcup_{\vec{q}\in S\uparrow_{q,i,a}} \{\vec{q}_{(1)}, \ldots, \vec{q}_{(n)}\} \cup \{act\}, g, f) \\ g = ((\mu(act) == 0) \wedge (\bigvee_{\vec{q}\in S\uparrow_{q,i,a}} \mu(\vec{q}) == 1)), \\ f(\mu) = \{\mu' \mid \forall p \in P \setminus \{q\}.\mu'(p) = \mu(p), \mu'(q) = 0\} \end{cases}$$

$$\textit{if } a \in \mathbb{R} \textit{ then } \begin{cases} IG(a_t) = (\bigcup_{\vec{q} \in S\downarrow_{q,i,a}} \{\vec{q}_{(1)}, \ldots, \vec{q}_{(n)}\} \cup \{act\}, g, f) \\ g = ((\mu(act) == id(\vec{a}) \wedge (\bigvee_{\vec{q} \in S\downarrow_{q,i,a}} \mu(\vec{q}) == 1)), \\ f(\mu) = \{\mu' \mid \forall p \notin \{q, act\}, \mu'(p) = \mu(p), \mu'(q) = \mu'(act) = 0\} \end{cases}$$

- $O = \{OG(a_t) \mid a_t \in A\}$ *s.t.* $\forall a_t \in A, t = (q, a, q') \in T_i, i \in \mathbb{N}^{<n}$. $OG(a_t) = (\{q', act\}, f)$ *where if* $a \in \mathbb{O}$ *then* $f(\mu) = \{\mu' \mid \forall p \in P \setminus \{q', act\}.\mu'(p) = \mu(p)\mu'(q') = 1, \mu'(act) = id(a)\}$ *else if* $a \in \mathbb{R}$ *then* $f(\mu) = \{\mu' \mid \forall p \in P \setminus \{q'\}.\mu'(p) = \mu(p), \mu'(q') = 1$
- $\forall a \in A.\gamma(a) = 1, \tau(a) =$ *Instantaneous,* $IG(a) = \{ig\}.\iota(ig) = a$,
- $\forall a \in A, OG(a) = \{og\}.o(og) = (a, 1)$.

The initial marking $[\![\mathcal{A}]\!]$ *is* $\mu_0(\vec{q}_0) = 1, \mu_0(act) = 0$ *and the set of final markings is* $\{\mu \mid \mu(\vec{q}) = 1, \vec{q} \in F, \mu(act) = 0\}$.

Example 1. In Figure 5 the AN $\mathcal{N} = [\![\mathcal{KS}_{\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}}]\!]$ computed through Definition 4 is depicted. For example, for the activity `off_ins` we have the input gate $ig11 = (P, g, f)$ where the places of the input gate are: $P = \{qH01, qH02, qQ0, qH12, qH11, act\}$; the guard is: $g = (\mu(act) == 0) \wedge (\mu(qH01) == \mu(qH02) == \mu(qQ0) == 1 \vee \mu(qH01) == \mu(qH22) == \mu(qQ0) == 1) \vee \mu(qH01) == \mu(qH12) == \mu(qQ0) == 1))$; and the change of marking is $f(\mu) = (\mu'(qH01) = 0, \forall p \in P \setminus \{qH01\}.\mu'(p) = \mu(p))$. Moreover, the output gate $og1 = (P, f)$ where its places are $P = \{qH11, act\}$ and the change of marking is $f(\mu) = \mu'(qH11) = 1, \mu'(act) = id(\overline{ins}), \forall p \in P \setminus \{qH11, act\}, \mu'(p) = \mu(p)$.

The following definition provides a mapping from transitions of CA to corresponding activities of AN, and will be useful in the following. Even though traces in strong agreement are only made by match transitions, we also consider offer and request transitions for a total mapping.

Definition 5. *The mapping from a transition t of a CA to the corresponding activity* a_t *of AN is defined as:*

$$[\![(\vec{q}, \vec{a}, \vec{q}')]\!] = \begin{cases} a_{(\vec{q}_{(i)}, \vec{a}_{(i)}, \vec{q}'_{(i)})} & \textit{if } \vec{a} \textit{ offer/request}, a_{(i)} \neq \bullet \\ a_{t1}a_{t2} & \textit{if } \vec{a} \textit{ match}, \vec{a}_{(i)} \in \mathbb{O}, \vec{a}_{(j)} \in \mathbb{R}, t1 = (\vec{q}_{(i)}, \vec{a}_{(i)}, \vec{q}'_{(i)}) \\ & t2 = (\vec{q}_{(j)}, \vec{a}_{(j)}, \vec{q}'_{(j)}) \end{cases}$$

The trace correspondence between the contract automaton $\mathcal{A}$ and the corresponding activity network $\mathcal{N}$ is proved below. Firstly, we need to define the notion of bisimulation between $\mathcal{A}$ and $\mathcal{N}$.

Definition 6. *Let* $\mathcal{A}$ *be a CA of rank n and* $\mathcal{N}$ *be an AN. We say that* $\mathcal{A}$ *and* $\mathcal{N}$ *are bisimilar, denoted* $\mathcal{A} \sim \mathcal{N}$ *iff there exists a binary relation* $B \subseteq Q \times M_P$ *such that* $(\vec{q}_0, \mu_0) \in B$ *and for any* $(\vec{q}, \mu) \in B$ *the following holds:*

1. $\forall \vec{q} \xrightarrow{\vec{a}} \vec{q}'$ *there exists* $\mu \xrightarrow{[\![(\vec{q}, \vec{a}, \vec{q}')]\!]}{}^{*} \mu'$ *s.t.* $(\vec{q}', \mu') \in B$, *and*
2. $\forall \mu \xrightarrow{a_{t1}a_{t2}}{}^{*} \mu'$ *there exists* $\vec{q} \xrightarrow{\vec{a}} \vec{q}'$ *s.t.* $[\![(\vec{q}, \vec{a}, \vec{q}')]\!] = a_{t1}a_{t2}, (\vec{q}', \mu') \in B$

The correspondence between a strongly safe controller of $\mathcal{A}$ and the corresponding activity network $\mathcal{N}$ is given below.

Lemma 2. *Let* $\mathcal{KS}_{\mathcal{A}}$ *be a (strongly safe) controller of* $\mathcal{A}$ *and* $\mathcal{N} = [\![\mathcal{KS}_{\mathcal{A}}]\!]$ *be the corresponding activity network, then:* $\mathcal{KS}_{\mathcal{A}} \sim \mathcal{N}$.

Example 2. In Section 2 we noted that the $\mathcal{KS}_{\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}}$ over-approximates the behaviour of the analysed system, and so does $[\![\mathcal{KS}_{\mathtt{H}_1 \otimes \mathtt{H}_2 \otimes \mathtt{Q}}]\!]$ by Lemma 2.

An important consequence of Lemma 2 is that an activity network computed through the mapping in Definition 4 is *convergent*.

Theorem 1. *Let $\mathcal{KS}_{\mathcal{A}}$ be a (strongly safe) controller of the CA $\mathcal{A}$, and $\mathcal{N} = [\![\mathcal{KS}_{\mathcal{A}}]\!]$ be the corresponding activity network, then $\mathcal{N}$ is* convergent

A refinement relation between two AN is introduced below. The refined network $\mathcal{N}'$ will have stricter conditions on guards of input gates, a subset or the same activities of $\mathcal{N}$ and a subset or the same set of places. Moreover, the functions of gates will be equal to the former network, except for the new places. Intuitively, $\mathcal{N}'$ will admit fewer behaviours (i.e. firing of activities) than $\mathcal{N}$.

Definition 7. *Let $\mathcal{N} = \langle P, A, I, O, \gamma, \tau, \iota, 0\rangle$ and $\mathcal{N}' = \langle P', A', I', O', \gamma', \tau', \iota', 0'\rangle$ be two activity networks, then $\mathcal{N}'$* refines *$\mathcal{N}$, written $\mathcal{N}' \preceq \mathcal{N}$, iff*

- *$P' \subseteq P$; $A' \subseteq A$;*
- *$ig = (P,g,f) \in I_{AP}, \iota(ig) = a$ implies $ig' = (P',g',f') \in I_{AP'}, \iota'(ig') = a$ where g' implies g and $\forall \mu, p \in P'.f(\mu)(p) = f'(\mu)(p)$ (written $ig' \preceq ig$);*
- *$og = (P,f) \in O_{AP}, o(og) = a$ implies $og' = (P',f') \in O_{AP'}, o'(og') = a$ and $\forall \mu, p \in P'.f(\mu)(p) = f'(\mu)(p)$ (written $og' \preceq og$);*
- *$\forall a \in A'.\gamma(a) = \gamma'(a), \tau(a) = \tau'(a)$, $\forall ig \in IG(a), ig' \in IG'(a), ig' \preceq ig.\iota(ig) = \iota'(ig')$, $\forall og \in OG(a), og' \in OG'(a), og' \preceq og.o(og) = o'(og')$*
- *the initial marking μ'_0 of $\mathcal{N}'$ is s.t. $\forall p \in P'.\mu'_0(p) = \mu_0(p)$, and the final markings of $\mathcal{N}'$ are $\{\mu' \mid \mu$ final marking of $\mathcal{N} \wedge \forall p \in P'.\mu'(p) = \mu(p)\}$.*

Simulation between two AN is now introduced, and will be used in the following.

Definition 8. *Let $\mathcal{N}, \mathcal{N}'$ be two AN, then $\mathcal{N}$ simulates $\mathcal{N}'$, written $\mathcal{N}' \leq \mathcal{N}$ iff there exists a binary relation $R \subseteq M_{P'} \times M_P$ such that $(\mu'_0, \mu_0) \in R$ and for any $(\mu', \mu) \in R$ it holds: $\forall \mu' \xrightarrow{a} \mu'_1$ there exists $\mu \xrightarrow{a} \mu_1$ s.t. $(\mu'_1, \mu_1) \in R$.*

The next lemma shows that $\preceq$ does not introduce unwanted behaviours in $\mathcal{N}'$.

Lemma 3. *Let $\mathcal{N}$ be an AN and $\mathcal{N}'$ be s.t. $\mathcal{N}' \preceq \mathcal{N}$, then it holds $\mathcal{N}' \leq \mathcal{N}$.*

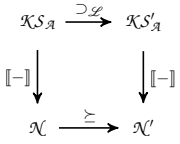
When refining networks it is possible to introduce deadlocks in the system, by disabling or removing all activities in a reachable marking. This could happen if, for example, we refine all predicates of input gates as $g' = g \wedge false$.
The following theorem uses a correspondence with the former mpc from which $\mathcal{N}$ was obtained and implies convergence of a refined network $\mathcal{N}' \preceq \mathcal{N}$.

Theorem 2. *Let $\mathcal{KS}_{\mathcal{A}}, \mathcal{KS}'_{\mathcal{A}}$ be respectively the mpc of $\mathcal{A}$ and the controller from Lemma 1. Moreover let $\mathcal{N} = [\![\mathcal{KS}_{\mathcal{A}}]\!]$, $\mathcal{N}' = [\![\mathcal{KS}'_{\mathcal{A}}]\!]$, then $\mathcal{N}' \preceq \mathcal{N}$.*

Example 3. The network $[\![\mathcal{KS}']\!] = \mathcal{N}'$ is, by Theorem 2, a refinement of $\mathcal{N}$, and is *convergent* by Theorem 1.

A commutative diagram explaining Theorem 2 is depicted below. Theorem 2 yields two procedures for generating a verified system that amounts to refine either the AN or the mpc. Both procedures start by (i) modelling the system as a composition of principals $\mathcal{A}$ in input, and (ii) compute the mpc $\mathcal{KS}_{\mathcal{A}}$. The first procedure `p1` refines the controller $\mathcal{KS}_{\mathcal{A}}$ to a controller $\mathcal{KS}'_{\mathcal{A}}$ (`p1-i`) and translates it to a network $\mathcal{N}' = [\![\mathcal{KS}'_{\mathcal{A}}]\!]$ (`p1-ii`). Alternatively, the second procedure `p2` generates the corresponding AN $[\![\mathcal{KS}_{\mathcal{A}}]\!] = \mathcal{N}$ (`p2-i`), and refines $\mathcal{N}$ to a network $\mathcal{N}''$ with stricter guards on the input gates (`p2-ii`).

While p1 yields a convergent network by Theorem 2, p2 does it only if ($\mathcal{N}' \sim \mathcal{N}''$) holds, that is both refinements of $\mathcal{N}$ and $\mathcal{KS}_{\mathcal{A}}$ result in removing the same unwanted behaviours. Note that in general $\mathcal{N}' \neq \mathcal{N}''$ could hold, and we remark that the AN refinement is an optional step.



6 Stochastic Continuous Aspects

The stochastic continuous behaviour related to the temperatures of the rail road track to allow the analysis of quantitative properties is now considered. As depicted in Figure 1, we will use the SAN formalism. Generally, for modelling CPS we need to add stochastic hybrid behaviour to the convergent network obtained either with p1 or p2. This can be obtained by extending the AN to a SAN model, as defined below. In the following let AN $\mathcal{N}'$ be a sub-network of $\mathcal{N}''$ only if $\mathcal{N}''$ contains all places and activities of $\mathcal{N}'$.

Definition 9. *Let $\mathcal{N}, \mathcal{N}', \mathcal{N}''$ be AN s.t. $\mathcal{N}' \preceq \mathcal{N}$ and $\mathcal{N}'$ is a sub-net of $\mathcal{N}''$, then given C, F, G the SAN $\mathcal{S} = \langle \mathcal{N}'', C, F, G \rangle$ is a* decorated $\mathcal{N}$*, written $\mathcal{S} \leftrightharpoons \mathcal{N}$.*

Example 4. The AN $\mathcal{N}$ will be decorated to a SAN $\mathcal{S}$ describing all stochastic continuous information related to the quantities that we want to evaluate, which in this case are the temperature of the rail road track and the weather conditions. In particular, an extended place $Temperature_i$ (i.e. the marking is a real number) describing the physical temperature of the rail road track is shared with the corresponding network $\mathtt{H}_i$. Moreover a stochastic process modelling the weather conditions and a differential equation modelling the physical evolution of temperature through induction heating are added to the SAN model $\mathcal{S}$ (see [4] for technical details). This decoration is such that $\mathcal{S}$ preserves all the logic described in $\mathcal{KS}'$, and can be used to quantitatively analyse the measures of interest. The guards g of input gates of activities $\mathtt{off\_ins_i}$ are refined as $g' = g \wedge (Temperature_i < T_{wa})$. The guards g of input gates of activities $\mathtt{off\_rem_i}$ are refined as $g' = g \wedge (Temperature_i > T_{wo})$. All the guards g' imply the corresponding g, and the network *Cyber Module* is a correct refinement of $\mathcal{N}'$ and the corresponding SAN is a correct decoration.
Now each guard depends also on the (continuous stochastic) marking of place $Temperature_i$, and generally the model checking problem for complex stochastic hybrid systems is undecidable [16]. Nevertheless, through our methodology it is possible to guarantee: (1) by Lemma 3 that $\mathcal{N}'$ simulates the synthesised controller, i.e. no unwanted behaviour is introduced by the refinement; (2) by assuming that $(Temperature_i < T_{wa})$ and $(Temperature_i > T_{wo})$ eventually hold, the network $\mathcal{N}'$ such that $\mathcal{S} \leftrightharpoons \mathcal{N}'$ is convergent. We remark that if the above assumptions are not verified then the switches will never fail and will never waste energy. Hence, the overall behaviour of the system is guaranteed to be safe. If, for example, a higher consumption of energy is detected, then it is formally proved that this is not due to a wrong interaction between a heater asking to be deactivated and the central control unit not receiving the request, but can only be related to the physical parameters instantiated in $\mathcal{S}$ (e.g. too high T_{wo}).

Note that Definition 7 does not pose any restriction on places used in gates of the refined network. Indeed, in the extension the guards of $\mathcal{N}'$ could use newly added

places. For a correct design, the new places and activities of $\mathcal{S}$ should model the stochastic hybrid behaviour, while all the discrete (verified) behaviours should be defined in the "embedded" network $\mathcal{N}'$.

7 Related Work and Conclusion

Related Work Several approaches for the verification and validation of stochastic hybrid models have been proposed in the literature. In particular, model checking [12] is a widely-used and powerful approach for the verification of finite state systems. However, the continuous stochastic nature of CPS is not always captured by finite state systems, and models as hybrid automata [15], hybrid Petri net [14], stochastic activity network [20] 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 [17], Kronos [21], Möbius [11]. When the continuous time behaviours of CPS are subject to complex and stochastic dynamics, the model checking problem is undecidable [16], and generally an approximation to more tractable models, as for example timed automata [2], is performed.
Statistical Model Checking (SMC) [19] uses results from statistics on top of simulations of a system to decide whether a given property is satisfied with some degree of confidence, and it represents a valid alternative to probabilistic model checking and testing, especially in the case of undecidability. UPPAAL-SMC [13] has been proposed as a tool that implements the above techniques. Compared to these previous works, we propose a hybrid qualitative and quantitative framework for analysing both critical quantitative properties and qualitative measures related to performance and dependability parameters, which is based on a formal relation between different formalisms to account for these analyses.
Conclusion We have proposed a modular approach to efficiently design and verify models of cyber-physical systems. These models are thought of as composed of a cyber (discrete) and a physical (stochastic continuous) part, that are modelled through different formalisms: (1) contract automata models for the cyber module and (2) stochastic activity networks for the physical module. A correct mapping from CA to SAN has been formalised, where CA are firstly mapped to AN and then decorated to SAN. Refinement relations from abstract to more concrete representations have been defined for all these formalisms, while retaining the correctness of the mapping. The proposed methodology has been applied to a realistic case study from the railway industry: a system of rail road switch heaters. This case study has been analysed in [5] to evaluate indicators of reliability and energy consumption. In this paper critical aspects related to the interactions of components implementing a policy of energy consumption have been verified through our methodology, and the corresponding SAN models have been automatically synthesised with further guarantees on the correctness of the control.
We are planning to implement our methodology as a toolchain by using existing tools (e.g. Möbius tool, CAT [7]), for generating correct SAN models starting from CA descriptions of components interactions, and to extend the approach to consider other existing formalisms and techniques [9]. [3]

[3] Acknowledgements: This work has been partially supported by the Tuscany Region project POR FESR 2014-2020 SISTER and H2020 2017-2019 S2R-OC-IP2-01-2017 ASTRail.

References

1. Abdollahi, M., Azgomi, Movaghar, A.: A modeling tool for a new definition of stochastic activity networks. IJST, Trans. B 29, 79–92 (2005)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183 – 235 (1994)
3. Balbo, G.: Introduction to generalized stochastic petri nets. In: Bernardo, M., Hillston, J. (eds.) FMPE, LNCS, vol. 4486. Springer (2007)
4. Basile, D., Di Giandomenico, F., Gnesi, S.: A Refinement Approach to Analyse Critical Cyber-Physical Systems: Extended Version. Tech. Rep. 2017-TR-005, ISTI–CNR (2017), http://puma.isti.cnr.it/rmydownload.php?filename=cnr.isti/cnr.isti/2017-TR-005/2017-TR-005.pdf
5. Basile, D., Chiaradonna, S., Di Giandomenico, F., Gnesi, S.: A stochastic model-based approach to analyse reliable energy-saving rail road switch heating systems. JRTPM 6(2), 163 – 181 (2016)
6. Basile, D., Degano, P., Ferrari, G.L.: Automata for Specifying and Orchestrating Service Contracts. LMCS Volume 12, Issue 4 (Dec 2016)
7. Basile, D., Degano, P., Ferrari, G.L., Tuosto, E.: Playing with our cat and communication-centric applications. In: FORTE 2016. pp. 62–73. Springer
8. Basile, D., Di Giandomenico, F., Gnesi, S.: Enhancing models correctness through formal verification: a case study from the railway domain. In: Amaretto, Modelsward 2017
9. Basile, D., Di Giandomenico, F., Gnesi, S.: Statistical model checking of an energy-saving cyber-physical system in the railway domain. In: SAC 2017
10. Bause, F., Kritzinger, P.S.: Stochastic petri nets: An introduction to the theory. SIGMETRICS Perform. Eval. Rev. 26(2) (1996)
11. Clark, G., Courtney, T., Daly, D., Deavours, D., Derisavi, S., Doyle, J.M., Sanders, W.H., Webster, P.: The möbius modeling tool. In: PNPM (2001)
12. Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge, MA, USA (1999)
13. David, A., Larsen, K.G., Legay, A., Mikuăionis, M., Poulsen, D.B.: Uppaal smc tutorial. Int. J. Softw. Tools Technol. Transf. 17 (2015)
14. David, R., Alla, H.: On hybrid petri nets. Discrete Event Dynamic Systems 11(1-2), 9–40 (Jan 2001)
15. Henzinger, T.A.: The theory of hybrid automata. pp. 278–. LICS '96, IEEE Computer Society (1996)
16. Henzinger, T.A., Ho, P.: Algorithmic analysis of nonlinear hybrid systems. In: CAV (1995)
17. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. Int. Journal on Software Tools for Technology Transfer 1 (1997)
18. Lee, E.A.: Cyber physical systems: Design challenges. ISORC '08, IEEE Computer Society (2008)
19. Legay, A., Delahaye, B., Bensalem, S.: RV 2010. Proceedings, chap. Statistical Model Checking: An Overview. Springer (2010)
20. Sanders, W.H., Meyer, J.F.: Stochastic activity networks: Formal definitions and concepts. In: FMPA (2000)
21. Yovine, S.: Kronos: A verification tool for real-time systems. (kronos user's manual release 2.2). JSTTT 1, 123–133 (1997)