Survivability evaluation of fluid critical infrastructures using hybrid Petri nets

Hamed Ghasemieh, Anne Remke and Boudewijn R. Haverkort
Design and Analysis of Communication Systems,
University of Twente, Netherlands
{h.ghasemieh, a.k.i.remke, b.r.h.m.haverkort}@utwente.nl

***Abstract*—In this paper we propose a formal, model-checking based procedure to evaluate the survivability of fluid critical infrastructures. To do so, we introduce the Stochastic Time Logic (STL), which allows to precisely express intricate state-based and until-based properties for an important class of hybrid Petri nets. We present an efficient model checking procedure which recursively traverses the underlying state-space of the hybrid Petri net model, and identifies those regions (subsets of the discrete-continuous state space) that satisfy STL formulae. A case study studying the survivability of a water refinery and distribution plant shows the feasibility of our approach.**

I. Introduction

Over the last 10-15 years, governments around the world have increased their efforts to ensure dependable, safe and secure operation of their national critical infrastructures, including the energy, water and gas distribution and supply to citizens and professional users [1], [2]. To ensure a continuous operation of critical infrastructures, a model-based evaluation of these systems' dependability [3] is of great use, as it allows to evaluate a wide variety of "what-if?" scenarios. We experienced that the evaluation of so-called survivability properties is most useful to operators of critical infrastructures, as the the notion of survivability comes close to typical questions put forward by practitioners [4]. Although multiple definitions are around, we view the survivability of a system as the probability that it recovers within a predefined amount of time to a predefined level of service, after the occurrence of some disaster [5], [6], [7]. We restrict ourselves in this sense to so-called GOOD models, for Given-Occurrence-Of-Disaster. This means that we do not model the failure process itself, but focus on the effect of the failure once occurred, on the behaviour of the system.

In what follows we focus on so-called fluid infrastructures, such as they appear in the water, gas and oil sectors. These infrastructural systems have as distinguishing feature that they encompass both continuous state components (levels of fluid, etc), as well as discrete state components (the states of discrete control components), hence, so-called hybrid models appear most applicable. Recently, hybrid Petri net models with a single general one-shot transition (HPnGs) have been introduced for modeling fluid critical infrastructures such as water refinery plants [8] where the general one-shot transition models stochastic failures or repair processes. HPnGs allow for a combination of discrete and fluid state variables, and four different types of state transitions (immediate, deterministically timed, generally distributed timed and fluid-style continuous), and allow for an efficient evaluation of a variety of interesting system properties, e.g., the probability of fluid reservoirs to become empty before certain events have happened. Following [8], in [9] a much more efficient algorithm to evaluate the state probabilities for such HPnGs has been proposed. The crux of that algorithm is to decompose the continuous part of the state space in convex *regions*, such that within such a region, the discrete part of the state space does not change. Using that algorithm, simple (transient) state-based system properties can be evaluated.

However, in order to formalize the notion of survivability a time-bounded until operator is needed. Hence, this paper adds a CSL-style [10] Until operator to the previously introduced state-based logic in [9], which together is called Stochastic Time Logic (STL) in the following. In this paper, we introduce syntax and semantics of STL, as well as a model checking procedure to check the until operator, hence, to evaluate system survivability properties. Our new algorithm makes extensive use of computational geometry and recursively traverses all regions in the underlying state space of the HPnG and identifies those parts of regions that satisfy sub-formulae.

The contribution of this paper lies in taking into account the evolution of the system over time which is needed for model checking the until operator, whereas previous work could only analyze state-based properties (probabilities). The logic we propose is similar to MITL [11] and STL in [12], however, instead of a signal we check nested properties that are based on discrete and continuous atomic properties that reflect the marking of the discrete and continuous places of the hybrid Petri net, respectively. Using the until operator, as introduced for HPnGs in this paper, we are not restricted to the evaluation of simple reachability properties. Instead, our approach allows us to compute so-called *reach-avoid* properties, using the until operator.

Related work. The class of hybrid Petri nets, as considered in this paper constitutes a sub-class of Stochastic Hybrid Models, where the continuous evolution is linear and determinsitic. Due to rate-adaption which takes place when a fluid place is at its upper or lower boundary, the continuous behaviour is however not fully controlled by the discrete part of the system.

The state representation which is used to compute measures of interest resembles related work in non-stochastic linear hybrid automata [13]. For an overview on the related work with respect to the state representation, we refer to [9]. For model checking linear hybrid systems different approaches exist, such as linear abstraction, used in HyTech/PHAVer [14],

[15] and flow pipe construction, as used, e.g., in SpaceEx [16]. However, our work differs from these approaches, since we partition the state-space according to time and the support of the general distribution instead of the values of the continuous variables. Hence, regions, as presented in [9] do not correspond to modes in hybrid automata (as used there). Due to the partitioning of the state-space into convex regions with the same characteristics, we are able to provide exact results for time-bounded reachability within few seconds.

This considerably differs from related work on Stochastic Hybrid systems (with deterministic continuous behaviour), where mostly over-approximation [17], [18], [19] is used to compute results. The practical applicability of these approaches, however, seems to be limited to special cases. Another option is to use Monte-Carlo simulations, as done in [20], [21].

For model checking Fluid Stochastic Petri nets [22] two comments are in place. First, the model class is different, since their analyzability is restricted to only a few fluid places. Second, we are not aware of a unifying approach that incorporates both discrete and continuous properties. Model checking (discrete) probabilistic hybrid automata has been shown to be decidable for probabilistic temporal logics [23].

This paper is further organized as follows. Section II summarizes the definition of our class of hybrid Petri nets, and revisits the partitioning of the state-space into the underlying regions. The stochastic time logic STL (syntax and semantics) is introduced in Section III, whereas Section IV explains the model checking procedure in detail. A case study addressing the survivability of a water refinery and delivery plant shows the feasibility of our approach in Section V. The paper is concluded in Section VI.

II. Model definition

Section II-A defines the syntax of an HPnG and indicates its evolution. Section II-B explains how the underlying state-space of an HPnG is partitioned into regions with the same characteristics.

A. Hybrid Petri-nets with general one-shot transition

An HPnG is defined as a tuple $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathbf{m}_0, \mathbf{x}_0, \Phi)$, where $\mathcal{P} = \mathcal{P}^D \cup \mathcal{P}^C$ is a set of *places* that can be divided into two disjoint sets $\mathcal{P}^D$ and $\mathcal{P}^C$ for the discrete and continuous places, respectively. The discrete marking $\mathbf{m}$ is a vector that represents the number of tokens $m_P \in \mathbf{N}$ for each discrete place $P \in \mathcal{P}^D$ and the continuous marking $\mathbf{x}$ is a vector that represents the non-negative level of fluid $x_P \in \mathbf{R}_0^+$ for each continuous place $P \in \mathcal{P}$. The initial marking is given by $(\mathbf{m}_0, \mathbf{x}_0)$. Four types of *transitions* are possible, as follows. The set of immediate transitions, the set of deterministically timed transitions, the set of general transitions, and the set of continuous transitions together form the finite set of transitions $\mathcal{T} = \mathcal{T}^I \cup \mathcal{T}^D \cup \mathcal{T}^G \cup \mathcal{T}^C$. Note that in this paper the number of general transitions is restricted to $|\mathcal{T}^G| = 1$. Also the set of *arcs* $\mathcal{A}$ consists of four sets: The set of discrete input and output arcs $\mathcal{A}^D$, connects discrete places and discrete transitions and the set of continuous input and output arcs $\mathcal{A}^C$ connects continuous places and continuous transitions. The set of inhibitor arcs $\mathcal{A}^I$ and the set of test arcs $\mathcal{A}^T$, both connect discrete places to all kinds of transitions and ensure that the transition is only enabled in case the discrete place that is connected via an inhibitor arc does not contain any token or in the case of a test arc, the transition is only enabled in case there are as many tokens in the connected place, as defined by the test arc.

The tuple $\Phi = (\phi_b^{\mathcal{P}}, \phi_p^{\mathcal{T}}, \phi_d^{\mathcal{T}}, \phi_f^{\mathcal{T}}, \phi_g, \phi_w^{\mathcal{A}}, \phi_s^{\mathcal{A}}, \phi_p^{\mathcal{A}})$ contains 8 *functions*. Function $\phi_b^{\mathcal{P}} : \mathcal{P}^C \to \mathbf{R}^+ \cup \infty$ assigns an upper bound to each continuous place. In contrast to the definition of HPnG in [8] in the following $\phi_p^{\mathcal{T}} : \mathcal{T}^D \cup \mathcal{T}^I \to \mathbf{N}$ specifies a *unique priority* to each immediate and deterministic transition to resolve firing conflicts, as in [24]. Deterministic transitions have a constant firing time defined by $\phi_d^{\mathcal{T}} : \mathcal{T}^D \to \mathbf{R}^+$ and continuous transitions have a constant nominal flow rate defined by $\phi_f^{\mathcal{T}} : \mathcal{T}^C \to \mathbf{R}^+$. The general transition is associated with a random variable s, representing its firing time, with a cumulative probability distribution function (CDF) $\phi_g(s)$, and its probability density function (PDF) is denoted $g(s)$. We assign to all arcs except continuous arcs the weight: $\phi_w^{\mathcal{A}} : \mathcal{A} \setminus \mathcal{A}^C \to \mathbf{N}$ which defines the amount of tokens that is taken from or added to connected places upon firing of the transition.

Graphical representation. The primitives of the hybrid Petri net formalism with general one-shot transitions are shown in Figure 1. A discrete place is graphically represented by a single circle and a fluid place is represented by two concentric circles. A general transition is drawn as an empty rectangle, a deterministic transition is drawn as a grey rectangle, a fluid transitions shown as an empty rectangle with double lines and an immediate transition is a thick black bar. The discrete input and output arcs are drawn as single arrows and fluid input and output arcs are represented with double lines. Inhibitor arcs are drawn with a small circle toward the transition and test arcs are drawn with two triangular arrowheads.

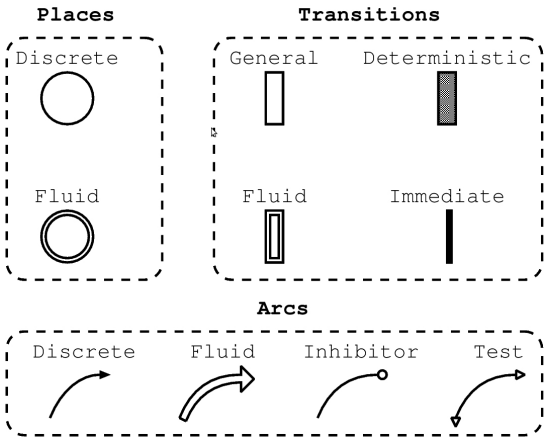

Figure 1. Graphical representation of primitives of HPnG.

System evolution. Markings are collected into two vectors, the discrete marking $\mathbf{m} = (m_1, \ldots, m_{n_d})$ and the continuous marking $\mathbf{x} = (x_1, \ldots, x_{n_c})$. The initial marking is composed of a discrete part $\mathbf{m}_0$ that describes the initial amount of tokens in all discrete places and a continuous part $\mathbf{x}_0$ that describes the initial amount of fluid in all continuous places.

The *state* of an HPnG is defined by $\Gamma = (\mathbf{m}, \mathbf{x}, \mathbf{c}, \mathbf{d}, \mathcal{G})$, where vector $\mathbf{c} = (c_1, \ldots, c_{|\mathcal{T}^D|})$ contains a clock c_i for

each deterministic transition that represents the time that T_i^D has been enabled. When a transition is disabled the clocks do not evolve, but the clock value is preserved until the transition is enabled again and clocks are only reset, when the corresponding deterministic transition fires. If the general transition has not fired yet, it can be considered as a deterministic transition, whose firing time is sampled from the corresponding general firing time distribution. This sampling happens only once per model execution, and it occurs when the general transition becomes enabled for the first time. Vector $\mathbf{d} = (d_1, \ldots, d_{|\mathcal{P}^c|})$ indicates the drift, i.e., the change of fluid per time unit for each continuous place. Note that even though the vector d is determined uniquely by x and m, it is included in the definition of a state to make it more descriptive. The general transition is only allowed to fire once, hence, the flag $\mathcal{G} \in \{0, 1\}$ indicates whether the general transition has already fired ($\mathcal{G} = 1$), or not ($\mathcal{G} = 0$). So, the initial state of the system is $\Gamma_0 = (\mathbf{m}_0, \mathbf{x}_0, \mathbf{0}, \mathbf{d}_0, 0)$. A system state can be seen as a snapshot of the system evolution at a specific time, and assumed general transition firing time; this is elaborated in more detail in the next section.

The firing rules of deterministic, general, immediate and fluid transitions differ. Whether a transition is allowed to fire depends (1) on the structure and the current marking of the Petri net (*concession*) and (2) on the type of the transitions.

Fluid transitions that have concession, are always enabled, and continuously transport fluid along fluid arcs. Conflicts in the distribution of fluid occur when a continuous place reaches one of its boundaries. To prevent overflow, the fluid input has to be reduced to match the output, and to prevent underflow the fluid output has to be reduced to match the input, respectively. The firing rate of fluid transitions is then adapted according to the share $\phi_s^{\mathcal{A}} : \mathcal{A}^C \to \mathbf{R}^+$ and priority $\phi_p^{\mathcal{A}} : \mathcal{A}^C \to \mathbf{N}$ that is assigned to the continuous arcs that connect the transition to the place. This is done by distributing the available fluid over all continuous arcs. Those with highest priority are considered first and if there is enough fluid available, all transitions with the highest priority can still fire at their nominal speed. Otherwise, their fluid rates are adapted according to the firing rate of the connected transitions and the share of the arc, according to [25]. The adaptation of fluid rates in these cases, results in a piecewise constant fluid derivative per continuous place.

Non-fluid transitions that have concession may be enabled, depending on their type. If an immediate transition has concession the marking is said to be *vanishing* otherwise the marking is said to be *tangible*. Immediate transitions have precedence above deterministic and general transitions. In a vanishing marking deterministic and general transitions are disabled and cannot fire. The clock of each enabled deterministic transition T_i^D evolves with time at rate $dc_i/d\tau = 1$ and when a clock reaches its firing time, i.e., $c_i = \phi_d^{\mathcal{T}}(T_i)$ transition T_i^D fires. Similarly, the enabling time of the enabled general transition, that has not fired yet, evolves with time at rate 1. The general transition then fires with probability $\phi_g(\tau + \Delta\tau) - \phi_g(\tau) = \int_\tau^{\tau+\Delta\tau} g(s)ds$ in any time interval $[\tau, \tau + \Delta\tau]$.

Whenever a non-fluid transition fires the marking evolves according to a firing rule, depending on the type of the transition. All discrete transition types, i.e. immediate, deterministic and general, change the discrete part of the marking $\mathbf{m}$ in a similar way. For a more detailed description of HPnGs and their evolution, we refer to [26].

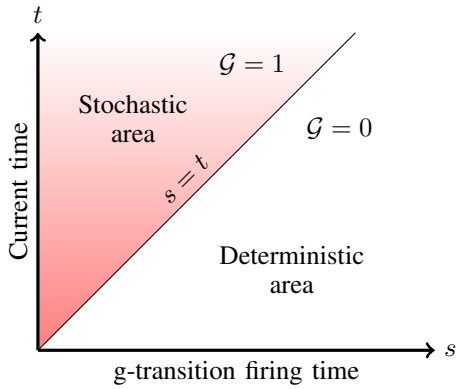

Figure 2. Generic presentation of STD.

B. Stochastic time diagram

The Stochastic time diagram (STD) introduced in [9], provides a genuine way of representing the evolution of a HPnG for a given initial state. The main reasoning behind this is that, for an initial state of an HPnG and a predefined value for the firing time of the general transition, denoted s, for all the future time instances t we can determine the state of the system. The STD is a two-dimensional diagram with t and s as its vertical and horizontal axis, respectively. Each point in this diagram is associated with a unique HPnG state, which is denoted by $\Gamma(s, t)$. A generic version of this diagram is shown in Figure 2. Two main areas can be distinguished in this figure. The area above the line $t = s$, called *stochastic area*, contains all the HPnG states for which the general transition has fired ($t > s$), whereas the area below the line $t = s$, called *deterministic area*, includes those states for which the general transition did not yet fire ($t < s$). To compute measures of interest for HPnGs, the STD needs to be deconditioned with the probability density function $g(s)$.

The main idea behind the proposed method in [9], is that instead of dealing with infinitely many points in the ts-plane, we can partition it into several *regions*. These regions exist, because the state of the system does not change until an *event* occurs. In each system state two types of potential events can occur: a continuous place reaching its lower/upper boundary or an enabled transition, either deterministic or general fires. The former imposes a change in the drift of the continuous place, due to rate adaptation [25], while the latter event type alters the discrete marking and the general transition flag. Hence, an event is a change in the discrete marking, a change in drift or a change in a general transition flag. We define a region as a set of states that while the system remains in them no event occurs, and by the occurrence of an event the system enters another region. This leads to the following definition.

Definition 1. *A region $\mathcal{R}$ is set of (s, t) points in a given STD,*

for which we have:

$$\forall (s_1,t_1),(s_2,t_2) \in \mathcal{R}: \begin{cases} \Gamma(s_1,t_1).\mathbf{m} & = \Gamma(s_2,t_2).\mathbf{m}, \\ \Gamma(s_1,t_1).\mathbf{d} & = \Gamma(s_2,t_2).\mathbf{d}, \\ \Gamma(s_1,t_1).\mathcal{G} & = \Gamma(s_2,t_2).\mathcal{G}. \end{cases}$$

Note that, while $\Gamma(s,t).\mathbf{m}$ is used to refer to the vector of discrete markings, $\Gamma(s,t).m_P$ is used to refer to the discrete marking of a specific place P. A similar notation is used for the continuous marking.

A possible partitioning of the state space into regions is shown in Figure 3 together with probability density function $g(s)$. Note that the shape of these regions depends on the structure of the model at hand. In [9] it is shown that, inside a region all continuous variables, i.e., the amount of fluid and the clock valuations, can be represented by simple linear equations in s and t. Intuitively, this is because in a region all continuous places are associated with a constant drift and clocks also have a constant drift (of one). Using this we infer that the boundaries between regions, which represent the occurrence of an event, are characterized by linear functions of s and t. Hence, each region in the STD is a polygon. The unique line segment corresponding to the event, causing the system to enter this region, is called the *underlying segment* of the region. In Figure 5, a generic region is shown and its underlying segment is depicted by l_u. To compute the probability to be in a specific system state at time τ, it suffices to find all regions intersecting the horizontal line $t = \tau$ that correspond to the specific system state and integrate $g(s)$ over the intersection. This idea is illustrated for a given partitioning in Figure 3.

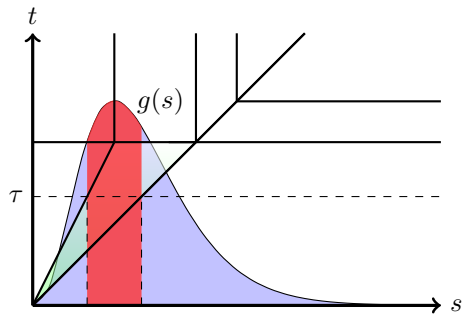

Figure 3. Deconditioning according to the probability density function $g(s)$.

Even though reachability computations on the STD are always performed for a given and finite time bound, there is still a possibility of having an infinite number of regions in the STD before the finite time bound. This happens whenever an infinite sequence of vanishing markings occurs. This problem is well-known for all Petri nets formalism that allow immediate transitions. However, if we require that models have to be bounded, infinite sequences of vanishing markings can only take place in the form of cycles of vanishing markings, which, however can be detected and removed. This ensures that we always reach a tangible marking in a finite number of steps and the number of regions in the STD before a finite time bound is also finite. Hence, for a bounded model and a finite time bound the algorithm always terminates.

III. Stochastic Time Logic

In [9] an algorithm has been presented to compute the probability that a certain (state-based) property holds at a specific time instance. However, to evaluate also path-based properties, such as (time-bounded) reachability, i.e., whether a certain property *becomes* valid within a certain amount of time, we enhance the logic presented in [9] by adding an *until* operator. The resulting logic is called Stochastic Time Logic (STL); it is used to reason about the underlying state space of a HPnG, namely the STD as described in Section II-B. Recall that an STD explicitly takes into account all possible values of s and t, hence, with the proposed logic it is possible to reason whether an STL formula holds for a certain system state $\Gamma(s,t)$. However, note that STL reasons on the deconditioned state-space of a HPnG, i.e., on the regions of an STD, which does *not* yet take into account the distribution of the general transition. To lift this to the level of an HPnG we introduce a probability operator that explicitly takes into account the density function of the general transition.

Definition 2 (Stochastic Time Logic)**.** *An* STL *formula* Ψ *is defined as*

$$\Psi := \mathtt{tt} \mid x_P \geq c \mid m_P = a \mid \neg\Psi \mid \Psi \wedge \Psi \mid \Psi \; \mathcal{U}^{[T_1,T_2)} \; \Psi,$$

where $T_1, T_2 \in \mathbb{R}^+$, $x \geq c$ *and* $m = a$, *with* $a \in \mathbb{N}, c \in \mathbb{R}^+$, *are called continuous and discrete atomic properties, respectively.*

Note that, although the above definition allows nested until formula, for this paper we only consider non-nested until formula.

In the following we define two different satisfaction relations, namely (i) $\models^{s,t}$ between a single system state $\Gamma(s,t)$ and an STL formula Ψ, and (ii) $\models^{t}$ between an interval on the support of the general transition and an STL formula Ψ. The different indices on the satisfaction relation are used to stress their dependencies on s and t.

The satisfaction relation $\models^{s,t}$ takes into account a single point in the STD, which corresponds to a single system state. The satisfaction relation is defined as follows:

Definition 3 (Satisfaction on system states)**.**

$$\begin{aligned}
&\Gamma(s,t) \models^{s,t} \mathtt{tt} && \forall t, s, \\
&\Gamma(s,t) \models^{s,t} m_P = a && \textit{iff } \Gamma(s,t).m_P = a, \\
&\Gamma(s,t) \models^{s,t} x_P \geq c && \textit{iff } \Gamma(s,t).x_P \geq c, \\
&\Gamma(s,t) \models^{s,t} \neg\Psi && \textit{iff } \Gamma(s,t) \not\models^{s,t} \Psi, \\
&\Gamma(s,t) \models^{s,t} \Psi_1 \wedge \Psi_2 && \textit{iff } \Gamma(s,t) \models^{s,t} \Psi_1 \wedge \Gamma(s,t) \models^{s,t} \Psi_2, \\
&\Gamma(s,t) \models^{s,t} \Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2 && \textit{iff } \exists \tau \in [t+T_1, t+T_2]: \\
& && \quad \Gamma(s,\tau) \models^{s,t} \Psi_2 \wedge (\forall \tau' \in [t,\tau]: \; \Gamma(\tau',s) \models^{s,t} \Psi_1).
\end{aligned}$$

For the STL until operator $\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2$ and a system state $\Gamma(s,t)$ we have to check, whether starting from time t and for a given sample s, the evolution of the system is such that a time point τ exists at which Ψ_2 holds and before which Ψ_1 holds. Note that for a given system state $\Gamma(s,t)$ the evolution over time is deterministic and coincides with a vertical line in the STD, starting at point (s,t). Hence the analysis of the STL until operator for a given system state boils down to checking, whether this line only intersects with regions where Ψ_1 holds

until a region is hit where Ψ_2 holds within the defined time interval.

As mentioned earlier STD combines system states with the same characteristics into regions. This can be exploited for checking sets of states instead of individual system states. Hence, we introduce a satisfaction relation $\models^t$ for intervals on the support of the distribution of the general transition, denoted $I_\psi \subseteq \mathbb{R}_{\geq 0}$, and STL formula Ψ, at time t, which allows for more efficient model checking procedures than checking each system state individually.

Definition 4 (Satisfaction on intervals)**.**

$$I_\psi \models^t \Psi \quad \textit{iff } \forall s \in I: \ \Gamma(s,t) \models^{s,t} \Psi.$$

Definition 5. *The set of satisfaction intervals $Sat^t(\Psi)$ is defined as the set of all intervals satisfying Ψ at time t, i.e., $Sat^t(\Psi) = \{I_\psi : I_\psi \models^t \Psi\}$.*

While the explicit dependency on s (or sets of s-values) is used for the efficient computation of properties, in the end we want to know whether a given STL formula holds at time t for the HPnG model of interest with a certain probability. Hence, we introduce a probability operator $\mathbb{P}_{\sim p}(\Psi)$ which is wrapped around an STL formula, where $p \in [0,1]$ is a real number and $\sim \in \{\leq, <, >, \geq\}$ a comparison operator. It abstracts from the possible values of s by deconditioning with the probability density function $g(s)$, as follows.

Definition 6. *Let $\Gamma(t) = \{\Gamma(s,t) | s > 0\}$ be the set of possible system states at time t, then the satisfaction relation for the probability operator $\mathbb{P}_{\sim p}$ is defined as:*

$$\Gamma(t) \models \mathbb{P}_{\sim p}(\Psi) \quad \textit{iff } Prob(\Psi, t) \sim p,$$

where

$$Prob(\Psi, t) = \sum_{I_\Psi \in Sat^t(\Psi)} \int_{I_\Psi} g(s) ds.$$

IV. Model checking Stochastic Time Logic

For model checking an HPnG with given initial state, we first have to generate the corresponding STD. Then the satisfaction set of intervals is formed over this STD. We emphasize that, due to the time-inhomogeneity of an HPnG, the validity of a certain formula depends on time. Model checking STL is based on computing the satisfaction sets of intervals as in Definition 5. Hence, the output of the model checking algorithms, as presented in this section, is the satisfaction set of intervals, which due to the time-inhomogeneity depend on the time we are checking the formula at, denoted t in the following.

Model checking STL formula without the until operator is briefly explained in [9]. Conjunctions and negations of atomic properties are state-based properties and independent of the further evolution of the system. Hence, we just need to identify those regions where the system can be in at time t, and find out in which part of those regions the given formula is satisfied. In contrast to state-based properties, the until operator is path-based and its validity depends on the future system evolution. We have to find those intervals of s-values for which the evolution of the system from a given starting time satisfies the formula.

This section is further organized as follows. Section IV-A introduces special notation for operations on sets of sets that is needed in the algorithms. Section IV-B explains the main idea of model checking the until operator and Section IV-C presents the model checking algorithms in detail.

A. Notation

The algorithm presented in this paper involves a special set of operations on sets of intervals and on sets of system states. Hence, we introduce a new *intersection* and *complement* operator between sets of sets.

Definition 7. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two sets of sets, the* intersection *operator between these two, and the* complement *operator of a set of sets, are defined as follows:*

$$\begin{aligned} S \in \mathcal{S}_1 \sqcap \mathcal{S}_2 \quad & \textit{iff } \exists S_1 \in \mathcal{S}_1, S_2 \in \mathcal{S}_2: \ S = S_1 \cap S_2, \\ S \in \neg\mathcal{S}_1 \quad & \textit{iff } \forall S' \in \mathcal{S}_1 : \\ & \quad S' \cap S = \emptyset \wedge \nexists S'' \in \neg\mathcal{S}_1 : S'' \subset S. \end{aligned}$$

According to the above definition we can derive the *union* and the *relative complement* operator as $\mathcal{S}_1 \sqcup \mathcal{S}_2 = \neg(\neg\mathcal{S}_1 \sqcap \neg\mathcal{S}_2)$ and $\mathcal{S}_1 \setminus \mathcal{S}_2 = \mathcal{S}_1 \sqcap \neg\mathcal{S}_2$, respectively. Intuitively, the union operator $\sqcup$, merges two sets of sets and ensures that the results are maximal and there is no duplication. Also the intersection operator $\sqcap$, computes the intersection of all the set members of two given sets of sets. For instance if we define two sets of intervals on real number as $\mathcal{S}_1 = \{[1,3],[4,5]\}$ and $\mathcal{S}_2 = \{[0,2],[3,5],[6,7]\}$, then we have, $\mathcal{S}_1 \sqcap \mathcal{S}_2 = \{[1,2],[4,5]\}$ and $\mathcal{S}_1 \sqcup \mathcal{S}_2 = \{[0,5],[6,7]\}$. These operations are also shown in Figure 4.

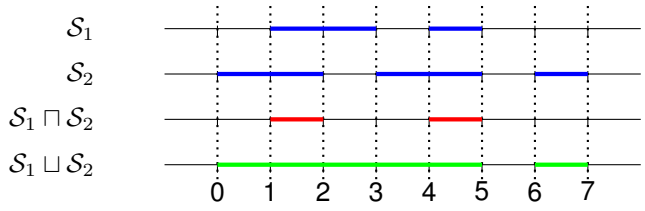

Figure 4. Presentation of intersection and union operators over sets of intervals.

Finally, since the result of the algorithm are sets of satisfaction intervals, we introduce a projection operator as follows:

Definition 8. *The operator Prj projects a set of points S in the ts-plane on the s-axis:*

$$Prj(S) = \{s | (s,t) \in S\}.$$

If the given set is a connected set, the result of the projection is an interval. If $\mathcal{S}$ is a set of connected sets its projection results in a set of intervals: $\text{Prj}(\mathcal{S}) = \bigsqcup_{S \in \mathcal{S}} \text{Prj}(S)$. This operator is used later in the model checking algorithms.

B. Model checking the until operator

Recall that the evolution of the system for a value of s, corresponds to a vertical line in STD. The main idea behind

checking formula $\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2$ is to find all intervals which satisfy Ψ_1 at the starting time. Then by moving upwards in the STD, we refine the initial intervals by omitting those parts which violate Ψ_1. We continue this until reaching either the time bound T_2 or a time point in $[T_1, T_2]$ for which Ψ_2 is satisfied. Due to the partitioning into regions of states with the main characteristics, this can be done efficiently by traversing the regions in the STD. In order to model check the bounded until operator at starting time t_0, for each possible firing time of the general transition, i.e., all possible values of s, we have to find a time point τ between $t_0 + T_1$ and $t_0 + T_2$, where Ψ_2 holds and before which Ψ_1 is not violated. Hence, the satisfaction set of intervals for a bounded until is defined as:

$$Sat^t(\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2) = \bigsqcup \{I \subseteq \mathbb{R}_{\geq 0} \mid \exists \tau \in [t_0 + T_1, t_0 + T_2] : I \models^{\tau} \Psi_2 \wedge \forall t' \in [t_0, \tau] I \models^{t'} \Psi_1\}.$$

The algorithm for model checking formula $\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2$ at a given time t_0, is based on recursively traversing all regions in the underlying state space of a HPnG, starting from time t_0. Depending on the position of each region, relative to time t_0 and the interval $[T_1, T_2]$, we investigate (i) what are the intervals in which Ψ_1 holds and (ii) whether Ψ_2 holds in this region. This is explained in more detail in the following.

According to the definition of regions, the value of discrete variables does not change within a region. Hence, a discrete atomic property either holds in the entire region or not at all. However, as stated earlier, each continuous variable can be represented by a linear equation in a region and the corresponding continuous property, say $x_P \geq c$, can be valid in only part of that region, where the line $x_P = c$ then is the boundary between the part where the property is valid, the so-called validity area, and the part where it is not valid.

Considering the linear boundaries of the region, the validity area of an atomic property in a region is a polygon. The validity area of the conjunction of two continuous atomic properties is the intersection of the corresponding areas, and the negation results in the complement of the validity area with respect to the considered region. Hence, in order to find the validity area of a nested formula, which only consists of negation and conjunctions, first we need to form the parse tree of the formula. Then, by repeated application of the respective operators the validity area is formed. Moreover, if we consider polygons as set of points, then any set theoretical operation between two polygons, results in a polygon. Therefore, the satisfaction area of formula Ψ is also a polygon. More formally, we denote the set of polygons in region $\mathcal{R}$, where formula Ψ is satisfied by $\mathcal{P}^{\mathcal{R}}(\Psi)$ and for the state-based STL operators they are defined as:

Definition 9. *The set of polygons within region $\mathcal{R}$, where a state-based STL formula holds is given by:*

$$\begin{aligned} \mathcal{P}^{\mathcal{R}}(m_P = a) &= \{(s,t) \in \mathcal{R} \mid \Gamma(s,t).m_P = a\}, \\ \mathcal{P}^{\mathcal{R}}(x_P \geq a) &= \{(s,t) \in \mathcal{R} \mid \Gamma(s,t).x_P \geq a\}, \\ \mathcal{P}^{\mathcal{R}}(\Psi \wedge \Psi) &= \mathcal{P}^{\mathcal{R}}(\Psi) \sqcap \mathcal{P}^{\mathcal{R}}(\Psi), \\ \mathcal{P}^{\mathcal{R}}(\neg \Psi) &= \mathcal{R} \setminus \mathcal{P}^{\mathcal{R}}(\Psi). \end{aligned}$$

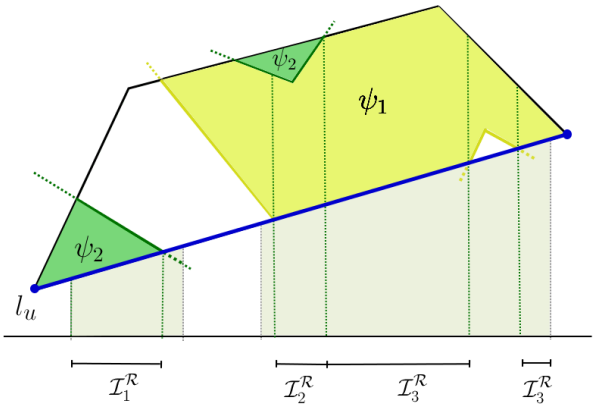

Figure 5. Determining intervals that satisfy formula $\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2$, in a generic region.

Note, that if Ψ is an atomic property, then $\mathcal{P}^{\mathcal{R}}(\Psi)$ is a single polygon. Figure 5 shows the validity area of properties Ψ_1 and Ψ_2 in a hypothetical region. As can be seen, the area corresponding to Ψ_1 is a non-convex polygon, and the area satisfying Ψ_2 consists of two polygons.

C. Algorithms for model checking the until operator

The procedure for model checking an until formula is presented in Algorithm 1. At first we determine all starting regions, i.e., all regions we can be in at a given time t. Then we iterate over all these regions and determine the intervals which satisfy the until formula, by calling the recursive function VISIT for each region, which is the core of the model checking procedure and is presented in Algorithm 2.

Algorithm 1 UNTIL$(\Psi_1, \Psi_2, [T_1, T_2], t)$

Require: Properties Ψ_1 and Ψ_2, time interval $[T_1, T_2]$, and time t.
Ensure: The satisfaction set $Sat^t(\Psi_1 \ \mathcal{U}^{[T_1,T_2]} \ \Psi_2)$.
1: $\mathcal{R}^t \leftarrow$ all possible regions we can be in, at time t
2: $Sat^t(\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2) \leftarrow \emptyset$
3: **for all** $\mathcal{R}_i \in \mathcal{R}^t$ **do**
4: $\quad Sat^t(\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2) \xleftarrow{\sqcup}$ VISIT$(\mathcal{R}_i, \Psi_1, \Psi_2, \{[0, \infty]\}, t, [t + T_1, t + T_2])$
5: **return** $Sat^t(\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2)$

Function VISIT takes as input the region to be visited, denoted $\mathcal{R}$, formulas Ψ_1 and Ψ_2, starting time t and interval $[T_1, T_2]$ of the until formula. Furthermore, it receives the set of intervals $\mathcal{I}_{\Psi_1}$, which satisfy Ψ_1 just before entering region $\mathcal{R}$. Hence, $\mathcal{I}_{\Psi_1}$ is the set of potential intervals which may satisfy the until formula. For the initial call of VISIT, this set contains the biggest interval in which the until formula may hold, which is $[0, \infty]$.

At first (lines 2-3) we form the set of polygons corresponding to Ψ_1 and Ψ_2, as defined in Definition 9. Then, for each region we have to find three sets of intervals: (i) ($\mathcal{I}_1^{\mathcal{R}}$), that is the set of intervals which satisfy the until formula upon entering the region; (ii) the intervals in ($\mathcal{I}_2^{\mathcal{R}}$) satisfy Ψ_1 and reach a Ψ_2 polygon in the region in time, therefore they also

satisfy the until formula; (iii) for the intervals in ($\mathcal{I}_3^{\mathcal{R}}$) Ψ_1 holds, but a Ψ_2 polygon cannot be reached in time within the region. These are the intervals for which the successive regions need to be considered to check whether they reach a Ψ_2 polygon in the future (but within the time bound). These three sets of intervals are illustrated in Figure 5.

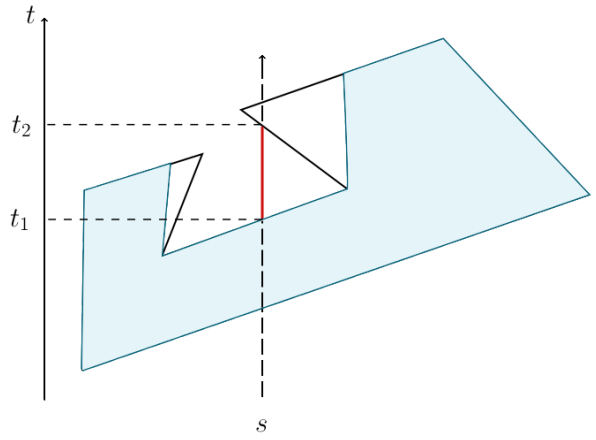

Figure 6. Reforming a polygon over a given interval.

As pointed out earlier, the position of a region w.r.t. the starting time t_0, and interval $[T_1, T_2]$ should be taken into account during the computation. If a region intersects the horizontal line $t = t_0$, only the part above this line needs to be considered. If a region is above the line $t = t_0$, but below the line $t = T_1$, according to the definition of the bounded until operator, we only need to consider those intervals, which satisfy Ψ_1. If a region intersects the line $t = T_1$, we have to split it into two parts. The part below this line only needs to be considered if Ψ_1 holds, and for the upper part we have to apply the general algorithm, as explained before. The same applies, if the region is between the lines $t = T_1$ and $t = T_2$. However for simplicity, Algorithm 2, only discusses the general case where the region is placed inside the interval $[T_1, T_2]$.

First set of intervals.: For finding the first interval set, $\mathcal{I}_1^{\mathcal{R}}$, we have to check if the polygons that form the validity area of Ψ_2 intersect the underlying segment of the region $\mathcal{R}$, denoted $\mathcal{R}.l_u$, and whether the intersection (or part of it) lies in one of the intervals that belongs to $\mathcal{I}_{\Psi_1}$. More formally, we denote the part of the underlying segment where Ψ_1 holds, i.e., that lies in $\mathcal{I}_{\Psi_1}$ as

$$\mathcal{R}.l_u.\mathcal{I}_{\Psi_1} = \{(s,t) \in \mathcal{R}.l_u \mid \exists I \in \mathcal{I}_{\Psi_1},\ s \in I\}.$$

Then, the first interval set is computed as the projection of the intersection of the validity area of Ψ_2 and the Ψ_1 part of the underlying segment:

$$\mathcal{I}_1^{\mathcal{R}} = \text{Prj}\left(\mathcal{P}^{\mathcal{R}}(\Psi_2) \sqcap \mathcal{R}.l_u.\mathcal{I}_{\Psi_1}\right).$$

This is done in lines 4-5 of Algorithm 2 and illustrated in the left part of Figure 5, where one of the Ψ_2 polygons intersects the underlying segment in one of the intervals which satisfies Ψ_1. Hence, the resulting intersection intervals, satisfy the until formula. Note that, since the polygons corresponding to an STL formula are not necessarily convex, they may have several intersections with the underlying segment.

Algorithm 2 VISIT($\mathcal{R}, \Psi_1, \Psi_2, \mathcal{I}_{\Psi_1}, t, [T_1, T_2]$)

Require: Region $\mathcal{R}$, properties Ψ_1 and Ψ_2, set $\mathcal{I}_{\Psi_1}$ containing all intervals in which Ψ_1 holds before entering this region, starting time t and absolute main interval $[T_1, T_2]$.

Ensure: The set of intervals which satisfy the the bounded until in region $\mathcal{R}$, and potential intervals which satisfy Ψ_1 passing from $\mathcal{R}$.

1: **if** $\mathcal{R}$ is above the line $\tau = t$ and $\mathcal{R} \in [T_1, T_2]$ **then**
2: $\quad\mathcal{P}^{\mathcal{R}}(\Psi_1) \leftarrow CreatePoly(\Psi_1, \mathcal{R})$
3: $\quad\mathcal{P}^{\mathcal{R}}(\Psi_2) \leftarrow CreatePoly(\Psi_2, \mathcal{R})$
4: $\quad$**for all** $p \in \mathcal{P}^{\mathcal{R}}(\Psi_2)$ and $I \in \mathcal{I}_{\Psi_1}$ **do**
5: $\quad\quad\mathcal{I}_1^{\mathcal{R}} \xleftarrow{\sqcup}$ Prj$(p \cap \mathcal{R}.l_u.I)$
6: $\quad$**for all** $p \in \mathcal{P}^{\mathcal{R}}(\Psi_1)$ **do**
7: $\quad\quad P \leftarrow$ REF(p)
8: $\quad$**for all** $p \in \mathcal{P}^{\mathcal{R}}(\Psi_1)$ and $p' \in \mathcal{P}^{\mathcal{R}}(\Psi_2)$ and $I \in \mathcal{I}_{\Psi_1}$ **do**
9: $\quad\quad\mathcal{I}_2^{\mathcal{R}} \xleftarrow{\sqcup}$ Prj$(p \cap p' \cap I)$
10: $\quad\mathcal{S} \leftarrow \emptyset$
11: $\quad$**for all** $\mathcal{R}_i \in \mathcal{R}.successors$ and $p \in \mathcal{P}^{\mathcal{R}}(\Psi_1)$ and $I \in \mathcal{I}_{\Psi_1}$ **do**
12: $\quad\quad\mathcal{S} \xleftarrow{\sqcup}$ Prj$(p \cap \mathcal{R}.l_u.I \cap \mathcal{R}_i.l_u)$
13: $\quad\mathcal{I}_3^{\mathcal{R}} \leftarrow \mathcal{S} - (\mathcal{I}_1^{\mathcal{R}} \sqcup \mathcal{I}_2^{\mathcal{R}})$
14: $\quad Sat^t(\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2) \xleftarrow{\sqcup}$
$\quad\mathcal{I}_1^{\mathcal{R}} \sqcup \mathcal{I}_2^{\mathcal{R}} \sqcup$ VISIT$(\mathcal{R}_i, \Psi_1, \Psi_2, \mathcal{I}_3^{\mathcal{R}}, t, [T_1, T_2])$
15: $\quad$**return** $Sat^t(\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2)$
16: **else if** $\cdots$ **then**
17: $\quad\cdots$

Second set of intervals.: The next step is to find the second set of intervals, $\mathcal{I}_2^{\mathcal{R}}$. For this at first, the set of Ψ_1 polygons $\mathcal{P}^{\mathcal{R}}(\Psi_1)$ is needed. However, since the polygons are not necessarily convex, we only consider that part of the region where Ψ_1 continuously holds over time. This is illustrated in Figure 6, where for some value of s, Ψ_1 does not hold between time t_1 and t_2, i.e., part of the evolution of s lies outside the Ψ_1 polygon.

Hence, the polygons are *reformed* to ensure that the Ψ_1 polygon only contains system states $\Gamma(s,t)$ where in the respective polygon the evolution until time t continuously satisfies Ψ_1. In other words, we make the polygons convex w.r.t the vertical lines. For this the operator REF over a polygon p is introduced as follows:

$$(s,t) \in \text{REF}(p) \Leftrightarrow$$
$$\forall (s,t') \in p, t' \leq t : \nexists t' < t'' < t : (s,t'') \notin p.$$

The reformed polygon is illustrated in Figure 6 by shaded area. Then $\mathcal{I}_2^{\mathcal{R}}$ can be computed by projecting the intersection of the reformed Ψ_1 polygon with the Ψ_1 part of the underlying polygon $\mathcal{R}.l_u.\mathcal{I}_{\Psi_1}$ and with a Ψ_2 polygon. The resulting intersection intervals then satisfy the bounded until formula:

$$\mathcal{I}_2^{\mathcal{R}} = \text{Prj}\left(\text{REF}\left(\mathcal{P}^{\mathcal{R}}(\Psi_1)\right) \sqcap \mathcal{R}.l_u.\mathcal{I}_{\Psi_1} \sqcap \mathcal{P}^{\mathcal{R}}(\Psi_2)\right).$$

This is done in lines 6-9 of Algorithm 2. At first, all the Ψ_1 polygons are reformed. Then we iterate over all reformed Ψ_1 polygons, the set of Ψ_2 polygons and the set of intervals in $\mathcal{I}_{\psi_1}$, and add their intersection to the second interval set i.e., $\mathcal{I}_2^{\mathcal{R}}$.

Third set of intervals.: We also need to consider those intervals, through which we can enter and leave a region without violating Ψ_1, which corresponds to the third category of intervals mentioned earlier. This is done in lines 11-13. We iterate over all possible successors of the region $\mathcal{R}$ and check for each successor region whether a reformed Ψ_1 polygon intersects with its underlying segment, in order to find all potential intervals that can be used to enter the new region. However, only those intervals that have an interval intersection with the former set $\mathcal{I}_{\Psi_1}$ are valid. We also, only consider those intervals which have not already satisfied the overall until formula. The latter point is considered in line 13, by subtracting the two computed sets $\mathcal{I}_1^{\mathcal{R}}$ and $\mathcal{I}_2^{\mathcal{R}}$.

Finally, we add $\mathcal{I}_1^{\mathcal{R}}$ and $\mathcal{I}_2^{\mathcal{R}}$ to the satisfaction interval set of until formula. We also call the VISIT function for the successive regions, with $\mathcal{I}_3^{\mathcal{R}}$, as input for potential satisfaction interval set.

V. CASE STUDY: WATER STORAGE

The case study addressed in this section shows the feasibility of our formalism and algorithms by addressing the survivability of a simple water storage model, such as depicted in Figure 7. Here we reuse a simple version of the case study in [9] to investigate survivability properties, which previously was not possible. Clearly, the HPnG formalism is not restricted to the above case study, it could be applied to other fluid critical infrastructures, like oil and gas. Moreover, it could be used to model production systems and communication systems, as done with FSPNs [27] and Hybrid Petri nets [28]. In our case study, we use a so-called *Given the Occurrence Of Disaster* (GOOD) model do not take into account the probability of a certain failure but assume that the disaster has just happened; such models allow us to evaluate the recovery process and study the impact of the disaster on the system operation.

The model consists of a water storage tank (place C_s) with a constant inflow at rate 1.7 (transition F_s) and an outflow during day with rate 2 $units/h$ (transition F_d) between 6:00 and 21:00, and a lower outflow during night, with rate 1 $units/h$ between 21:00 and 6:00 the next day. The day-night pattern is governed by transitions T_n and T_d; a finer scale durial pattern can easily be made, see also the example in [9].

Under normal operation the amount of water in the storage fluctuates between 3.5 and 8 $units$ and stays at its maximum between 3:00 and 6:00. During this time span the inflow is reduced to 1 $units/h$ in order to match the outflow and to prevent the storage from overflowing. By introducing a single failure, occurring at different times of the day, we turn off the inflow to the tank, i.e., introduce a disaster, via the firing of the deterministic transition T_b. The repair period that follows has stochastic length, modelled by transition G_r, of which the duration can follow any possible distribution.

To analyze the *survivability* of the system, we check whether the system recovers within a time bound T to a predefined service level without violating a preset safety condition with probability at least 0.8. The system is defined as being recovered when the storage holds at least 3 $units$ of water and the inflow is restored, i.e., the discrete place P_i holds a token. We require that during the recovery process the amount of water never drops below 0.1 $units$, which can be seen as a safety threshold. This translates to checking whether the following formula holds expressing the system survivability:

$$\mathbb{P}_{\geq 0.7}\left((x_{C_s} \geq 0.1)\ \mathcal{U}^{[0,T]}\ (x_{C_s} \geq 3 \wedge m_{P_i} = 1)\right).$$

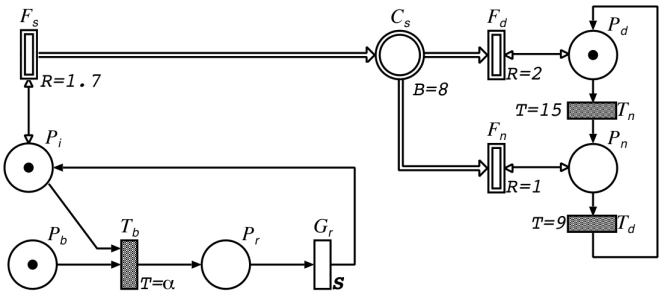

Figure 7. Model of a water storage with different demands during night and day

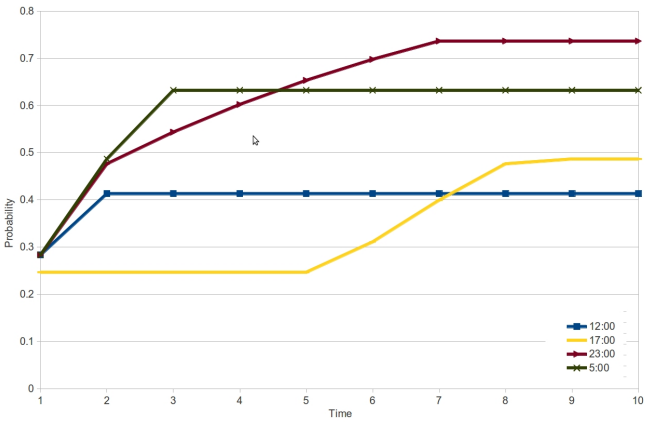

Figure 8. Survivability probability for different failure times with negative exponential distributed repair (mean =3)

Figure 8 shows the probability for the system to recover before a certain time bound T (depicted on the x-axis), for four different failure occurrence times. The repair process is assumed to be exponential with mean 3. Note that, we start the model checking at the time of failure, so due to the definition of the until operator, time on the horizontal axis is relative to the failure occurrence time. As can be seen in the figure, the survivability of the system highly depends on the time of the occurrence of the failure; a failure which occurs later during the day recovers with smaller probability. This is due to the fact that during the day the buffer is emptied, hence, the probability to hit the safety threshold in the storage becomes larger. However, in case the failure occurs during the night, the survivability is higher, since the time window between 21:00 and 24:00 is used to refill the storage. Note that, for the presented occurrence times of failure the probability to recover is only larger than 0.7 when the failure occurs at 23:00; hence, the overall formula is not generally satisfied.

Figure 9 shows the probability for the system to recover for different repair distributions, namely, Gamma, Normal and Exponential distributions. Note that the Normal distribution is truncated for the values smaller than zero and renormalized, and we use the Gamma distribution with shape parameter

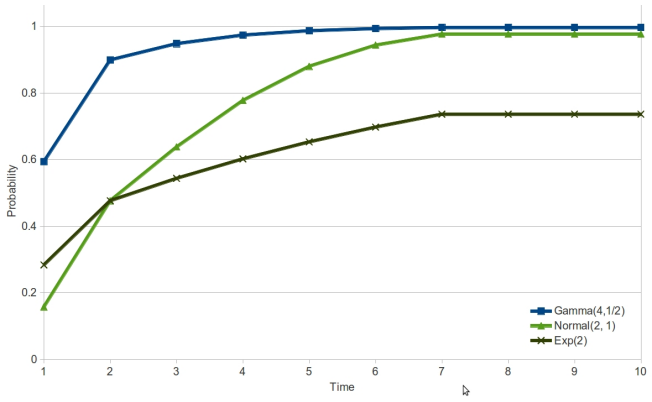

Figure 9. Survivability probability for different repair distributions for a failure at 23 : 00

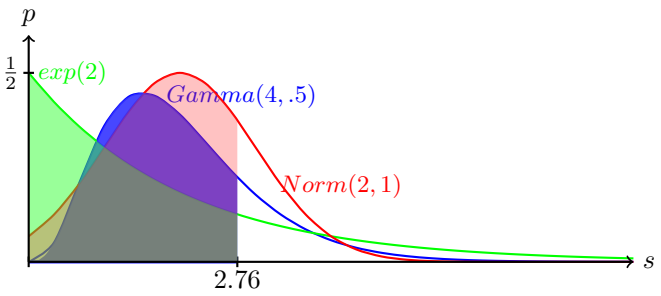

Figure 10. Representation of probability mass on the interval $(0, 2.76)$, for three different probability distributions.

4, and scale parameter 0.5. All the distributions have the same mean, i.e. 2, but with different variance. As can be seen, the exponentially distributed repair time yields a lower probability for the survivability property to hold than the Normal and Gamma-distributed repairs. Figure 9 clearly shows the importance of being able to model general transitions with different distributions.

For computing the probabilities, according to Definition 6, the model checking algorithm first computes the set of satisfaction intervals $Sat^t(\Psi)$ which can be interpreted as the maximum time window for repair in order to not violate the survivability condition. Table I summarizes these intervals for the four different failure occurrence times and six different time bounds. For instance, for a failure occurrence time of 23:00, these time-dependent intervals are $(0, 2.76)$ and $(0, 3.58)$ for 4h and 6h, respectively. Future work will investigate how we can further use the satisfaction set of intervals to schedule optimal repair strategies.

The Gamma distribution has most of its probability mass in the corresponding intervals, hence, yields the highest survivability. This is depicted in Figure 10. It can easily be seen, that the probability mass accumulated in the interval $(0, 2.76)$ is smaller for the Exponential distribution comparing to the Normal and, the Gamma distribution.

Table I. SATISFACTION INTERVALS FOR DIFFERENT FAILURE TIMES (LEFTMOST COLUMN) AND TIME BOUNDS FOR REPAIR (COLUMNS 2-6)

T	1h	2h	4h	6h	8h	10h
5:00	(0,1)	(0,2)	(0,3)	(0,3)	(0,3)	(0,3)
12:00	(0,1)	(0,1.6)	(0,1.6)	(0,1.6)	(0,1.6)	(0,1.6)
17:00	(0,0.85)	(0,0.85)	(0,0.85)	(0,1.12)	(0,1.94)	(0,2)
23:00	(0,1)	(0,1.94)	(0,2.76)	(0,3.58)	(0,4)	(0,4)

To obtain the results presented in this case study the algorithms introduced in this paper have been implemented in C++ and been executed on a machine equipped with a 2.0 GHz intel® CORE™ i7 processor, 4 GB of RAM, and Ubuntu 12.04. Computation times vary between 10 ms and 50 ms, depending on the time bound.

VI. CONCLUSIONS

In this paper we introduced a model checking algorithm for the evaluation of survivability properties for fluid critical infrastructures. To describe such properties, we introduced the logic STL that reasons on the underlying state space of an HPnG, the class of hybrid Petri nets we use to describe our systems. In STL, we introduced a time-bounded until operator that reasons on the STD, as well as a probability operator which lifts the expressivity of STL to the level of a HPnG state at a given time, by deconditioning account the probability distribution of the general transition.

This paper builds on earlier work [9] which introduced a partitioning algorithm for the state space of an HPnG into so-called regions with the same characteristics. We would like to stress that even though our earlier work contained part of the logic STL, it only dealt with state-based properties and was not suitable to argue about the evolution of the system in time, that is, about survivability properties. With the algorithm presented in this is now possible using the new until operator. Now for a restricted class of linear stochastic hybrid models, we are able to compute exact survivability measures in a couple of seconds.

The current work presents a detailed algorithm for model checking time-bounded until properties on HPnGs. The algorithm is based on grouping, per region, all system states that satisfy a given STL formula into sets of polygons. Note that, due to negation and conjunction operations, the resulting validity area is not necessarily a convex polygon. In the field of computational geometry this problem is referred to as *polygon clipping* [29], [30], [31]. The worst case complexity for finding intersection of two polygons is $O(n \cdot m)$, in which n and m are the number of polygons edges. Note that the number of edges of polygons in our algorithm corresponds to the number of continuous atomic properties in the given STL formulas. In the worst case, we may have to visit all regions, so that the final complexity of the algorithm is dependent on the number of regions times the number of continuous atomic properties. The number of regions in the STD is dependent on the structure and the initial marking of the model. More precisely, the way that model components are connected to each other and the initial distribution of tokens and fluid levels in discrete and continuous places, heavily influence the number of regions, and hence, it can not be determined a priori.

A case study on the survivability of a water refinery and distribution system under time-varying load shows the feasibility of our approach. On the one hand, using the new operators

it is possible to check whether the probability that the system recovers in time, matches a certain probability bound. On the other hand, using the satisfaction set of intervals that stem from the STL model checking, it is possible to reason about the time windows which is available for repair, and hence, allows a more operational evaluation of the system. For instance, among all the considered failure times we showed that 17:00 is the most vulnerable time for the system. Moreover, we showed that a repair strategy with the Gamma distribution has the best recovery probability among others.

In future work, we will develop the full model checking procedure, also for nested until formula. We plan to add content-dependent and time-dependent control on continuous places. Furthermore, we work on adding more general transitions to the formalism, which would at least add one more dimension per general transition to the STD.

VII. Acknowledgements

The authors would like to thank Erika Ábrahám , Joost-Pieter Katoen (RWTH Aachen), Mohammad Amin Fazli (Sharif University of Tech.) and Holger Hermanns (Saarland University) for fruitful discussions on the topic. This work has been supported by the ROCKS project through the NWO grant DN 63-257. Anne Remke is funded by a NWO Veni grant.

References

[1] "Tweede inhoudelijke analyse bescherming vitale infrastructuur (in dutch)," Ministry for Justice and Homeland Security, Dutch government, Tech. Rep., February, 2010.

[2] "Official website of department of homeland security, 'http://www.dhs.gov/critical-infrastructure'."

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, 2004.

[4] A. Avritzer, F. Di Giandomenico, A. Remke, and M. Riedl, "Assessing dependability and resilience in critical inrastructures: Challenges and opprotunities," *Resilience Assessment and Evaluation of Computing Systems*, pp. 41–63, 2012.

[5] L. Cloth and B. Haverkort, "Model checking for survivability!" in *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems, 2005*. IEEE, 2005, pp. 145–154.

[6] P. E. Heegaard and K. S. Trivedi, "Network survivability modeling," *Computer Networks*, vol. 53, no. 8, pp. 1215–1234, 2009.

[7] J. C. Knight and K. Sullivan, "On the definition of survivability," University of Virginia, Tech. Rep., 2000.

[8] M. Gribaudo and A. Remke, "Hybrid Petri Nets with General One-Shot Transitions for Dependability Evaluation of Fluid Critical Infrastructures," in *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*. IEEE CS Press, Nov. 2010, pp. 84–93.

[9] H. Ghasemieh, A. Remke, B. Haverkort, and M. Gribaudo, "Region-Based Analysis of Hybrid Petri Nets with a Single General One-Shot Transition," in *Formal Modeling and Analysis of Timed Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7595, pp. 139–154.

[10] C. Baier, B. Haverkort, H. Hermanns, and J. Katoen, "Model-checking algorithms for continuous-time markov chains," *Software Engineering, IEEE Transactions on*, vol. 29, no. 6, pp. 524–541, 2003.

[11] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proceedings of the 2nd international conference on Formal modeling and analysis of timed systems*, ser. Lecture Notes in Computer Science, vol. 3253. Springer Berlin Heidelberg, 2004, pp. 152–166.

[12] D. Nickovic and O. Maler, "AMT: a property-based monitoring tool for analog systems," in *Proceedings of the 5th international conference on Formal modeling and analysis of timed systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 304–319.

[13] T. Henzinger, "The theory of hybrid automata," in *Proceedings Eleventh Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1996, pp. 278–292.

[14] T. Henzinger, P. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," *Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 110–122, 1997.

[15] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, vol. 3414. Springer Berlin Heidelberg, 2005, pp. 258–273.

[16] G. Frehse, C. Le Guernic, A. Donze, R. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *International Conference on Computer Aided Verification*, ser. LNCS. Springer Berlin Heidelberg.

[17] E. M. Hahn, A. Hartmanns, H. Hermanns, and J.-P. Katoen, "A compositional modelling and analysis framework for stochastic hybrid systems," *Formal Methods in System Design*, 2012.

[18] L. Zhang, Z. She, S. Ratschan, H. Hermanns, and E. Hahn, "Safety verification for probabilistic hybrid systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6174, pp. 196–211.

[19] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini, "Approximate model checking of stochastic hybrid systems," *European Journal of Control*, vol. 16, p. 624641, Dec. 2010.

[20] A. Julius, "Approximate abstraction of stochastic hybrid automata," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3927, pp. 318–332.

[21] M. Bujorianu, J. Lygeros, and R. Langerak, "Reachability analysis of stochastic hybrid systems by optimal control," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 4981, pp. 610–613.

[22] M. Gribaudo, A. Horvath, A. Bobbio, E. Tronci, E. Ciancamerla, and M. Minichino, "Fluid Petri Nets and hybrid model-checking: a comparative case study," *Reliability Engineering and System Safety*, vol. 81, no. 3, pp. 239–257, 2003.

[23] J. Sproston, "Decidable Model Checking of Probabilistic Hybrid Automata," in *Formal Techniques in Real-Time and Fault-Tolerant Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, vol. 1926, pp. 31–45.

[24] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, 1st ed. John Wiley & Sons, Inc, 1994.

[25] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*, 2nd ed. Springer Berlin Heidelberg, 2010.

[26] M. Gribaudo and A. Remke, "Hybrid Petri nets with general one-shot transitions: model evolution," University of Twente, Tech. Rep., 2010, http://wwwhome.cs.utwente.nl/ anne/techreport/hpng.pdf.

[27] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi, "Fluid stochastic Petri nets: Theory, applications, and solution techniques," *European Journal of Operational Research*, vol. 105, no. 1, pp. 184–201, 1998.

[28] R. David and H. Alla, "On hybrid Petri nets," *Discrete Event Dynamic Systems*, vol. 3, pp. 9–40, 2001.

[29] K. Weiler and P. Atherton, "Hidden surface removal using polygon area sorting," in *Computer Graphics*, 1977, pp. 214–222.

[30] B. Vatti, "A generic solution to polygon clipping," *Commun. ACM*, vol. 35, no. 7, pp. 56–63, 1992.

[31] G. Greiner and K. Hormann, "Efficient clipping of arbitrary polygons," *ACM Transactions on Graphics*, vol. 17, no. 2, pp. 71–83, 1998.