Hybrid Petri nets with multiple stochastic transition firings

Hamed Ghasemieh
University of Twente
h.ghasemieh@utwente.nl

Anne Remke
University of Twente
a.k.i.remke@utwente.nl

Boudewijn R. Haverkort
University of Twente
b.r.h.m.haverkort@utwente.nl

ABSTRACT

This paper introduces an algorithm for the efficient computation of transient measures of interest in Hybrid Petri nets in which the stochastic transitions are allowed to fire an arbitrary but finite number of times. Each firing increases the dimensionality of the underlying discrete/continuous state space. The algorithm evolves around a partitioning of the multi-dimensional state-space into regions, making use of advanced algorithms (and libraries) for computational geometry. To bound the number of stochastic transition firings the notion of control tokens is newly introduced. While the new partitioning algorithm is general, the implementation is currently limited to only two stochastic firings. The feasibility and usefulness of the new algorithm is illustrated in a case study of a water refinery plant with cascading failures.

Keywords

Stochastic hybrid Petri-nets, Transient analysis, Computational geometry

1. INTRODUCTION

Recently the framework of Hybrid Petri-nets with General one-shot transitions (HPnG) has been introduced for the analysis of, e.g., fluid critical infrastructures [13]. Efficient algorithms have been introduced for investigating, among others, reachability properties in the presence of a single stochastic transition firing [12, 11]. Such HPnGs are very well able to capture the rather deterministic evolution of a physical process with many continuous variables, as for example present in the application area of water management. In this application field we have conducted a capacity analysis for a real sewage treatment facility in [10]. However, other application areas with more intricate failure scenarios, like, e.g., cascading failures, require the presence of more stochastic transition firings in the model. In this paper, we tackle the above limitation of existing HPnG analysis algorithms to one stochastic transition firing and generalise the state space representation and the analysis algorithm, cf. [12], to an arbitrary but finite number of stochastic variables. This is done by introducing the concept of *control tokens*, which restrict the total number of firings of general transitions. Note that each stochastic transition firing introduces a new random variable and corresponds to adding a new dimension to the underlying discrete/continuous state space. Hence, dealing with higher dimensions becomes inevitable. Moving to higher dimensions increases the complexity of the existing algorithms. More specifically, instead of dealing with line segments and polygons in two dimensions as done in [12], we have to deal with hyperplanes and polytopes, respectively, in multiple dimensions. Hence, the proposed method in this paper highly depends on algorithms from computational geometry, especially for *half space intersection* [20] and *hyperplane arrangement* [15]. For both of these operations feasible solutions from the field of computational geometry exist. However, to the best of our knowledge, there is no implementation of hyperplane arrangement for more than two dimensions, which corresponds to the presence of two stochastic variables in our case. The contribution of this paper, i.e., the formalization of the multi-dimensional state-space and the presented algorithm for its partitioning, is applicable to multiple stochastic transition firings. However, due to the limitations of existing computational geometry libraries, at the moment the implementation is limited to only two stochastic firings.

The paper is further organized as follows: Section 2 covers the basic definition and formalism of HPnGs along with the extension for support of multiple general transitions. Section 3 discusses the main idea for state space representation. Section 4 proposes the algorithm for generating and segmenting the state space. Section 5 discusses the method for computing measures of interests, and finally, Section 6 investigates a working example to show the feasibility of the proposed method and Section 7 concludes the paper.

2. HYBRID PETRI NETS

In this section we recall, intuitively, the HPnG model. Like many other Petri-net models also the formalism of HPnGs is user friendly and close to real life applications. Hence, this section stresses the graphical representation of HPnGs, and refers to [13, 10], for a detailed discussion of syntax, semantics, and the modelling formalism. In contrast to earlier works, this paper allows multiple firings of one or more general transitions. In order to formalize a restriction on the total number of firings, this paper additionally introduces the concept of *control places* and *control tokens*.

2.1 Modelling formalism

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
VALUETOOLS 2014, December 09-11, Bratislava, Slovakia
Copyright © 2015 ICST 978-1-63190-057-0
DOI 10.4108/icst.valuetools.2014.258204

An HPnG model is designed to depict real systems containing both discrete and continuous variables, combined with stochastic behaviour. It consists of three main sets of components: (I) *places* (discrete and continuous), which model different modes of the system, (II) *transitions*, which allow changes between different modes of the system, and finally (III) *arcs* (connecting places and transitions), which determine how the other two sets are related to each other, i.e, how a transition between different modes of the system can take place. Each of these three sets contains different types, and graphical representations, as illustrated in Figure 1.

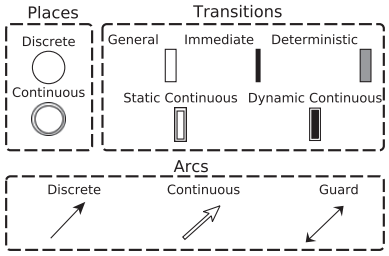


Figure 1: Graphical representation of primitives of HPnG.

The set of **places**, $\mathcal{P}$, contains two disjoint sets of *discrete* ($\mathcal{P}^D$) and *continuous* ($\mathcal{P}^C$) places. The former keeps track of discrete variables of the system, e.g., the the number of spare parts, and the latter contains the continuous state of the system, e.g., the amount of fluid in a container. A discrete place may contain a number of *tokens*, while a continuous place is assigned with a real number, representing the level of fluid residing in it. We later refer to the content of places as the *marking* of the system.

Transitions will trigger a change in the state of the system, i.e., they may change the content of place(s), provided that all the required resources are available. In this case we say that the transition is *enabled* and may *fire*. The set of transitions, $\mathcal{T}$, consists of four disjoint sets. Immediate ($\mathcal{T}^I$), deterministic timed ($\mathcal{T}^D$), and general ($\mathcal{T}^G$) transitions, all referred to as discrete transitions, are responsible for changing the discrete part of the system, whereas, continuous transitions ($\mathcal{T}^C$) change the content of continuous places. An immediate transition will fire at the very moment it is enabled, and a deterministic transition will fire at a specific time after it has been enabled. Each deterministic transition, T_i^D, is associated with a clock c_i, which evolves with drift $dc_i/d\tau = 1$, if the transition is enabled. When a clock reaches its firing time, transition T_i^D fires, and the clock is reset to zero. A general transition will fire according to an arbitrary probability distribution after it has been enabled. More specifically, a general transition T_k, associated with the probability distribution $g_k(s)$, fires with probability $\int_{\tau}^{\tau+\Delta\tau} g_k(s)ds$, in the time interval $[\tau, \tau + \Delta\tau]$. Note that the execution policy of the general transitions, i.e., their enabling/disabling semantics, is of type *race model with age memory*, i.e., the clock of a general transition is preserved upon disabling and resumed with enabling [2]. The total a general transition is enabled before it fires is drawn independently from its respective probability distribution. There may be a dependence between the actual time of firings due to the time transitions have been disabled; which depends on the structure of the Petri net. Continuous transitions, as their name suggests, will fire continuously, according to an assigned rate and change the content of continuous places, provided that they are enabled. Moreover, a continuous transition can be *static* or *dynamic*, meaning that, it will either fire with constant *nominal rate*, or its rate can dynamically depend on the rates of other static continuous transitions. The set of **arcs**, $\mathcal{A}$, characterizes how transitions and places are related to each other. Discrete arcs, $\mathcal{A}^D$, connect discrete places to transitions, on the following way. If a transition fires, it will remove tokens from places connected to it via input arcs, and add tokens to the places that are connected via output arcs. The number of tokens being removed or added are determined by the weights assigned to the arcs. Continuous arcs, connect continuous places and transitions. Therefore, when a continuous transition fires, it will remove content of its input places and add to the content of its output places, with a specific rate assigned to the transition. The set of guard arcs, $\mathcal{A}^G$, connects discrete transitions to both discrete and continuous places. These arcs ensure that a transition is only enabled in case the number of tokens (in case of a discrete place) or the amount of fluid (in case of a continuous place) fulfils a certain condition that is specified on the guard arc.

Figure 2, shows a simple HPnG model, with two general transitions. The continuous place P_r, models a water reservoir with capacity 10. This place is being filled using two different producer pumps, modelled as continuous transitions T_1 and T_2, with nominal rates 1 and 2, respectively. The water in the reservoir is being consumed with the demand pump, T_d, at rate 2. We know that after 5 hours, the demand will stop, i.e., the deterministic transition T_s will fire at $t = 5$. Two general transitions G_1 and G_2, are modelling the possibility of failure in the production. They fire according to predefined probability distributions (not provided here).

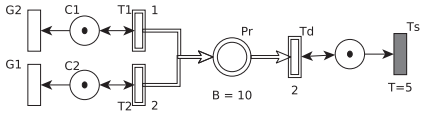


Figure 2: A reservoir model with two different production rates.

Rate adaptation

In the HPnG modelling formalism, we associate with each continuous place a lower and upper boundary. Conflicts between continuous transitions 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. This means that the rates of inputs/outputs transitions should be adapted. The newly adapted rates of continuous transitions are called *actual rates*, in contrast to their preassigned nominal rates. This rate for each transition is determined based on the *priorities* and *shares* assigned to the arcs, connecting continuous transitions and the place. This process is called *rate adaptation*. For further details on rate adaptation we refer to [4]. For example, if the reservoir P_r in Figure 2 reaches its upper boundary and we assume that the arcs connecting transitions T_1 and T_2, have the

(a) A one-shot general transition (b) A two-shot general transition

Figure 3: Representation of control places and tokens.

same share and priority, then both transition rates will be adapted to match the output from the place P_r, such that both together fire at rate 1.

2.2 Control places and control tokens

Each possible firing of a general transition will introduce a new stochastic variable to the system. In order to analyse HPnGs, we need to be able to count and impose a restriction on the number of stochastic variables. For this purpose we introduce *control tokens* and *control places*. [1] Control places are isolated discrete places, which can only be connected to one general transition via an input arc. A control place may contain one or more control tokens. When a general transition fires it will consume one control token from its connected control place. A control place can not be refilled with tokens. Note that, for simplicity in this paper, we assume that there is a one to one relationship between control places and general transitions, i.e., a control place can be connected to exactly one general transition and vice versa. Although it is possible to have several general transitions sharing a control place and its tokens, we reserve this for future work. Therefore, with the current setting we can ensure that the number of firings of general transitions is smaller than or equal to the number of available control tokens. Hence, the complexity and the dimension of the state space, as will be shown later, is determined by the number of control tokens. Figure 3, illustrates the idea of control places and tokens. Control places are indicated by dashed boxes. Formally, control places are a subset of the discrete places. Figure 3a shows a general *one-shot* transition, the case that we previously addressed in [13]. Figure 3b, illustrates a general *two-shot* transition, i.e., a general transition which can fire two times at most. Also, note the presence of control places, C_1 and C_2, in the example of Figure 2, which allow each general transition to fire at most once.

2.3 State of the system

Markings i.e., the content of places, are collected into two vectors, the discrete marking $\mathbf{m} = (m_1, \ldots, m_{|\mathcal{P}^D|})$ and the continuous marking $\mathbf{x} = (x_1, \ldots, x_{|\mathcal{P}^C|})$. Note that, since the control places form a subset of discrete places, $\mathbf{m}$ also includes the control tokens. 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 overall *state* of an HPnG is defined by $\Gamma = (\mathbf{m}, \mathbf{x}, \mathbf{c}, \mathbf{d}, \mathbf{g})$, where the 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. Clocks are only reset when the corresponding deterministic transition fires. Vector $\mathbf{d} = (d_1, \ldots, d_{|\mathcal{P}^C|+|\mathcal{T}^D|})$ indicates the drift of all continuous variables. For continuous places it indicates the change of fluid per time unit, and for deterministic transitions it is the clock drift one for enabled and zero for disabled transitions, respectively. Note that even though the vector $\mathbf{d}$ is determined uniquely by $\mathbf{x}$ and $\mathbf{m}$, in combination with the condition of guard arcs, it is included in the definition of a state for the ease of analysis. Finally, the vector $\mathbf{g} \in \mathbf{N}^{|\mathcal{T}^G|}$ indicates the number of times that each general transition has already fired. Hence, the sum of the elements of $\mathbf{g}$, is equal to the total number of present stochastic variables in the system.[2] The initial state of the system is $\Gamma_0 = (\mathbf{m}_0, \mathbf{x}_0, \mathbf{0}^{|\mathcal{T}^D|}, \mathbf{d}_0, \mathbf{0}^{|\mathcal{T}^G|})$, where $\mathbf{0}^m$, is the vector with m zero elements. A system state can be seen as a snapshot of the system evolution at a specific time, which given all stochastic firing times uniquely determines the future evolution of the system. This is elaborated in more detail in the next section. For the example given in Figure 2, the initial state is given by $\mathbf{m}_0 = (1, 1, 1)$, since there is a token in all the discrete places. The amount of fluid in the reservoir is expressed by $\mathbf{x}_0 = (5)$), and the drift is $\mathbf{d}_0 = (+1)$, i.e., the difference between input and output rates to and from the reservoir.

3. STATE SPACE REPRESENTATION

The Stochastic Time Diagram (STD) introduced in [12], provides a genuine way of representing the evolution of a HPnG for a given initial state. The main reasoning is that, for a given initial state of an HPnG and given all stochastic firing times, the evolution of the system is deterministic. Let $\mathbf{s} = (s_1, \cdots, s_n)$ be the vector of n random variables, representing the firing time of the general transitions. If n stochastic variables are present in a system, the STD will have $n + 1$ dimensions, n of which are associated with stochastic variables and the $(n + 1)$th dimension is associated with time t. Each point in the STD is associated with a unique HPnG state, which is denoted by $\Gamma(\mathbf{s}, t)$. Figure 4 illustrates a generic STD with two stochastic variables.

The main idea behind the method in [12], 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 a so-called *event* occurs. In each system state, three types of potential events can occur: (i) a continuous place reaching its lower/upper boundary, (ii) a continuous place reaches the weight of the guard arc connected to it, and (iii) an enabled transition, either deterministic or general, fires. Event type (i) imposes a change in the drift of the continuous place, due

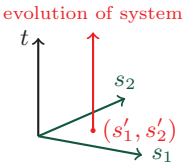


Figure 4: Generic presentation of STD in 3 dimensions.

[1] Control tokens and places have the same functionality as discrete tokens and places, however, we choose to separate them to underline their different role, and to allow syntactic checks.

[2] Note that, since in this paper we are not allowing general transitions sharing a control place, the vector $\mathbf{g}$ can be determined from the initial and current marking of the control places.

to rate adaptation [4], and event type (ii) will enable or disable a transition. In case of an immediate transition, it will fire and alter the discrete marking immediately, and if it is a deterministic transition its clock drift will be set to one, thereby changing a continuous variable. Finally, event type (iii) alters the discrete marking $\mathbf{m}$, or the general transitions vector $\mathbf{g}$. In any case, an event may cause a change in the discrete marking, a change in drift (either for clocks or fluid levels) or a change in the vector indicating the number of firings of each general transition. We define a region as a maximal set of states, that while no event occurs, the system remains inside, i.e., discrete marking, drift of continuous variables and general transitions vector remain unchanged, *within* a region. Moreover, at the occurrence of an event the system enters another region. This leads to the following definition.

DEFINITION 1. *A region $\mathcal{R}$ is a maximal connected set of $(\mathbf{s}, t)$ points in a given STD, for which we have:*

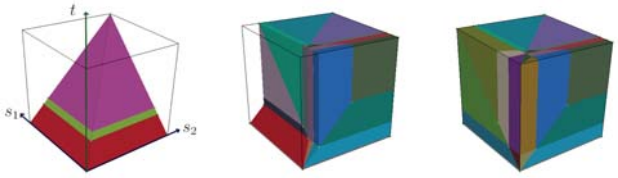
$$\forall(\mathbf{s}_1,t_1),(\mathbf{s}_2,t_2)\in\mathcal{R}:\begin{cases}\Gamma(\mathbf{s}_1,t_1).\mathbf{m} &= \Gamma(\mathbf{s}_2,t_2).\mathbf{m},\\ \Gamma(\mathbf{s}_1,t_1).\mathbf{d} &= \Gamma(\mathbf{s}_2,t_2).\mathbf{d},\\ \Gamma(\mathbf{s}_1,t_1).\mathbf{g} &= \Gamma(\mathbf{s}_2,t_2).\mathbf{g},\end{cases}$$

where $\Gamma(\mathbf{s},t).\mathbf{m}$ refers to the vector of discrete markings, and $\Gamma(\mathbf{s},t).m_P$ refers to the discrete marking of a specific place P. A similar notation is used for the continuous marking. Note that the above definition is different from the definition in [11], since here, $\mathbf{g}$ is a vector containing the number of firings of each general transition. Note that vector $\mathbf{d}$ contains both drifts of continuous places and clocks for deterministic transitions. The reason for this is that because of guard arcs, a deterministic transition can be enabled or disabled for the same discrete marking, due to a change in the continuous marking. This is an event type (ii), hence, represents a move to another region.

The shape of the regions depends on the structure of the model at hand. In [12], for the case of one stochastic variable, 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 of s and t. Adding more stochastic variables, does *not* influence the linear characterization of continuous variables, as will be shown in Proposition 1. 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 or zero). Using this we infer that the boundaries between regions, which represent the occurrence of an event, are characterized by linear functions of $\mathbf{s}$ and t, which represents a hyperplane in $n+1$ dimensions. Hence, each region in the STD can be considered as a polyhedron, in $n+1$ dimensions. Introducing dynamic fluid transitions does not change this fact, because their nominal rates depend on the actual rates of other static continuous transitions, which are constant, *within* each region. Hence, we can safely treat dynamic transitions as static transitions, for that matter.

Even though reachability computations on the STD are always performed for a given and finite time bound, there is still the possibility of having an infinite number of regions in the STD before a finite time bound. This happens whenever an infinite sequence of vanishing markings occurs. This problem is well-known for all Petri net formalisms 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 can be detected and removed. This ensures that we can 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, a finite number of general transition firings, and a finite time bound our algorithm will always terminate. In Section 4, we discuss the proposed algorithms for generating the state space.

Figure 5, shows the 3-dimensional Stochastic Time Diagram for the example previously introduced in Figure 2. Each region is depicted with a different color. Figure 5a, illustrates the phase in which no general transition has fired, yet. As can be seen the formation of regions is independent of the value of s_1 and s_2, i.e., they can be characterized by planes parallel to plane s_1s_2. Moreover, the planes $t = s_1$ and $t = s_2$, which correspond to the firing of general transitions, are clearly visible. Figure 5b, represents a later phase, whereas, Figure 5c, depicts the complete STD of the reservoir example. From these pictures it is apparent how the shape of regions above the two planes $t = s_1$ and $t = s_2$ depends on the values of s_1 and s_2.



(a) No g-trans. fired. (b) A middle phase. (c) Complete STD

Figure 5: STD for the reservoir example.

4. GENERATING THE STATE SPACE

As mentioned in Section 3, to partition the state space into regions, we need to determine the next events in each stage in the system evolution. Since events depend on the value of continuous variables (either clock value of a deterministic transition or fluid level in a continuous place) in the system, the first thing is to find the equations that characterise these continuous variables. As mentioned earlier, a continuous variable can be represented as a linear combination of the current time t and the general transition firing times, i.e., vector $\mathbf{s}$. Intuitively, this is because in each system state, a continuous variable evolves with a constant drift.

PROPOSITION 1. *At each time point t during the evolution of the system, the value of the continuous variables and the occurrence time of the next events can be characterised as a linear equation of t and $\mathbf{s}$.*

PROOF. Let x be the value of a continuous variable:

$$x = a_{n+1}t + \sum_{k=1}^{n} a_k s_k + a_0 = a_{n+1}t + \mathbf{a}.\mathbf{s} + a_0. \qquad (1)$$

Assume the previous event has occurred at time $t_0 = \alpha.\mathbf{s} + \alpha_0$, in which α is a vector of n scalars. If no general transition

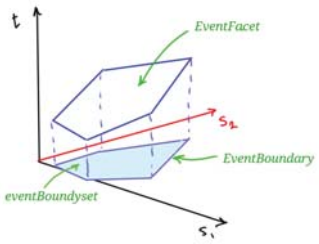


Figure 6: Demonstration of an event facet, in 3 dimensions.

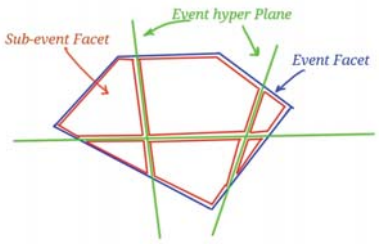


Figure 7: Top view of possible intersection of an event hyperplane with event facets, and formation of sub-facets.

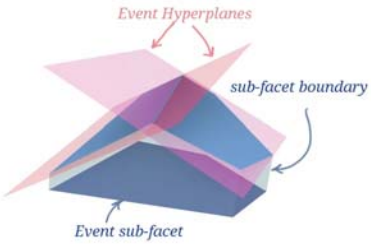


Figure 8: Formation of a hyper-region over an event facet.

has fired yet, t_0 is a constant. Moreover, $t_0 = s_k$ corresponds to the firing time of the k-th general transition. We calculate the occurrence of the next event due to a continuous place reaching its boundary (other event types are simpler version of this one). For this case, the place with x amount of fluid, which changes with drift d, will reach its upper boundary B, according to the following equation:

$$B = x + \Delta t.d = x + (t - t_0)d,$$

in which $\Delta t = t - t_0$ is the relative time to the occurrence of the event, and t is the absolute occurrence time of the event. Hence, we have:

$$t = -(1/d)x + t_0 + B/d,$$

which is a linear equation of $\mathbf{s}$ and t. Now each continuous variable can be updated based on Δt and their drift, which results in a linear equation. □

4.1 Facets and regions

Since at each point in the state space the occurrence time of the next events is a linear function of all stochastic firing times, each event in the state space can be represented by a hyperplane, in $(n+1)$-dimensions. These hyperplanes form the boundaries of regions of the partitioned state space, as mentioned in Section 3. Therefore, in order to partition the state space, after the occurrence of each event, we have to find the equations for all *potential next events*, and take the minimum over them. Geometrically, this corresponds to finding the *lower envelope* of a set of hyperplanes (visualized in Figure 8), which results in a set of *facets*. A facet is a confined version of a (hyper)plane, i.e., it is limited by its set of borders. Facets in higher dimensions, correspond to segments in two dimensions. While a segment is represented by a line and an interval, a facet is characterized by a hyperplane and a set of boundaries. Each event is associated with an *event facet*, as shown in Figure 6 for a model with two stochastic transition firings:

Definition 2. *An event facet is defined as the following structure:*

EventFacet*:*
EventHyperPlane*:* $t = \sum_{k=1}^{n} a'_k s_k + a'_0$
EventBoudarySet*:* $\{\sum_{k=0}^{n} b_k s_k + b_0 \leq 0\}$

The EventHyperPlane shows the time t at which the event corresponding to this facet is happening. The hypervolume that contains vector $\mathbf{s}$ is EventBoudarySet and the corresponding event to this facet EventHyperPlane will happen at time t. These concepts are visualized in Figure 6 for 3 dimensions (when two general transition firings are assumed).

We also define a *hyper-region* as the maximal area surrounded by a set of neighbouring event facets. The concept of hyper-regions is the same as regions in two dimensions [12]. This means that for all points in a hyper-region, the possible values of a continuous variable can be represented by a linear equation of time and the firing times of general transitions, as shown in Equation 1.

4.2 State space generation and partitioning

The algorithm for partitioning the state space is given in Algorithm 1. The algorithm is called with the initial marking Γ_0, at time $t = 0$ (the initial event facet). The function ComputeNextEvents solves a set of linear equations, as suggested in Proposition 1, and returns a set of hyperplanes, where, each of its elements corresponds to the occurrence time of a potential next event. Since we are interested in finding the next occurring events, which depend on $\mathbf{s}$, we have to find the minimum over all occurrence times of these potential events. The function CreateHyperRegions finds this minimum over the given hyperplane set, which is returned by the function ComputeNextEvents, and then creates the set of hyper-regions formed above the given event facet, (we will provide a detailed description of this function below). Subsequently, we iterate over the sets of facets forming the hyper-regions and update the system state Γ, by calling the function update, which updates the values of all the continuous variables, based on the time difference of $\mathcal{F}_0$ and the new event facet. Moreover, if the new event facet corresponds to the firing of a general transition, the vector $\mathbf{g}$ is updated. Finally, we recursively call the function PartitionAboveEventFacet over each new facet, with the updated system state.

The function CreateHyperRegions embodies the implementation of the main challenge for handling multiple general transitions, which is presented in Algorithm 2. Since the given hyperplanes may intersect with the underlying event facet, there is the possibility of having a set of hyper-regions. This is depicted in Figure 7, for the case of two general transitions being present in the system. Note that, the intersection of a hyperplane with a facet in three dimensions is a line. As illustrated in the figure, the facet intersects with three hyperplanes, and as a result, six *sub-facets* are formed. Above each of these sub-facets we have to form a hyper-region. This formation has been covered in detail in

Algorithm 1 PARTITIONABOVEEVENTFACET($\mathcal{F}_0, \Gamma$)

Require: $\mathcal{F}_0$, the event facet above which we want to partition the state space, Γ, the current HPnG state, and $\mathcal{R}^{\mathcal{H}}$ as the global set in which all the hyper-regions are saved.
Ensure: Returns set of all hyper-regions above the given event facet.
1: $E^{\mathcal{H}} \leftarrow$ COMPUTENEXTEVENTS($\mathcal{F}_0, \Gamma$)
2: $\mathcal{R} \leftarrow$ CREATEHYPERREGIONS($\mathcal{F}_0, E^{\mathcal{H}}$)
3: $\mathcal{R}^{\mathcal{H}} \leftarrow \mathcal{R}^{\mathcal{H}} \cup \mathcal{R}$
4: **for all** $\mathcal{R}_i \in \mathcal{R}$ **do**
5: **for all** $f_j \in \mathcal{R}_i$ **do**
6: $\Gamma_{new} \leftarrow$ UPDATE($\mathcal{F}_0, f_j, \Gamma$)
7: PARTITIONABOVEEVENTFACET(f_j, Γ_{new})
8: **return** $\mathcal{R}^{\mathcal{H}}$

Algorithm 2 CREATEHYPERREGIONS($\mathcal{F}, E^{\mathcal{H}}$)

Require: $\mathcal{F}$, the event facet, $E^{\mathcal{H}}$, set of potential event hyperplane.
Ensure: Creates and returns the set of direct hyper-regions above the given event facet.
1: $\mathcal{F}_{sub} \leftarrow$ CREATESUBFACETS($\mathcal{F}, E^{\mathcal{H}}$)
2: $\mathcal{R} \leftarrow \emptyset$
3: **for all** $f_i \in \mathcal{F}_{sub}$ **do**
4: $\mathcal{R} \leftarrow \mathcal{R} \cup$ FORMHYPERREGION($f_i, E^{\mathcal{H}}$)
5: **return** $\mathcal{R}$

[12], for the case of one general transition. The problem of forming these sub-facets in two dimensions is known as *arrangement of lines*, and in higher dimension as *arrangement of hyperplanes* [15]. This is an essential problem in computational geometry, since many other problems can be reduced to it [5].

After having determined all sub-facets, we have to form the hyper-regions above each of them. This, basically, is done by taking the minimum over all the event hyperplanes. This problem can be interpreted as the intersection of a set of half spaces. More formally, let $\mathcal{F}$ be a sub-facet, with hyperplane $t = \mathbf{a}.\mathbf{s}+a_0$, and the set of m boundaries $\{\mathbf{b}^i.\mathbf{s}+b_0^i = 0\}_{i=0}^{m-1}$. Moreover, let $\{t = \mathbf{e}^j.\mathbf{s}+e_0^j\}_{j=0}^{l-1}$ be the set of l potential event hyperplanes. Then the hyper-region formed above the event facet $\mathcal{F}$ is the intersection of the following half spaces:

$$\begin{cases} t - \mathbf{a}.\mathbf{s} + a_0 > 0 & \\ \mathbf{b}^i.\mathbf{s} + b_0^i \bowtie 0, & 0 \le i < m, \\ t - \mathbf{e}^j.\mathbf{s} + e_0^j < 0, & 0 \le j < l, \end{cases}$$

in which, $\bowtie \in \{<, >\}$ (depends on the sub-facet). This can be determined while creating the sub-facets, in the previous phase. The formation of a hyper-region is visualised in Figure 8. As can be seen, a hyper-region (the transparent volume), is formed as the interior space of an event sub-facet, sub-facet boundaries, and potential event hyperplanes. Informally, these can be interpreted as, floor, columns, and roofs, respectively.

The presented algorithm approaches the problem for the general case of n stochastic variables. However, as mentioned earlier, the introduced algorithm depends on two well-known computational geometry problems, known as *half-space intersection* and *hyperplane arrangement*. The function FORMHYPERREGION embodies the former problem, and the latter is present in the function CREATESUBFACETS. The problem of half-space intersection, is dual to the convex-hull problem, which can be solved in order $O(m \log m)$, where m is the number of half-spaces [20].

However, the complexity of the existing algorithm for hyperplane arrangement, is exponential in dimension d, i.e., $O(m^d)$, where m is the number of hyperplanes [6], which in our case, is equal to the number of stochastic transition firings plus an extra dimension for time. To the best of our knowledge, there is no implementation of hyperplane arrangement, for more than two dimensions. Hence, we are currently restricted to the case, where only two stochastic variables are available in the system. For both of the above problems there are reliable implementations and libraries, among which we have used Computational Geometry Algorithm Library (CGAL), which provides extensive implementation for most of the existing algorithms, [18] ,[19], [21].

5. COMPUTING MEASURES

To compute the probability to be in a specific system state at time τ, it suffices to find all regions intersecting the horizontal hyperplane $t = \tau$. Then we project the intersection result over the **s**-plane, and integrate all probability density functions $g_i(s_i)$ over the resulting area. In order to define properties we use the same logic as in [12]:

$$\psi = \neg\psi \mid \psi \wedge \psi \mid n_i = a \mid x_k \le b, \tag{2}$$

where, n_p is the number of tokens in the discrete place P_i, and x_k is the fluid level in the continuous place P_k. An atomic discrete property $(n_i = a)$, either holds in the entire region or not at all. An atomic continuous property $(x_k \le b)$, depends on the value of vector **s**. More specifically, at a given time τ, the value of continuous variable x is a linear function of the vector **s**. Therefore, the validity space of an atomic continuous property, would be the half space $\Gamma(\mathbf{s}, \tau).x \le b$, i.e., all the $(\mathbf{s}, \tau)$ points for which their associated system state $\Gamma(\mathbf{s}, \tau)$, satisfies the given property.

Finally, conjunction and negation of atomic properties correspond to boolean operations on half spaces. When two or three stochastic variables are present, the problem reduces to boolean operations on polygons or 3D polyhedrons, for which CGAL [7], [14] provides efficient implementations. After obtaining the validity space of a certain property for a given time τ, we need to integrate the density functions over the validity space in order to find the probability that a given property holds. For the case of two stochastic variables the probability that ψ holds at time τ is given by:

$$\pi^{\psi}(\tau) = \int_{A_2} \int_{A_1} g_1(s_1) g_2(s_2) ds_1 ds_2, \tag{3}$$

Where A_1 and A_2 encode exactly the time periods in which at least one general transition is enabled. This together with the independence of random variables associated with the firing time of general transitions, ensures that the joint probability distribution is the product of the individual probability distributions. The arbitrary nature of the integration area, requires numerical solutions. Each multiple integration can recursively be converted to a sequence of single definite integration, for which reliable and nearly exact algorithms exist [9].

6. CASE STUDY

This section provides an application example, to illustrate the feasibility of the proposed method. We investigate the survivability of a, so-called, GOOD (Given the Occurrence of Of Disaster) model, where the occurrence of a disaster is assumed at a certain point in time and the focus is on the recovery process, after that point in time. Figure 9, depicts the model of a water cleaning facility, where raw water is taken in via two entries f_1 and f_2 with rates 4 and 2, respectively and then cleaned in two separate cleaning streets. This setup increases the dependability of the system: in case the upper cleaning street fails, half of its input is rerouted to the lower cleaning street, which is then able to operate at a higher speed (modelled via the static fluid transition f_e and the dynamic fluid transition f_4, respectively). However, since transition f_4 is handling twice of its normal load while the system is not repaired, yet, it may fail as well. This is modelled by the general transition T_f. Note that the guard arc connecting T_f and D_1 ensures that the failure can occur only if the repair process is not accomplished yet. Also note that, both of the general transitions can fire only once, which is guaranteed through control places, D_1 and D_2. We start the analysis assuming that the pump f_3 has failed, and the repair process has been initiated (modelled by the general transition T_r, which is now enabled) and all the tanks are full initially. While the upper cleaning street is not repaired yet, there is a token in place D_1, transition f_e is enabled and pumps additional water to the lower cleaning street at rate 2. The rate of the dynamic transition f_4 is then equal to the sum of the rates of f_e and f_2.

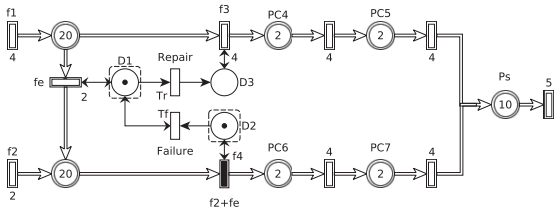
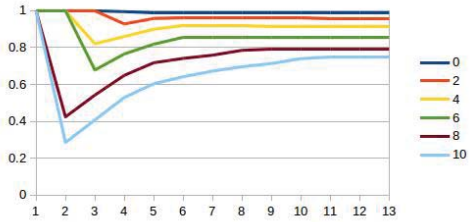


Figure 9: HPnG model of a water treatment facility with two parallel cleaning streets.

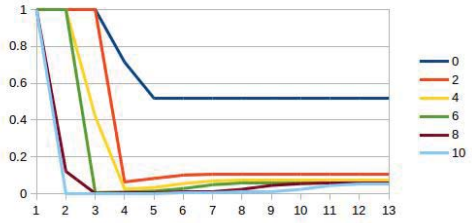
Since we want to ensure that customers can reliably be served also in the presence of a failure in the system, we will compute the probability of having at least a certain amount of fluid in place P_s, which models a cleaned water storage, i.e., $x_{P_s} \geq b$, for varying values of b. Figure 10, shows the transient probability that the above property holds at different times, τ, and for fluid levels, b, and different probability distribution for failure and repair, as provided in Table 1. The horizontal axis depicts the time of operation (τ), while the vertical axis is the probability of having more water in the reservoir (x_{P_s}) than the given value, b (color coded, see the legend), and each curve corresponds to a specific fluid level (b). In Figure 10a the repair process will take place earlier than the failure with a high probability. In contrast, Figure 10b, shows the results for a situation where a failure of the second cleaning street is likely to occur before the system is repaired. Finally, in Figure 10c the expected time of occurrence for repair and failure are almost at the same time, namely 3 and 4, respectively.

Table 1: Different parameters of probability distributions.

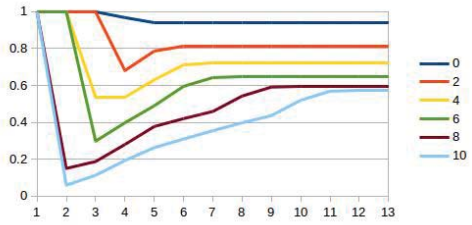
	Repair	Failure
	folded normal	gamma
(a)	$\mu = 2, \sigma = 1$	$k = 3, \theta = 2$
(b)	$\mu = 5, \sigma = 1$	$k = 2, \theta = 1$
(c)	$\mu = 3, \sigma = 1$	$k = 2, \theta = 2$



(a) Early repair.



(b) Late repair.



(c) When failure is likely to happen slightly after repair.

Figure 10: Probability of having more than a given water level for a certain time.

These figures show that the ability of the system to recover from the former disaster highly depends on the chosen probability distributions. In case (a) the system is highly survivable, since it quickly refills the water storage. However, the setting in case (b) cannot be considered survivable, as it reaches a predefined storage level only with a probability of approximately 0.1. In case (c), where the failure is likely to happen slightly after the repair, the system recovers to a certain degree however, a positive probability of keeping an empty final storage remains. These experiments show how the analysis of Hybrid Petri nets with two general one-shot transitions helps to obtain insight in the recovery process of a system with stochastic failure and repair processes. Our analysis shows how important the maintenance of the pump with the additional load is. Since as can be seen, if the failure of this pump is likely to happen before the repair, there is no possibility of recovery from the first failure. The STD for this model has 85 regions, and its generation takes less than half a second. Each diagram in Figure 10, consists of 6 curves, each consists of 25 points. For each point a separate probability computation and integration needs to take place. In total, the generation of each curve has taken less than 5 seconds. All computations have been performed on a machine with a Core i7 processor and 4GB of memory.

7. CONCLUSIONS AND RELATED WORK

This paper strives to tackle the main constraint of Hybrid Petri-nets with general one-shot transitions, by developing an algorithm for the analysis of HPnGs with multiple general transition. To do so, we extended the concept of HPnGs by adding control tokens that restrict the overall number of stochastic transition firings. Adding more of such firings adds extra dimensions to the underlying state-space and the developed algorithm makes heavy use of existing algorithms for two well-known problems from computational geometry.

Although the approach presented in this paper is general and could be applied for an arbitrary but finite number of firings of general transitions, the presented implementation is currently restricted to two firings of general transitions. This is because, to the best of our knowledge, no implementation of the hyperplane arrangement problem exists for more than two dimensions, which corresponds to two firings of general transitions in our setting. The contribution of this paper lies in the efficient combination and integration of involved computational geometry algorithms into the evaluation framework for HPnGs. Given the lack of analytical methods for such models, having an exact state-space representation for models with two firings of general transitions and the very efficient and precise numerical integration procedure is already very useful in the application domain.

Most related analysis methods for Stochastic Hybrid Models compute an over- or under approximation of the desired probability [1, 22]. In contrast, we provide exact validity intervals for given properties, and only the integration step requires numerical solutions. The problem of finding a state-space representation that allows for an efficient computation of e.g., reachability probabilities is well known in the area of Hybrid models [16], and several libraries exist that support the basic computations for polyhedra [3]. Tools like Hytech [17] and PHAVer [8] support hybrid automata with linear differential equations, using linear abstraction methods. However, note that we partition according to time and the support of the general transitions instead of the values of the continuous variables.

8. REFERENCES

[1] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, 16:624–641, 2010.

[2] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic petri nets. *Software Engineering, IEEE Transactions on*, 15(7):832–846, Jul 1989.

[3] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

[4] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2010.

[5] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

[6] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. In *24th Annual Symp. on Foundations of Computer Science*, pages 83–91, 1983.

[7] E. Fogel, R. Wein, B. Zukerman, and D. Halperin. 2D regularized Boolean set-operations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 2014.

[8] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.

[9] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, F. Rossi, and R. Ulerich. *GNU Scientific Library*. 2013.

[10] H. Ghasemieh, A. Remke, and B. R. Haverkort. Analysis of a sewage treatment facility using hybrid petri nets. In *7th Int. Conf. on Performance Evaluation Methodologies and Tools*, pages 165–174, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[11] H. Ghasemieh, A. Remke, and B. R. Haverkort. Survivability evaluation of fluid critical infrastructures using hybrid petri nets. In *19th Pacific Rim Int. Symp. on Dependable Computing*, pages 152–161. IEEE, 2013.

[12] H. Ghasemieh, A. Remke, B.R. 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*, volume 7595 of *LNCS*, pages 139–154. Springer, 2012.

[13] M. Gribaudo and A. Remke. Hybrid Petri Nets with General One-Shot Transitions for Dependability Evaluation of Fluid Critical Infrastructures. In *12th Int. Symp. on High Assurance Systems Engineering*, pages 84–93. IEEE, 2010.

[14] P. Hachenberger and L. Kettner. 3D Boolean operations on Nef polyhedra. In *CGAL User and Reference Manual*. CGAL Editorial Board, 2014.

[15] D. Halperin. Arrangements. In *Handbook of Discrete and Computational Geometry*, pages 529–562. CRC Press LLC, 2004.

[16] T.A. Henzinger. The theory of hybrid automata. In *11th Annual IEEE Symp. on Logic in Computer Science*, pages 278–292. IEEE, 1996.

[17] T.A. Henzinger, P. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1-2):110–122, 1997.

[18] S. Hert and S. Schirra. 3D convex hulls. In *CGAL User and Reference Manual*. CGAL Editorial Board, 2014.

[19] S Hert and M Seel. dD convex hulls and Delaunay triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 2014.

[20] F.P. Preparata and D.E. Muller. Finding the intersection of n half-spaces in time $o(n \log n)$. *Theoretical Computer Science*, 8(1):45 – 55, 1979.

[21] R. Wein, E Berberich, E. Fogel, D. Halperin, M. Hemmer, O. Salzman, and B. Zukerman. 2D arrangements. In *CGAL User and Reference Manual*. CGAL Editorial Board, 2014.

[22] L. Zhang, Z. She, S. Ratschan, H. Hermanns, and E. Hahn. Safety verification for probabilistic hybrid systems. In *Computer Aided Verification*, volume 6174 of *LNCS*, pages 196–211. Springer, 2010.