Region-Based Analysis of Hybrid Petri Nets with a Single General One-Shot Transition

Hamed Ghasemieh[1], Anne Remke[1],
Boudewijn Haverkort[1,2], and Marco Gribaudo[3]

[1] Design and Analysis of Communication Systems, University of Twente, The Netherlands
{h.ghasemieh,a.k.i.remke,b.r.h.m.haverkort}@utwente.nl
[2] Embedded System Institute, Eindhoven, The Netherlands
[3] Dipartimento Di Elettronica E Informazione, Ingegneria dell'Informazione, Italy
marco.gribaudo@polimi.it

Abstract. Recently, hybrid Petri nets with a single general one-shot transition (HPnGs) have been introduced together with an algorithm to analyze their underlying state space using a conditioning/deconditioning approach. In this paper we propose a considerably more efficient algorithm for analysing HPnGs. The proposed algorithm maps the underlying state-space onto a plane for all possible firing times of the general transition s and for all possible systems times t. The key idea of the proposed method is that instead of dealing with infinitely many points in the t-s-plane, we can *partition* the state space into several regions, such that all points inside one region are associated with the same system state. To compute the probability to be in a specific system state at time τ, it suffices to find all regions intersecting the line $t = \tau$ and decondition the firing time over the intersections. This partitioning results in a considerable speed-up and provides more accurate results. A scalable case study illustrates the efficiency gain with respect to the previous algorithm.

1 Introduction

In a recent study we have evaluated the impact of system failures and repairs on the productivity of a fluid critical infrastructure, in particular, a water treatment plant [8]. In that study, we have developed an analysis algorithm for a class of Hybrid Petri nets [6] with a single general one-shot transition. Despite the current restriction to a single general one-shot transition this class turns out to be very useful for this application field. However, the algorithm proposed in [8] requires a discretization of the support of the distribution that determines the firing time of the general transition. This is on the one hand computationally very expensive for small step sizes, and on the other hand may lead to less accurate results for larger step sizes.

This paper presents a considerably more efficient algorithm that partitions the underlying state space of an HPnG into regions with equivalent markings, depending on the current time t and the firing time of the general transition s. We provide a graphical representation of these regions, a so-called Stochastic Time Diagram (STD), which consists of two main parts, namely, the deterministic and

M. Jurdziński and D. Ničković (Eds.): FORMATS 2012, LNCS 7595, pp. 139–154, 2012.

the stochastic part. In the first part, the evolution of the system and the continuous marking solely depends on t, since the general transition has not fired, yet. In the second part, however, the continuous marking and the remaining firing time of deterministic transitions may depend linearly on the values s and t. The main advantage of our new method is that for each system time, we can easily find all possible states of the system. Instead of dealing with infinitely many points in the t-s plane, for computing the probability to be in a specific system state at time τ, it suffices to find all regions intersecting the line $t = \tau$ and decondition the firing time of the general transition over the intersections that correspond to a the specific system state. The partitioning into regions with the same system state avoids accuracy problems of the old algorithm that stem from a discretization with fixed step sizes and also significantly decreases the computation time.

The idea of a partitioning the underlying state-space of hybrid systems is not new and, e.g., the underlying state-space of Timed Automata (TA) [3,2] can be partitioned into zones, where each zone represents a symbolic system state. However, due to the fact that all real-valued clocks have an identical drift of 1 the shape of a zone is much more restricted than the shape of the regions resulting for HPnGs. Similarly, a partitioning of the state-space of hybrid systems has been introduced in [1]. The difference to our work is that we partition according to time and the support of the general transition instead of the values of the continuous variables. Also for Dense-Time Reactive Systems [12,11] that are a generalization of Timed Petri Nets [5] a partitioning of the state space into state classes has been introduced. whereas, such systems do not include continuous variables they allow to equip timed transitions with an interval indicating their firing time. Again, since time evolves linearly with derivative 1, the shape of state classes is similarly restricted as zones for TA. Dynamical Systems having Piecewise-Constant Derivatives (PCDs) [4] represent a class of hybrid systems where the evolution of the continuous variables is piecewise-linear and the control component of a state is fully determined by the values of the continuous variables. This also results in a set of regions, where each region is associated with a constant vector field which identifies the rates at which the various variables change. Similarly to HPnGs, PCDs allow different slopes for the continuous variables within one region and more general guards for discrete transitions. However, in contrast to PCDs, the discrete state of an HPnG is not fully described by the values of the continuous variables and hence allows for a more general discrete component.

This paper is further organized as follows: In Section 2 we provide a brief description of the modelling formalism and system evolution of HPnG and introduce some notation that is used throughout the paper. In Section 3, we intuitively describe the idea behind our new algorithm, and illustrate the idea with a simple example. In Section 4, we formalize the details of the algorithm, and prove that the partitioning results in polygons. Section 5 addresses the computation of measures of interests, e.g., the probability of having an empty storage at a given time. To study the efficiency of the new algorithm, Section 6 compares the run time with the existing algorithm [8] on a scalable case study.

2 Hybrid Petri Nets with General One-Shot Transitions

A 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.

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 [10]. 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 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.

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 [6]. The adaptation of fluid rates in these cases, results in a piecewise constant fluid derivative per continuous place.

The *state* of an HPnG is defined by $\Sigma = (\mathbf{m}, \mathbf{x}, \mathbf{c}, \mathbf{d}, \mathcal{G})$, 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. 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 this vector d is determined uniquely by x and m, it is included in the definition of a state to make it more descriptive. A general transition is only allowed to fire once, hence, the flag $\mathcal{G} \in \{0, 1\}$ indicates whether the general transition has already fired. So, the initial state of the system is $\Sigma_0 = (\mathbf{m}_0, \mathbf{x}_0, \mathbf{0}, \mathbf{d}_0, 0)$. For a more detailed description HPnGs and their evolution, we refer to [9].

3 Graphical Representation of the System Evolution

The evolution of an HPnG can be represented by a so-called *Stochastic Time Diagram* (STD) that illustrates the system state at each time conditioned on the firing time of the general transition. Section 3.1 introduces STDs and shows how to generate this diagram for a simple example in Section 3.2.

3.1 Stochastic Time Diagram

Given an initial state of an HPnG and a predefined value for the firing time of the general transition (denoted s) the state of the system can be determined for all the future times t starting from a given initial state. Hence, in order to characterize the system state, we consider a two-dimensional diagram with t on the vertical axis and s on the horizontal axis. Each point in this diagram is associated with a unique system state. A generic version of this diagram is shown in Figure 1. The *stochastic* area contains all states for which we assume that the general transition has fired, i.e., the current system time is larger than the firing time of general transition, $t > s$. The *deterministic* area, in contrast, represents all states where the general transition has not fired yet, i.e., $t < s$. In this area the evolution of the system is independent of parameter s.

To compute measures of interest for HPnGs, the state space needs to be deconditioned with a probability density function $g(s)$. The main idea of the proposed method is that instead of dealing with infinitely many points in the t-s-plane, we can *partition* the state space into several regions, such that all points inside one region are associated with the same system state. More formally a *system state* Γ is defined as a set of HPnG states with the same discrete marking m, drift d and general transition flag $\mathcal{G}$, where the continuous marking and the clock values linearly depend on s and t according to the same equations. Then to compute the probability to be in a specific system state at time τ, it suffices to find all regions intersecting the 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 2.

While the generic STD from Figure 1 holds for every HPnG, the partitioning into invariant regions as shown in Figure 2, depends on the structure of the model at hand. These invariant regions exist, because the state of the system does not change until an *event* occurs. At each system state two types of potential events should be considered: a fluid place reaching its lower/upper boundary or an enabled deterministic transition reaching its firing time. Both events induce

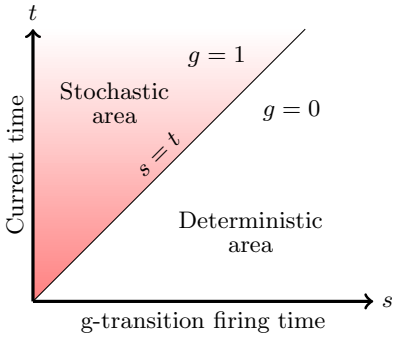

Fig. 1. Generic presentation of STD

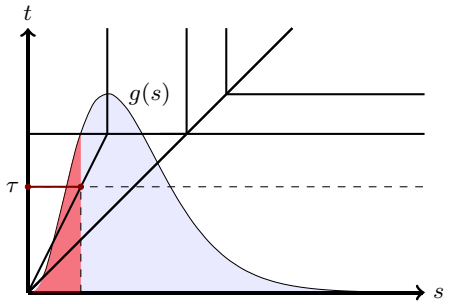

Fig. 2. Deconditioning according to the probability density function $g(s)$

a state change, i.e., the system enters another region. Therefore, the boundary between regions represents the occurrence of an event. Section 4 presents an algorithm that constructs the STD for a given HPnG and proves that for HPnGs all boundaries are linear.

3.2 Reservoir Example

In order to illustrate the above concepts, we construct the STD for an example HPnG taken from [8]. Using the same graphical representation as in [8], Figure 3 shows an HPnG model of a reservoir that is filled by a pump and drained due to some demand. The reservoir C_r can contain at most 10 units of fluid (say, m^3) and as long as the discrete places P_p contains a token and the reservoir is not full, fluid is pumped in with rate 2. As long as P_d contains a token and the reservoir is not empty, fluid is taken from the reservoir at rate 1. The demand is deterministically switched off after 5 time units by transition D_e and the pump fails according to an arbitrary probability distribution. At $t = 0$, the reservoir is empty.

Assuming that the general transition has not been fired, i.e., $s > t$, there are two possible events: either transition D_e fires at time 5 or reservoir C_r reaches its upper boundary. Since the overall rate of change (drift) of fluid into C_r is 1 in this sense, it takes 10 time units to become full. So the first occurring event is D_e, firing at time 5. This event is represented in Figure 4 by the horizontal line $t = 5$, labelled D_e. Then, since transition F_d is no longer enabled, the drift at C_r becomes 2. Since the reservoir contains 5 units of fluid, it takes 2.5 time units to reach its upper boundary, which occurs at time 7.5. In Figure 4, this is shown by line $t = 7.5$, labelled C_r. After entering the area above this line, no deterministic event is possible anymore, i.e., we reach an *absorbing* region.

After partitioning the deterministic area, the line $t = s$, is divided into three segments, as illustrated in Figure 4. Then, in order to partition the stochastic area, all of these segments have to be considered as possible firing times of the

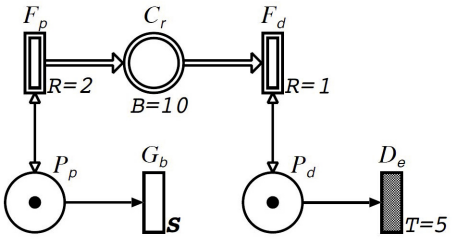

Fig. 3. Reservoir model

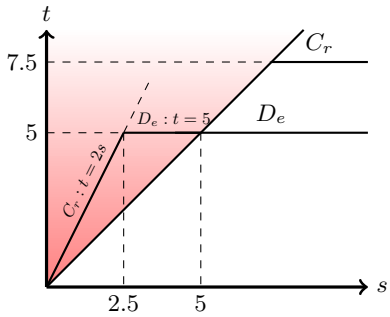

Fig. 4. Polygon over the interval $[0, 5)$

general transition. Starting from the initial state of the system, first consider that the general transition fires at time $s \in [0, 5)$. Hence, by passing the line $t = s$, the system enters the stochastic area. Then two events are possible: either D_e fires or C_r reaches its lower boundary. Before the general transition fired, place C_r had drift 1 and since s time units have passed it now contains s units of fluid. After the general transition has fired, the transition F_p is disabled, and the drift at C_r becomes -1.

Now, either the reservoir becomes empty or the deterministic transition fires, which stops the demand. To find out which of these events is going to occur first, we have to compare their occurrence time t, which may depend on s. Let Δt be the time needed for C_r to become empty, we have: $\Delta t = s$. The previous event has occurred at time s, so $\Delta t = t - s$ and the reservoir becomes empty at $t = 2s$. Since transition D_e fires at time 5, the occurrence time equation of this event is simply $t = 5$ and does not depend on s. The minimum of both equations then determines the next event, as shown in the shaded area in Figure 4. The procedure forms a polygon over the segment $t = s$ for $s \in [0, 5)$. Note that each side of this polygon represents the occurrence time of an event, hence, the procedure can be repeated recursively for each of them ,i.e., we can form another polygon over each side, and continue this procedure until we have obtained the complete partitioning of the stochastic area, up to the maximum analysis time. Figure 2 shows the complete STD for the reservoir example with nine different regions in the stochastic area. After all regions have been determined, measures of interests can be computed by deconditioning over the distribution function $g(s)$.

4 Generating the Diagram

We now present a formal algorithm for the generation of the STD. It consists of two main phases: partitioning the deterministic area (described in Section 4.1), and partitioning the stochastic area (described in Section 4.2).

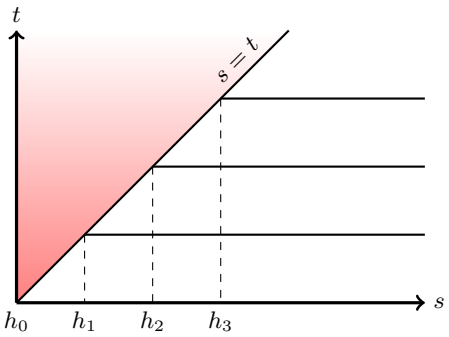

Fig. 5. Deterministic regions

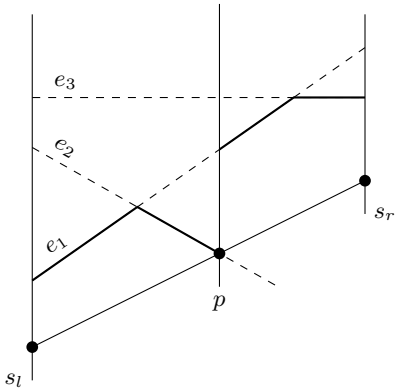

Fig. 6. Formation of invariant polygons

4.1 Partitioning the Deterministic Area

In this phase the evolution of the system is purely deterministic, since by assumption the general transition has not fired yet. Therefore the so-called *deterministic regions* are constructed with lines parallel to the s-axis, as shown in Figure 5. Each deterministic region $\mathcal{R}_i$ is determined uniquely by the interval $[h_i, h_{i+1})$, where h_i is the occurrence time of the event that changes the state of the system from region $\mathcal{R}_{i-1}$ into region $\mathcal{R}_i$.

The procedure for partitioning the deterministic area is outlined in Algorithm 2. Until the system reaches T_{max}, the procedure FINDNDTEVENT, c.f. Algorithm 3, provides the next event for each marking of the system. During the evolution of the system at each point, two types of events are possible: an enabled deterministic transition reaches its firing time or a continuous place reaches a boundary. We iterate over all continuous places, and find the time at which each reaches its lower and upper boundaries (lines 2-8, Algorithm 3). Also for all enabled deterministic transitions we have to find the remaining time to fire (lines 9-13, Algorithm 3). Finally, the next event is the one with the smallest remaining time to occur, denoted e and Δt_e, respectively. A new region $\mathcal{R}_i$ is created and added to the set of deterministic regions $\mathcal{R}^D$(lines 6-8, Algorithm 2). Then the current system sate is updated (Algorithm 6) and the current system time is advanced by Δt_e time units (lines 9-10).

4.2 Partitioning the Stochastic Area

In this phase we partition the area above the line $t = s$. As shown in Figure 5, in the previous phase the line $t = s$ has been segmented into several line segments, by deterministic regions. We iterate over all these line segments, and if the general transition was enabled in the system state of the corresponding deterministic region, it is fired, and the area above the corresponding line segment is further

Algorithm 1. genDiagram()

$\mathcal{P}^{\mathcal{S}}, \mathcal{R}^{\mathcal{D}} \leftarrow \emptyset$
$\mathcal{R}^{\mathcal{D}} \leftarrow$ partDtrmArea(Γ_0)
for all $\mathcal{R}_i \in \mathcal{R}^{\mathcal{D}}$ **do**
    if T^G in enabled in the $\mathcal{R}_i.\Gamma$ **then**
        $\delta.[s_l, s_r) \leftarrow R_i.[h_i, h_{i+1})$
        $\delta.eq \leftarrow t = s$
        $\Gamma' \leftarrow$ update($T^G, \Delta t = s - R_i.h_i, R_i.\Gamma$)
        $\mathcal{P}^{\mathcal{S}} \leftarrow \mathcal{P}^{\mathcal{S}} \cup$ partStochArea(δ, Γ')

Algorithm 2. partDtrmArea(Γ_0)

Require: Initial system state Γ_0.
Ensure: Set of deterministic regions.
$\Gamma \leftarrow \Gamma_0$
$\mathcal{R}^D \leftarrow \emptyset$
$t \leftarrow 0$; $i \leftarrow 0$
while $t < t_{max}$ **do**
    $(e, \Delta t_e) \leftarrow$ findNDtEvent(Γ)
    $R_i.[h_i, h_{i+1}) \leftarrow [t, t + \Delta t_e)$
    $R_i.\Gamma \leftarrow \Gamma$
    $\mathcal{R}^D \leftarrow \mathcal{R}^D \cup \{R_i\}$
    $\Gamma \leftarrow$ update($e, \Delta t = \Delta t_e, \Gamma$);
    $t \leftarrow t + \Delta t_e$; $i \leftarrow i + 1$

Algorithm 3. findNDtEvent(Γ)

Require: The current system state Γ.
Ensure: Next event and its remaining time to occur.
$\Delta t_{min} \leftarrow \infty$
for all $P_i \in \mathcal{P}^C$ **do**
    if $\Gamma.d_i > 0$ **then**
        $\Delta t_e \leftarrow \frac{\phi_b^{\mathcal{P}}(P_i) - \Gamma.x_i}{\Gamma.d_i}$
    if $\Gamma.d_i < 0$ **then**
        $\Delta t_e \leftarrow \frac{\Gamma.x_i}{\Gamma.d_i}$
    if $\Delta t_e < \Delta t_{min}$ **then**
        $(e, \Delta t_{min}) \leftarrow (P_i, \Delta t_e)$
for all $T_i \in \mathcal{T}^D$ **do**
    if T is enabled **then**
        $\Delta t_e \leftarrow \phi_d^{\mathcal{T}}(T_i) - \Gamma.c_i$
        if $\Delta t_e < \Delta t_{min}$ **then**
            $(e, \Delta t_{min}) \leftarrow (T_i, \Delta t_e)$
return $(e, \Delta t_{min})$

partitioned. An arbitrary segment δ is defined by equation $\delta.eq : t = \alpha s + b$ and endpoints $\delta.[s_l, s_r)$. Each segment δ corresponds to an event in a way that the equation $\delta.eq$ represents its occurrence time, and the general transition had to fire between the endpoints $\delta.[s_l, s_r)$. For example if the general transition fires when the system is in the deterministic region $\mathcal{R}_i$, we enter the area above the segment with equation $t = s$ and endpoints $[h_i, h_{i+1})$.

Proposition 1. *After firing the general transition at time s, for each system state the occurrence time of the next events are linear functions of s.*

Proof. We prove the proposition by structural induction on associated segments of events, with firing of the general transition as the basis. W.l.o.g. assume that before firing the general transition, the system is at the deterministic region $\mathcal{R}_i$. When the general transition fires at time s, we have been in this region for $\Delta t = s - h_i$ time units, so at the firing time of the general transition the fluid level of continuous place P_k^C linearly depends on s, as follows $x'_k = d_k(s - h_i) + x_k$. Also the clock value of an enabled deterministic transition T_k^D is $c'_k = (s - h_i) + c_k$. Therefore, at the very moment after firing of the general transition, all fluid levels and clock values are linear functions of s, and hence the occurrence time of the next events are linear function of s too, as shown for the general case below.

As the inductive step, w.r.t. induction hypothesis, suppose that an event has occurred at time $t = \alpha s + \beta$ for $s \in [s_l, s_r)$. Recall, that two events are possible: a continuous place reaches its lower / upper boundary or an enabled deterministic transition reaches its firing time. Let the fluid level in a continuous place P_k^C be $x_k = a_k^p s + b_k^p$. The amount of time this place needs to reach one of its boundary is denoted by Δt_k^p and can be calculated as follows:

$$d_k \Delta t_k^p = \begin{cases} \phi_b^{\mathcal{P}}(P) - x_k & \text{if } d_k > 0, \\ -x_k & \text{if } d_k < 0. \end{cases}$$

According to the occurrence of the previous event, we have $\Delta t_k^p = t_k^p - (\alpha s + \beta)$ for $s \in [s_l, s_r)$, where t_k^p is the occurrence time of this fluid event. As a result, above the line $t = \alpha s + \beta$ and $s \in [s_l, s_r)$, the considered fluid event occurs at time t_k^p as follows:

$$t_k^p = \begin{cases} (\alpha - \frac{a_k^p}{d_k})s + (\frac{-b_k + \phi_b^{\mathcal{P}}(P_k^C)}{d_k} + \beta) & \text{if } d_k > 0, \\ (\alpha - \frac{a_k^p}{d_k})s + (-\frac{b_k}{d_k} + \beta) & \text{if } d_k < 0. \end{cases} \quad (1)$$

The firing time for a deterministic transition can also be derived in a same way. Let the clock value of the deterministic transition T_k^D be $c_k^t = a_k^t s + b_k^t$. The firing time can be calculated as follows:

$$t_k^t = (\alpha - a_k^t)s + (\phi_d^{\mathcal{T}}(T_k^D) + \beta - b_k^t). \quad (2)$$

Therefore, occurrence time of both types of events linearly depend on s. Now, with the same argument, if we set $\alpha = 1$ and $\beta = 0$ the basis of the induction is also satisfied, and hence the proof is complete. □

As an immediate result of the above proposition we can state that all regions in the stochastic area are polygons. Since the system state in these polygons does not change, we call them *invariant polygons* in the following. We present the algorithm for partitioning the area above an arbitrary *underlying segment* δ. The algorithm for this procedure is outlined in Algorithm 4. At first, we check whether the maximum analysis time has not been reached (line 1). Then we identify the potential events that can occur in the current marking of the system. For this we consider all continuous places and enabled deterministic transitions.

The procedure for finding all potential events, at each system state, is called findPotEvents and shown in Algorithm 5. It uses the same arguments as in the proof of Proposition 1. In lines 2-7 we iterate over all continuous places, and for each with a non-zero drift, the time for reaching its boundary is computed, according to Equation (1). Also in lines 8-11 this is repeated for all enabled deterministic transitions, according to Equation (2). Finally the set of all events and the equations of their occurrence time is returned.

In order to find the occurrence times of the next events conditioned on the value of $s \in [s_l, s_r)$, we have to take the minimum over all these linear equations. Taking the minimum over a set of lines results in several convex polygon(s) over the underlying segment δ. Note that, these equations are only valid in the area above the underlying segment δ. An example with three possible events: e_1, e_2 and e_3 is presented in Figure 6. Event e_2 intersects with the underlying segment at point p, so e_2 can not occur for $s > p$ and in the minimum taking procedure after this point e_2 does not have to be considered any more. As a result two polygons will be formed over the underlying segment.

The procedure that identifies the set of next events over an underlying segment, is called findNEvents, it simply iterates over all lines indicating the firing time of potential events and for each $s \in [s_l, s_r)$ finds the minimum line. It returns a set of segments from which the invariant polygon(s) over the underlying segment can be formed by iteration over the set of segments, this is done in procedure createPolygons. These two procedures are described in detail in [7].

The procedures findPotEvents, findNEvents and createPolygons, are called in lines 3-5, in Algorithm 4. Now, having obtained the set of segments of all next events, we can partition the area above each of these segments. Through line 6-8 we iterate over all these segments, and recursively call the function partStochArea for each segment, after updating the system state.

The procedure for updating the system state is provided in Algorithm 6. This procedure needs the event e and the amount of time Δt that can possibly depend on s, to advance the marking. In lines 2-3, for each continuous place P_i it alters the fluid level in that place according to Δt and the fluid drift d_i. Also in lines 4-5, it adds Δt to the clock value of each enabled deterministic transition. Moreover, if the event e is a transition it is fired, in line 8, to update the discrete marking. Finally we update the fluid drifts for the new marking, by calling the function

Algorithm 4. partStochArea(δ, Γ)

Require: The underlying segment δ and the current system state Γ.
Ensure: Partitioning of the area above the given underlying segment δ.

if $\delta.s_l > T_{max}$ **then**
  return
potentialSet $\leftarrow$ findPotEvents(δ, Γ)
nextEvSet $\leftarrow$ findNEvents(potentialSet, δ)
polygonSet $\leftarrow$ polygonSet $\cup$ createPolygons(nextEvSet, δ, Γ)
for all $(e, \delta_e) \in$ nextEvSet **do**
  $\Gamma' \leftarrow$ update(e, $\Delta t = \delta_e$.eq $-\delta$.eq, Γ)
  polygonSet $\leftarrow$ polygonSet $\cup$ partStochArea(δ_e, Γ')
return polygonSet

Algorithm 5. findPotEvents(δ, Γ)

Require: The underlying segment δ and the current system state Γ.
Ensure: Set of potential events.

potSet $\leftarrow \emptyset$
for all $P_i \in \mathcal{P}^C$ **do**
  if $\Gamma.d_i > 0$ **then**
    $eq \leftarrow t = \delta.eq - \frac{\Gamma.x_i}{\Gamma.d_i} + \frac{\phi_b^{\mathcal{P}}(P_i)}{\Gamma.d_i}$
  if $\Gamma.d_i < 0$ **then**
    $eq \leftarrow t = \delta.eq - \frac{\Gamma.x_i}{\Gamma.d_i}$
  potSet $\leftarrow$ potSet $\cup\{(P_i, \text{eq})\}$
for all $T_i \in \mathcal{T}^D$ **do**
  if T is enabled **then**
    $eq \leftarrow t = \delta.eq - \Gamma.c_i + \phi_d^{\mathcal{T}}(T_i)$
    potSet $\leftarrow$ potSet $\cup\{(T, \text{eq})\}$
return potSet

Algorithm 6. update($e, \Delta t = as + b, \Gamma$)

Require: The event e to be committed, the s-dependent equation Δt of time to advance, and the marking Γ to be updated.
Ensure: Advancement of system marking for the specified time.

$\Gamma' \leftarrow \Gamma$
for all $P_i \in \mathcal{P}^C$ **do**
  $\Gamma'.x_i \leftarrow \Gamma.x_i + \Delta t \times \Gamma.d_i$
for all $T_i \in \mathcal{T}^D$ **do**
  if T_i is enabled **then**
    $\Gamma'.c_i \leftarrow \Gamma.c_i + \Delta t$
if e is a transition **then**
  $\Gamma' \leftarrow fire(\text{e}, \Gamma)$
$\Gamma' \leftarrow$ upadteDrifs(Γ')
return Γ'

updateDrifts. This is done according to the new discrete and continuous marking and rate adaptation in rules described in [9].

Finally, Algorithm 1 generates the STD. First the procedure PARTDTRMAREA is called, and the deterministic regions are saved in the set $\mathcal{R}^D$. Then for each region $\mathcal{R}_i$, if the general transition is enabled, the segment with equation $t = s$ with the interval $[h_i, h_{i+1})$ is created (lines 4-6). The marking of the system is updated in line 7, by calling procedure UPDATE. Since the general transition should be fired we pass T^G as argument. Also the time that has passed after entering region $\mathcal{R}_i$ until firing the general transition, is $\Delta t = s - h_i$, as it is passed as second argument. Finally the procedure PARTSTOCHAREA is called with two arguments the created segment and updated system state.

5 Computing Measures

After the STD has been generated, the state of the system depends on the distribution of the firing time of the general transition, $g(s)$ and on the system time. By deconditioning s over the values of $g(s)$, the state probability distribution can be derived, as briefly sketched in Section 3.1. In order to compute more sophisticated measures of interest, we introduce property ψ, see below, which is defined as a combination of discrete and continuous markings. Note that this property is an extended version of what has appeared in [8]. The main difference is that we add negation which makes it complete and the result more expressive:

$$\psi = \neg\psi \mid \psi \wedge \psi \mid n_p = a \mid x_k \leq b. \tag{3}$$

To compute the probability of being in a system state for which property ψ holds at time τ, at first we have to identify all invariant polygons and deterministic regions the system can be in, i.e. all regions intersecting line $t = \tau$. Then we verify whether property ψ holds for any of these regions, and if so determine the intervals in which the property is satisfied. Finally, $g(s)$ is integrated over all these regions.

An atomic property which reasons about discrete places, either holds in the complete invariant polygon or not at all. Recall, that the amount of fluid in a continuous place may linearly depend on s. Hence, an atomic property explaining the amount of fluid in a continuous place, may be valid only in a certain part of the considered polygon. More specifically, let the amount of fluid in the place P_k^c be $x_k = \alpha s + \beta$. For the computation of the probability to be in a system state for which $x_k \leq b$ holds, $s^* = (b - \beta)/\alpha$ defines the threshold value of s where the validity of the property changes. In case s^* lies inside a given polygon, depending on the sign of α the property is satisfied either before or after s^*.

Let the line $t = \tau$ intersect the polygon $\mathcal{P}_i$ in the interval $[s_1^i, s_2^i]$, then to negate a property we need to find the complement of the interval within $[s_1^i, s_2^i]$ for which the original property holds. For the conjunction of two properties we need to find the intersection of the two intervals that are associated with the two original properties. Therefore, a nested property ψ may be satisfied in a set of intervals.

Let $\mathcal{P}^\tau$ be the set of all invariant polygons intersecting the line $t = \tau$. For a given invariant polygon $\mathcal{P}_i \in \mathcal{P}^\tau$, the set of intervals in which the property ψ holds is indicated by $\mathcal{S}^i$ and each interval in this set is denoted $[s_l^i, s_r^i)$. Trivially $\mathcal{S}^i$ is empty if the property is not satisfied in $\mathcal{P}_i$. Let $\mathcal{R}_i^\tau$ denote the deterministic region for which $\tau \in [h_i, h_{i+1})$. Also, let $\mathcal{I}^\psi(s,\tau)$ be the characteristic function for condition ψ at the point (s,τ), which evaluates to 1 or 0 whether ψ holds or not, respectively. Furthermore, $\mathcal{I}^\psi(\mathcal{R}_i^\tau)$ indicates whether condition ψ is satisfied in $\mathcal{R}_i^\tau$. So, the probability distribution to be in a system state for which property ψ is satisfied at time τ can be computed as follows:

$$\begin{aligned}
\pi^\psi(\tau) &= \int_0^\infty \mathcal{I}^\psi(s,\tau) g(s) ds \\
&= \int_0^\tau \mathcal{I}^\psi(s,\tau) g(s) ds + \int_\tau^\infty \mathcal{I}^\psi(s,\tau) g(s) ds \\
&= \sum_{\mathcal{P}_i \in \mathcal{P}^\tau} \int_{\mathcal{S}^i} g(s) ds + \mathcal{I}^\psi(\mathcal{R}_i^\tau) \int_\tau^\infty g(s) ds \\
&= \left(\sum_{\mathcal{P}_i \in \mathcal{P}^\tau} \sum_{[s_l^i, s_r^i) \in \mathcal{S}^i} \left(\phi_g(s_r^i) - \phi_g(s_l^i) \right) \right) + \mathcal{I}^\psi(\mathcal{R}_i^\tau)(1 - \phi_g(\tau)) \qquad (4)
\end{aligned}$$

The above set of equations shows how the partitioning into regions can be used for smarter deconditioning. Equation (4) consists of two terms. The first term expresses the probability of holding ψ at time τ, in the stochastic area, by simply iterating over all invariant polygons intersecting the line $t = \tau$ and summing the probability over all intervals in which the property holds. The second term expresses the probability of being in $\mathcal{R}_i^\tau$ if the property ψ holds in it.

6 Case Study

The complexity of the proposed algorithm clearly depends on the structure of the model. The process of computing measures of interest also linearly depends on the number of regions. In the following we show the scalability and efficiency of the proposed method using the case study as in [8]. We scale the number of transitions and continuous places in the case study, and discuss its influence on the number of regions and the computation time of the algorithm.

Figure 7, presents a model of a water treatment facility with different phases. The continuous place C_i represents the storage of a water softening phase. By design this is a slow process with large storage. The continuous place C_f and transition F_f, represent a generic water filtering phase. Opposed to the softening phase this filtration phase is a fast process with small storage. The continuous place C_s, represents the final storage from which water is distributed to the customers with different rates, depending on the time of the day. The deterministic transition T_b represents a failure at time α, in the softening phase. When it fires, the continuous transition F_i is disabled and the general transition G_r becomes enabled. G_r models the time it takes to repair the system failure according to

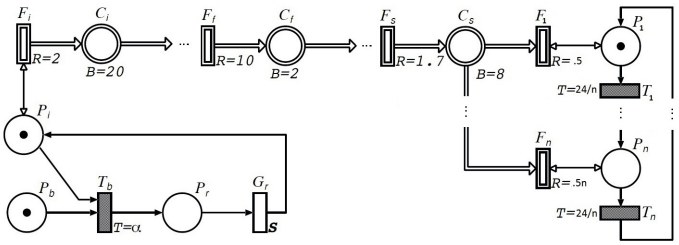

Fig. 7. HPnG model for a symbolic water treatment facility

Table 1. Scaling the filtration phases. All times in milliseconds.

	Region-based			Param. reach.	
#Filters	#Region	STD	MCT	Tree	MCT
1	327	43	161	10	11106
2	433	80	239	19	13153
3	539	69	294	19	15415
4	663	77	373	22	17188
5	769	86	461	25	19352
6	903	95	509	26	21501
7	1026	106	586	30	23385
8	1159	121	662	31	25875

Table 2. Scaling the demand rates. All times in milliseconds.

	Region-based			Param. reach.	
#Demands	#Region	STD	MCT	Tree	MCT
2	202	26	104	15	32348
3	403	76	202	21	43874
4	909	72	431	23	52526
5	1204	82	538	38	66793
6	1624	91	711	49	79479
7	1797	90	681	30	69484
8	2225	115	1004	99	115542
9	2776	125	1195	102	120129
10	3457	143	1451	133	136896

the arbitrary probability density $g(s)$. Note that discrete place P_b restricts the model such that the failure can occur only once.

The model presented in Figure 7 is made scalable in two ways. First, by cascading more filtration phases, and second by dividing the day into more intervals with different demand rates. In order to show the efficiency of our algorithm, we scale the model in these two ways, and for each instance, compute the probability distribution for the amount of fluid in place C_s. This is an important measure of interest, because an empty final storage C_s means failure to deliver water to the consumers. Moreover, to provide a comparison with the parametric reachability algorithm, as presented in [8], we also calculate this probability distribution using this algorithm. All the computations have been performed on a machine equipped with a 2.0 GHz intel® CORE™ i7 processor, 4 GB of RAM, and Windows 7. The results are shown in Tables 1 and 2.

Scaling in both dimensions increases the number of regions, as shown in the second column of both tables. The time needed to construct the STD and the tree with all parametric locations, are given in the third and fifth columns of both tables. The time needed to compute the measures of interest is denoted MCT (Measure Computation Time). When scaling the number of filters the generation of the STD takes about 3 to 4 times longer than the construction of the parametric locations. This is due to the more involved computations that are necessary to construct the polygons in the STD. When scaling the number of demands the generation of the STD takes about 2 to 3 times longer than the construction

of the parametric locations. This is, however, more than compensated for when the measures of interest are computed. The new algorithm is, depending on the size of the model, between 20 and 100 times faster than the algorithm in [8]. Apparently, for smaller models the speed up is larger than for bigger models. This is because the complexity of the old algorithm is logarithmic in number of parametric locations. Furthermore, in case a closed form of the CDF exists, the choice of the distribution does not influence the complexity of the region-based algorithm. Clearly, the MCT of the parametric reachability algorithm depends on the chosen discretization step. The results presented in the tables have been obtained for a discretization step of 0.005. A larger discretization step reduces the MCT, but also decreases the accuracy of the results. For a discretization step of 0.005, the maximum difference between the results from both algorithms is 0.5%. Running the parametric location algorithm with a discretization step of 0.2 leads to approximately the same MCT with both algorithms, the resulting maximum relative error, however is 3%.

7 Conclusions

This paper presents an algorithm for the analysis of HPnGs that partitions the state space into regions, where all the states in a given region have the same deterministic marking and the continuous marking and the remaining firing time for all states in the same region follow the same linear function of s and t.

The restrictions of the model class to a single one-shot transition and the requirement of a unique priority assignment to each deterministic and immediate transition ensure that the computed partitioning is a single two-dimensional STD. Relaxing the requirement of the unique priority assignment potentially leads to concurrency between timed transitions. In [8] this has been resolved by a probabilistic choice between transitions with the same minimum firing time. Since the firing of different transitions leads to a different further evolution of the system, a different STD is needed. To compute measures of interest, the deconditioning then needs to take several STDs into account and weight them according to the probabilities assigned to the firing of each transition. Future work will investigate how this can be done efficiently. Also allowing more general transitions or relaxing the one-shot restriction will change the resulting STD. Each firing of a general transition will add an extra dimension to the STD and the deconditioning then needs to be done for several dimension. This is also an interesting line for future research.

Even though the model class currently is restricted in several ways, it is still very useful for the application field of fluid critical infrastructures, since the physical processes are fairly deterministic and stochasticity is only needed to model failures and repairs. To the best of our knowledge no analyzable model class exists that allows for an arbitrary amount of continuous places without resetting the amount of fluid upon discrete changes, as needed in this field. Furthermore, we would like to emphasis that the presented algorithm presents an enormous improvement with respect to the parametric reachability analysis

in [8], it allows for a much quicker analysis and due to the partitioning the obtained results are also more accurate.

Acknowledgement. This work has been supported by the ROCKS project through the NWO grant DN 63-257 and . Anne Remke is funded by a NWO Veni grant.

References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Yovine, S., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J.: The algorithmic analysis of hybrid systems. Theoretical Computer Science 138, 3–34 (1995)
2. Alur, R., Courcoubetis, C., Henzinger, T.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. Hybrid Systems 736, 209–229 (1993)
3. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
4. Asarin, E., Maler, O.: Reachability analysis of dynamical systems having piecewise-constant derivatives. Theoretical Computer Science 138(1), 35–65 (1995)
5. Berthomieu, B.: Modeling and verification of time dependent systems using time Petri nets. IEEE Transactions on Software Engineering 17(3), 259–273 (1991)
6. David, R., Alla, H.: Discrete, Continuous, and Hybrid Petri Nets, 2nd edn. Springer (2010)
7. Ghasemieh, H., Remke, A., Haverkort, B., Gribaudo, M.: Region-based analysis of hybrid Petri nets with a single general one-shot transition: extended version. Technical report, University of Twente (2012), `http://wwwhome.cs.utwente.nl/~anne/techreport/std.pdf`
8. Gribaudo, M., Remke, A.: 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, pp. 84–93. IEEE CS Press (November 2010)
9. Gribaudo, M., Remke, A.: Hybrid petri nets with general one-shot transitions: model evolution. Technical report, University of Twente (2010), `http://wwwhome.cs.utwente.nl/~anne/techreport/hpng.pdf`
10. Kartson, D., Balbo, G., Donatelli, S., Franceschinis, G., Conte, G.: Modelling with Generalized Stochastic Petri Nets, 1st edn. John Wiley & Sons, Inc. (1994)
11. Vicario, E.: Static analysis and dynamic steering of time-dependent systems. IEEE Transactions on Software Engineering 27(8), 728–748 (2001)
12. Vicario, E., Sassoli, L., Carnevali, L.: Using stochastic state classes in quantitative evaluation of dense-time reactive systems. IEEE Transactions on Software Engineering 35(5), 703–719 (2009)