Fluid Survival Tool: A Model Checker for Hybrid Petri Nets

Björn F. Postema, Anne Remke,
Boudewijn R. Haverkort, and Hamed Ghasemieh

Centre for Telematics and Information Technology
Design and Analysis of Communication Systems
University of Twente, the Netherlands
{b.f.postema,a.k.i.remke,b.r.h.m.haverkort,h.ghasemieh}@utwente.nl

Abstract. Recently, algorithms for model checking Stochastic Time Logic (STL) on Hybrid Petri nets with a single general one-shot transition (HPNG) have been introduced. This paper presents a tool for model checking HPNG models against STL formulas. A graphical user interface (GUI) not only helps to demonstrate and validate existing algorithms, it also eases use. From the output of the model checker, 2D and 3D plots can be generated. The extendable object-oriented tool has been developed using the Model-View-Controller and Facade patterns, Doxygen for documentation and Qt for GUI development written in C++.

1 Introduction

HPNGs allow to model continuous and discrete variables of a system with a single stochastic event. Therefore, HPNGs can be used to model fluid critical infrastructures [2] like water, gas and oil networks. Since critical infrastructures may fail and their continuous operation is of utmost importance for both industry and society, survivability is an important property for these systems.

The logic STL [6] has recently been introduced to easily formulate measures of interest for HPNG models, such as survivability measures. To automate the evaluation of STL formula for HPNGs, we developed the Fluid Survival Tool (FST) [9] for model checking HPNG models against an STL specification.

FST has an extendable software design based on the software engineering principles of the Model-View-Controller and Facade patterns, and uses Doxygen for documentation. The GUI has been implemented with Qt and the tool has been written in C++. It implements the region-based algorithm for the transient analysis of HPNGs [5] and for model checking HPNGs against STL [6].

The paper is organised as follows. First, the HPNG modelling formalism and STL specification are discussed in Section 2. Section 3 describes the tool functionality and design, as well as the Graphical User Interface (GUI) and presents results for a running example.

2 Hybrid Petri nets and Stochastic Time Logic

An HPNG is formally defined as a tuple $\langle \mathcal{P}, \mathcal{T}, \mathcal{A}, \boldsymbol{m_0}, \boldsymbol{x_0}, \Phi \rangle$, which consist of a set of places $\mathcal{P}$, a set of transitions $\mathcal{T}$, a set of arcs $\mathcal{A}$, the initial marking vectors $\boldsymbol{m_0}$ and $\boldsymbol{x_0}$, and a tuple of functions Φ that define additional parameters. The discrete and continuous dynamics of HPNGs are formally defined in [7]. Figure 1 presents an HPNG model of a water tower that consists of a reservoir, represented by continuous place Cr (a circle with a double border), with an input and an output pump, represented by the continuous transitions Fp and Fd (rectangles with double border), that are connected to the reservoir using continuous arcs (white arrows). The remaining part of the HPNG describes how the inflow stops when the input pump breaks after a random amount of time, modelled by a general transition (a rectangle with a single border) and how the outflow stops deterministically after 5 hours, modelled by a deterministic transition (a grey rectangle). Discrete arcs (black arrows) connect discrete places (a single circle) to deterministic and to general transitions and vice versa. When transition Gb fires, a token is removed from the discrete place Pp. This disables the continuous transition Fp, since it is connected to place Pp with a test arc (arc with two arrowheads). Similarly, the outflow stops when transition Fd becomes disabled due to the removal of the token in place Pd.

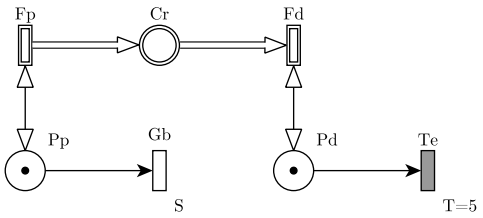


Fig. 1. Water tower HPNG model

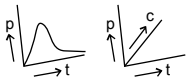
Measures of interests for HPNGs can be described using the logic STL [6]:

$$\Psi := \; tt \;\mid\; x_{\mathcal{P}} \leq c \;\mid\; m_{\mathcal{P}} = a \;\mid\; \neg\Psi \;\mid\; \Psi_1 \wedge \Psi_2 \;\mid\; \Psi_1 \; \mathcal{U}^{[T_1,T_2]} \; \Psi_2,$$

where $T_1, T_2, c \in \mathbb{R}^{\geq 0}$; $a \in \mathbb{N}^{\geq 0}$. Atomic formulas $x_{\mathcal{P}} \leq c$ and $m_{\mathcal{P}} = a$ compare continuous and discrete markings with a predefined constant. STL formulas can be negated ($\neg\Psi$) and combined with conjunction ($\Psi_1 \wedge \Psi_2$). The until operator $\Psi_1 \; \mathcal{U}^{[T_1,T_2]} \; \Psi_2$ describes that during the evolution of the system, property Ψ_1 should hold, until in the time interval $[T_1, T_2]$ property Ψ_2 becomes true. In [7] the formal semantics for STL on HPNGs is given.

Using algorithms from [6] and [9], so-called satisfaction intervals are computed for all but nested until formula. These represent all intervals from the support of the random variable that defines the firing time of the general one-shot transition for which the resulting evolution of the HPNG fulfils the STL property (at a certain time τ). Then, by integrating the probability density function of the random variable over the satisfaction intervals the probability that the STL property holds at a given time τ is computed.

Table 1. In- and output with corresponding functionalities

Functionality	**Input**	**Output/Button**
Model checking [6] [9]	HPNG model STL formula Time to check t	
Stochastic time diagram [5]	HPNG model	
Transient probabilities [5]	HPNG model Time range Continuous atomic property A constant/constant range	

3 Fluid Survival Tool Overview

Table 1 summarizes the functionality of FST. HPNG models (textually specified) can be model checked against an STL specification for a specific time (t), in order to generate a model checking verdict. FST provides insight in the time-based evolution by generating graphical stochastic time diagrams from the HPNG models and transient probabilities can be computed. Also 2D and 3D plots can be automatically generated.

The model checking tool FST has an extendable object-oriented design with a GUI. This has been achieved through the composition of a Software Development Kit (SDK) consisting of the C++ compiler GCC, the GUI library Qt and the Integrated Development Environment (IDE) Qt Creator [3]. Additionally, the SDK contains Doxygen [8] for documentation, SVN [1] for version control and a project page for information, releases and bug tracking [4]. Moreover, the Model-View-Controller and Facade software patterns are added for designing extendable GUI. So, to run a functionality with these software patterns, the user interacts with the view class that evokes the controller to pass a request to the Facade class that evokes the model checking algorithms. Then, feedback is provided to the controller class such that the view can be updated. Figure 2 shows a screenshot of the tool FST where the HPNG model of the water tower from Figure 1 is loaded as (textual) input. The tool consists of a menu bar **(1)**, a button for each functionality **(2)**, a textual HPNG model editor **(3)** and a logger to provide textual feedback **(4)**. When the functionality buttons are clicked a configuration dialog pops up where all the input variables from Table 1 are requested and the output is provided, accordingly.

The 2D plot in Figure 3 shows the probabilities that the amount of fluid in the reservoir is smaller or equal to 0.4 m^3 for the example model over a range of time, where the input pump breaks according to an exponential distribution (with mean 10). Time in hours is on the x-axis and the probability is on the y-axis. This probability is calculated for the time range $[0, 100]$ with a time step of 1. The 3D plot in Figure 3 additionally parametrises the maximum amount of fluid in the reservoir, i.e., the amount of fluid that should not be exceeded, over the range 0.0 to 8.0 m^3 with a step size of 0.2 m^3.

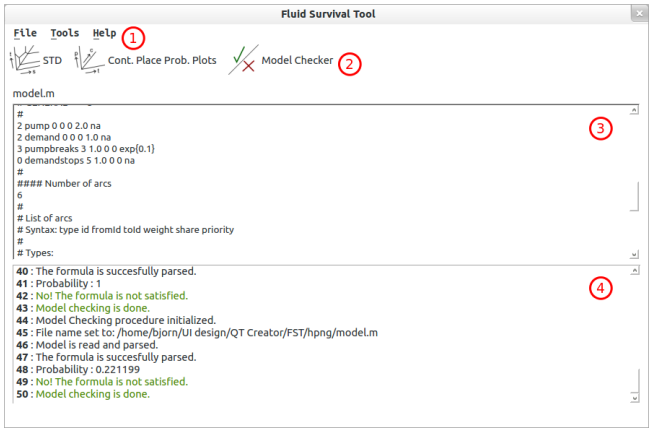


Fig. 2. FST main screen

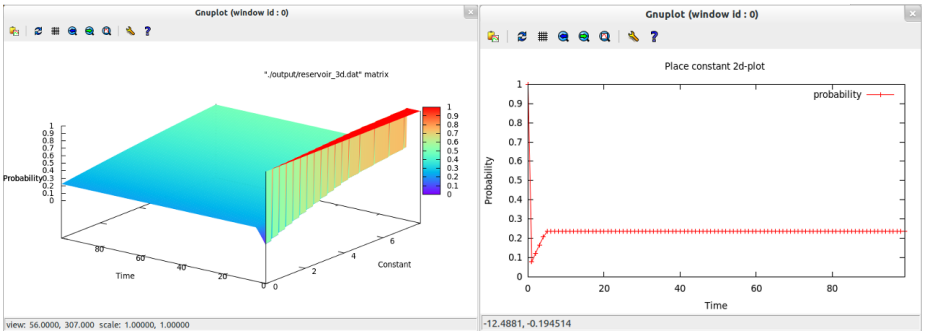


Fig. 3. 3D and 2D output of FST

References

1. Apache Software Foundation: Subversion. http://subversion.apache.org/. (2000)
2. S.H. Conrad, R.J. LeClaire, G.P. O'Reilly, H. Uzunalioglu: Critical national infrastructure reliability modeling and analysis. *Bell Labs Technical Journal* 11(3): 57–71. (2006)
3. Digia: Qt and Qt Creator. http://qt-project.org/. (2009)
4. FST Projectpage. https://code.google.com/p/fluid-survival-tool/. (2013)
5. H. Ghasemieh, A. Remke, B.R. Haverkort, M. Gribaudo: Region-based analysis of hybrid Petri nets with a single general one-shot transition. *Proc. 10th Int'l. Conf. FORMATS*, LNCS 7595: 139–154. (2012)
6. H. Ghasemieh, A. Remke, B.R. Haverkort: Survivability evaluation of fluid critical infrastructure using hybrid Petri nets. *19th IEEE Int'l. Symposium PRDC*. (2013)
7. M. Gribaudo, A. Remke: Hybrid Petri nets with general one-shot transitions: model evolution. Technical report, University of Twente. http://130.89.10.12/~anne/pub/tecrep.pdf. (2012)
8. D. van Heesch: Doxygen. http://www.stack.nl/~dimitri/doxygen/. (2001)
9. B.F. Postema, *Fluid Survival Tool: A model checker for Hybrid Petri nets.* MSc thesis, Department of Computer Science, University of Twente. (2013)