Modeling and Analyzing Hybrid Systems Using Hybrid Predicate Transition Nets

Dewan Mohammad Moksedul Alam
Xudong He
School of Computing and Information Sciences
Florida International University
Miami, Florida 33199, USA
dalam004@fiu.edu, hex@cis.fiu.edu

William (Cheng-Chung) Chu
Department of Computer Science
TungHai University, Taichung, Taiwan
cchu@thu.edu.tw

***Abstract*— Hybrid systems, especially in the form of cyber physical systems, have become ubiquitous and are playing critical roles in the functioning of society, however their design and implementation are extremely difficulty, especially regarding their dependability. In this paper, we propose a hybrid high level Petri net formalism, hybrid predicate transition nets (HPrTNs), for modeling and analyzing hybrid systems. We discuss some critical concepts and features of HPrTNs. We demonstrate the applicability of HPrTNs through several well-known benchmark hybrid systems and compare our results with other relevant methods. HPrTNs are fully supported in the tool environment PIPE+.**

Keywords— formal methods; high-level Petri nets; hybrid Petri nets, differential Petri nets

I. INTRODUCTION

Hybrid systems refer to the systems that arise out of the interaction of continuous dynamics and discrete dynamics. Many modern embedded systems, especially cyber physical systems are hybrid systems that contain physical devices having continuous dynamics and computational control processes with discrete behaviors. Hybrid systems have become ubiquitous and are playing critical roles in the functioning of society, however their design and implementation are extremely difficulty, especially regarding their dependability.

Hybrid systems have been the focus of intense research in the past few decades since they provide a convenient framework to accurately model a wide range of engineering systems and provide the flexibility to abstract complex physical behaviors away and to model dynamics having varying scales. Following the early works on the verification of digital circuits, many formalisms, methods and tools have been proposed to model and verify more complex embedded systems such as air traffic control systems, automotive control, bioengineering, process control, real-time communication protocols, manufacturing control, etc. One early prominent work is the hybrid automata [1] that provided a concrete mathematical framework for the analysis and verification of hybrid systems. Hybrid automata integrate diverse models such as differential equations and state machines in a single formalism with a uniform mathematical semantics and novel algorithms for multi-modal control synthesis and for safety and real-time performance analysis [2]. However, despite providing powerful methods to analyze hybrid systems, the major inconvenience of hybrid automata is the dramatical increase of model dimensions for complex systems due to the intrinsic global state configurations and sequential behaviors of automata.

DOI reference number: 10.18293/SEKE2018-158

Petri nets, a concurrent and distributed formal models, provide a great flexibility to model complex systems. Petri nets have evolved in the past half century in many directions: including continuous Petri nets [3], fluid stochastic Petri nets [4]. Continuous Petri nets have further been extended to hybrid Petri nets [5] for modeling hybrid systems. Hybrid Petri nets inherit all the advantages of the Petri net model such as the ability to capture distributed behaviors, concurrency, synchronization and conflicts. Similar concepts have also been extended to high-level Petri nets [6, 7, 8] to model data dependent hybrid systems.

In this paper, we present our results in introducing continuous features into predicate transition nets (PrTNs) [9] for modeling hybrid systems. Specifically, we introduce two different kinds of places and transitions namely continuous places and continuous transitions with differential and difference equations. Our approach has a well defined priority rule to resolve the conflict of firing enabled discrete and continuous transitions. We have implemented the whole hybrid PrTN framework in our modeling tool PIPE+ [9]. We demonstrate how to model some classic examples of hybrid systems using PIPE+ and compare the modeling and simulation experience and performance with some existing and well known hybrid Petri net modeling tools.

II. RELATED WORK

The concept of extending Petri net formalism to provide means to model continuous and hybrid systems was first presented in [10]. Based on this concept, several other extended Petri net formalisms were proposed. In the following subsections some of related formalisms and their applications and supporting tools are discussed.

A. Hybrid Petri net Formalism

In [10], the authors combined a continuous Petri net representing continuous dynamics with a discrete Petri net capturing discrete behaviors. Subsequently, the authors extended their formalism to provide distinction between

autonomous and timed hybrid Petri nets and provided rules to resolve conflicts between continuous and discrete part [5, 11]. Hybrid Petri nets are based on low level Petri nets where tokens in continuous places are numerals and change rates associated with continuous transitions are simple difference equations. A slightly different approach was introduced in [12]. Here new kind of places and transitions were introduced, namely differential places and differential transitions. Differential places constitute the continuous state of the system being modeled. Differential transitions are always enabled and associated with a firing frequency, where first-order ordinary differential equations are used to represent the evolution rules. Another class of hybrid Petri nets is fluid stochastic Petri nets introduced in [4], which extended stochastic Petri nets to model hybrid stochastic systems. Apart from these, several other prominent works were published to extend other classes of Petri nets, batch Petri nets, hybrid flow nets, etc., to support modeling of hybrid systems.

Along with the research on the extension of the low-level Petri nets, several classes of high-level Petri nets have also been extended for modeling hybrid systems. One of such early approaches was proposed in [8], where a method was presented to extend timed hierarchical object-related nets (THORNs). In this extension, the author introduced real data type to THORNs to represent the continuously changing state variable and continuous transitions to capture the continuous evolution. In this approach, a continuous transition was enabled or disabled by inhibitor arcs and the evolution was specified using ordinary differential equations. However, this approach was not fully developed and supported by any tool. Among other classes of high-level Petri nets, Colored Petri nets were studied extensively and several approaches for extending them to model hybrid systems were proposed in [6, 7, 8, 13].

B. Modeling and Analysis Tools

Although, both low-level and high-level Petri nets have been undergone rigorous studies and many extensions are proposed to model hybrid systems, not many efforts are made to provide proper tool support. Among low-level hybrid Petri net tools HYPENS [14], SimHPN [15] and HISim [16] are worth mentioning. Both HYPENS and SimHPN are not native Petri net tool, and are based on MATLAB and Simulink. They do not provide proper net editing capabilities. A user needs to use MATLAB/Simulink components to specify the semantics of the Petri net model of the system being modeled. HISim on the other hand integrates modeling and simulation in a unified tool but is functionally incomplete. In [7], the authors proposed a different approach to create a model using MATLAB components for simulation and provided a methodology to translate that into CPN for analysis. Among the tools in this context, Snoopy [13, 17] provides a unified experience of creating graphical model, simulation and analysis; but focuses on modeling biological systems. This tool supports several hybrid low-level and high-level Petri nets.

Our work provides a unified framework for system modeling and analysis using Hybrid high-level Petri nets leveraging our tool environment PIPE+.

III. HYBRID PREDICATE TRANSITION NETS

In the following sections, we provide a formal definition of hybrid predicate transition nets (HPrTNs) by extending the definitions of PrTNs [18].

Definition 1. A HPrTN is a tuple $N = (P, T, F, \Sigma, \alpha, \beta, \gamma, M_0)$, where

(1) $P = P_d \cup P_c$ is a non-empty finite set of discrete places P_d and continuous places P_c (graphically represented by circles and double circles respectively);

(2) $T = T_d \cup T_c$ is a non-empty finite set of discrete transitions T_d and continuous transitions T_c (graphically represented by bars and boxed bars respectively), which disjoins P, i.e. $P \cap T = \emptyset$;

(3) $F \subseteq P \times T \cup T \times P$ is a flow relation (the arcs of *N*) such that $\forall p \in P_c, t \in T_d. ((p,t) \in F \Leftrightarrow (t,p) \in F)$;

(4) $\Sigma = (St, Op, Eq)$ is the underlying algebraic specification with sorts St, operations Op, and equations Eq. Σ defines the set *Token* of tokens, the set *Label* of labels, and the set *Constraint* of constraints of *N*. In our tool environment, the Σ-algebra is instantiated with a subset of Java data types and their associated operations and laws;

(5) $\alpha: P \to \wp(St)$ associates each place p in P with a subset of sorts in St such that $p \in P_d \Rightarrow real \notin \alpha(p)$ and $p \in P_c \Rightarrow real \in \alpha(p)$. The above constraints refer to projected component when $\alpha(p)$ is a composite type;

(6) $\beta: T \to Constraint$ associates each transition *t* in *T* with a first order logic formula that defines the enabling condition (precondition) and the processing result (post-condition) of *t*;

(7) $\gamma: F \to Label$ associates each flow relationship *f* in *F* with a label denoting the data flow of a relevant transition satisfying $\forall p \in P_c, t \in T_d. (\gamma(p,t) = \gamma(t,p))$ and $\forall p \in P_d, t \in T_c. (\gamma(p,t) = \gamma(t,p))$, i.e. discrete transition can only read but not change continuous place, and continuous transition cannot change discrete place. Thus, this restriction corresponds to the concept of elementary hybrid Petri nets in [11], which does not allow the conversion between discrete and continuous markings;

(8) $M_0: P \to \wp(Token)$ is a sort-respecting initial marking such that $\forall p \in P_c. (M_0(p) \neq \emptyset \wedge |M_0(p)| = 1)$, i.e. each continuous place contains one and only one token.

The dynamic semantics of HPrTNs are defined on the concept of markings (states) $M: P \to \wp(Token)$ that are mappings from places to tokens.

Definition 2. A transition *t* in *T* is *enabled* in a marking *M* if $\forall p \in P. (\bar{\gamma}(p,t): \theta \subseteq M(p) \wedge \beta(t): \theta)$, where $\bar{\gamma}(p,t)$ is a generalization of γ such that $(p,t) \notin F \Rightarrow \bar{\gamma}(p,t) = \emptyset$. $e: \theta$ is the result of instantiating all arc variables with tokens in *p* according to substitution θ.

The enabling condition of a continuous transition here is similar to the strongly enabled concept in [11].

Definition 3. An enabled transition t in marking M with substitution θ can *fire* and results in a new marking M' defined by: $\forall p \in P. (M'(p) = M(p) \cup \bar{\gamma}(t,p):\theta - \bar{\gamma}(p,t):\theta)$.

Two enabled transitions may fire at the same time as long as they are not in conflict, i.e. the firing of one them disables the other. Furthermore, a discrete transition has priority over a continuous transition if they are in conflict. All enabled continuous transitions are fired in a single round to reflect a snapshot of the time passage. The dynamic semantics (behavior) of a HPrTN is the set of all possible transition firing sequences starting from the initial marking. Various conflict situations involving continuous transitions are further discussed in the following sections.

IV. Modeling Hybrid Systems

In this section, we present new features implemented in PIPE+ to model hybrid systems. PIPE+ provides full support to model, simulate, and model check (using external dedicated model checkers) discrete event systems [9]. In the following sub-sections, we discuss only the features related to modeling continuous and hybrid systems.

A. Modeling Continuous Components

To provide support for modeling continuous components, two different elements, continuous places and continuous transitions, are introduced, which are significant different from their discrete counterparts in terms of both structure and dynamic semantics.

1) Continuous Places

As in [5], continuous places represent the continuous part of the state space of the hybrid system being modeled. However, it is possible to design these places to represent more than one continuous attributes in HPrTNs. In other words, during modeling the continuous dynamics, the modeler can design these places to represent as many continuous dynamics as needed as long as they satisfy the constraints imposed on the dynamic semantics of the continuous components. This capability is quite useful to group together related dynamics instead of scattering those across multiple continuous places. We provide more insights on this in section V.

Continuous places share similar behavior as discrete places. The only differences between these are their data types. The data type of a continuous place (1) must be a singleton (or not a power set in our implementation), and (2) must have at least one number element. Thus, a continuous place can hold only one token and at least one element of this token is a numeral capturing the dynamic changing value of the system. A useful guideline is that a continuous place is modified by only one continuous transition. This helps to avoid (1) conflict between continuous transitions, and (2) inconsistent behavior. This restriction may be overcome by using more efficient scheduling mechanism of continuous transitions. Finally, continuous places are represented by double circle in PIPE+.

2) Continuous Transition

Continuous transitions are used to model the continuously changing behavior of the hybrid system being modeled. Unlike other hybrid Petri nets, the behavior of continuous transitions in HPrTNs is strategically different in many ways, including (1) continuous transitions are always enabled unless the discrete parts are a part of their preconditions, (2) the continuous transitions can modify discrete places, (3) the marking of the continuous places can control the speed of changes made by the continuous transitions. These strategies offer several benefits. The first strategy allows us to map the behavior of the continuous transitions analogous to the semantics of the hybrid systems, where the continuous part continuously changes the state of the system and the discrete part only controls the speed of the change. For example, when a car is stopped, the engine is running and consumes fuel and produces power; but its speed and acceleration remain zero since the transmission is detached. Thus, depending on the discrete control, some part produces positive/negative changes while other parts do not change. The second strategy provides modeling flexibility and the third allows us to model feedback mechanism. As an example, consider the movement of a pendulum. At every point, the dynamics (acceleration and speed) of the next depends on the dynamics of the current.

In HPrTNs, the constraints of continuous transitions are defined the same way as discrete transitions, and consist of preconditions and post-conditions specified in a first order logic formula. Generally, the preconditions of the continuous transitions consist of the data flow of the discrete input places attached to the transition in question. However, there is no restriction to use tokens from continuous places in the precondition, which is needed to model conditional branches (emulating if-else conditions) to compute new marking. This flexibility keeps the overall net size smaller.

The post-conditions of continuous transitions are similar to those of discrete transitions. However, some new concepts are introduced. First order ordinary differential equations can be used to compute continuous dynamics. These equations are used as part of integral operator. Listing A shows the format of ODE used with integral operation. One example of integral equation is shown there with changing variable *q*. The lower and upper limits of the equation is specified using the tokens of some place.

Table 1 – (A) Example of using differential equations

Format	∫ (*ode, lower_limit, upper_limit*) ∂*change_var*
Example	∫ (q/0.98,d1[1],d[1])∂q)

Post-conditions can be made dependent on time. An approximation of logical time is introduced for this purpose which is discussed in sub-section IV.C. A special operator, τ (tau), is used to represent the current logical time. The following expression shows an example of using logical time.

$$y=x+ \int (5,\tau-1,\tau)\partial\tau$$

3) Dynamic Semantics

Historically, PrTN is assumed to be autonomous without explicit timing information. HPrTNs follow the same concept. The evolution of its discrete components is the same as that of PrTNs. To compute the evolution of the continuous part, a slightly different approach is adopted. Since the evolution of the continuous part may depend on the time and/or other continually

changing components. Hence, some representation of time is needed. We discuss the modeling of time in HPrTNs in the next sub-section (IV.B).

All the continuous transitions are assumed to be always ready and to have equal firing speed. Thus, in each execution step, all the continuous transitions are evaluated to test whether they are enabled or not. All enabled continuous transitions are fired in every execution step. However, this strategy may result in conflict if two continuous transitions have the same input places but have different enabling conditions. The resolution of such conflict is discussed in sub-subsections (IV-C). The evaluation of the preconditions and post-conditions are the same as the discrete part as discussed in our earlier works [9].

Another interesting aspect of the dynamic behavior is the execution order of discrete and continuous parts. The discrete component takes the precedence. After all enabled discrete transitions fire once, the enabled continuous transitions start firing.

B. Modeling Time

The continuous dynamics of hybrid systems is generally specified using differential or difference equations. To evaluate these equations, a reference clock is needed. Continuous dynamics is computed with respect to this reference clock. In PIPE+ there are several ways to specify this reference clock. This may be a random number, a function or a set of values specified in a place. This can be achieved by adding a pair of continuous place and a continuous transition. The place would hold the reference clock value and the transition would be responsible for the coherent evolution of the reference clock value. One can choose whatever one needs as the basis of change. Alternatively, a convenient and simple way is also provided for the basis of change, which approximates a logical clock. The clock simply counts the number of steps the execution performed so far. The step size should be assumed by the modeler, which can be an hour, one second, one milli-second or even one Nano-second. The current time provided by the clock can be accessed using the operator τ.

C. Conflict Resolution

In the context of Petri nets, a conflict arises when multiple transitions are enabled and firing one of them disables others. Conflicts can be categorized according to three common causes: (1) between two discrete transitions, (2) between one discrete and one continuous transitions and (3) between two continuous transitions. The conflict between discrete transitions is resolved using non-determinism. In this case, one of the enabled transitions is chosen non-deterministically to be fired and it is ensured that only one discrete transition is fired. This type of conflicts is already discussed in our prior work. The other two categories are interesting in the context of hybrid Petri net and are discussed below.

1) Conflict Between Discrete and Continuous Transitions

This type of conflict arises when a discrete and a continuous transition are connected to the same input place, either discrete or continuous. In our approach, both discrete part and continuous part have their own separate methods of execution and we allow the execution of both these two parts in the same step. We have a well-defined precedence between these two components that ensures that these two components do not modify the same place at the same time, which nicely resolves this type conflict.

2) Between Continuous Transitions

There are three different ways that can lead to this type of conflict: (1) the transitions have the same input discrete place, (2) the transitions have the same input continuous place, and (3) they both have same output place. In the first case, there is conflict when any of the following conditions is true – (a) the satisfiability of the preconditions depends on the token from the common input place, and (b) the common place is a power set and the token satisfying the preconditions of the conflicting transitions is the same. To resolve this conflict, a token from a discrete place used in the evaluation of the precondition of a continuous transition must be returned to the place unaltered. This means that continuous transitions cannot modify such places. In the second case, since continuous places can hold only one token, the removal of that token in the process of execution of one of the conflicting transitions makes other transitions disabled. To resolve this, a different method of executing continuous transition is used. The output places of continuous transitions are updated once the preconditions are evaluated and post-conditions are computed for all the continuous transitions. The third case is an undesired situation, and should be avoided. It is the modeler's responsibility to ensure that no unexpected results can be produced. Although, the above is an undesired situation, no restriction is implemented for simplicity. Furthermore, it is very convenient when one continuous place is designed to store multiple continuous attributes with separate continuous transitions are used to access those attributes.

D. Analysis

In PIPE+, only simulation and evolution graph are supported to analyze a hybrid system model. Model checking is not supported yet due to complexity of numerical functions used in modeling hybrid states. The simulator executes the net following the dynamic semantics of HPrTNs, and stores the state sequences of the system during the execution that can be used later to analyze the evolution. Simulation can be done using one step at a time for better understanding of the evolution or multiple steps in a single run to quickly have an overall picture of the system behavior. However, both these two-execution methods support (1) configuring the evolution graphs, and (2) exporting of the snapshots of the states for statistical analysis using other sophisticated tools.

1) Evolution Graph

Evolution graphs show the evolution of the continuous attributes over time. Before starting simulation, a chart configuration UI is provided to select the attributes for evolution to be shown in charts. For each of these attributes a separate chart window is created. It is also possible to configure multiple attributes to be shown in the same chart window for better comparison. By default, the evolution of the attributes over time is shown in the charts. It is also possible to configure the charts to plot the evolution of one attribute again another.

2) Export of Results

Evolution graph provides simple means to show how the value of some continuous attributes changes over time. It does not provide support to generate any other insight. To address this problem, simulation results can be exported for more sophisticated analysis. Generally, after each simulation step the state of the whole net along with the inputs and generated outputs are stored in a file. These can be exported to some external data analysis tools to better understand the behavior of the system.

V. Case Study

We have applied HPrTNs to model and analyze several well-known benchmark hybrid systems [19], including bouncing ball, thermostat, robotic motion controller, and obstacle avoidance. Due to space limit, we only show two systems here.

A. Bouncing ball

In this model, the physics of a bouncing ball, i.e. its motion before, during and after the impact against another surface is modeled. In this model, the state of the ball is captured when it falls freely from a place 10 meters above the surface assuming 0.75 coefficient of restitution. Here, only the effect of gravity is considered. Figure 1 shows a pictorial diagram of the hybrid Petri net model. Table 2 lists the inscriptions of the net. Here, the continuous place *Dynamics* is used to store the velocity and height, real valued numbers as reflected in its datatype definition shown in Table 2. The continuous transition *Compute* computes these dynamics by following the basic laws of motion of freely falling objects as shown by $\beta(Compute)$ in Table 2. The discrete transition *Change* simply changes the direction of the motion by negating the velocity whenever the height falls below zero.

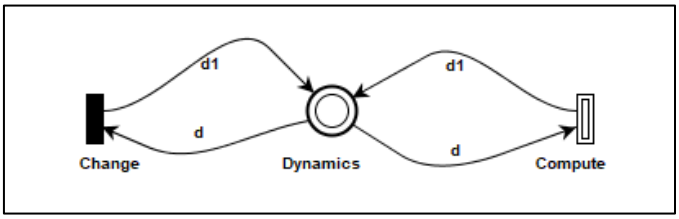

Figure 1 – Pictorial diagram of the bouncing ball model

Table 2 – Net inscription of the model in Figure 1

$\alpha(Dynamics) = \langle number, number \rangle$
$\beta(Change) = d[2] \leq 0 \wedge d1[2]=0 \wedge d1[1]= -d[1]*0.75$
$\beta(Compute) = d1[1]=d[1]-0.98*\tau \wedge d1[2]=d[1]*\tau - 0.49*\tau*\tau + d[2]$

This model is simulated with various initial conditions, i.e. initial speed and height. Figure 2 shows the result of a simulation run when a ball is dropped from a height of 10m. Initial marking for this case is $M_0(Dynamics) = \langle 0, 10 \rangle$. The left chart in the figure shows the evolution of the velocity of the ball and the right chart shows the evolution of height with respect to time.

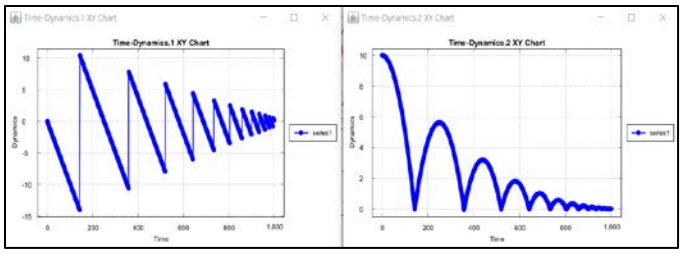

Figure 2 – Simulation result of the model in Figure 1

B. Air Traffic Collision Avoidance

In air traffic control, collision avoidance maneuvers are used to resolve conflicting flight paths that arise during free flight [20]. These are very important and complex applications. A great number of different successful maneuvers are proposed and verified in the literature, many of them are also used in practice. As a case study, we model one of these maneuvers – straight line maneuver with instant turn. This maneuver involves a series of linear movement of the aircrafts. These movements can be controlled either from a central command center or from the approaching aircrafts' local control system. In our model, a central control system is used. Figure 3(a) shows the required movements of the aircrafts participating the straight-line collision avoidance maneuver, and Figure 3(b) shows the pictorial diagram of the hybrid PrTN model.

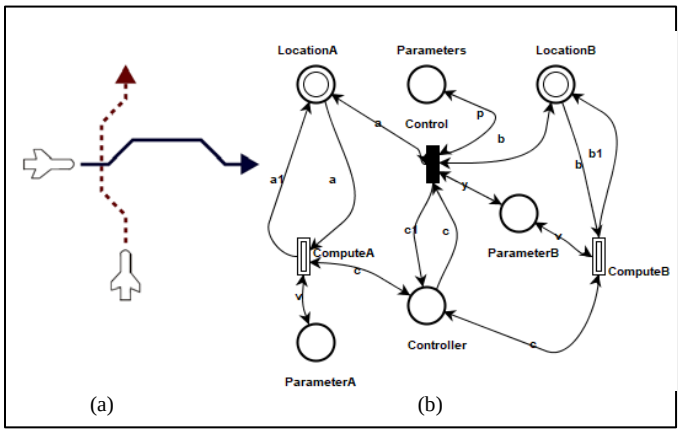

Figure 3 –(a) The movement of the aircrafts in straight line maneuver with instant turn. (b) A pictorial diagram of the hybrid PrTN model

In this model, two aircrafts *A* and *B* are participating the straight line with instant turn collision avoidance maneuver. Here, the place *Controller* stores the parameters to control the directions of the aircrafts participating the maneuver. The transition *Control* generates these control parameters depending on the state. The control parameter here is basically an angle that dictates the direction of the aircrafts. The place *ParameterA* stores the velocity and the angle of direction of the aircraft *A*. *LocationA* stores the location of *A*. The transition ComputeA computes the location of *A* using its parameters and the control parameter. *ParameterB*, *LocationB* and *ComputeB* do the same for aircraft *B*. The place Parameter defines the safe horizontal and vertical distances. Due to space limit, the detailed net inscription is omitted here. We have simulated this model with different sets of initial conditions, i.e. initial locations, velocity and directions of the aircrafts, different safe distances. Figure 4

shows the result of a simulation run where the aircraft A starts from the location (0,0) along X-axis and aircraft B starts from (18, 0) towards the opposite direction of A. Both have equal ground speed of 200 m/s. The safe horizontal and vertical distances are 12 and 2 kms respectively. In Figure 4, the charts (a) and (b) show the entrance and exit of the collision avoidance maneuver of the aircrafts A and B respectively. Initially both A and B move towards each other, when A reaches just after 3 and B reaches 15, the vertical distance falls below the safe distance. Both planes turn left and follow that direction until they reach the safe vertical distance. When the safe distance is reached, they turn right and follow their own course. An error of 200m from the original course is allowed as shown according to the constraint of transition *Controller*.

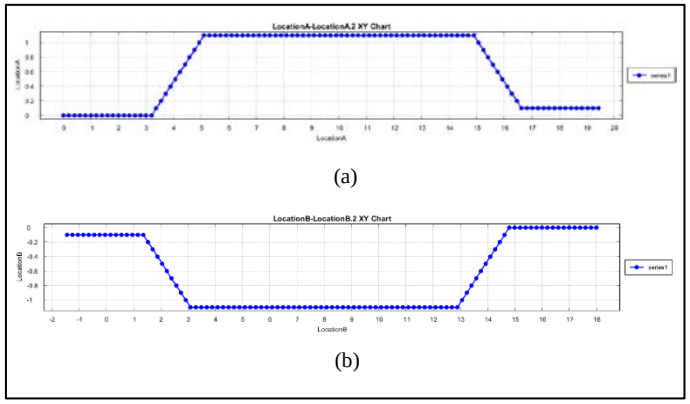

Figure 4 – Simulation results of the model in Figure 3(b).

VI. Conclusions and Future Directions

In this paper, we presented hybrid predicate transition nets (HPrTNs) to model and analyze hybrid systems. The whole framework is implemented in our tool environment PIPE+. We have shown how the new features of PIPE+ can be utilized to model and analyze hybrid systems through several well-known hybrid systems. We have studied several other tools providing modeling and analyzing capability of hybrid Petri nets. Most of these tools are tailored to specific applications such as bio-medicine. Some of these tools use other modeling languages, like MATLAB, to generate Petri net models. PIPE+ provides a unified environment for modeling and analyzing high-level Petri nets including HPrTNs.

This work is an initial attempt to extend PrTN towards hybrid system modeling and analysis. We will study the explicit time representation as in timed Petri nets. We will investigate new and improved scheduling algorithms of discrete and continuous transitions to cover more realistic and sophisticated applications.

Acknowledgement

Alam and He were partially supported by AFRL under FA8750-15-2-0106. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

[1] Henzinger T.A., The Theory of Hybrid Automata. In: Inan M.K., Kurshan R.P. (eds) Verification of Digital and Hybrid Systems. NATO ASI Series (Series F: Computer and Systems Sciences), vol 170. Springer, Berlin, Heidelberg, 2000.

[2] J. Lygeros, C. Tomlin, and S. Sastry, “Hybrid Systems: Modeling, Analysis and Control”, December 2008, retrieved from http://www-inst.cs.berkeley.edu/~ee291e/sp09/handouts/book.pdf

[3] H. Alla and R. David, “A modeling and analysis tool for discrete events systems: continuous Petri net”, Performance Evaluation, 33(3), pp. 175-199, August 1998.

[4] Trivedi, K. S. and Kulkarni, V. G. 1993. FSPNs: Fluid stochastic Petri nets. *14th International Conference on Application and Theory of Petri Nets*, Chicago.

[5] R. David and H. Alla, “Discrete, Continuous, and Hybrid Petri Nets”, Springer Berlin Heidelberg, Springer, 2010.

[6] M. Herajy, F. Liu and C. Rohr, “Coloured hybrid Petri nets for systems biology”, *Biological Process and Petri Nets*, 1159, pp. 60-76, June 2014.

[7] D. Bera, K. van Hee and H. Nijmeijer (2015), ”Modeling Hybrid Systems with Petri Nets”, *In: Obaidat M., Ören T., Kacprzyk J., Filipe J. (eds) Simulation and Modeling Methodologies, Technologies and Applications.* Advances in Intelligent Systems and Computing, vol 402. Springer, Cham

[8] R. Wieting, "Modeling and Simulation of Hybrid Systems Using Hybrid High-level Nets", In: *Proceedings of the 8th European Simulation Symposium* (ESS'96), Vol. II, pages 158-162, October 1996, Genoa, Italy.

[9] D. Alam and X. He, “A method to analyze high level Petri nets using SPIN model checker”, in *Proceedings of the 29th International Conference on Software Engineering & Knowledge Engineering*, pp. 161-166, July 2017.

[10] R. David, H. Alla, Continuous Petri nets, in: *Proc. 8th European Workshop on Application and Theory of Petri Nets*, Zaragoza, Spain, 1987.

[11] R. David and H. Alla, "On Hybrid Petri Nets", *Discrete Event Dynamic Systems*, 11, pp. 9-40. January 2001, doi: 10.1023/A:1008330914786

[12] I. Demongodin, N.T. Koussoulas, Differential Petri nets: Representing continuous systems in a discrete-event world, *IEEE Trans. Automat. Control* (1998).

[13] M. Heiner, M. Herajy, F. Liu, C. Rohr, M. Schwarick, "Snoopy – A Unifying Petri Net Tool", *Application and Theory of Petri Nets. PETRI NETS 2012. Lecture Notes in Computer Science*, vol 7347. Springer, Berlin, Heidelberg, 2012.

[14] F. Sessego, A. Giua, C. Seatzu, ”Simulation and Analysis of Hybrid Petri Nets using the Matlab Tool HYPENS”, *SMC08: 2008 IEEE Int. Conf. on Systems, Man, and Cybernetics (Singapore)*, October 2008.

[15] J. Júlvez, C. Mahulea, and C. R. Vázquez, "SimHPN: A MATLAB toolbox for simulation, analysis, and design with hybrid Petri nets", Nonlinear Analysis: Hybrid Systems, 6(2), pp. 806-817, March 2012.

[16] A. Amengual, “A Specification of a Hybrid Petri Net Semantics for the HISim Simulator”, accessed from http://www.icsi.berkeley.edu/pubs/techreports/TR-09-003.pdf, 2009.

[17] M. Herajy, F. Liu, C. Rohr and M. Heiner, "Snoopy’s hybrid simulator: a tool to construct and simulate hybrid biological models", *BMC Systems Biology*, 11:71, July 2017.

[18] X. He: “A Comprehensive Survey of Petri Net Modeling in Software Engineering”, International Journal of Software Engineering and Knowledge Engineering - IJSEKE, vol. 23, no. 5, 2013, 589-626

[19] R. Alur: “Principles of Cyber-Physical Systems”, The MIT Press, 2015.

[20] A. Platzer, “Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics”, Springer-Verlag Berlin Heidelberg 2010