Development and Improvement of Scene Transition Nets(STN) GUI Simulator for Discrete-continuous Hybrid Systems

Takeshi Tateyama
Tokyo Metropolitan University, Tokyo, Japan

Hiroshi Chin
The University of Tokyo, Tokyo, Japan

Seiichi Kawata
Advanced Institute of Industrial Technology, Tokyo, Japan

Yoshiki Shimomura
Tokyo Metropolitan University, Tokyo, Japan

Abstract

Scene Transition Nets(STN) is a very useful modeling method for discrete-continuous hybrid systems. However, it is difficult to write STN in standard object-oriented programming languages because STN programming requires much skill of object-oriented programming and high knowledge of STN of designers. To overcome this problem, the authors have developed a useful GUI simulator software for modeling and simulations of STN. The experimental results of a transport system including an AGV showed the availability of our software.

Key Words: Scene Transition Nets(STN), discrete-continuous hybrid systems, GUI simulator

1. Introduction

Many of complex systems(e. g. manufacturing systems, chemical plant systems) need to be modeled as discrete-continuous hybrid systems. Petri net[1] is useful method to model discrete event systems and differential equations are often used in order to model continuous systems. However, there are not many studies about modeling and simulation method for hybrid systems. In some studies of modeling hybrid systems[2, 3], Scene Transition Nets(STN)[4, 5] is a very useful modeling method for discrete-continuous hybrid systems in order to model and simulate complex and dynamic hybrid systems in which many systems perform in parallel and interact with each other. It is a modeling method to depict hybrid systems graphically based on the concepts of Petri net. It is known that this method is effective in the modeling and simulation of complex manufacturing systems and chemical plant systems. However, it

* Corresponding author:
Tel: +82-(0)2-2290-047
Fax: +82-(0)2-2292-047
Homepage: http://www.university.ac.kr/~kdhong/
Email: kdhong@university.ac.kr
is difficult to implement STN in standard object-oriented programming languages (e.g. Smalltalk, JAVA) because STN programming requires considerable skill in object-oriented programming and excellent knowledge on the part of STN designers. Because of this background, a useful graphical user interface(GUI) modeling and simulation tool is required so that designers can easily edit and simulate STN models. The purpose of our study is to develop a GUI simulator software useful for modeling and simulation of STN[6]. In this paper, we present a new version of an STN GUI simulator written in JAVA. The experimental results of a slab carrier system including a battery-powered automated guided vehicle(AGV) demonstrated the usefulness of our software.

2. Scene Transition Nets(STN)

This section describes the outline and components of STN.

2.1 The Outline of STN

STN is a graphic modeling method for discrete-continuous hybrid systems: it uses the concepts of "actors" and "scenes". It is based on Petri net which is a modeling method for discrete event systems. STN can express hybrid systems by using the concept of Petri nets and by writing differential equations in the scenes. In STN, an actor corresponds to a subsystem of a hybrid system. Designers can simulate interactions between the sub systems that act in parallel by describing the models by using object-oriented programming languages(e.g. Smalltalk, JAVA).

2.2 STN Components

STN is constructed by actors, scenes, transitions, and arcs(Fig. 1). Details of these components are described below.

![Fig. 1. STN Components](image)

2.2.1 Actor Classes and Actors

Actors in STN correspond to the tokens in Petri net. However, actors have state variables whose values change dynamically unlike tokens. An actor is one of the objects in STN that belongs to the "actor class". Actors belonging to the same actor class have common data structures(same types of constants and
variables) and are called instances of the actor class. In STN, actors are defined as "subsystems" of a whole system, which is defined as an "observed system" and an observed system is a set of actors that interact with each other. Through these interactions, the states of actors and state variables change by the computing dynamics described in the scenes, as explained in the following section.

2.2.2 Scenes, Castings, and Performers

Scenes in STN correspond to the places in Petri net. In STN, scenes are combinations of "activities" defined in discrete event systems and dynamics for changing variables of actors in the activities. Fig. 1 shows STN components including a scene and an actor using the description format of STN. A scene is represented as a box divided into three parts. The name of the scene(Scene 1) is indicated at the top. The circle A1 shown at the bottom of the scene indicates the location of actor named A1. The circle A at the middle of the scene indicates "casting" of the scene. A casting of a scene indicates an actor class whose instances(actors belonging to the actor class) can transit to the scene. A casting has a parameter which is called "capacity". A capacity of a casting in a scene denotes the maximum number of the actors which can transit the scene at the same time. In addition, an actor located in a scene is called a "performer" of the scene. Designers write dynamics using differential equations in each scene in order to dynamically change variables of the performers of the scene.

2.2.3 Transitions and Arcs

"Transitions" in STN correspond to those in Petri net and indicate "scene transition boundaries" that correspond to "events" of discrete event systems. A transition is indicated by a bold line in Fig. 1. Transitions and scenes are connected by "arcs". Transitions connected to scenes with input arcs leading into the scenes are called "input transitions" of the scenes. In contrast, transitions connected to scenes with output arcs exiting from the scenes are called "output transitions" of the scenes. In a similar way, transitions have some "input scenes" and "output scenes". Designers write conditions for which the actors in the input scenes transit to the output scenes in the transitions and they write transit rules for the state variables.

2.3 Hybrid System Simulation Using STN

Designers can analyze discrete-continuous hybrid systems using object-oriented programming languages. They can analyze discrete event systems by observing the scene transitions of actors and can analyze continuous variable systems by observing the change in the state variables of actors.

3. Definition of Capacities of Castings

In STN researches, we have not defined a maximum number of actors which can transit to a scene at the same time. However, we need this definition when we simulate a queue or an AGV whose carrying capacity is limited. Therefore, in this paper, the authors propose that
a casting has a parameter which is called “capacity”. A capacity of a casting in a scene denotes the maximum number of the actors which can transit to the scene at the same time. Fig. 2 shows the concept and outline of capacities of castings. In addition, when we simulate a queue, we need to specify transition rules of queued actors. In this study, we introduce a concept of two transition rules: FIFO (First In First Out) and LIFO (Last In First Out). Users can select one of the two transition rules in each scene. This new function enables us to simulate complex systems including queues and complex transition rules.

Fig. 2. The concept of capacities of castings in scenes

4. STN GUI Simulator

Fig. 3. Overview of STN GUI Simulator
STN is a very useful modeling and simulation method for hybrid systems. However, it is difficult to construct STN models and execute simulation in standard object-oriented programming languages because STN programming requires considerable skill in object-oriented programming and excellent knowledge on the part of STN designers. Considering this background, the authors have developed an STN GUI simulator so that designers can easily edit and simulate STN models. This chapter describes the simulator. Fig.3 shows an overview of the GUI simulator written in the JAVA programming language. This simulator consists of the STN edit toolbar, workspace, simulation toolbar, and graph windows. Details of these components are described below.

4.1 STN Edit Toolbar

The STN edit toolbar which is located at the top of the screen includes the following ten icon buttons. Designers can easily edit STN models by using this GUI tool.

4.1.1 Actor Class, Scene, Arc, and Transition Button

These buttons are used to place STN components into the workspace.

4.1.2 Casting and SetActor Button

These buttons are used to set castings or performers by clicking on an actor class in the workspace and dragging and dropping it into a scene.

4.1.3 Global Variable Button

This button is used to set global variables. Designers can set the names, max/min values, and initial values of the variables and determine whether the simulator displays their graphs. In addition, they can set the dynamics by writing differential equations.

4.1.4 Property Button

The button is used to set the following properties of STN objects.
- Actor class names and their variable names and attributes(integer or real)(Fig.4)
- Actor names, their variables (max/min/initial values), and determining whether the simulator displays their graphs(Fig.5)
- Scene names and dynamics of their performers(Fig.6)
- Transition names, their fire conditions, and transit rules(Fig.7)
- Capacities of castings and priorities of actors' transitions(First In First Out, FIFO / Last In First Out, LIFO)(Fig.8)

4.1.5 Move and Delete Buttons

Designers can move and delete the STN objects in the workspace.
4.2 Workspace

The workspace is the space in the middle of the simulator. Designers draw STN models in this space and can analyze behaviors of the networks by observing the animation in the space in simulation phases.

Fig. 4. Setting Window of an Actor Class’s State Variables

Fig. 5. Setting Window of an Actor’s State Variables

Fig. 6. Setting Window of a Scene
4.3 Simulation Toolbar

In a simulation phase, designers control the simulation using the simulation toolbar.

4.3.1 Step Size

Designers set the step size and time scale for the Runge-Kutta method by writing in the text box.

4.3.2 Compile Button

When the compile button is clicked, the simulator analyzes the variables and equations (dynamics, fire conditions, and transit rules) inputted by the designers using a syntax analysis algorithm. The analysis algorithm will hereinafter be described in detail.

4.3.3 Start/Stop and Reset Button

The designers can start or stop the simulation using the start/stop button. The reset button is used to initialize the global variables, actor's state variables, and simulation time.
4.3.4 Simulation Speed Setting Bar

Designers can change the animation speed of the simulation by using the speed setting bar.

4.4 Syntax Analysis Algorithm for Equations

The simulator includes a compiler to analyze the text data of the equations (dynamics of state variables, fire conditions, and transit rules of transitions) using a syntax analysis[7] algorithm. When the compile button is clicked, the compiler divides the text data of the equations into values, variables, signs, and operators. Designers can use following symbols as operators:

- monadic operators (functions): sin, cos, tan, exp, log, -(sign)
- infix operators: +, -, *, /, %(remainder), ^(power)
- comparison operators: ==, >, <, <=, >=, !=
- logic operators: &&(AND), ||(OR)
- assignment operator: =
- differential operator: $d(variable)/dt$

The equation format is similar to those in C, C++, and JAVA. In addition, the symbol "t" can be used to denote time. After dividing the equations, the compiler converts the equations written using the infix notation into on written using the postfix notation to facilitate the computer calculations. In the simulation phase, the simulator calculates the dynamics of the state variables, equations for fire conditions, and the transit rules of the transitions using the calculation method for postfix notations.

4.5 Sequence of STN Simulations

![Sequence of STN Simulations Diagram](image_url)

**Fig.9. One Step Sequence of STN**
Fig.9 shows the sequence of STN simulation. First, the simulator searches the transitions which can fire in the network. In STN, a transition can fire only when there are performers of all casting in all input scenes of the transition and these performers' state variables meet the fire conditions of the transition. When some transitions fires, the actors which are performers of the input scenes of the transitions transit to the output scenes and their state variables are changed according to the transit rules of the transitions. Next, all global variables and all actors' state variables are updated using Runge-Kutta method. After that, simulation clock is updated. Above 1-step cycle is repeated until termination of the simulation.

4.6. Graph Displays of Actors' State Variables and Global Variables

In the simulation phases, it can display some graphs which show dynamical changes of designated actors' state variables and global variables(Fig.3). The designers can analyze the behaviors of continuous variable systems by observing these graphs and the behaviors of discrete event systems by observing the animation that show the transitions of the actors.

5. An Example: An Automatic Transport System Using Battery-Powered AGVs

This section describes an example of the modeling and simulation of an automated transport system including one or multiple battery-powered AGVs. Fig.10 shows the outline of the example. In this example, the purpose of the system is to carry parts from the buffer 1 to the process machinery using the battery-powered AGV. The AGV can move between the buffer 1 and the process machinery (distance:50[m]) and between the buffer 1 and the battery changer(distance:10[m]). The AGV can carry \( N_p \) sets of parts at a time. After the AGV throws the parts in the process machinery, it returns to the buffer. When the AGV reaches the buffer 1, it checks the voltage of its battery. If the voltage is higher than \( V_{th} \), it continues to carry other parts; otherwise, it goes to the battery changer in order to replace its battery. The
authors define the dynamics of the AGV and batteries as follows:

\[
\begin{bmatrix}
    x \\
    \omega \\
    V_a
\end{bmatrix}
= \begin{bmatrix}
    0 & \pm R & 0 \\
    0 & -\psi^2 / R_a J & \psi / R_a J \\
    0 & a\psi / R_a & -a / R_a
\end{bmatrix}
\begin{bmatrix}
    x \\
    \omega \\
    V_a
\end{bmatrix}
+ \begin{bmatrix}
    0 \\
    -T l / J - R_c \\
    0
\end{bmatrix}
\]  

(1)

Here, \(x(-10 \leq x \leq 50)[m]\): current position of the AGV, \(\omega\) [rad/sec]: angular velocity of the AGV’s motor, \(V_a\)[V]: voltage of the battery, \(R\): final reduction ratio, \(R_a\)[\Omega]: armature circuit resistance, \(J\)[kgm]: the total moment of inertia of the rotational system, \(\psi\)[Nm/A]: torque constant of armature, \(T l\)[Nm]: counter torque to the motor shaft, \(R e\)[Ns/m]: viscous frictional drag, \(a\)[V/As]: characteristic constant of battery. The purpose of this simulation is to observe the AGV’s positions, changes in the battery voltage and efficiency of transportation of the parts. In our simulation, the initial value of \(V_a\) is 100[V] and \(V_{th}\) is 45[V]. Fig.11 shows the STN model of the system.

In the first experiment, we set the number of AGVs to one and set the capacity of the AGV, \(N_p\) to one set of the parts. In order to set the capacity of the AGV to one, we set the capacities of the castings of the parts in the scenes the AGV transits with the parts to one. Fig.12 shows the changes in the position of the AGV \(x[m]\) and the voltage of the battery \(V_a\)[V]. These graphs indicate that after the AGV carried the parts thirteen times, it went to the battery changer because \(V_a\) was lower than \(V_{th}\). After changing the battery, it carried another parts once again. In addition, we can observe that the AGV gradually slowed down because of the battery’s voltage reduction. Fig.13 shows the changes of the positions of parts No.1-5. Each graph in the figure shows when the parts arrived at the buffer No.2.

In the second experiment, we set the value of \(N_p\) to two. To simulate this condition, we change
the capacities of the castings of them from one to two. Fig.14 shows the changes of the positions of parts. This figure shows that the efficiency of transportation is improved compared with the results of the first experiment.

**Fig. 12.** Changes in the position of the AGV(left) and the voltage of the battery(right)

**Fig.13.** Changes of the positions of the parts(1 AGV, \( N_p = 1 \))

In the third experiment, the number of AGVs increases from one to two and we set the value of \( N_p \) to one. Fig. 15 shows the changes of the positions of the parts. These graphs in this figure shows that the efficiency is improved compared the results of the first experiment. However, the efficiency is down a little compared with the results of the second experiment. As for the reason for the results, we assumed that only one AGV can go through the course from the buffer 1 to the process machinery at a time.
Therefore, while an AGV is running the course, another AGV has to wait until it reaches the buffer 2.

These experimental results show that we can easily observe the behaviors of the system and analyze them by using the STN GUI simulator.

![Fig.14. Change of the positions of the parts(1 AGV, $N_p = 2$)](image)

![Fig.15. Change of the positions of the parts(2 AGVs, $N_p = 1$)](image)

6. Discussion

In the experiments, we could model STN of a parts carrier system including AGVs whose carrying capacities are limited by using the new concept, “capacities of castings”. In addition, we could easily and graphically observe the state transitions as a discrete event system and state variables of the AGVs as a
continuous system. There experimental results show that the proposed modeling and simulation tool is very useful in order to simulate complex discrete-continuous hybrid system.

7. Conclusion

This paper describes a new version of the STN GUI simulator. The new functions implemented in this system are “capacities of castings” and transit rules for queues of actors. In addition, the authors demonstrate the usefulness of this tool and the new functions by the experiments simulating the automated transport system using the battery-powered AGV.

8. Acknowledgements

This research was partially supported by the Ministry of Education, Science, Sports, and Culture through a Grant-in-Aid for Scientific Research (B), 18360079

References