Development of Practical Software for Micro Traffic Flow Petri Net Simulator

Noboru Kimata[1), Keiich Kisino[2), Yasuo Siromizu[3)

[Abstract] Recently demand for microscopic traffic flow simulators is increasing both from changes of problem situations we encounter in transportation planning and intensity of public involvement (PI) movement in public planning. We have been developing a new type simulator based on Coloured Timed Petri Net. This type simulator has potential to be put into practical use as PI support system. In this paper we present a more practically strengthened prototype system on Windows-OS platform and show its active supportability for development of individual simulations and accountability of their usage to a wide range of people involved. We demonstrate especially how newly developed menus such as drawing traffic-flow-net, tracing token behaviors on the net, making the individual places/transitions/arcs visible or invisible by selection, work and improve readability of Petri net description for microscopic traffic flows and contribute to putting the simulator into practical usage.

[Keyword] simulation, Coloured Timed Petri nets, microscopic traffic flows, transportation planning, Public Involvement (PI)

1. Introduction

In urban area, it has been difficult to implement a drastic plan like a new road construction as a resolution for traffic congestion. So we are forced to find out other practicable resolutions based on individual conditions of actual road configuration and available technologies. It would be advantageous for us to do such task if we could be supported by a microscopic traffic flow simulator that can deal with such factors fully and concretely. Support by such a microscopic traffic flow simulator would be effective also for evaluation of new alternatives like ITS, P&R system, LRT, etc.

Public involvement movement in public planning field is another important background of a great demand for microscopic traffic flow simulators. Under PI oriented planning, it is indispensable requisite that accountability is fulfilled for a wide range of people involved in the disputed plan. Microscopic traffic simulator could contribute to this requisite as a competent tool to explain details of relationships between traffic flows and proposed alternatives both from logical and experiential points of view.

In these situations we now encounter, some powerful and visual simulators such as NETSIM, AVENUE, PALAMICS, tiss-Net, etc. are developed as planning support systems. We also propose a microscopic traffic flow simulator based on the methodology called coloured timed Peri net.

In this methodology, (1) traffic flows are represented by a visual graph, and this graphical net itself serves directly as illustrative presentation of simulated outputs. (2) Any nets are essentially driven by common and simple rules that we could easily follow by hand. (3) The nets have common structure and

1)Member of JSCE, Professor, Department of Civil Engineering, Kanazawa University (2-40-20 Kodatsuno Kanazawa-shi, Tel 076-234-4914)

2)Member of JSCE, Chuo Fukken Consultants Co., Ltd. (1-8-29 Nisimiyahara Yodigawaku, Osaka-)

3)Member of JSCE, Chuo Fukken Consultants Co., Ltd. (2-11 Oodenmacho Nihonbashi Chuoku Tokyo)

are conjunctive each other. From (1) and (2), we can develop highly flexible driving software for traffic flows Petri nets. Under this software and from futures of (1) and (3), we can easily perform to expand and refine our description of disputed traffic flows by step-by-step wisely. And constructed net models of disputed traffic flows and construction processes of these nets could be understood both by following the logic manually and by confirming vehicle's movements visually only given fundamental knowledge of Petri nets and their driving rules.

We expect certain potential to our simulator as PI support system from these properties and then try to expand it to some more practical level. In this paper, we first summarize our Petri net type simulator and explain some fundamentals like collision safety in simulated traffic flows, renewal rule of vehicle's speed, etc. Afterward, we present its more practical oriented and converted version from UNIX to Windows-OS platform, and demonstrate menus newly devised for PI style usage.

2. Petri nets Methodology for Traffic Simulation

2-1 Peri nets representation of vehicle traffic flows.

Essentially in Peri net method, objective system is represented by a graph with two kinds of nodes: places and transitions. In the nets, arc exists only from a place to a transition or from a transition to a place and is classified into input, output, or inhibitor arc. And states of the system are indicated by markings of tokens at the places.

States are dynamically transformed by "firing" of enabled transitions. As mentioned above, the firing rule is simple and common: a transition fires if and only if at least one token is marking at all input places of the transition and no token exists at any inhibitor places. And after the transition fires, its associated tokens at its input places are removed, and a token is deposited at each of its output places. Here we avoid non-determined situation with "conflict" in traditional rule because we must validate traffic safety. We will explain this point in section 2-2(1).

Fig.1 is an example of Peri net representations of vehicle traffic flows around a signalized intersection focusing on the traffic flow from left to right in this figure. The modeling road is a single lane for each way and set right-turn pocket with capacity five near the intersection. As shown in Fig.1, a system net of traffic flows can be constructed by combining several basic Petri nets for fundamental traffic flows. In this case the system net consists of five such basic sub nets as follows: ① vehicle's arrival sub-net, ② vehicle's progress sub-net, ③ lane changing sub-net, ④ traffic signal sub-net, ⑤ right turn behavior sub-net. We have developed other basic sub nets like merging, pedestrian crossing, etc.

Vehicle's arrival sub-net generates vehicles at time intervals which follow exponential distribution with given mean time μ. It is needless to say that 1/μ is the expected mean of generated traffic volume. In Fig.1, this type sub nets are put at both the left and the right hand end as generators for the main and the opposite traffics respectively.

Sub net ① is combined with vehicle's progress sub-net as shown in Fig.1. As presented in Fig.2, vehicle's progress net is a bonded net of sub nets each consists of a paired places and two transitions and corresponds to a block on the road defined as a space limited only for one vehicle to occupy safely.

Here we define the transition as an event of vehicle's progress to the adjacent block on the road. While we devise the definition of the paired places as one of places is "vehicle's presence in the block" (denoted by a black token) and another is "non-occupation of the block" (denoted by a white token). And we connect them to the transitions by arcs of such input / output relationships as presented in Fig.2 (a). Traffic safety is autonomously validated by these net modeling and the firing rule described above.

Firstly, two types of places are complement

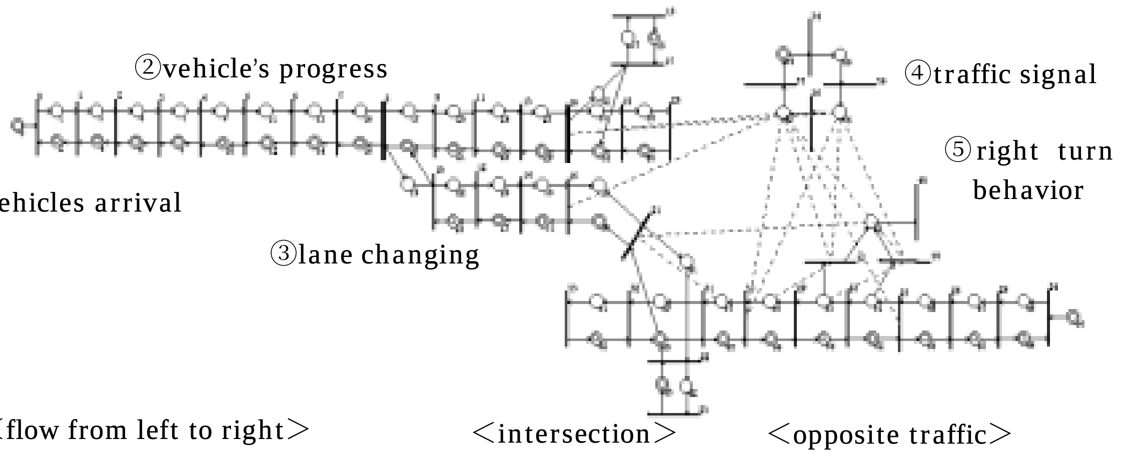


Fig.1 Petri-net Representation of Traffic Flows around Signalized Intersection

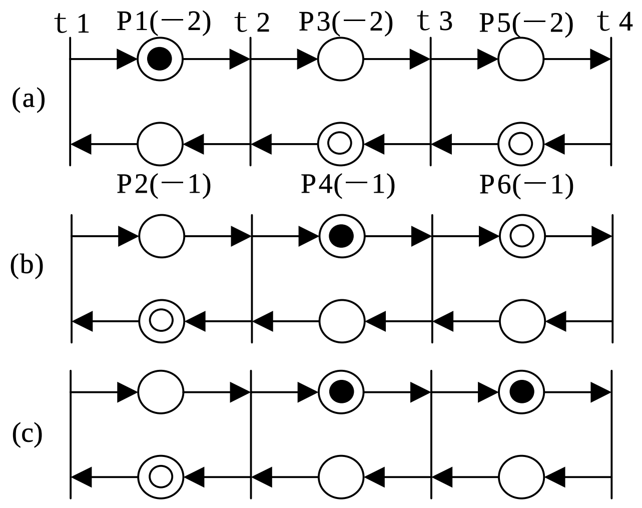
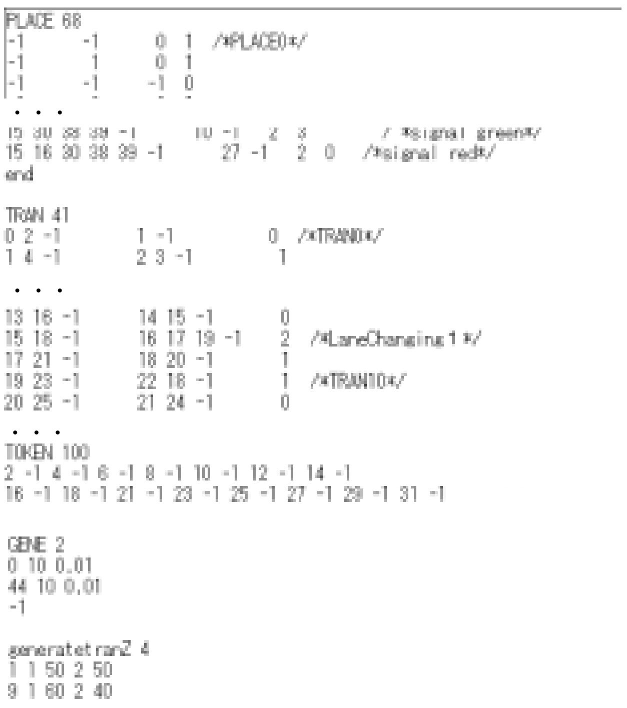


Fig.2 Basic Sub Net of Vehicle's Progress

relations with each other and then always only one token is put on either one of them. In Fig.2 (a), places inscribed with (-2) are first type ones and places inscribed with (-1) are second types. So tokens are put on at only ones of paired places, saying P1(-2), P4(-1), and P6(-1). Here a black token indicates existence of a vehicle and a white token non-occupation of the block. Then the marking of Fig.2 (a) represents a state of traffic flow that one vehicle exists at the first block while its forward two blocks are free.

Next, applying the firing rule to this net marking, we can identify only transition t2 is enabled because its all input places P1 and P4 are marked by tokens but for other transitions one of their input places is not marked. Then

Table 1 Sdata Statement of System Net of Fig.1



```
PLACE 68
-1       -1      0  1   /*PLACE0*/
-1        1      0  1
-1       -1     -1  0
 . . .
15 30 38 39 -1        10 -1   2  3        / *signal green*/
15 16 30 38 39 -1       27 -1   2  0   /*signal red*/
end

TRAN 41
0 2 -1          1 -1            0  /*TRAN0*/
1 4 -1          2 3 -1           1
 . . .
13 16 -1        14 15 -1         0
15 18 -1        16 17 19 -1      2  /*LaneChanging1*/
17 21 -1        18 20 -1         1
19 23 -1        22 18 -1         1  /*TRAN10*/
20 25 -1        21 24 -1         0
 . . .
TOKEN 100
2 -1 4 -1 6 -1 8 -1 10 -1 12 -1 14 -1
16 -1 18 -1 21 -1 23 -1 25 -1 27 -1 29 -1 31 -1

GENE 2
0 10 0.01
44 10 0.01
-1

generatetran2 4
1 1 50 2 50
9 1 60 2 40
```

transition t2 fires and the state of the net shifts to the marking as shown in Fig.2 (b). It is easy to understand that this marking represents the vehicle progressed safely to the adjacent forward block.

While Fig.2 (c) represents a traffic state that two vehicles are running on neighboring blocks. We can also assure that collision never occur in this situation. Since transition t3 does not satisfy the firing condition mentioned above, then it does not fire until the forward vehicle progresses to the next block and a token is returned at P6. That is, in vehicle's progress net vehicles run autonomously as following their forward ones and rear-end collision is avoided.

Lane changing sub net is modeled by devising two expanded transitions as follows. One is generative transition and equips tokens with colors that represent straight, left turn, or right turn vehicles. Another one is selective transition that selectively determines its output place according to token-colors. In Fig.1, GT denotes the former and ST denotes the later. Tokens (vehicles) equipped with colors by GT are deposited at one of correspondent output places (lanes) of ST when transition ST fires.

In sub nets of traffic signal and right turn behavior, we design usage of inhibitor arcs to intentional traffic control. Inhibitor arcs from the place of red signal to the transition stop vehicles to enter the intersection until the token stays at the red place because of the firing rule. We could design inhibitor arcs according to phases of the signal we want to adopt. We can also use inhibitor arcs to ensure critical gaps between right turn vehicle and ones running opposite lanes. In Fig.1, we use a traffic signal sub net with four phases, green, right turn only, yellow, and red, and a right turn behavior sub net with space of five blocks as driver's acceptable critical gap.

The individual system net is detailed by a numerical data format called Sdata. At Table 1, we show a part of Sdata statement for the system net of Fig. 1.

The statement of Sdata consists of five parts, PLACE, TRAN, TOKEN, GENE, and generatetranZ. In the part of PLACE, we describe related inhibit transitions, types of place, and parameter for drawing/non-drawing at simulation to each of all places which appear in the system net. In TRAN, we do specification to all of transitions in same manner. The part of TOKEN is used to set an initial marking of the system net by describing place numbers and initial stay timers at the places. GENE defines parameters for traffic volumes we want to generate at the vehicle's arrival sub-nets mentioned above, and generatetranZ sets ratios of colors to transition GTs in order to simulate more realistic and reliable designated traffic flows. As demonstrated lately, Sdata representation is indispensable to our computer simulation and its wide range of applicability and visual readability.

2-2 Simulation Algorithm of Traffic Petri Nets

(1) Driving algorithm by firing of transitions.

As mentioned in 2-1, Petri nets are dynamically driven by the "firing" rule of transition. We essentially adopt this rule as our simulation algorithm. That is, a transition fires if and only if

r-1: at least one token is marking at all input places of the transition and

r-2: no token is put at all inhibitor places, then

r-3: it's associated tokens on its input places are removed, and

r-4: a token is deposited on each of its output places.

In traditional rule, if enabled transitions, from viewpoints of r-1 and r-2, are "conflict" with each other, one of them is selected in probabilistic manner and only the selected transition fires. But in our driving algorithm all of enabled transitions fire simultaneously at the same instance as mentioned in 2-1.

Fig. 3 shows an example of "conflict" situation in traffic Petri nets. Judging from viewpoints of r-1 and r-2, both of transition t1 and t5 are enabled. And it is obvious that if one of them fires then the another becomes not enabled because of net transaction by r-3 and r-4. This is the definition that transition t1 and t5 are "conflict" with each other.

In the case of traffic simulation, this means that always one of the vehicles located at P1 and P5 gives way another vehicle and collision accidents are never occurred. But it is too optimistic in this case. In our algorithm, both transition t1 and t5 fire simultaneously. Therefore two vehicles enter the space P3 and a collision will occur.

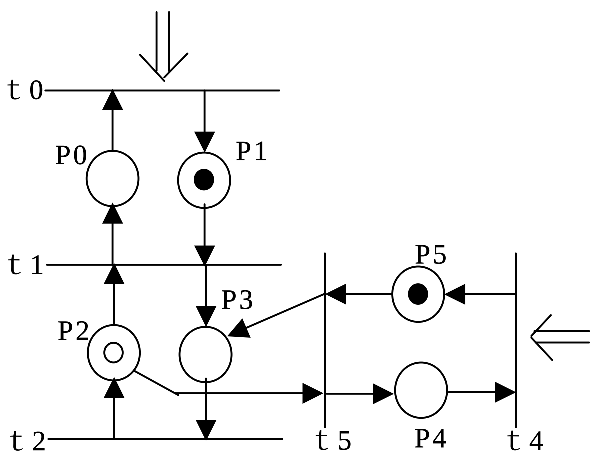


Fig.3 Conflict State in Traffic Peri-nets

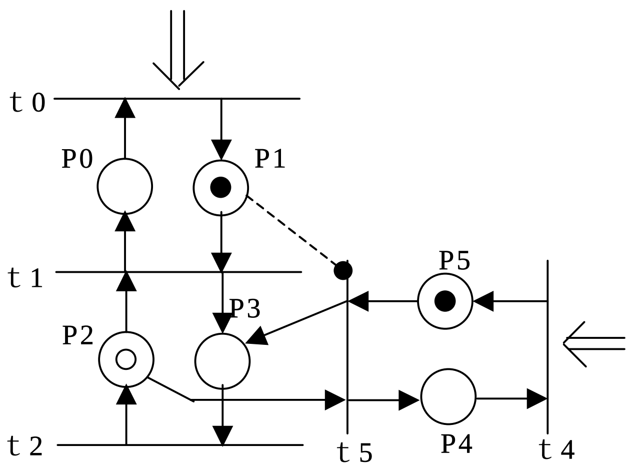


Fig.4 Priority Setting by Inhibitor Arc

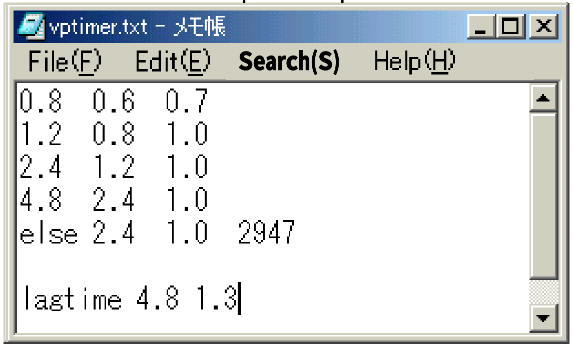
Of course it will be very rear to occur such simultaneous firing of transitions even in our algorithm. But we need avoid the risk intentionally by setting priority to main road. Fig.4 presents such a net in which priority is given to the vertical road against horizontal one using inhibitor arc. In this net, t5 can not fire so for as the token stays at P1 because of r-2 mentioned above. And the token passes the P3 and a token is returned to P2, then t5 becomes to fire. So in our algorithm, head to head collision can be avoided by introduction of some intentional rule.

This modeling technique can be applied to sub nets of traffic signal, right turn behavior, merging, pedestrian crossing, etc. It is also possible to devise a kind of killer sub nets for safety mechanism of these nets and to construct a system net for collision simulation by combining them to traffic flow net.

(2) Renewal rule of vehicle's desired speeds

The notion of time in timed Petri nets are given by "timers" associated with places or tokens. For instance, duration times of each phases of traffic signal are represented by timers set on corresponded places. And these timers are fixed at constants if not actuated control case. To such type of sub nets we can apply the firing rule directly and change only the state of net after the laps of the timers. That is we need no treatment of timers themselves. But concerning to vehicle's progress sub nets that we discussed at Fig. 2,

Table 2 Example of Vptimer Table



```
0.8  0.6  0.7
1.2  0.8  1.0
2.4  1.2  1.0
4.8  2.4  1.0
else 2.4  1.0  2947

lastime 4.8 1.3
```

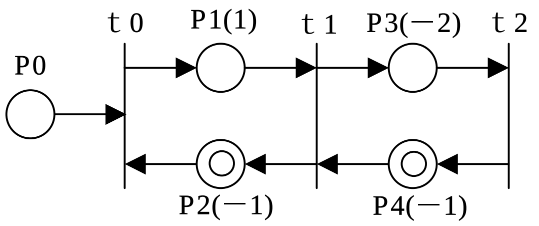


Fig.5 Combined Net of Vehicle's Arrival and Progress

we need additional procedure to renew timers of associated tokens.

The tokens located at places of type (-2) represent vehicles on road. Their timers are time that the tokens stay at the places and mean times that the vehicles journey the one block that we define at 2-1. That is, these timers vary depending on currently realized speeds of vehicles and traffic conditions. Then for tokens on the place of type (-2), we define their timers as a new type and incorporate additional procedure to renew them under an assumption that drivers will be tendency to put up their desired speed if possible.

Table 2 is an example of reference data for renewal of desired speeds named as the “vptimer” table. The table is described using stay time values calculated by (the length of a block) / (desired speed). In this table, we also assume that the length of a block is 6.7m and maximum speed limit is 40kph.

The first column of this table represents realized vehicle’s stay time at the current place and the second desired stay time (reverse of desired speed) at the next place the vehicle progresses. In third column, we could assume probabilities that drivers would select acceleration shown in the table. At the bottom line of this table, we could also set lag-time when vehicles start from state of stopping. The numerical values following “lag time” are stay time for judgement of state of stopping and delayed time of firing.

Incorporation of the vptimer would be informative for simulation of traffic flows. But it requires stay time at the current place as reference data to renew timers of token corresponded to vehicle. Fig. 5 reveals a problem arisen from this requirement and its resolution.

Fig. 5 is the combined net between the sub nets of vehicle’s arrival and vehicle’s progress. The P0 is programmed as a generation place where vehicles are generated as tokens at time intervals with an exponential distribution. Then place P0 holds generated times of tokens but does not hold any stay times of tokens. When we bond P0 to the vehicle’s progress sub net without any amendment, P1 becomes as type of (-2) instead of (1) denoted in Fig. 5.

If P1 is a place of type (-2), it is impossible to determine the timer attached to the token that progresses to the place P1(-2) because there is no stay time at current place P0 to be referred at the vptimer table. The net shown in Fig. 5 resolves this problem by defining the place P1, adjacent to the generation place P0, as special type of (1) and devising a fixed timer on this special place.

The tokens that progressed to P1 stay there for a fixed duration given by this timer. Then how many tokens are generated densely at P0, the minimum of firing intervals of transition t1 is limited to this fixed duration. In other words, the timer setting on this place defines the maximum volume of traffic generated by our vehicle’s arrival sub net. It is said that the maximum capacity per a lane is 2200～2500 (pcu/h). We use 1.6 second by default as the fixed timer for places denoted (1) in traffic Petri nets. The maximum volume is then calculated as 2250 (pcu/h). Fig. 5 is seen as artificial at glance, but we insist as supported by traffic reality.

3. Development of Practical Prototype for Petri Net Traffic Simulator.

3-1 Requisites for PI usage of Petri net traffic simulator.

We advocate a PI style usage of our newly developed simulator for urban traffic planning. Here we consider fundamental requisites to evolve a practical prototype from the simulator to this end. The process of simulation usage for PI style planning is generally divided into the following three phases: description phase of disputed traffic situation by Petri nets, construction and execution phase of the net on computer, and application phase of simulation to the planning. We insist above that the most competitive potential of our simulator is users could develop their own nets corresponding to their individual cases and validate them for their planning bases almost by themselves. But in reality still user’s working load may be heaviest at the first phase. So it is especially vital to lighten the working load at this phase to the PI style usage success.

We try to achieve this requisite by implementation of the first phase working itself on computer, which is supposed to be classified as the second phase at above recognition. During the first phase, we do trial and error to get suitable net to simulate disputed traffic situations. First of all we must endeavor to realize step-by-step wisely construction and execution procedure of Petri nets model development and facilitate trial and error performance on our prototype system. As discussed above, direct execution of any nets without any reprogramming is another competitive future of Petri nets simulator. So we can develop such a prototype by devising user friendly interface to this end.

In the trial and error process, advocators of nets try to interpret their nets modeling to other participants and participants try to contribute to construction of nets for practical usage by understanding the model structure, expressing their opinions and proposing ideas. In Petri net methodology, the net itself serves directly illustrative presentation of simulated outputs. This property provides valuable assistance to fostering communication between advocators and other participants and achieving more useful net construction. So we must endeavor to enrich visual and illustrative presentation ability of nets.

The net showed at Fig. 1 is so simple that simulated token's moves are traced easily on this style of presentation. But in practical cases, it is easy to predict that nets become more complicated and become difficult to trace token's moves or to follow logical structure if we present them directly. For instance increasing of number of places and transitions increases number of arcs among them and nets themselves will become rapidly unreadable because of high density of drawn arcs. Then we may say that it is also vital how to elaborate the net representation interface in order to ensure readability of drawn nets on computer and trace complicated tokens behaviors in the nets.

3-2 Windows version interface of Perti net simulator

From considerations in section 3-1, we focus on following three points: (1) step-by-step wisely construction and execution of net development for practical usage, (2) selective representation of net components to ensure visibility and readability of nets drawn on computer, and (3) coloring tokens to trace complicated their behaviors. Fig.6 is the main menu developed as user interface in order to realize these points on our prototype system. Here left side three of buttons on the main menu and right side two buttons are almost same as standard ones equipped with the Microsoft Windows system. Other three are ones added for Petri net traffic flows simulator. Fig.6 shows a situation that following selection of “Drawing Net”, “Figure the Net” is clicked and a pullout menu appears which contains sub menu to select for ensuring visibility and readability of nets drawn on computer, and for tracing complicated tokens behaviors. We give outline of usage and functions of them.

First, we assume that a net model is proposed and its Sdata file is made up. Now, we can open the Sdata file by clicking “File” on Fig.6 and selecting the file name. Construction of the net on computer is executed using “Drawing Net”. As shown in Fig. 6, “Drawing Net” has sub menu which consists of “Layout P and T”, “Figure the Net”, and “Sketch the Road”. Here we select “Layout P and T” and click it. Fig. 7 appears on screen as support windows to lay out places and transitions of which the net consists. Then we select a place or a transition from the lists shown at the left side sub windows and click the put-button. After that we move the cursor on screen to the position where we want to lay out the place or the transition and click the mouse at the point. As shown at the right side of Fig.7, a circle or a bar is drawn at the position corresponding to the selected component.

Continuing this procedure, we construct the net on computer. In section 3-1, we pointed out rapidly increasing of arcs between places and transitions in practical usage. So in order to reduce user's load we adopt the algorithm that all of arcs are drawn automatically by referring their relational data to the Sdata. Also we provide at right side another sub windows which display relevant components to the selected one and guide user's construction procedure of net.

Of course, we could rearrange easily positions of components by drag and drop of them. This rearrangement procedure can be

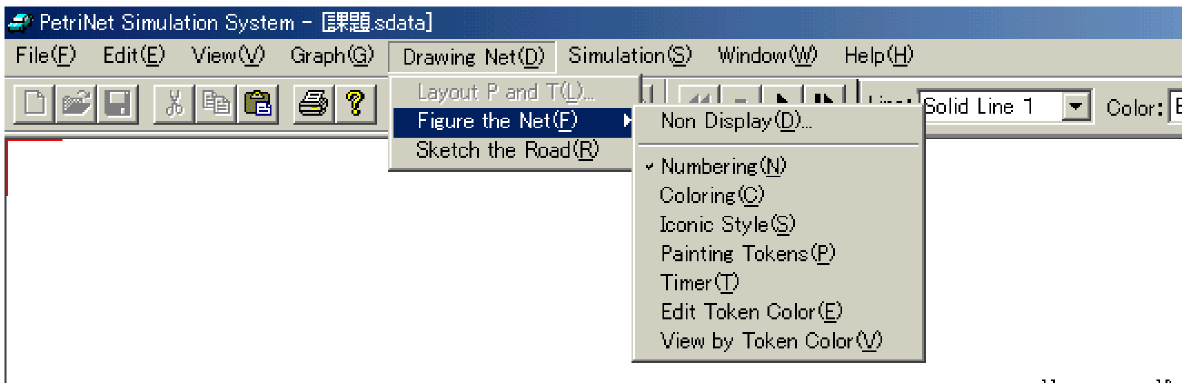
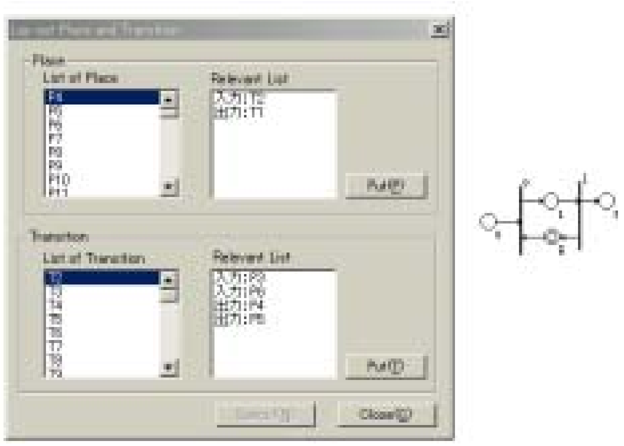


Fig.6 Basic Menus of Prototype Petri-net Simulator (Windows version)



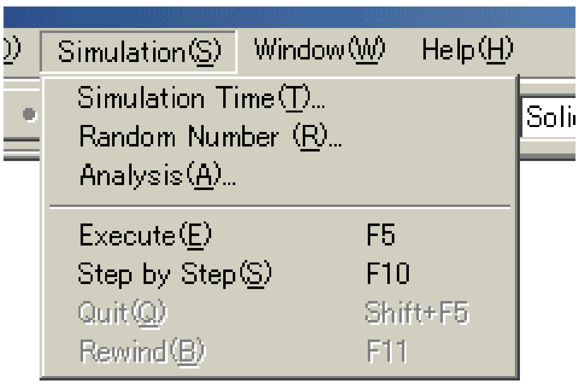


Fig.7 Interface for "Layout Places and Transitions"

Fig.8 Menus for Simulation Execution

applied to sub nets by selecting them as an entity. By adopting strategy to lay out in wider space at the earlier stage of net construction and ensure visibility and readability of the net, users can check more easily the logical relationships between places and transitions even if the net is complicated. At the latter stage, users can get more realistic configuration of the net by rearrangement to overlap some parts and/or move some parts to positions to well fit the real space.

The layout data of the net is saved as Ndata file paired with the Sdata by using "File" menu. After this, we open the Sdata then the net is drawn on computer. Users can do step-by-step wisely construction and execution of the net modeling supported by these functions.

"Figure the Net" supports users to ensure visibility and readability of the net and to trace token's movement on the net from more detail viewpoints. "Non Display" in its sub menu enables to draw components selectively as designated in the Sdata and simplify the representation of complex net. "Numbering", "Coloring", and "Iconic Style" help users discuss the net structure. Fig.1 is the net drawn by giving checks to "Numbering" and "Iconic Style" in this sub menu. Two special transitions GT and ST, introduced in 2-1, are displayed as bold bar and double bar respectively and become distinguishable at glance as iconic style.

"Painting Tokens" and "View by Token Color" help users trace visually token movement by coloring tokens. By "Edit Token Color", we design colors given to tokens in order to attach realistic colors or emphasize particular token movement intentionally. We show later an example of coloring of tokens.

Fig.8 is sub menu of "Simulation". Here we can set "Simulation Time" and initial values of "Random Number" and design "Analysis"

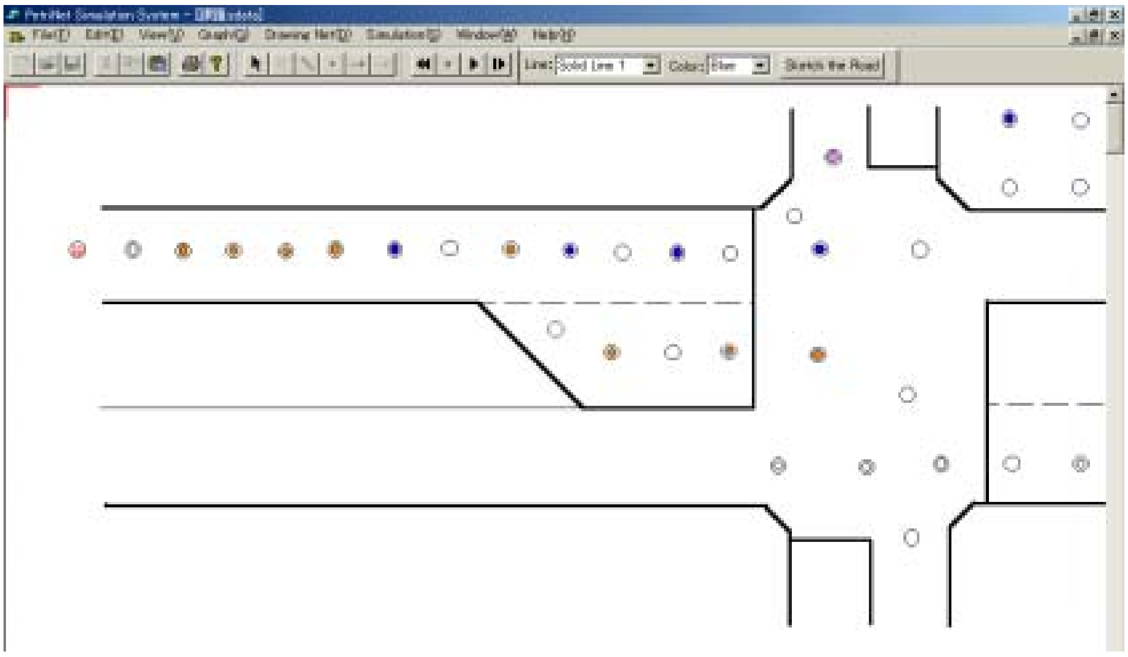


Fig.9 Representation of Traffic Flows Simulation Shots near Signalized Intersection

format. Implementation of simulation can be done in two modes, "Execute" and "Step by Step", and its rewinding is also possible by "Rewind".

Finally, we demonstrate an example of practical traffic simulation image at Fig.9. This is a shot of simulation of the net model shown at Fig.1. In this case, we designate the Sdata as only places inscribed as (-2) and places corresponding to traffic signal in Fig.1 are "visible" and all of others like transitions, arcs, places inscribed as (-1), etc. are "invisible". And using "Sketch the Road", we outline the road configuration on the drawn net and get the simple representation shown at Fig.9 where only real space and real entities are presented and conceptual elements in the net are eliminated.

We execute this net by giving check to "View by Token Color" and cut off a shot of simulated net markings at Fig.9. In this representation, tokens are colored according to their entities such as traffic signal, or cars of straight forward, left turn, or right turn. The shot shows us with these coloring of tokens the current phase of traffic signal, number of cars on each of lanes and their locations, and indicates visually adequacy and some causes of the current traffic situation. Step-by-step execution of this form and its rewind will facilitate to interpret and understand proposed Petri net model for practical planning.

4. Conclusion

In this paper, we described some fundamentals of our micro traffic flow Petri net simulator and proposed practical prototype system for its PI style usage. And we outlined the prototype focusing on menus equipped with its user- friendly interface.

First, we explained that "Drawing Net" has sub menu which consists of "Layout P and T", "Figure the Net" and "Sketch the Road", and they help users construct the net on computer, ensure readability and real presentation for complicated nets, and trace visually token behaviors on it by coloring them. Next, we showed that "Simulation" contains fundamental functions to set initial conditions for "Simulation Time" and "Random Number" and to design the plan of "Analysis" and revealed that two modes execution by "Execute" or "Step by Step" and "Rewind" mode are realized.

Finally, using these menus, we demonstrated representation of simulated shot of a Petri net model for traffic flows around a signalized intersection and showed the representation is highly informative to interpret and understand traffic Peri net models. We confirm that the prototype system will facilitate user's net model development and contribute to PI style usage success.

For future study, we need improve the execution speed of simulation and expand the scale of simulation to practical network level. Further more, we need provide more various and more refined sub nets which describe fundamental traffic flows.

5. References

1) W.Reisig: A Primer in Petri Net Design, Springer-Verlang, 1992.
2) K. Jensen: Coloured Petri Nets, vol.1～3, Springer,1997,
3) Noboru Kimata, etc.: Development of Traffic Flow Simulation System by Petri-Net Model, Infrastructure Review, No.12, pp.691～699, 1995 (in Japanese).
4) Noboru Kimata, etc.: Fundamental Study on Validation of Petri Net Simulator for Micro Traffic Flows, Proc. of Infrastructure Planning, No.23(1), pp.415～418, 2000(in Japanese).
5) T.Holvoet, P.Verbaeten: Using Agents for Simulating and Implementing Petri Nets, 11th Workshop on Parallel and Distributed Simulation, 134～137, 1997.