A train traffic model based on coloured Petri Nets and its application to a train schedule planning system

T. Sakaguchi, N. Tomii
Railway Technical Research Institute,
2-8-38, Hikari-cho, Kokubunji-shi, Tokyo 185, Japan

Abstract

We propose an approach to making train schedules by simulation and a train traffic model suitable for simulation. This model is developed based on a Coloured Petri Net (CP-net), an enhancement of Petri Net which is known as a graphic tool applicable to the analysis of concurrent, asynchronous and distributed systems. Some of the advantages of our CP-net based train traffic model are that it is a universal model suitable for various kinds of software which deal with train schedules; simulation on this model always produces a feasible train schedule and it is helpful in making the software more compact, robust, re-usable, efficient and enhanced. In this paper, we introduce the model together with an outline of a train schedule planning system developed using the model.

1 Introduction

Train Schedules are by far the most fundamental information for railway administration. It should be considered for a commercial product of railway companies. They are making continuous efforts seeking for a better train schedule by refining a current train schedule or by renovating facilities such as rolling stocks, railway facilities and so on. Also, they are eager to keep their transportation as stable as possible, hence, when train traffic is disrupted by accidents or some other troubles, they immediately try to normalize the traffic by changing the current schedule. This means a new schedule has to be made as soon as possible. All of these jobs seek one thing in common, that is, to be able to get a good train schedule in a short time.

In this paper, we propose an approach to making a train schedule by simulation, and a train traffic model which is suitable for simulating train traffic. This model is a sort of universal model for train traffic, which is effectively applicable for various kinds of computer systems which deal with train schedules. This model also has advantages in making software, which is easy-to-develop, portable, reusable and enhanced.

Concerning train setting in making train schedules, a sequential approach has been conventionally employed. That is, users set trains one by one, and the compatibility of a new train with other trains which are already set is checked every time a new train is set. Computers could inform them of errors but a lot of trials and errors are involved in order to fix the errors. Since it is not so easy even for experts to make a complete and consistent schedule, this approach

requires a long time until a satisfactory result is obtained.

We propose here an approach to making train schedules by simulation. First, users set trains using their intuitions and imaginations. This is a draft schedule which may contain contradictions and may be impossible to realize, in other words, this may be an infeasible schedule. Then, a simulation of train traffic is conducted on this draft schedule. The simulator detects and resolves the contradictions or modifies the draft into a more reasonable one. These operations are done all at once with the scope of the whole train schedule, hence, the number of trials and errors is greatly reduced. Another merit of the simulation approach lies in the fact that we can handle train schedules apart from constraints imposed by facility conditions and operational rules. Thus, even when some of them are changed (for example, a new station is constructed or a number of tracks is added in some station), by performing simulation on the current schedule under the new facility data, we can get a feasible and reasonable train schedule which is adaptable to the new situation. This is not the case in the sequential approach, because train schedules and the constraints are firmly mixed and difficult to segregate.

We believe that a model for train traffic plays a quite important role in the simulation approach. The model should satisfy the following requirements:

1. *Simulation results on this model are always feasible.* Getting a feasible schedule is not an easy job, thus to install functions to avoid infeasibility in a software is not a good idea, because this makes the software too complicated, too difficult to maintain and reuse, and furthermore, quite probably it will contain deficiencies. But all of these problems disappear if the model is designed in such a way that simulation results on it are always feasible.

2. *The model is universal.* Railway companies have various kinds of jobs in which train schedule has to be handled. Some of them are; train schedule planning, examination of facilities including cars and track facilities to get a better schedule, enhancement of the schedule to operate extra trains, control of urban dense traffic so that trains run smoothly, rescheduling of disrupted train traffic and so on; the purposes, processes, environments of these jobs are all different. If a train traffic model is universal, i.e. if it is applicable to all of these jobs, it will be quite beneficial.

We introduce a train traffic model which satisfies these requirements. This model is based on **Coloured Petri nets**. Petri nets are design and analysis tools used for discrete events systems[1], and Coloured Petri nets (**CP-nets**) are enhancement of Petri nets[2]. In our model, we represent physical constraints, which are restrictions imposed by facilities, by means of the structure of the net. Then, we represent control of train traffic (logical constraints) by means of **occurrence selection logic**. This logic selects an event which should be put into practice from workable events given by the CP-net structure

We have developed a train schedule planning system using this CP-net based model. We realized that the software was completed in quite a short time compared with our last experiences. Also, we are now developing other systems

which deal with train schedules using this model.

In chapter 2, we briefly explain Petri nets and Coloured Petri nets, and in chapter 3, we introduce our train traffic model. Then in chapter 4, a mechanism which enables our model's universality is introduced. In chapters 5 and 6, we will describe our train schedule planning system and our model's application to other systems respectively.

2 Petri nets and Coloured Petri nets

2.1 Petri nets

Petri nets are a graphical and mathematical modeling tool applicable to many systems. They are a suitable tool for describing information processing systems characterized as being concurrent, asynchronous, distributed, parallel and non-deterministic. Petri nets (**PT-nets**) are composed of places, input/output arcs, transitions and tokens (Figure 1). When each input place of a transition contains at least one token(we say this transition is **enabled**), this transition can occur. If this transition occurs, tokens are removed from the input places and added to the output places. So, states of PT-nets (**markings**) vary with occurrences of transitions, and this feature is used to simulate behavior of dynamic and concurrent systems.

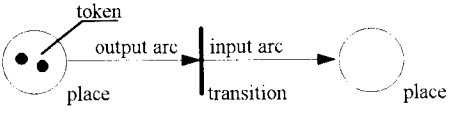


Figure 1: Components of Petri net

2.2 Coloured Petri nets

Coloured Petri nets (**CP-nets**) are an enhancement of PT-nets. An example of CP-nets is illustrated in Figure 2. Each token is equipped with an attached data value — called **token colour**. For a given place all tokens must have token colours that belong to a specified type — called **colour set** of the place. Token colours can be inspected by transitions, which means that enabling of a transi-

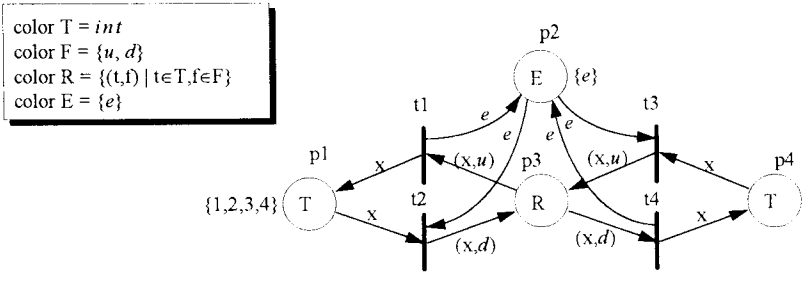


Figure 2: Example of CP-net

tion may depend upon the colours of its input tokens. **Arc expressions** are used to describe these conditions. So, in general, arc expressions contain variables. A binding for a transition is a set of instances assigned for the variables of input arcs of the transition. For each binding, we can check whether the transition is enabled or not. A pair (t,b) where t is a transition and b is a binding for t is called a **binding element**. In Figure 2, binding element ($t2$, $<x=1>$) is enabled in this marking.

One of the most essential differences between PT-nets and CP-nets is that in PT-nets the results of transition occurrences are uniquely determined, whereas in CP-nets, they are different depending upon binding elements. This fact means that CP-nets have an advantage over PT-nets in that the overall network becomes compact and simple, and that it is easy to grasp the behavior of the whole system.

The reason why we adopted CP-nets for modeling train traffic system is that a network model by PT-nets was suspected to become too large and complicated if we modeled a real railway network in which several tens of stations and thousands of trains are included.

3 CP-net Model of Train Traffic

3.1 Outline

Trains have to be operated observing various kinds of rules. Some of them come from facility conditions. The rule "*a train cannot pass another train between stations*" is a restriction of this type. We call restrictions imposed by facility conditions **physical constraints**. In our CP-net model, physical constraints are transformed into the structure of the CP-net, thus simulation using this model always produces a feasible schedule.

The other type of restrictions are the ones imposed by rules of operation. We call this type of restrictions **logical constraints**. "*Trains must not depart earlier than their scheduled time*" is an example of logical constraint. Sometimes logical constraints have to be observed and sometimes they should be broken in order to get a more reasonable schedule. In our CP-net model, logical constraints correspond to occurrence selection logic. By adopting and combining appropriate occurrence selection logic, we can give versatility to our simulator.

3.2 Transitions and places

Train traffic is considered to be made up of events and statuses. Events mean trains' arrivals and departures. Examples of statuses are : a block section is occupied by a train, a block section is empty, a train is stopping in a track of a station and so on. In a CP-net model, events are described by transitions and statuses are expressed by places. But unlike PT-nets, there is a lot of freedom in CP-net modeling. One extreme is depicted in Figure 3. In this model, trains, stations and tracks are considered tokens, and mutual interactions and conflicts between them are not well described in the structure of the net, i.e. they are

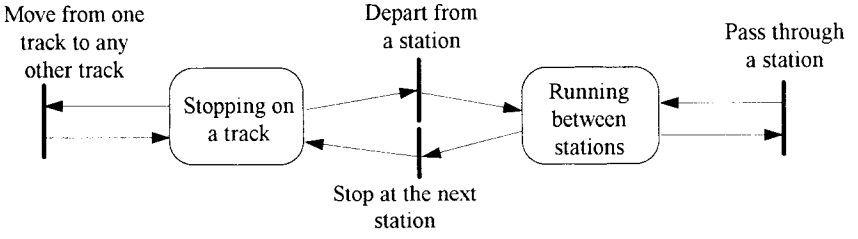


Figure 3: A reduced model of train traffic

hidden behind it. So obviously this modeling is not suitable to our purpose.

We made a model whose net structure exactly corresponds to a physical image of the railway line. In our model, events expressed by transitions are: trains' departure, trains' arrival, trains' passage at stations and so on.

Places represent statuses of elements or a pair of elements. In our model, trains, blocks, tracks and leading tracks are elements. Thus, places comprise —

P1: a train T is stopping on a track F. (train, track)

P2: a train T is staying on a leading track L. (train, leading track)

P3: a train T is running between stations S1 and S2. (train)

P4: there are n non-occupied blocks between stations S1 and S2. (block)

P5: a track F of a station S is empty (track)

and so on.

3.3 Event occurrence

Dynamics of train traffic is described by event occurrences in the model. Events may occur when a condition of the event occurrence is satisfied and this corresponds to trains movement. Event occurrence conditions relating with physical conditions are as follows:

(1) Train departure condition For a train to be able to depart from a station, the next block section from the station must be free. Using CP-net terminology, this corresponds to a condition that the place is available. After this event occurs, the place of the track where the train existed becomes empty, and the number of free section blocks between the stations is reduced by one.

(2) Train arrival condition Trains running between stations can arrive and stop at the next station in a time specified as minimum running time between the stations. The arrival event can occur if the place of the track has an "empty" status.

(3) Temporal condition There are two types of temporal conditions. The first one is a condition that a status has to be maintained for a certain time, and this is expressed by a place which has time delay. Constraints concerning minimum running times of trains, minimum stopping time of trains at stations are examples of this type. The other type is a condition concerning a chronological conflict among events. When one event occurs, the occurrence of another event

which conflicts with the former has to be delayed. Some of the examples of this condition are; time interval conflicts (a certain time interval has to be kept between two departing trains) and crossover conflict which happens between a departing train and an arriving train both of which use the same facility such as a crossing point.

In Figure 4, we show a part of our train traffic model. This example is illustrating trains movement which run through Stations A, B and C.

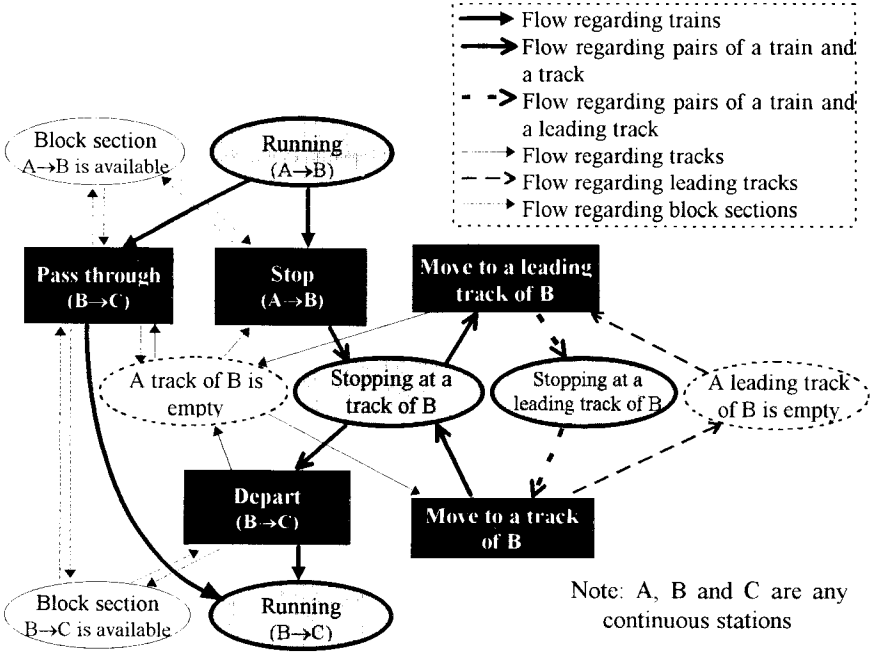


Figure 4: An example of our train traffic model

4 Occurrence Selection Logic

We have devised a variation of occurrence selection logics. An appropriate logic should be used depending on the purpose of train traffic simulators. We now cite a few examples of occurrence selection logics suitable for train schedule planning system.

(1) Logic 1: Occurrence selection logic which does not allow trains to depart and arrive earlier than their scheduled times This logic is appropriate when users want to give a slight modification to finish almost completed schedule. This is because, if trains are allowed to start without restrictions of scheduled times, the whole schedule may be changed, which is usually not desirable when users have almost completed their draft.

(2) Logic 2: Occurrence selection logic which allows trains to depart and arrive earlier than their scheduled times This logic plays an important role when users want to give a significant change to a schedule. For example, let us assume that users want to examine how much the current train schedule would be improved by introducing new cars of high performance. In this case, if Logic 1 is employed, trains which consist of new cars cannot run at their full performance as illustrated in Figure 5(A). So, in such cases Logic 2 should be used (Figure 5(B)).

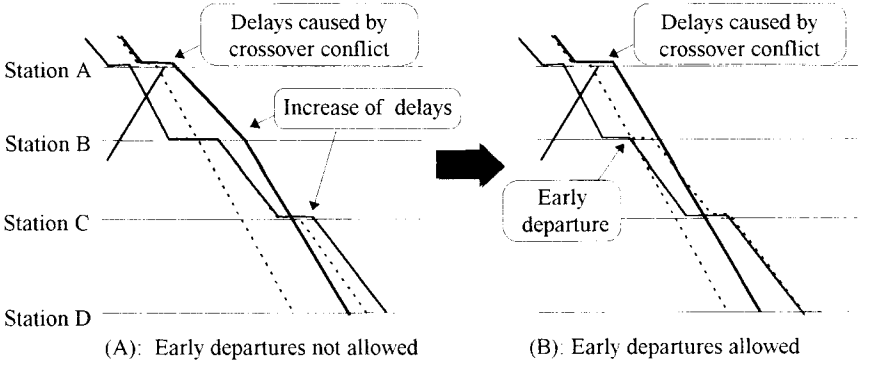


Figure 5: Improvement of schedule

(3) Logic 3: Occurrence selection logic which imposes constraints on departing orders of trains When this logic is applied, departing orders of trains are kept unchanged as long as no contradictions occur. This logic is effectively used combined with Logic 2. Let us assume that the departing order of trains T1 and T2 is specified as T1-T2. Then, even if train T2 is allowed to start earlier than its scheduled time, it cannot depart earlier unless T1 starts, and when T1 starts; train T2 can depart accordingly. So, even when early departures are allowed, the whole schedule will not be essentially changed.

(4) Other occurrence selection logics Other than Logics 1 - 3, we have installed a logic for track selection, a logic for delayed departure, etc. The logic for track selection works to select an appropriate track in a station when the originally assigned track for the train is in trouble such as a deadlock. The logic for delayed departure is useful to keep intervals between trains as equal as possible. A combination of logics which apparently look like contradictory often works very well. In Figure 5(B), The logic of early departure and the logic of delayed departure are used. You can see that these logics working very well to get a reasonable schedule.

5 Train Schedule Planning System — an Application of the Train Traffic Model

5.1 Outline

We have developed a train traffic schedule planning system using our train traffic model. This system consists a man-machine interface part and a simulator part which performs simulation based on the given draft schedule. Draft and final schedules are kept in a database system. Users examine the results of the simulator and they gradually brush up their draft schedules into complete ones by adding new trains or modifying unsatisfactory parts of the simulation results. This process is depicted in Figure 6.

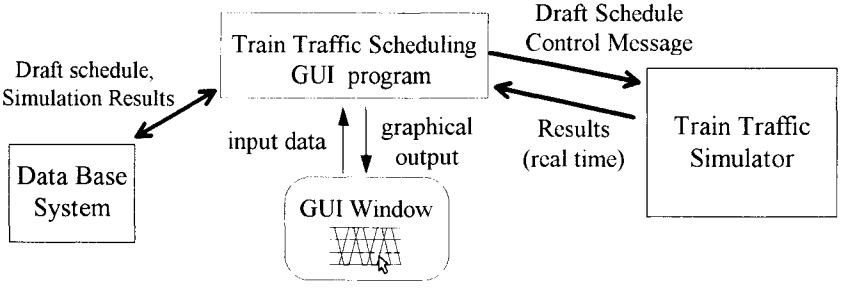


Figure 6: System configuration

5.2 Examples

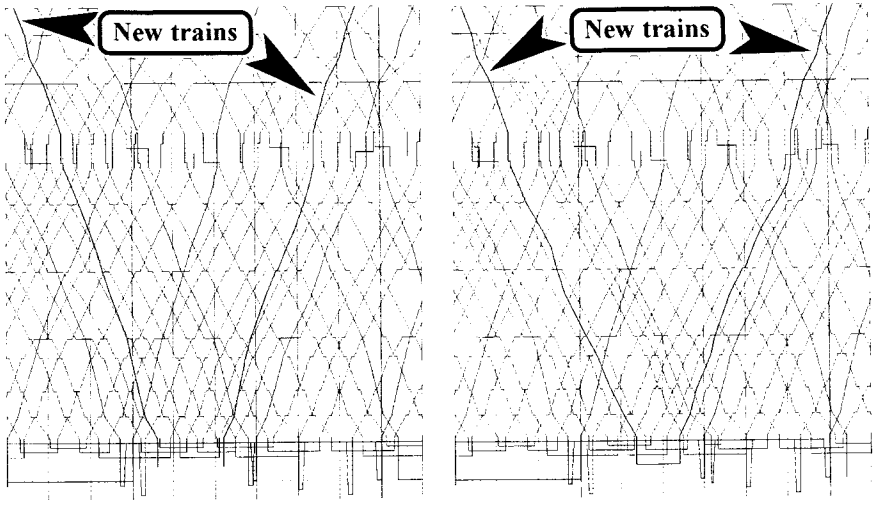
In Figure 7 we show an example which illustrates how to use our train traffic schedule planning system. A case where several express trains have to be newly created is assumed. Figure 7(a) shows a draft train diagram just after those trains are set. You can see the schedule infeasible because there are a lot of conflicts and contradictions. By conducting simulation, these contradictions are resolved and adjustment of departing times and change of train departing orders are also done and a feasible and more reasonable schedule is obtained as shown in Figure 7(b). Please note that the track of a train is also changed to avoid a conflict.

5.3 Evaluations

We realized that we can make feasible and reasonable schedules quite easily on our prototype system. This means our simulation-based approach to schedule planning was proved effective. Simulation time is about 90 seconds for 24 hour train schedule of an urban railway line near Tokyo (the number of simultaneously running trains is about 40), which we think is quite acceptable for practical use.

Another thing we want to insist on is that it took quite a short time until we implemented a satisfactory prototype system. This is an effect of CP-net model-

ing which separates the treatment of physical constraints and that of logical constraints. Programmers do not have to write even one single line of code concerning observance of physical constraints. They can also use modules of occurrence selection logic which are already prepared and ready to use. We believe all of these lead to a reliable software with versatile functions and realize a short development time.



(a) Infeasible diagram (b) Simulated output

Figure 7: Example of simulation

6. Other Applications of CP-net based Train Traffic Model

We are now developing the following systems using our train traffic model.

(1) Train traffic rescheduling system The purpose of this system is to normalize a disrupted train traffic schedule. The whole framework of our train schedule planning system can be used. But as for occurrence selection logics, a logic to change departure orders is useful, and logics for adjustment of train intervals and early departures should not be used.

(2) Train traffic simulator for urban traffic control This simulator is designed to investigate a traffic controlling algorithm for railway lines of dense traffic. On such lines one of the most important things is to secure a stable transportation and normalize congestion of trains. The following occurrence selection logics seem to be helpful to keep proper intervals of trains: 1) Logic 1 described in chapter 4 (early departures are not allowed), 2) Adjustment of train intervals, 3) To keep departing orders (normally, scheduled departing orders should be maintained, but in case traffic is disrupted, logic to change departing

orders should be used.).

(3) Simulator for diagram-less train operations On some railway lines where a lot of trains are operated, passengers do not care about train schedules. All they need is that trains which are not too crowded come with proper intervals. In order to realize this, there are a lot of things to be investigated. The simulator we are now developing is expected to be useful for these investigations. We think the following occurrence selection logics will be promising: 1) Adjustment of train intervals. Not only trains' departure times but trains' arrival times should be changed by adjusting trains' running time. 2) Adjustment of the number of trains which is suitable for the necessary volume of transportation.

(4) Decision making support system As we mentioned earlier, train schedule should be considered a commercial product of railway companies. They want to introduce new cars which run at a higher speed, renovate facilities so that trip times become shorter, or sometimes construct new stations so that their railway can attract more passengers. These are all done with an intention to get a better train schedule. Thus, train schedule exists at the core of these decision makings, and a decision making support system with which users can instantaneously know how much a train schedule would be improved by facility renovation is needed. We are now developing such a decision making support system by combining the train schedule planning system and train performance evaluating system SPEEDY which we have already developed and which is in use[3].

7. Conclusions

We proposed a simulation approach to making a train schedule and introduced a train traffic model based on Coloured Petri net. We also introduced a train schedule planning system using the model. Then we showed the model' universality, which means it is applicable to various kinds of systems related with train schedules.

Acknowledgments

The authors wish to express their sincere gratitude to Mr. Ikeda, Mr. Nozue and Mr. Hasegawa for their useful comments and discussions.

References

1. MURATA, T. Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE*, 1989, 77, 541-580.
2. JENSEN, K. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, Springer-Verlag, Berlin and New York, 1992.
3. HIRANO, J. et al. Development of a train performance computation system on an engineering workstation, *Proceedings of COMPRAIL92*, 1992.