Duke University

Petri Net

Xianjue Huang

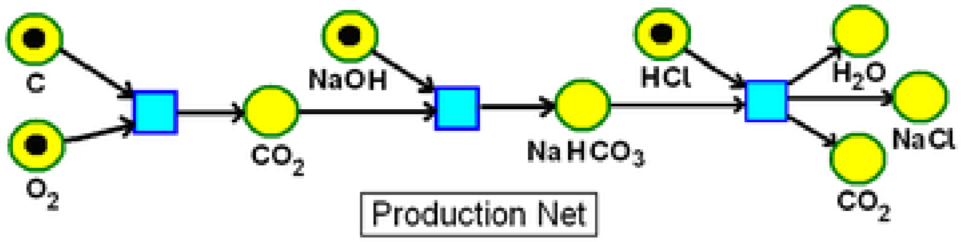
Math of the Universe

Hubert Bray

31 July 2017

Introduction

It would be really interesting when there exists one thing that can simulate lots of different situations in different areas, and that thing is called the "Petri Net". Imagine you are studying a chemical reaction (assume that you do not know the chemical elements that will be involved in the action). What normal researcher will do is that they will probably try hundreds of time and then write down every single chemical elements that involved in the reaction on a piece of paper. But the "Petri Net" can simulate the situation in a very simple form of expression that involved with the substances and their actions (see the picture below as an example). With the same idea, many other situations like the production line of a car, the progress of an online searching request, and even the routine that one person goes to his/her work and then goes home!

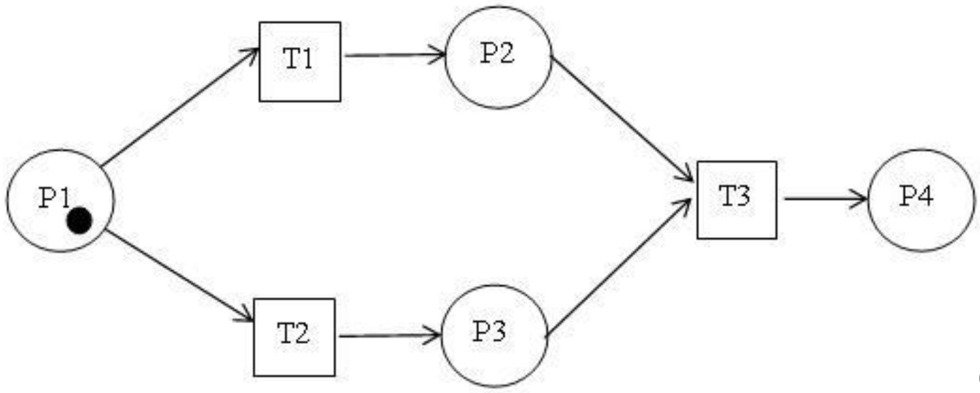


(1)

This paper will focus on what is the "Petri Net", how it works, and some of its applications in other fields specifically, the application on the elevator problems.

What Is The Petri Net And How The Net Works

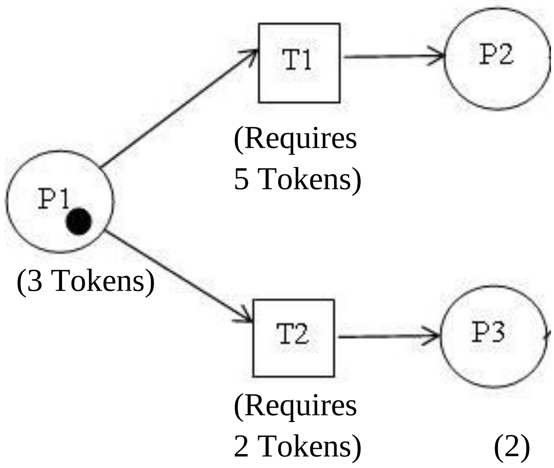
In the 1960s, one German mathematician as well as computer scientist called Carl Adam Petri stated his idea of "Petri Net" for a graphical tool that can be used to describe and analyze a concurrent process in a system with lots of components. Mr. Petri first came up with this idea when he was a 13 years old boy with the purpose of describing the chemical processes. One good thing about this net is that according to Petri, the net is fitted with the "Special Relativity" and that might be the reason for it fitting with many complex real cases. (1)



(2)

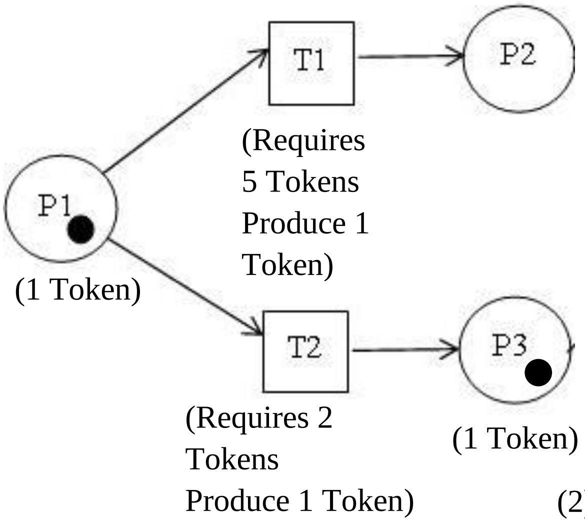
The graph above shows a typical "Petri Net". In the graph, we can tell that there are 2 kind of blocks, one is circle like, and the other is square like. The circle like block is called the "Place", whereas the square like block is called the "Transition". Besides the blocks there are many arrows, named as "Arc". The block dot in the circle "P1" is the "Token". In theory, the "Petri Net" is a static net, which means after any changes, there will not show up a new "Place", a new "Transition", or a new "Arc"; whereas, the "Token" is the dynamic part that will move around the net. Furthermore, the “Arc” has its direction, limiting the direction of each movement. An important thing about “Petri Net” is that the “Arc” never lines up between two “Places” or two “Transitions”, meaning the structure must follow the pattern of one “Place” connect with one “Transition”. Before each actions, each “Place” can be seen as an “Input Place”, which contains any number of “Tokens”; however, each “Place” can be also seen as an “Output Place”, since each of them will have a chance to be involved in an action. The action, which is called as “Fire”, can only take place when the number of the “Tokens” inside the “Input Place” matches the lowest requirement. After each “Fire”, the “Tokens” in the “Input Place” will decrease by the number of the requirement, and the number of the “Tokens” in the “Output Place” will increase by 1. For example, in the graph above, the “Token” can only fire into “Place” P2 or “Place” P3, and if the “Transition” “T1” is activated, the “Token” in “P1” will be fired in to “P2” but not into “P3”. (3)

In the picture on the right, I cut the original picture shown above and made some changes so that each “Transition” has requirements, and the number of the “Tokens” is also changed. In this situation, the “Transition” “T1” will not get



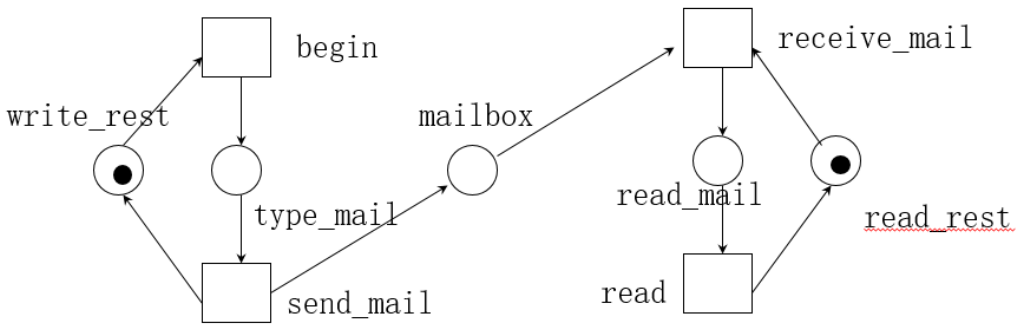
(2)

activated again because there are only 3 “Tokens” in “P1”, and the firing of “T1” requires 5 “Tokens”. By contrast, the “T2” is enabled, so that two of the “Tokens” in “P1” will be fired through “T2”. So after the action took Place, the number of the “Tokens” remained in “P1” will be 1, and 1 for “P3”.



(2)

Another interesting thing about this net is that each net can have a matrix form. The picture below shows how the net is converted in to a matrix. The column represents different “Place”, and the row represents different “Transition”. Take row

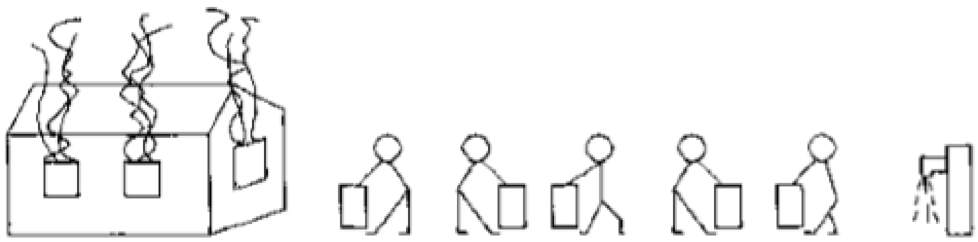


	bgn	sdm	rvm	red
wr	−1	1	0	0
tm	1	−1	0	0
mb	0	1	−1	0
rm	0	0	1	−1
rr	0	0	−1	1

(3)

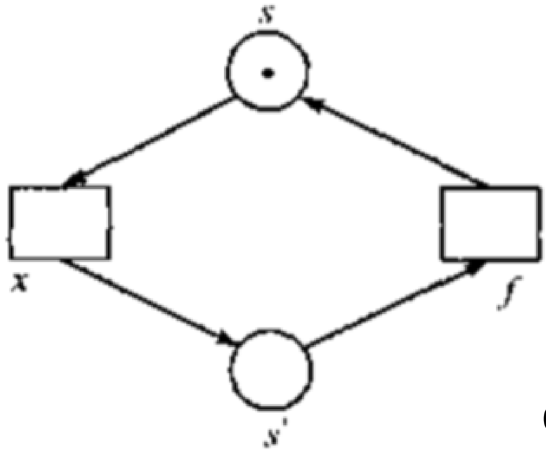
started with “wr” (write_rest) as an example. If the “Token” is fired from “wr” to “tm” (type_mail), then it lose one “Token”. Because it is fired through “Transition” “bgn” (begin), it shows a -1 on the block where “wr” meets “bgn”. By contrast, “tm” will lose one if the “Token” is fired from it to “wr” through “sdm” (send_mail), so there is a 1 on where “wr” meets “sdm”. (3)

The paragraphes above shows the basics of the “Petri Net”, but now, I want to talk about how this net can be used to simulate real situations. Imagine there is a fire case happening at a house, and the fire fighters are carrying buckets to hold and pass water. (4)



(4)

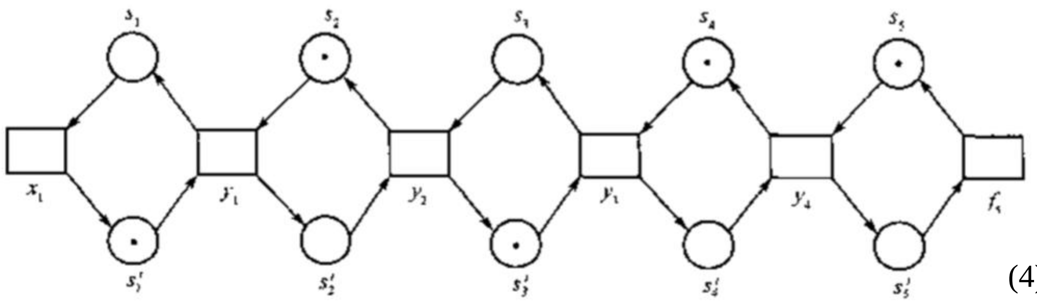
Let there be only one firefighter, so that what the net should express is the situation of the man, meaning whether he is carrying buckets with water or not, the “Transitions” of getting water from source and fire fighting. The graph below shows this case, where “s” is the situation that the firefighter is carrying water, and “s’ ” means he/she is carrying an empty bucket. If “Transition” “x” is activated, then the water in the bucket has been used to against the fire, similarly, if “Transition” “f” is activated, then the bucket has been refilled with water. The “Token” in “s” shows the situation that right now the firefighter is carrting the water.



(4)

This kind of model will be simulated as there is only one person (maybe the house keeper) is fighting the fire. (4)

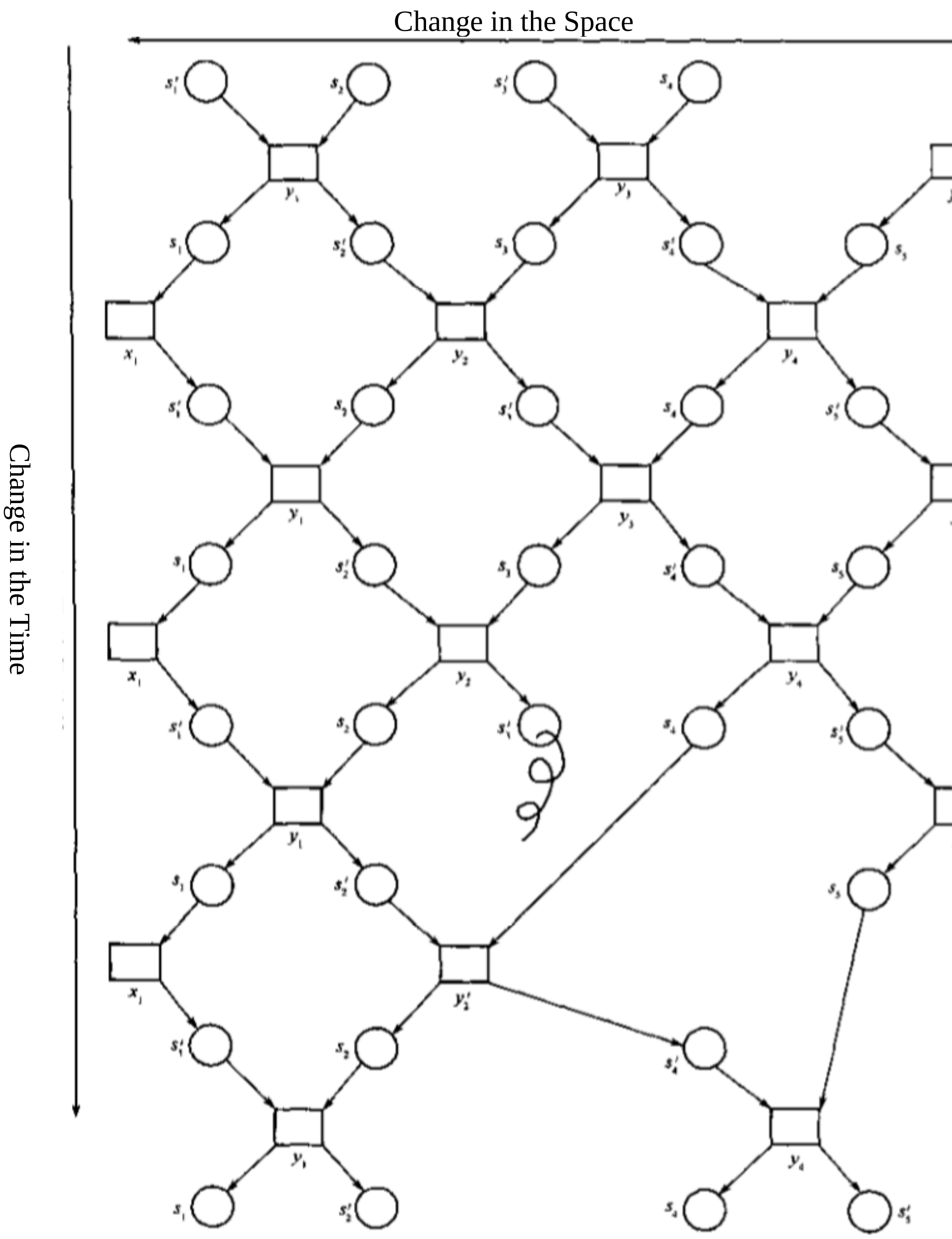
Now, let us look at a more complicated situation that is all the five firefighters got the SOS call and got to that house and all of them are involved in fire fighting. With the similar idea, each “Place” will represent the situation of individual firefighters. Also, a new kind of “Transition” appears, and that is the simulation of one passing the bucket to the one next to him/her. (4)



(4)

The picture above shows that situation. The “Place” lined up in the same column still represents the situation of the same firefighter, and the “Transition” “x” and “f” has the same function. Where are there are four new

“Transitions” with label “y_x”. How these new “Transitions” work is that if, the “Transition” is activated, assume we just talk about “y_1”, then, the empty bucket will be passed from “s_1’ ” to “s_2’ ”, and the water bucket will be passed from “s_2” to “s_1”. With this idea, the whole net can be used to simulate the fire fighting situation at every second with a clear and simple expression. The picture above shows the record of this fire fighting. (4)

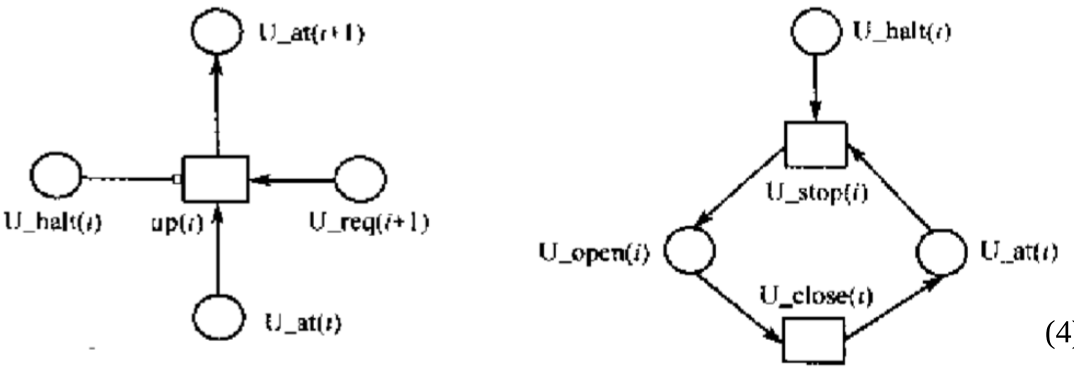


(4)

Application Of The Elevator Problem Using The Petri Net

Sometimes, there might be some situation that the elevator is close to you enough but it does not come to take you, so here, I intend to find out a “Petri Net” system that can maximize the usability for both the elevator administrators and the users. So first, let us assume that there are M floors for a building, and there are N elevators in the building. Inside each elevator, there are M buttons that make up the one-to-one relation with M floors, so that whatever the input the user give, the elevator should go to that floor. Then on each floor, there should be two additional button, (except for the first and the toppest floor) the button to go up and the button to go down. The Petri Net is the best choice to maximize the efficiency, because of its web-like-structure, and its partial-determinacy. Also, to maximize the efficiency, first, each elevator should be self-governing, then, each elevator should serve with the idea of “serving along my way”, in another way meaning, an elevator only goes to find request when there is no request on its way. However, if there is no request along its way or its serving direction, it should change its direction, and the direction should always be up or down, without the “no direction” form. Moreover, the number of forward serving elevator should be 1 for each request. Lastly, the existence of the forward serving elevator does not affect the other elevators to serve along their way, and each elevator should not be the forward serving elevator when there are two or more than two requests along its way so that other elevators can give services. (4)

Before we start to define the net for the elevator, let us first define some thing. Let *I(i)* be the button inside each elevator, where *i = 1,2,3,•••, M*, the floor numbers. Then, let *Fu(i)* (goes up) and *Fd(i)* (goes down) be the two buttons on each floor. And here, we define that every element that relate to “Place” will be represented by capital letters like “U” for ascending, “D” for decending, whereas every element that relate to “Transition” will be represented by normal letters. Lastly, because there exists symmetry between ascending and decending, so here we will only talk about the ascending one, and different floors are just different numbers so that we can simplify the problem into first find out what one floor will look like and then apply it to all the floors. (4)

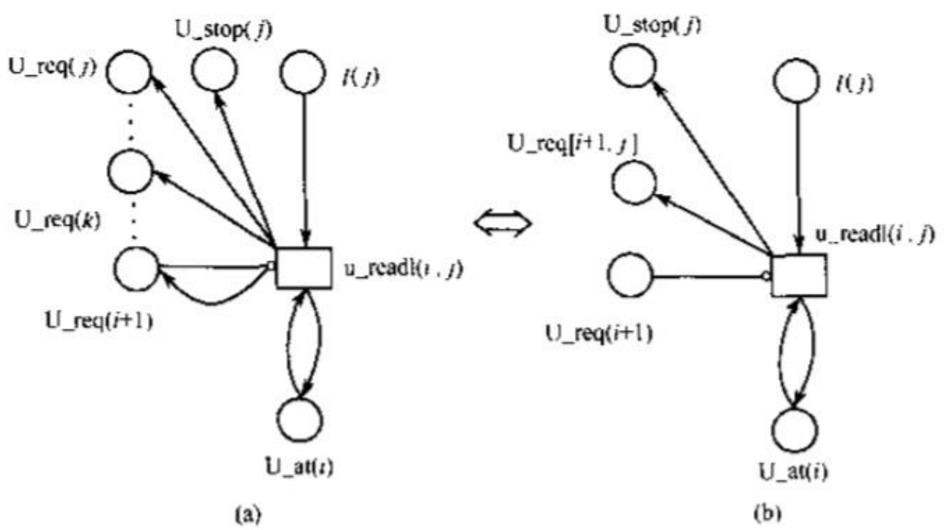


(4)

The pictures above show how the “Transition” “*up(i)*” and relavant inside “Transitions” “*U_stop(i)*” and “*U_close(i)*” should be lined up. In the leftern graph, the “*U_req(i+1)*” represent the request for going up one floor. For example, if the elevator is currently at floor 3, and the user requires it to go to floor 5 and pick him up, then we could write the following to represent the request: “*U_req(4)*”+“*U_req(5)*”+“*U_halt(5)*”. So much for that, the “*U_at(i)*” in the leftern graph means that the elevator is currently at floor *i,* and "*U_halt(i)"* is a restrictive "Place", so that if there is a "Token" in side it, the elevator should stop and the net in the rightern graph should be activated. So what the rightern graph reqresent is that if there is a "Token" inside "*U_halt(i)*", "*U_stop(1)*" and "*U_close(i)*" should be activated so that the elevator will open its gate and then after some seconds, close its door, and then it could activate the "Transition" "*up_(i)*". Now, let us see how the elevator should analyze the requests from inside-elevator buttons. (4 (same as the formula))

$$U\_at(i) \wedge \neg U\_req(i+1) \Rightarrow l(j) = U\_req(i+1) + \cdots + U\_req(j) + U\_stop(j)$$

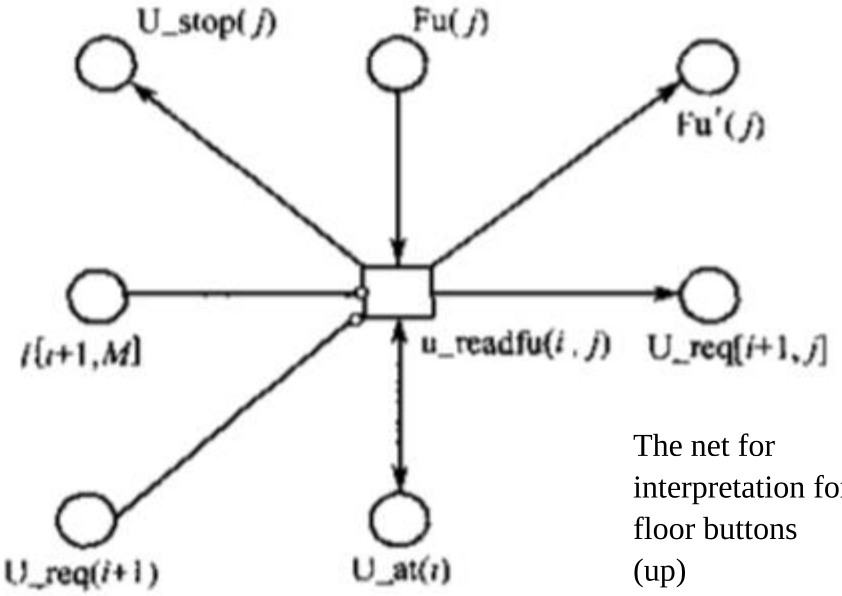
Assume the elevator is currently at floor *i* and the request is to go to floor *j* , then the interpretation for *l(j)* is defined as the formula above. The reason for having a "U_stop(j)" is to make sure that the elevator will be able to stop there and do further actions, and keep the elevator safe when it is functioning as well. So the corresponding "Petri Net" for this interpretation should be as follow: (4)



(4)

The graph (b) is the simplified version of graph (a), where "*U_req(i+1,j)*" represents all the request for going up for one floor. After defining the net for interpreting inside buttons, remembere that elevators should also go and search for the request on each floor, that is, to find "*Fu(j)*". However, as I mentioned before, the number of forward serving elevator should only be 1, so once one elevator finds the request and claims that it is the forward serving elevator, other elevators should not compete with that elevator, what they could do is just offering help along their way, and this offering is defined as "*Fu'(j)*". So the interpret formula of the elevators is defined as where "*U_req(i+1, j)+U_stop(j)*" is the part that what the forward serving elevator decides to be blocked, and the "*Fu'(j)*" is the non-blocked part; the condition on the first line assure that the elevators will only interpret the floor buttons when there is no request from elevator buttons: (4)

$$U\_at(i) \wedge \neg U\_req(i+1) \wedge \neg(\exists k: I < k \leqslant M: l(k))$$
$$\Rightarrow Fu(j) = U\_req(i+1) + \cdots + U\_req(j) + Fu'(j) + U\_stop(j)$$

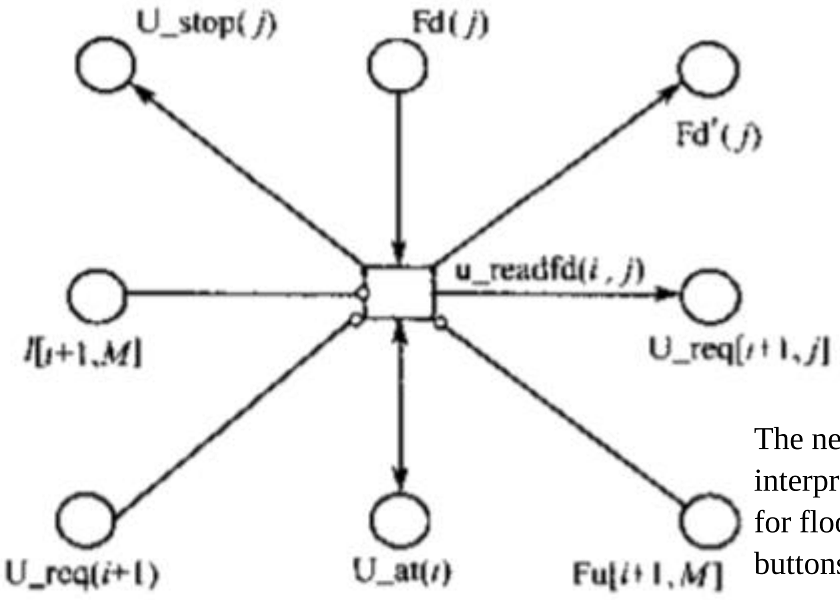


The net for interpretation for floor buttons (up)

(4)

When there is no request for the elevator that fits the condition, first, the request emits from higher floor compare to the elevator; second, the request is requesting to go up, then the elevator should search for the request on the higher floor to go down. Similar idea will be applied to this situation, where the interpretation formula for this will be as followed (one noticable thing is that in this situation, if the forward serving elevator keeps on ascending, then it might become the forward serving elevator for another “Fu” or “Fd”). (4)

$$U\_at(i) \wedge \neg U\_req(i+1) \wedge \neg(\exists k : i < k \leqslant M : l(k) \vee Fu(k))$$
$$\Rightarrow Fd(j) = U\_req(i+1) + \cdots + U\_req(j) + Fd'(j) + U\_stop(j)$$



(4)

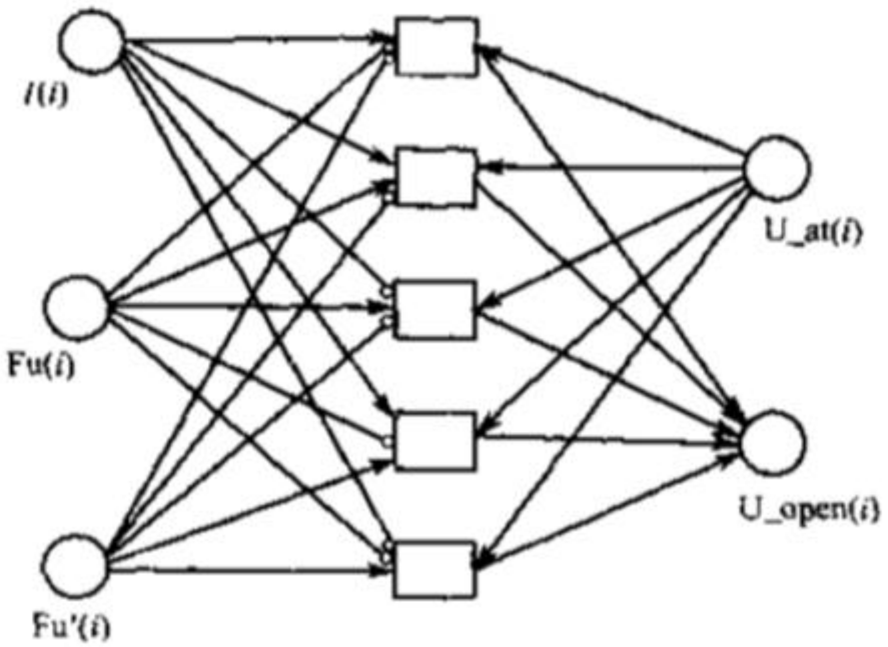
The net for interpretation for floor buttons (down)

Now I want to discuss about what should the “Petri Net” be like when the elevator make its way to the floor. At first, the elevator should stop, and then it should open its door. After that, the door should close, and then the elevator can go on and do other actions. (4)

So, let us assume that the request is to let the elevator go to floor *i* and stop. This means that the condition for "*u_stop(i)*" to be activated is that the "Places" "*l(i)*", "*Fu(i)*", and "*Fu'(i)*" should have at least 1 "Token" (see above paragraphes and graphes for the meaning of the three "Places"). When it is activated, the "Tokens" will be consumed totally, and there will be only one "Token" created in the "Place" "*U_open(i)*". Meanwhile, "*Fu(i)*" and "*Fu'(i)*" are in opposite situations, so that they cannot hold "Token" at the same time. So the number of combination of the "Places" and "Tokens" will just be 5, which are: (4)

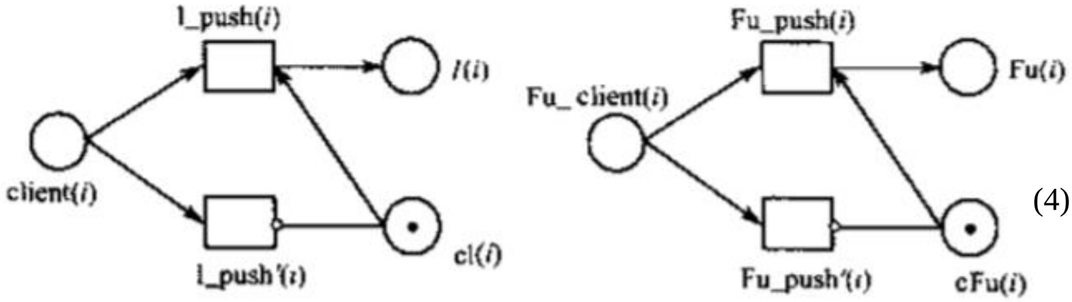
$$l(i), \mathrm{Fu}(i), \mathrm{Fu}'(i), l(i) \wedge \mathrm{Fu}(i), l(i) \wedge \mathrm{Fu}'(i) \quad (4)$$

The net for this is placed below:



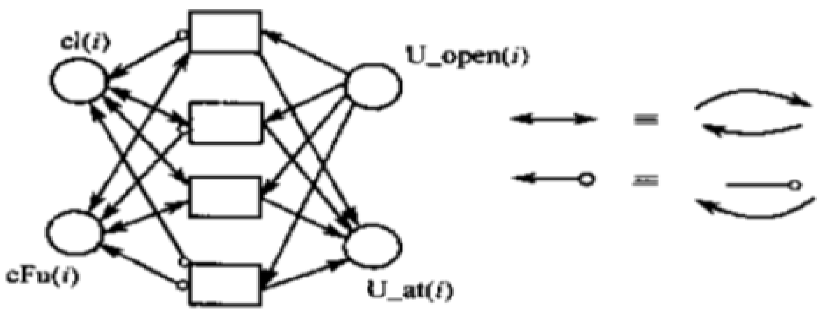
(4)

Here I want to point out an important thing: there might exist the situation that a user puch the same button more than one time, so that there must be a safety “Place” to prevent the situation when more than one “Token” is created. So here I define that if there is a “c” in front of the ”Place”, for example “*cl(i)*”, it means that this “Place” is the control “Place” (see the example below). (4)



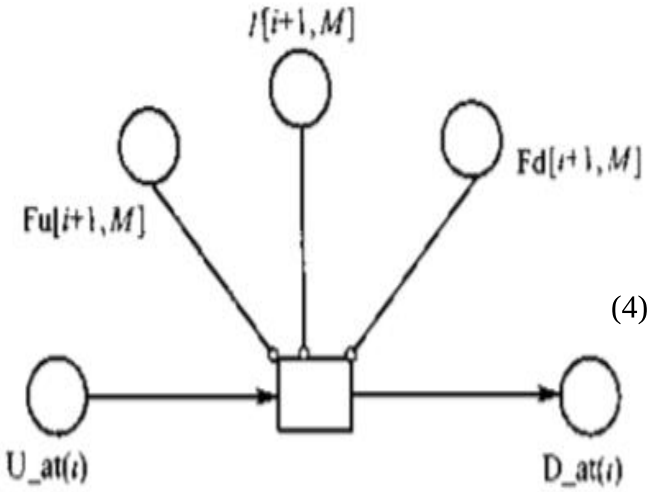
(4)

The example both shows that if the “*client(i)*” is activated, the “Transition” with “*_push(i)*” suffix will be activated so that “*l(i)*” and “*Fu(i)*” will get the “Token” and then the “Transition” with “*_push(i)*” will not be able to be activated again, then what will be activated is the “Transition” with “*_push’(i)*” suffix, and then “*l(i)*” will not get any “Token” so that the safety has been assured. Then this process should be continuable right? So what step we have after open the door should cover this action. Therefore, during the closing process, the “Petri Net” should in clude the action for giving back on “Token” to those c-“Places”. The net for the “*u_close(i)*” is made up with four “Transitions”, each relate itself with one kind of the combination of “*cl(i)*” and “*Fu(i)*”, and the net is listed below: (4)

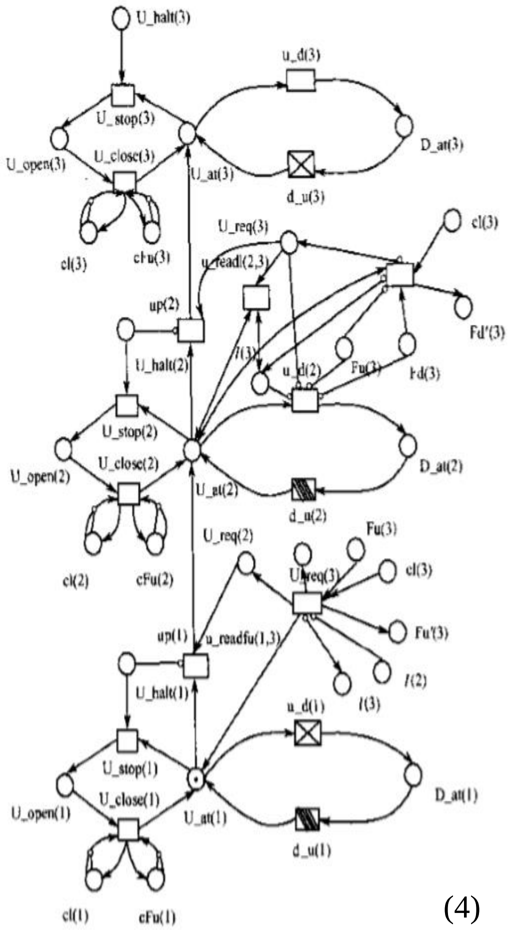


(4)

The last thing that should be mentioned is how the net enables the elevator change its direction (because the symmetry for going up and down, only shows change from going up to down here). So the basic idea is that only when the elevator cannot find any other request in the ascending form, the elevator can and should change from searching upward to searching downward. So the restrictive "Places" should be "*I(i+1,M)*", "*Fu(i+1,M)*", and "*Fd(i+1,M)*". (4)



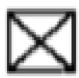
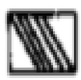
(4)

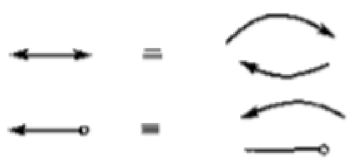


(4)

With all these nets, the net system for elevators could be set up by cloning the down side net as well and combining them together. (see the net on the left for M = 3) (4)

Conventional Signs:

	Transition that never works
	Transition with incomplete "Places"



(4)

Conclusion

Petri Net is a mathematical modeling language for the description of distriibutions. Even though it has some weakness, like sometimes this net might be too complicated to understand or to read, but it is still a good and useful modeling language since lots of situations can be simulated by it. I believe that the Petri Net will be the solution towards more and more difficult problems we will meet in the future.

Bibliography

Scholarpedia, Petri net, July 7/28 2017,
http://www.scholarpedia.org/article/Petri_net (1)

Wikimedia Foundation, Petri net, Wikipedia, July 28 2017,
https://en.wikipedia.org/wiki/Petri_net (Eng)
https://zh.wikipedia.org/wiki/Petri%E7%BD%91 (CHN) (2)

闻立杰 (Professor of Tsinghua University more Information : http://ise.thss.tsinghua.edu.cn/~wenlijie/) , January 26 – January 30, Field – Trip-Lecture on Petri Net, Tsinghua University School of Software, January 26 2017. (3)

袁崇义, 2005, Publishing House of Eletronics Industry, Chapter 1,2 and 12, July 7/27/2017. (4)