The algorithm description of the shortest possible sequence of transitions in Petri Nets

Goharik, Petrosyan

International Scientific-Educational centre of NAS RA
Yerevan, Armenia
e-mail: petrosyan_gohar@list.ru

ABSTRACT

This work is dedicated to several structure features of Petri Nets and the reachable tree. There is detailed description of appropriate access in Petri Nets and reachable tree mechanism construction. The reachable tree is retrieved, corresponding to Petri Nets. Then the infinite reachable tree is replaced with "finite" tree, by introducing an item, which replaces the idea of an infinite. There is algorithm description of the minimal sequence of possible transitions. The designed algorithm gets the shortest possible sequence for the net advance state, which brings the mentioned net state into covering state.
There is theorem, which states that through the describing algorithm, the number of transitions in covering state is in minimal.

Keywords

Petri Nets, reachable trees, transition, position, set, covering condition, token, capacity.

1. INTRODUCTION

Construction of discrete systems models need system components, with its operations in the abstract, such as, the program operator action, trigger transition from one state to another, interruptions in the operating system, machine or conveyer action, project phase completion etc. In general, the same system can operate differently in different conditions, bringing a multitude of processes, which means operating not deterministically. The real system operates in certain time, cases occur in certain periods and last for certain time. In synchronic models of discrete systems, the events are clearly associated with certain moments or pauses, during which all the components make simultaneous change in the system state, which is interpreted as a change in the system state. State conditions change successively. Alongside, these large systems, modeling approach has several drawbacks.

The condition is consistent with the existence of situations such as the operation of the system modeled data: any registry of computer equipment, parts availability on line.

Terms defined combinations allow to implement any of the cases (cases precondition), and the implementation of changes create certain conditions (cases post condition), which means the cases co-incident with the terms and conditions of the case.

Therefore, it is natural that many systems are suitable as discrete structures, consisting of two elements: the type of events and terms. The cases and terms in Petri Nets are disjoint sets with each other, respectively, called transitions and positions sets. Transitions are depicted in a graphical representation of Petri Nets (vertical lines), and places, with circles [1-3].

2. PETRI NETS, REACHABLE STATES, REACHABLE TREES

Definition 1: Petri Nets are in $M(C,\mu)$ pair, where $C=(P,T,I,O)$ is the net structure and μ is the net condition. In structure C of P -positions, T -transitions are finite sets. $I:T\to P^{\infty}, O:T\to P^{\infty}$ are input and output functions, respectively, where P^{∞} are all possible collections (repetitive elements) of P . $\mu:P\to N_0$ is the function of condition, where $N_0=\{0,1,...\}$ is the set of integers.

Saying net state, we will understand the following:

$$(\mu(P_1),\mu(P_2),...,\mu(P_n)),\ n=|P|,\ P=\{P_1,...,P_n\}$$

Suppose we have $M=(C,\mu)$.

We will say that in μ state $t_j\in T$ transition is allowed to implement if for $\forall P_i\in I(t_j)$ there is:

$\mu(P_i)\geq\#(P_i,I(t_j))$. Suppose in μ state t_j transition is allowed to implement and it is actually acted. In this case the net will appear in its new state, μ', which is solved in the following way:

$$\forall P_i\in P, \mu'(P_i)=\mu(P_i)-\#(P_i,I(t_j))+$$
$$+\#(P_i,O(t_j))$$

The meaning of coverage problem is for μ' decide whether it is reachable to $\mu''\geq\mu'$. The coverage problem can be solved through the reachable tree. At first for μ we will build the reachable tree. Then we will search x peak in the way that $\mu[x]\geq\mu'$. If there is no such a peak, then μ' marking isn't covered with any reachable marking, if it is located in $\mu[x]$ and gives a reachable marking which covers μ'-ը [4-6].

Let's build Petri Nets reachable tree in picture 1. The natural state of this net is (1101), which shows the presence of tokens in the net at that moment. The tokens that are in picture 1 with little dots, correspond with the presence of resources in the net. The state of the net is due to the move of the tokens.

Let's correspond states in the edges of the peaks, and transitions in the sides. The root is corresponded with the first stat of the net.

Picture 1. is corresponded with picture 2., in which the reachable tree is infinite. Let's put limitations, for the tree to be finite. If any peak is locked, then we will name it as terminal. If there is a state in any peak and there is another peak in the tree with the same state which is already developed, then we will name the new peak as repeated and will not develop it.

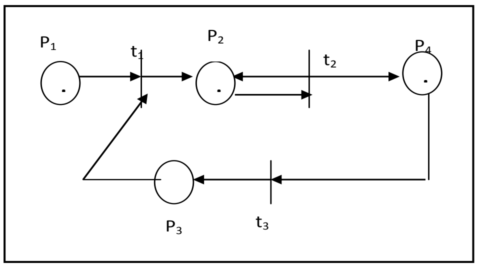
If there is /*/ type way in the tree, then the way through the second peak can be repeated and the states will grow.

Let's introduce the idea of infinite much as ω :

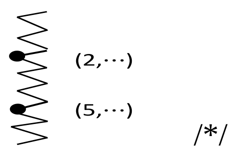
$\omega \geq \omega$, $\omega + a = \omega$, $\omega - a = \omega$, where $a = const$:

For example, instead of (5,⋯), we will write (ω,⋯). In this case the tree will become as finite [1], and we will have loss of information.

Let's give several definitions, which will be used in entire work.



Picture 1. An example of Petri Nets.



A way in the tree.

Definition 2: The peak is called as boundary if it is a subject of processing.

Definition 3: The peak is called as terminal if it doesn't content a sub tree.

Definition 4: The peak is called internal, if it is already processed.

Definition 5: The Boundary Peak is repeated if there is an internal peak with the same state.

Let's describe the structure of the algorithm of the reachable tree.

Suppose x peak is the next Boundary Peak. The state relating to it, will be marked as $\mu[x]$. Let's mark with $\mu[x]_i$ the i – state condition vector.

1. If x is terminal or repeated peak, then when processing, it will become as second peak and go to the second step.

2. For $\forall t_j$ transition $t_j \in T$, so that $\delta(\mu[x], t_j)$ is solved, then do the following: Add the tree a new side coming from x , and mark the side as t_j and name the peak as z .

The $\mu[x]$ will be solved with the following way.

If from the tree root, on its way to bring z , there is y peak, that

$\mu[y] \leq \delta(\mu[x], t_j) \,\&\, \mu[y]_i < \delta(\mu[x], t_j)$,

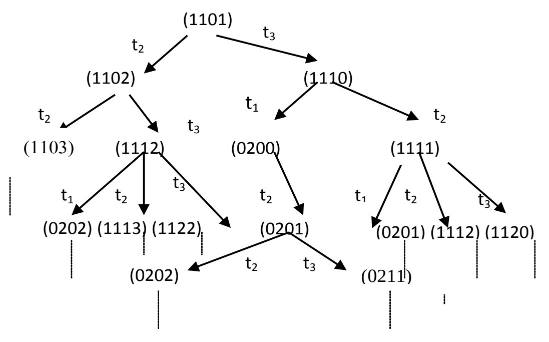
then $\mu[z]_i = \omega$.

If $\mu[x]_k = \omega$, then $\mu[z]_k = \omega$. In the opposite case: $\mu[z]_r = \delta(\mu[x], t_j)_r$.

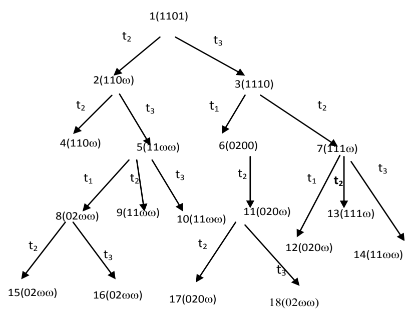
The second step is repeated for t_j .

3. If the number of the peaks are more, then the algorithm finishes its work.

With the help of this algorithm, we will build (as in picture 1.) the Petri Net reachable tree (picture 3.).



Picture 2. The Petri Net Reachable Infinite Tree.



Picture 3. The Reachable Tree of Petri Net.

3. DESCRIPTION OF THE ALGORITHM FOR FINDING THE MINIMUM NUMBER OF TRANSITIONS IN THE STATE OF COVERAGE

Suppose we have Petri Nets in picture 1, and the corresponding TT (picture 3) reachable tree.

Let's mark as P , and the set of states in Petri Nets as T^* from TT root till y, transition succession with, G the succession of the peaks in T^*.

Suppose we have $\mu[x] = (0,1,15,13)$ state. Let's find a y peak in the reachable tree that $\mu[y] \geq \mu[x]$:

Suppose such peaks are $y_1,\ldots,y_m$. Let's choose one peak among the peaks on which we will use the algorithm.

For every y_i peak, we profile $\mu[y_i]$.

Suppose in $\mu[y_i]$ there is ω in $\mu[y_i]_{i1},\ldots,\mu[y_i]_{ik}$. For each $\mu[y_i]$ we count $S = \sum_{j=i_1}^{i_k} z_j(t)$, where

$$z_j(t) = \#(P_j, I(t_k)), \forall t_k \in T^*:$$

We take the y_i for which the S is the minimum. If for any peak, these numbers are equal, then we take the y_i in which T^* height is the minimum.

For our example $\mu[x]$ we will cover the following peak:

$y_1 \quad \mu[y_1] = (1,1,\omega,\omega), \quad T^* = \{t_2, t_3\}$

$y_2 \quad \mu[y_2] = (0,2,\omega,\omega), \quad T^* = \{t_2, t_3, t_1\}$

$y_3 \quad \mu[y_3]=(1,1,\omega,\omega), \quad T^*=\{t_2,t_3,t_2\}$
$y_4 \quad \mu[y_4]=(1,1,\omega,\omega), \quad T^*=\{t_2,t_3,t_3\}$
$y_5 \quad \mu[y_5]=(1,1,\omega,\omega), \quad T^*=\{t_3,t_2,t_3\}$
$y_6 \quad \mu[y_6]=(0,2,\omega,\omega), \quad T^*=\{t_3,t_1,t_2,t_3\}$

$S(y_1)=1$	$S(y_4)=2$
$S(y_2)=2$	$S(y_5)=2$
$S(y_3)=1$	$S(y_6)=3$:

We found out that in minimum number: $S(y_1)=S(y_3), |T^*(y_1)|=2, |T^*(y_3)|=3 \Rightarrow$ we take the y_1 peak. After choosing the covering peak, we go to the usage of the algorithm. Suppose the y is the covering peak.

1. We take the way, which connects the tree root with y and T^* for our example t_2,t_3 let's mark $t_i^{'}=t_j, 1 \le i \le |T^*|$, $t_j \in T^*$. In this case: $t_1^{'}=t_2$, $t_2^{'}=t_3$:

2. For each chosen transitions, the $t_i^{'}$, is corresponded with a_i numbers in the following way:

- If for $t_i^{'}$ transition $\exists \hat{U} 1 \le j \le |P| \hat{U}$ in the way that $\delta(\mu[y^{'}], t_i^{'})_j = \omega$, $y^{'} \in G$ then $a_i=\mu[x]_j$ in which case $t_i^{'}$ transition corresponds with P_j position.
- If for the same $t_i^{'}$ transition $\exists \hat{U} 1 \le k \ne j \le |P| \hat{U}$ in the way that $\delta(\mu[y^{'}], t_i^{'})_k = \omega$, then $a_i=\max\{\mu[x]_j, \mu[x]_k\}$.

Moreover, for $t_i^{'}$ transition we will correspond P_j and P_k positions. If instead of $t_i^{'}$ $\sigma = t_{i_1}^{'} \dots t_{i_k}^{'} (t_{i_k}^{'} = t_i^{'})$ for $\exists \hat{U} 1 \le j \le |P| \hat{U}$ in the way that $\delta(\mu[y^{'}], \sigma)_j = \omega, y^{'} \in G$, then we will correspond a_i with σ and $a_i=\mu(x)_j$:

In this case, we will correspond σ with P_j position. In the opposite case, if there is no $t_i^{'}$ transition for $1 \le j \le |P|$ in the way that $\delta(\mu[y^{'}], t_i^{'})_j = \omega$, then $a_i=1$, in which case there is no related position for $t_i^{'}$.

For our example:

$$a_1 = 13 \qquad a_2 = 15$$
$$t_1^{'} \sim P_4 \qquad t_2^{'} \sim P_3:$$

3. Now, we will define the following action for a_i:

$$a_i = \begin{cases} \dfrac{a_i - n}{m}, & \text{if } (a_i - n)\text{mod} m = 0 \\ \left[\dfrac{a_i - n}{m}\right] + 1, & if (a_i - n)\text{mod} m \ne 0 \end{cases}$$

Where n to $t_i^{'}$ or in σ corresponding P_j position, the number of tokens are in their first position, and m from $t_i^{'}$ or σ to the number of the arrows in the state: $\#(P_j, O(t_i^{'}))$.

If for $t_i^{'}$ transition $P_1,\cdots,P_k$ positions correspond, then we will take the P_1 position for which: $\mu[x]_1 = a_i$. In this case: $n=\mu_0[P_1]$, $m = \#(P_j, O(t_i^{'}))$.

If there is no corresponding position for $t_i^{'}$ transition, then we will leave a_i to remain the same.

For our example:

$$a_1 = (a_1 - 1)/1 = (13-1)/1 = 12$$
$$a_2 = (a_2 - 0)/1 = (15-0)/1 = 15.$$

4. Let's mark $b_i^1 = a_i$. For our example:

$$b_1^1 = a_1 = 12$$
$$b_2^1 = a_2 = 15.$$

5. Cumulative move.
    1. We will take T^* last transition or the succession of transition, fix it and mark as t_α. t_α corresponding b_i^1 is marked as α which we also fix. The fixed b_i^j doesn't change in the next moves.
    2. We consider all T^* items from right to left, starting from t_α.

Suppose the $t_k^{'}$ is the considered transition or the transition succession and the P_1 is the corresponding position of $t_k^{'}$.

If $P_1 \in I(t_\alpha)$, then $t_k^{'}$ corresponding b_i^j in the next move will get the following value: $b_i^{j+1} = b_i^j + \alpha \cdot l$, where l from P_1 position t_α is the number of arrows.

Suppose $t_k^{'}$ corresponds with $P_1,\cdots,P_l$ positions. If $\exists\ 1 \le j \le l$ in the way that $P_j \in I(t_\alpha)$, then, $b_i^{j+1} = b_i^j + \alpha \cdot l$, where $l = \#(P_j, I(t_\alpha))$. In the opposite case $b_i^{j+1} = b_i^j$.

3. Now we will fix t_α the previous action of transition and mark it as t_α.

The new t_α corresponding b_i^j, we will mark as α and move again to the second step. The algorithm will implement its work if T^* fixes its first item.

So, for every $t_i^{'}$ transition or transition succession, there will be a corresponding fixed b_i^j number, which will mark for $t_i^{'}$ transition or transition succession implementation number. For our example:

$t_1^{'}$	$t_2^{'}$
12	$\underline{15}$
$\underline{27}$	$\underline{15}$.

We got that t_1' transition must be implemented for 27 times, and t_2', 15 times.

Returning to our appointment, we will get that $\mu[x]=(0,1,15,13)$ for covering the state $\mu[y]=(1,1,\omega,\omega)$. On the way to reach the state we need to implement t_2 transition 27 times, and the t_3 transition for 15 times.

Let's assign $t_s(y)=\sum_{i=1}^{l} b_i^j$, $l=\left|T^*(y)\right|$. Which means $t_s(y)$ is y the number of enabled transitions.

Lemma 1: Suppose there are y_1 and y_2 peaks in the way that $\mu(y_1)\geq\mu[x]\,\&\,\mu(y_2)\geq\mu[x]$. In that case $t_s(y_1)\leq t_s(y_2)$.

Lemma 2: Suppose y_1 and y_2 are covering peaks. There is $S(y_1)=S(y_2)\,\&\left|T_1^*\right|<\left|T_2^*\right|$: in this case $t_s(y_1)<t_s(y_2)$.

Theorem: Through the above mentioned number of covering state transition algorithm is in its minimal state.

Proof: Suppose y is the covering peak in our algorithm and $t_1',\cdots,t_k'$ is the succession of transitions. We will show that the number of $t_1',\cdots,t_k'$ move is in minimal state. For this reason, we need to show that $\nexists y'$ covering peak has less number of transitions than the number of $t_1',\cdots,t_k'$.

Let's consider two cases:

1. $y\neq y'$.

Suppose the transition number of y' is less than $t_1',\cdots,t_k'$ implementation number. According to the algorithm: $S(y)<S(y')$ or

$$S(y)=S(y')\,\&\left|T_1^*\right|<\left|T_2^*\right|.$$

- If $S(y)<S(y')\Longrightarrow$ according to lemma 1: $t_s(y)\leq t_s(y')$. We've come into a controversy.
- If $S(y)=S(y')\,\&\left|T_1^*\right|<\left|T_2^*\right|\Rightarrow$ according to lemma 2: $t_s(y)\leq t_s(y')$. We've come into a controversy.

2. $y=y'$.

Suppose succession transitions of y' is $s_1,\cdots,s_r$. As the tree doesn't contain cycle:

$y=y'\Longrightarrow t_1',\cdots,t_k'$ and $s_1,\cdots,s_r$ are the same $\Longrightarrow$ $t_s(y)\leq t_s(y')$. The theorem is proved.

4. CONCLUSION:

The above mentioned studies and the proved theorem brings out several important features of Petri Nets in optimization perspective, according which, if Petri Nets are used in technical devices, then the idea of succession transition passages brings resources and saves time.

REFERENCES

[1] J. L. Peterson,
"Petri Net Theory and the Modelling of Systems", Prentice Hall. ISBN 0-13-661983-5, 1981.

[2] T. Murata, "Petri nets: Properties, Analysis and Applications", Proc. of the IEEE, 77(4), 1989.

[3] V. Ye. Kotov, "Petri Nets", Moscow, World 1984.

[4] D. Knut, "The Art of Programming", T1, T2, T3, Moscow, Mir 1976.

[5] S. A. Orlov, "Technology of Software Development", textbook for universities, Petersburg, 2002.

[6] A.V. Gordeev, A. Yu. Molchanov "System Software", textbook, St. Petersburg, 2002.