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 conveyor 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 $\mu$ is the net condition. In structure $C$ of $P$-positions, $T$-transitions are finite sets. $I : T \rightarrow P^*, O : T \rightarrow P^*$ are input and output functions, respectively, where $P^*$ are all possible collections (repetitive elements) of $P$. $\mu : P \rightarrow N_0$ is the function of condition, where $N_0 = \{0, 1, \ldots\}$ is the set of integers.

Saying net state, we will understand the following:

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

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

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

$$\mu(P_j) \geq \#(P_j, I(t_j))$$

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

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

The meaning of coverage problem is for $\mu'$ decide whether it is reachable to $\mu' \geq \mu$. The coverage problem can be solved through the reachable tree. At first for $\mu$ 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 $\mu'$ marking isn’t covered with any reachable marking, if it is located in $\mu[x]$ and gives a reachable marking which covers $\mu'$ in $[4-6]$.

Let’s build Petri Nets reachable tree in picture 1. The natural state of this net is $1(101)$, 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 \( a \cdot b \):
\[ \omega \geq \omega, \quad \omega + a = \omega, \quad \omega - a = \omega, \quad \text{where} \quad a = \text{const} \]
For example, instead of \((5, \ldots),\) we will write \((\omega, \ldots)\). 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.

**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 \( \mu[X] \) 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 \( \forall i \), transition \( i \in T \), so that \( \delta(\mu[X], t_i) \) is solved, then do the following: Add the tree a new side coming from \( X \), and mark the side as \( i \) 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
\[ \mu[y] \leq \delta(\mu[X], t_j) \quad \mu[y] < \delta(\mu[X], t_j), \]
then \( \mu[z_k] = \omega \).

If \( \mu[z_k] = \omega \), then \( \mu[z_k] = \omega \). In the opposite case:
\[ \mu[z_k] = \delta(\mu[X], t_j), \]
The second step is repeated for \( i \).

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 1. An example of Petri Nets.**

**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 \( \mu[X] = (0,115,13) \) 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 \( \omega \) in \( \mu[y_i] \). For each \( \mu[y_i] \) we count
\[ S = \sum_{j=1}^n z_j(t), \]
where
\[ z_j(t) = \#(P_j, I(t)), \forall \forall t \in T^*: \]

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:

\( y_1 \quad \mu[y_1] = (0,115,13) \quad T^* = \{t_2, t_1, t_1\} \)
\( y_2 \quad \mu[y_2] = (0,2,0,0) \quad T^* = \{t_2, t_1, t_1\} \)
\[ y_3 = \mu[y_3] = (1.0, o, o), \quad T^* = \{t_2, t_3, t_2\} \]
\[ y_4 = \mu[y_4] = (1.0, o, o), \quad T^* = \{t_2, t_3, t_3\} \]
\[ y_5 = \mu[y_5] = (1.0, o, o), \quad T^* = \{t_3, t_2, t_3\} \]
\[ y_6 = \mu[y_6] = (0.2, o, o), \quad T^* = \{t_3, t_1, t_2, t_3\} \]
\[ S(y_1) = 1 \quad S(y_3) = 2 \quad S(y_5) = 2 \quad S(y_6) = 3 : \]

We found out that in minimum number: \[ S(y_1) = S(y_3), \quad T^*(y_1) = 2, \quad T^*(y_3) = 3 \Rightarrow \text{we take the y}_1 \text{ peak.} \]
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 \( t^*_i = t_{i.j} \), \( 1 \leq i \leq |T^*|, t_j \in T^* \). In this case: \( t^*_1 = t_2, \quad t^*_2 = t_3 \).

2. For each chosen transition, the \( t^*_i \) is corresponded with \( a_i \) numbers in the following way:
   • If for \( t^*_i \) transition \( \exists \Omega \leq j \leq |H| \) in the way that \( \delta(\mu[y'_{t_1}], y_{t_1}) = o, \quad y \in G \) then \( a_i = \mu[x] \) in which case \( t^*_i \) transition corresponds with \( P_j \) position.
   • If for the same \( t^*_i \) transition \( \exists \Omega \leq k \neq j \leq |H| \) in the way that \( \delta(\mu[y'_{t_1}], y_{t_1}) = o, y \in G \) then \( a_i = \max(\mu[x], \mu[x]) \).

   Moreover, for \( t^*_i \) transition we will correspond \( P_j \) and \( P_k \) positions. If instead of \( t^*_i \) \( \sigma = t_{i_1} \cdots t_{i_k} (i_k = t^*_i) \) for \( \exists \Omega \leq j \leq |H| \) in the way that \( \delta(\mu[y'_{t_1}], y_{t_1}) = o, \quad y \in G \), then we will correspond \( a_i \) with \( \sigma \) and \( a_i = \mu(x) \).

   In this case, we will correspond \( \sigma \) with \( P_j \) position. In the opposite case, if there is no \( t^*_i \) transition for \( 1 \leq j \leq |H| \) in the way that \( \delta(\mu[y'_{t_1}], y_{t_1}) = o \), then \( a_i = 1 \), in which case there is no related position for \( t^*_i \).

   For our example:
   \[ a_1 = 13, \quad a_2 = 15 \]
   \[ t^*_1 \sim P_4, \quad t^*_2 \sim P_3 \]

3. Now, we will define the following action for \( a_i \):
   \[ a_i = \begin{cases} a_i/n, & \text{if } (a_i - n) \text{mod} m = 0 \\ a_i/n + 1, & \text{if } (a_i - n) \text{mod} m \neq 0 \end{cases} \]

   Where \( n \) to \( l_i \) or in \( \sigma \) corresponding \( P_j \) position, the number of tokens are in their first position, and \( m \) from \( l_i \) or \( \sigma \) to the number of the arrows in the state: \( \#(p_i, l_i) \).

   If for \( t^*_i \) transition \( P_1, \cdots, P_k \) positions correspond, then we will take the \( P_i \) position for which: \( \mu[x] = a_i \). In this case: \( n = \mu[p_i], \quad m = \#(p_i, l_i) \).

   If there is no corresponding position for \( t^*_i \) transition, then we will leave \( a_i \) to remain the same.

   For our example:
   \[ a_1 = (a_1 - 1)/|l_1| = (13 - 1)/12 = 1 \]
   \[ a_2 = (a_2 - 0)/|l_2| = (15 - 0)/15 = 1. \]

4. Let’s mark \( b_i^* = a_i \). For our example:
   \[ b_1^* = a_1 = 12 \]
   \[ b_2^* = a_2 = 15 \]

5. Cumulative move.

   1. We will take \( T^* \) last transition or the succession of transition, fix it and mark as \( t_{a.i} \). \( t_{a.i} \) corresponding \( b_i^* \) is marked as \( \alpha \) which we also fix. The fixed \( b_i^* \) doesn’t change in the next moves.

   2. We consider all \( T^* \) items from right to left, starting from \( t_{a.i} \).

   Suppose \( t_{a.i}^* \) is the considered transition or the transition succession and the \( P_i \) is the corresponding position of \( t_{a.i}^* \).

   If \( P_i \in l_{(a.i)} \), then \( t_{a.i}^* \) corresponding \( b_i^* \) in the next move will get the following value: \( b_i^{a.i+1} = b_i^* + \alpha \cdot l \), where \( l \) from \( P_i \) position \( t_{a.i} \) is the number of arrows.

   Suppose \( t_{a.i}^* \) corresponds with \( P_i, \cdots, P_k \) positions. If \( \exists 1 \leq j \leq l \) in the way that \( P_j \in l_{(a.i)} \), then \( b_i^{a.i+1} = b_i^* + \alpha \cdot l \), where \( l = \#(p_j, l_{(a.i)}) \).

   In the opposite case \( b_i^{a.i+1} = b_i^* \).

   3. Now we will fix \( t_{a.i} \) the previous action of transition and mark it as \( t_{a.i} \).

   The new \( t_{a.i} \) corresponding \( b_i^* \), we will mark as \( \alpha \) 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 \( b_i^* \) number, which will mark for \( t^*_i \) transition or transition succession implementation number. For our example:
   \[ t^*_1, \quad t^*_2 \]
   \[ 12, \quad 15, \quad 27, \quad 15. \]
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 \( \mu[y] = (1,1,0,0,0) \). 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 \( t_s(y) = \sum_{i=1}^{|y|} h_i \), \( t = |r^*(y)| \). Which means \( t_s(y) \) is the number of enabled transitions.

**Lemma 1:** Suppose there are \( y_1 \) and \( y_2 \) peaks in the way that \( \mu(y_1) \geq \mu[x] \) and \( \mu(y_2) \geq \mu[x] \). 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 \( S(y_1) = S(y_2) \) and \( |r_1^-| < |r_2^-| \). Then 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 \( I_1, \cdots, I_k \) is the succession of transitions. We will show that the number of \( I_1, \cdots, I_k \) move in minimal state. For this reason, we need to show that \( \exists y' \) covering peak has less number of transitions than the number of \( I_1, \cdots, I_k \).

Let's consider two cases:

1. \( y \neq y' \).

Suppose the transition number of \( y' \) is less than \( I_1, \cdots, I_k \) implementation number. According to the algorithm: \( S(y) < S(y') \) or

\[
S(y) = S(y') \land |r_1^-| < |r_2^-|
\]

- If \( S(y) < S(y') \) according to lemma 1: \( t_s(y) \leq t_s(y') \). We've come into a controversy.
- If \( S(y) = S(y') \) and \( |r_1^-| < |r_2^-| \) 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:

\( y = y' \Rightarrow I_1, \cdots, I_k \) and \( s_1, \cdots, s_r \) are the same \( \Rightarrow t_s(y) \leq t_s(y') \). 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**


