INTERNATIONAL JOURNAL OF COMPUTERS
Issue 2, Volume 5, 2011

Rewriting Petri Nets as Directed Graphs

A. Spiteri Staines

***Abstract*—** This work attempts to understand some of the basic properties of Petri nets and their relationships to directed graphs. Different forms of directed graphs are widely used in computer science. Normally various names are given to these structures. E.g. directed acyclical graphs (DAGs), control flow graphs (CFGs), task graphs, generalized task graphs (GTGs), state transition diagrams (STDs), state machines, etc. Some structures might exhibit bi-similarity. The justification for this work is that Petri nets are based on graphs and have some similarities to them. Transforming Petri nets into graphs opens up a whole set of new interesting possible experimentations. Normally this is overlooked. Directed Graphs have a lot of theory and research associated with them. This work could be further developed and used for Petri net evaluation. The related works justifies the reasoning how and why Petri nets are obtained or supported using graphs. The transformation approach can be formal or informal. The main problem tackled is how graphs can be obtained from Petri nets. Possible solutions that use reduction methods to simplify the Petri net are presented. Different methods to extract graphs from the basic or fundamental Petri net classes are explained. Some examples are given and the findings are briefly discussed.

***Keywords*—** Directed Graphs, Graphs, Petri nets, Transformation, Reduction

I. Introduction

PETRI nets are expressive graphical formalisms that serve to model discrete event behavior that takes place in different systems [12]-[15]. They are designed to model system behavior like: sequential behavior, concurrency, mutual exclusion, non-determinism, choice and conflict. Petri nets are classified into different classes ranging from elementary nets to higher order nets, colored Petri nets and object oriented nets. All these classes can be converted to time Petri nets.

Ordinary Petri nets have a 'dual identity' they can be represented graphically or by using equations. These can be analyzed using mathematical models. Petri nets have at least three decades of use. Normally speaking, the analysis of Petri nets is based on i) structural properties and ii) behavioral properties [6]. The structural properties of Petri nets are suitable to understand the basic underlying structure. If the Petri net is viewed, basic structural features can be seen. E.g. the Petri net can be cycle free (acyclical) [9]. It could have bounded places, etc. On the other hand behavioral properties explain the behavior of the Petri net. These properties cannot be applied to all types of Petri nets especially if the net is unbounded or improperly designed. Some basic behavioral properties are i) reachability, ii) boundedness, iii) safeness, iv) conservativeness, v) liveness, vi) reversiblility, vii) repetitiveness, viii) home states. Sometimes these properties are also called structural properties by some authors [6], [7]. Other properties like persistence and synchronic distance could be included. From these main properties others like partially conservative, structurally bounded, partially repetitive, etc. can be defined. These properties can be found using reachability methods such as the marking graph or place and transition invariants or the analysis of the Petri net incidence matrix [6],[7]. Other analysis is based on the siphon and trap method. Most of these forms of analysis are applicable to structurally bounded Petri net structures with the exception of reachability which can be solved to provide for some unbounded states. Simulation is another method by which the Petri net can be tested and functionally verified. This should normally be done after the behavioral properties have been checked and verified. In general for the purpose of analysis, Petri net structures can be classified into two categories: i) unsolvable and ii) solvable. The structurally limited Petri nets are normally solvable whilst those that are not structurally limited and have state space explosion problems are not simple to solve. One possible solution is to reduce the structure.

One of the salient points for using Petri nets is precisely the ability to transform them or obtain them from other formalisms or notations. Petri nets are classified as directed bi-partite graphs, definitely sharing some common properties with graphs. This means that they could be transformed into graphs and analyzed from this point of view. This could serve to generate many new ideas. E.g. the static structure or topological features are examinable. Structure is easier to control and understand than behavior. This is because normally the structure should remain fixed in relation to time, whist behavior can be modified or applied differently, being dynamic. Another important aspect is the reduction of the Petri net model. Even though normally reduction implies fusion/augmentation of places, transitions etc.; in the wider sense a higher order net can be reduced to a simple place transition net by keeping the graphical outline structure and removing other information.

The work in this paper is restricted to the basic or fundamental classes of Petri nets.

II. Background

A normal Petri net is basically defined as directed bipartite graph or bipartite digraph that can be basically represented as

Manuscript received Feb 23, 2011. Anthony (Tony) Spiteri Staines, is with the Department of Information Systems, Faculty of ICT, University of Malta, (corresponding phone: 00356-21373402,e-mail: toni_staines@yahoo.com)

a five tuple (P, T, I,O,M_0) where P is a finite set of places, T is a finite set of transitions, I $\subseteq$ (P x T) Input arcs, and O $\subseteq$ (T x P) Output arcs, P $\cup$ T $\neq \phi$ and P $\cap$ T = ϕ , M_0 represents the initial marking.

Normal Petri nets are very simple and convenient to use for a variety of purposes. Similar to them are elementary nets and augmented marked graphs which are a special subclass of Petri nets. Normal Petri nets have a reduced or limited state space.

There are problems to find simple ways for understanding and analyzing Petri nets. Another aspect is that certain Petri net models that are created are just too complex to analyze and verify using the traditional approaches. Other fundamental properties of Petri nets are normally not applicable to certain classes of Petri nets like higher order nets. An interesting idea, that is often overlooked, is to analyze the structural properties of Petri nets by representing the Petri net as a graph. The graph although static can be used for different objectives such as visual inspection, etc.

Some works are listed below. These just confirm the importance of Petri nets for supporting other forms of graphical structures and the possible transformation or support of Petri nets using graphs. A vast amount of literature is available in this respect.

Obviously the transformation or mapping is done informally or formally or it just happens. In [6] it is shown how a Petri net having exactly one incoming arc and exactly one outgoing arc with unit weight can be directly represented as directed marked graph where directed edges correspond to places and nodes to transitions. The augmented marked graphs presented in [4] seem to share similar properties to this. The same Petri net can also be represented as a state machine. Augmented Marked Graphs are a special sub-class of Petri nets [4]. These are structurally bounded Petri nets that preserve certain properties. In [10] Petri nets are obtained from transfer resource graphs (TRGs). Here Petri nets are used to model a system at a higher level of abstraction. In [1] Petri net elements are defined as TGG rules from project or object elements. This is a form of formal mapping. In [2] it is proposed to use controlled time Petri nets (CtlTPNs) for RT systems dynamic modeling. Control class graphs (CCGs) are defined to explain CtlTPNs. Systolic Petri net structures [3] are based on restricted Petri net structures. These structures, whilst sharing some similarities to graphs, evidence the use of this type of transformation.

Again, different classes of Petri nets can possibly be intuitively bi similar. It is possible to use a symbolic reachability graph for cases where the reachability graph cannot be normally generated. Other works are the generation of Petri nets from UML diagrams or vice-versa. UML diagrams are graphical structures [8].

III. MOTIVATION

The evidence presented shows that Petri nets are strongly associated with graphs and graph theory. Most of the work about Petri nets does not usually consider them from the graph point of view. This opens up a lot of new exciting possibilities for analyzing Petri nets. If Petri nets are transformed into graphs, then they can be analysed using graph theory. The graphs can also serve for visualization. It can be shown that some notations like control flow graphs (CFGs), state transition diagrams (STDs), etc. can be obtained directly from certain Petri net types.

IV. PROBLEM FORMULATION

The main problem that is dealt with in this paper is to try to examine how Petri nets can be converted into graphs for the purpose of analysis. It is possible to transform Petri nets into graphs. There are different ways how to obtain graphs from Petri nets. To obtain graphs from the Petri net the Petri net should have a reduced structure and has to be bounded. A possible solution it to reduce the class and structure of the Petri net before applying analysis methods and transformation of the Petri net structure into a graph. Sometimes there can be various issues, especially if the Petri net is too complex. It has to be reduced. I.e. it can be reduced structurally to a simpler model or class reduction could be performed. E.g. a more complex class can be reduced to a lower class by replacing or eliminating some properties or information.

V. PROBLEM SOLUTION

There are two aspects of the solution i.e. i) reduction of the Petri net and ii) explaining the possible transformations that can be done. Reduction might imply i) class reduction and ii) structural reduction.

Once the Petri net is reduced it is possible to transform the Petri net to a graph by simply replacing the nodes and edges in the Petri net.

Another simple way of obtaining a graph from a Petri net is by generating the marking graph or the reachability graph. Other possible methods could include replacing the Petri net node and edges.

A. Reducing the Petri net

Two possibilities are given. I.e. i) class reduction and ii) structural reduction. As previously stated, the best Petri nets used to obtain graphs have to be structurally reduced and limited.

B. Class Reduction

Class reduction can be defined as transformation of a higher order Petri net or net into a simpler net or a more basic class type. There are various ways how this transformation can be carried out.

Normally class reduction necessitates the loss of information [11]. The resulting Petri net is simplified and it is more comprehensible and simple for transforming it into a directed graph. The basic structural features of the Petri net should still be retained. This might not always be the case.

C. Basic Structural Reduction

According to well known Petri net theory [5]-[7] it is possible to classify five or six main rules for Petri net reduction whilst retaining the main properties. Basically a subnet or structures of the Petri net are reduced or simplified. These basic reduction rules obtained in part from [5] are shown in fig. 1-5. The rules in fig. 1-5 are applicable to various Petri net classes. Normally applying them results in the loss of information. One issue is that these rules are applicable to basic constructs. If there are advanced constructs like inhibitor arcs, test arcs, conflict, choice, etc. reduction is not so simple.

Other more complex rules can be specified if required. It is possible to use ideas from decomposition with Petri nets. I.e. a place can contain an entire subnet at a lower level.

i) Serial Place Fusion/Reduction:
This rule refers to combining two sequential or serial places into one single place. Normally the given pattern is that of a place followed by an output arc to a transition. An outgoing arc connects to another place. This rule is described as follows:

$$\exists p_1, p_2 \in P,\ out(p_1) = in(p_2) \cap (|in(p_2)| = 1)$$

then replace the places and shared output/input transition with one single place.

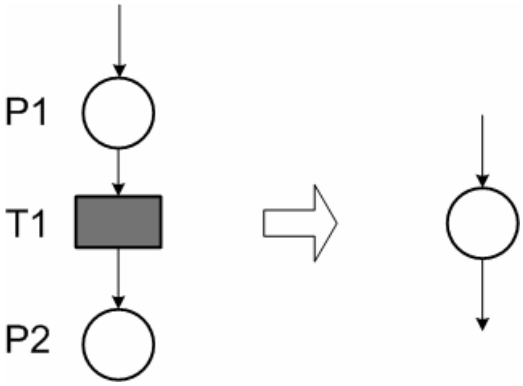


Fig. 1 serial place reduction

ii) Serial Transition Fusion/Reduction:
This rule is similar to the Serial Place Fusion/Reduction but in this case serial transitions are considered. Here two sequential transitions are combined into a single transition. Normally the given pattern is that of a transition connecting to a place which connects to another transition in series. This rule is described as follows:

$$\exists t_1, t_2 \in T,\ out(t_1) = in(t_2) \cap (\ |out(t_1)| = 1\ ) \cap (\ |in(t_2)| = 1\ )$$

then replace the transitions and shared output/input place with one single transition.

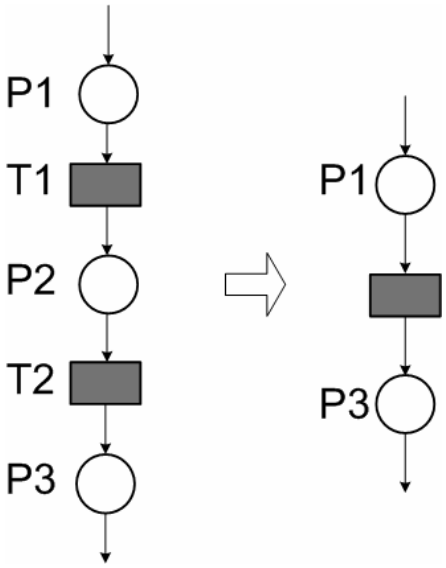


Fig. 2 serial transition reduction

iii) Parallel Place Fusion/Reduction:
This rule is applied to two places which connect to the same output transition and the same input transition. I.e. they are parallel places. This rule can be extended to more than two places which might be in parallel. This rule is described as follows:

$$\exists t_1, t_2 \subseteq T, in(t_1) = out(t_2) \cap \forall p \in in(t_1), in(p) = |t2| \cap out(p) = |t_1|$$

then transform the subgraph by replacing all bounded parallel places with a single place which is the output place of the first transition and input place of the second transition.

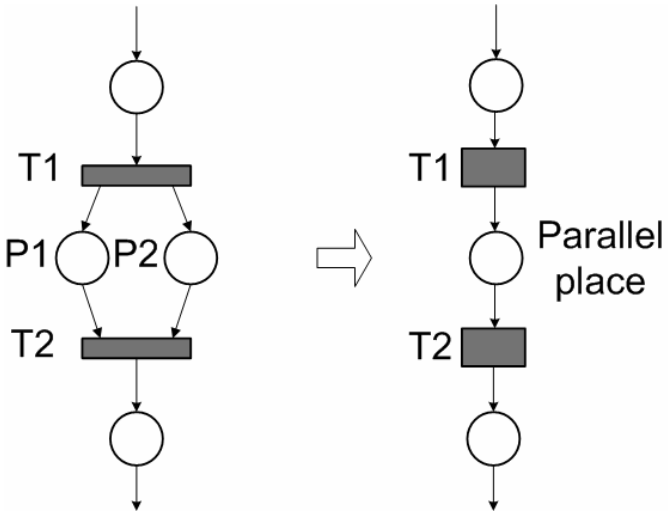


Fig. 3 parallel place reduction

iv) Parallel Transition Fusion/Reduction:
The idea behind this rule is similar to iii) Parallel Place Fusion/Reduction. Here instead of parallel places, parallel transitions are considered. Parallel transitions normally connect to the same input and output places. This rule can similarly be extended to more than two transitions in parallel.

This rule is described as follows:
$\exists p_1, p_2 \subseteq P, \forall a,b \in p_1, out(a) = out(b) \cap \forall a,b \in p_2, in(a) = in(b)$
$\cap \exists a \in p_1, b \in p_2, out(a) = in(b)$ then transform the subgraph by replacing all transitions bounded by sets of places with a single transition having one input/output arc from every place connected to the original transitions.

v) Self Loop Place Removal
This rule is quite simple and refers to reducing the Petri net by removing a place that is looped onto the same transition. This place does not really affect the marking graph, so it can be removed along with its connections. This rule is described as follows.
$\exists p_1 \in P, out(p_1) = in(p_1)$ remove place with self-loop connections.

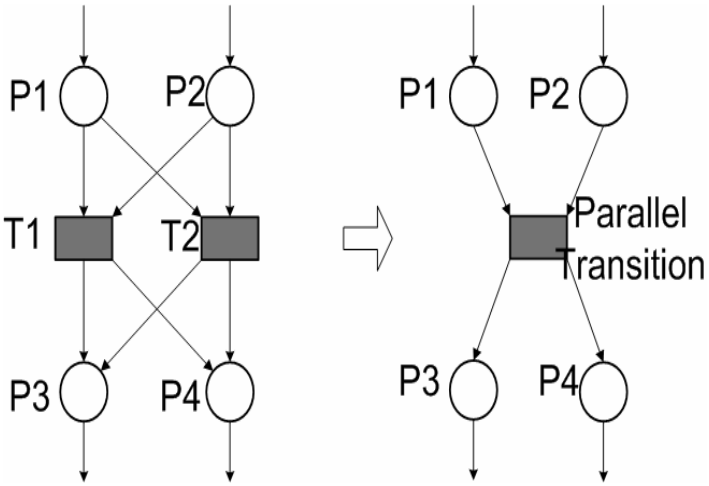


Fig. 4 parallel place reduction

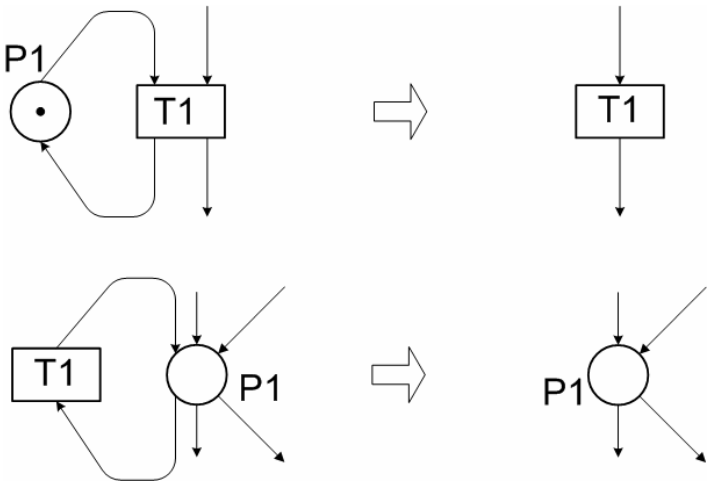


Fig. 5 removal of self loops

vi) Self Loop Transition Removal
This rule is similar to v) Self Loop Place Removal. The difference is that here we have a transition looped onto a place. This transition does not really affect the marking of the Petri net. I.e. the marking is left unchanged if this transition fires, even though the transition can be shown in the marking graph. This transition can be removed. This rule is described as follows:
$\exists t_1 \in T, out(t_1) = in(t_2)$ remove transition with self-loop connections.

D. Other Forms of Reduction

Other forms of reduction could be extraction or removal of subnets from the Petri net, removing loops, cycles, conflict or choice, etc. Extraction of the main subnet could make sense if this represents a system's main functionality. However these reductions will not necessarily preserve the main properties of the net.

VI. Conversion to Directed Graphs

Four different ways to convert a Petri net to a directed graph are listed and explained below.

A. Transitions as nodes. Places, Input/Output Arcs as Edges

Here the Petri net is converted into a graph and the transitions are replaced using nodes, whilst the places are their connecting input and output arcs are replaced with a single graph edge. The transformation of places into nodes and connecting transitions to edges works well only if places have single input points and single output (exit) points. For this type of conversion, it must be clearly explained that every place in the Petri net should have exactly one input arc and one output arc. If these conditions do not exist it is not possible to create a proper directed graph.

B. Places as nodes. Transitions, Input/Output Arcs as Edges

Here the Petri net is converted into a graph. The transitions and their connecting arcs are replaced using a single edge, places are replaced with nodes. For this type of conversion it must be clearly stated that a transition must have exactly one input arc and one output arc. If these conditions do not exist it is not possible to create a proper directed graph.

The transformation of places into nodes and connecting transitions to edges works well only if places have single input points and single output (exit) points. Thus augmented marked graph Petri nets (AMG) or state-machine like Petri nets, etc. can easily be converted. Other more complex Petri net classes have to be reduced for conversion to be possible. Reduction methods presented should simplify the structure of the net.

C. Places and Transitions as Nodes , Input/Output Arcs as Edges

In this approach, when converting the Petri net the input and output arcs are replaced as graph edges and the places and

transitions are replaced as nodes. It can be important to properly label the nodes.

I.e. in this case, there is not really any complex structural change in the Petri net. It is just that the places and transitions are represented as similar nodes. This implies that there is not any real change to the Petri net, it is only the structural representational notations of the net that have been modified.

D. Marking Graph

If the Petri net is structurally limited and its state space is limited (i.e its markings are finite) it is possible to construct the marking graph for the net [6],[7]. Different algorithms can be used to construct the marking graph. In simple terms the marking graph represents all the possible states of the Petri net. The reachability tree or better known coverability tree for a restricted Petri net is easily constructed.

The directed graph obtained from the Petri net can be used for different forms of analysis, which is often overlooked. This type of graph can become quite large if the Petri net has over one hundred states. A possible solution is to reduce the Petri net model and eliminate ambiguity.

The marking graph or reachability tree is a simple directed graph or digraph where the nodes or vertices represent a marking whilst the directed edges represent the transitions used to reach a particular marking. The reachability tree can be drawn as a marked directed graph of the form G=(V,E), where E = edges representing transitions and V= States or markings. There are various forms of the marking graph having different names but in essence they are similar. The delay of a transition can be represented on the edges.

For the marking graph an adjacency matrix can be constructed.

VII. EXAMPLES

Some simple examples illustrating the four conversion methods are given. A specific Petri net is taken and a corresponding directed graph is constructed using each one of the four methods.

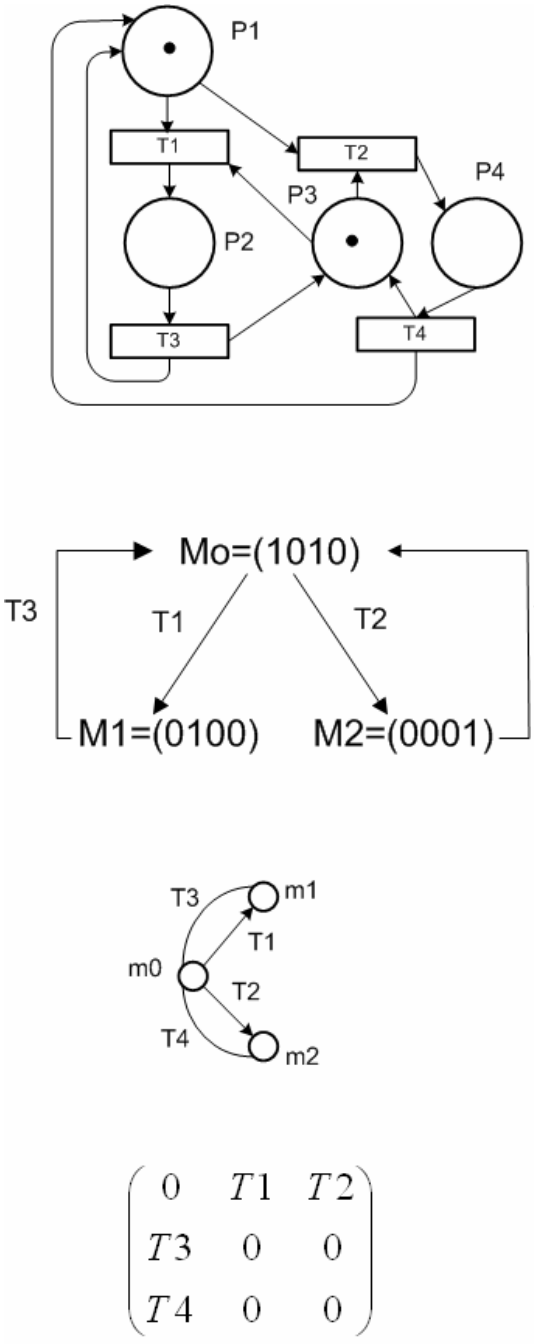
It can be assumed that the structural reduction rules previously defined, have been applied as required. These are quite simple to comprehend and are self explanatory. Note that the resulting graph can obviously be drawn as required, i.e. the node or edge layout could be drawn aesthetically in different ways for visualization e.g. using rounded or flat edges, circles for nodes, etc.

Fig. 6 shows a Petri net, complete with its marking graph below and a compacted or reduced form of marking graph. Below the reduced marking graph the adjacency matrix for the marking graph has been given. For the marking graph in fig. 6 the adjacency matrix is easily constructed. It is assumed that edges represent transitions. The adjacency matrix is shown in fig 14.

Fig. 7 and 8 show two examples where transitions are treated as nodes and places and input/output arcs are treated as edges.

Fig. 9 shows an example where places are treated as nodes and transitions and their connecting input and output arcs are treated as edges.

Fig. 10 shows an example where places and transitions are shown as nodes and input/output arcs are treated as edges. This is not much different from a typical Petri net except for representation of nodes.



$$\begin{pmatrix} 0 & T1 & T2 \\ T3 & 0 & 0 \\ T4 & 0 & 0 \end{pmatrix}$$

Fig. 6 Petri net, marking graph and simplified/concise marking graph and adjacency matrix

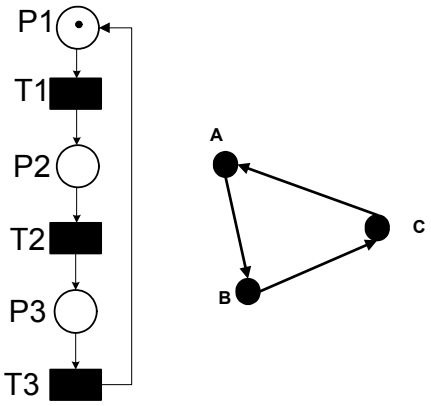


Fig. 7 conversion example 1

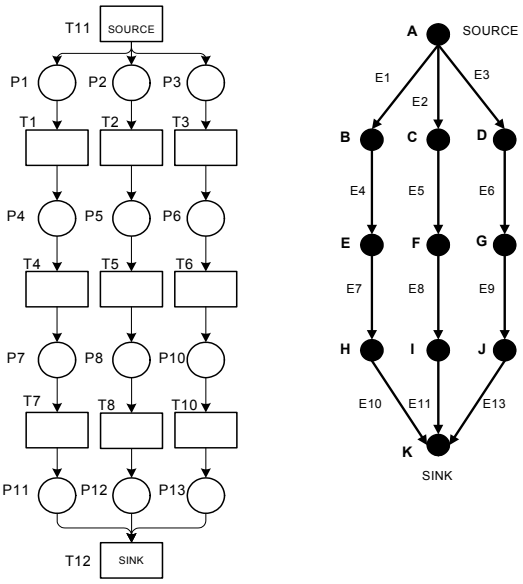


fig. 8 conversion example 2

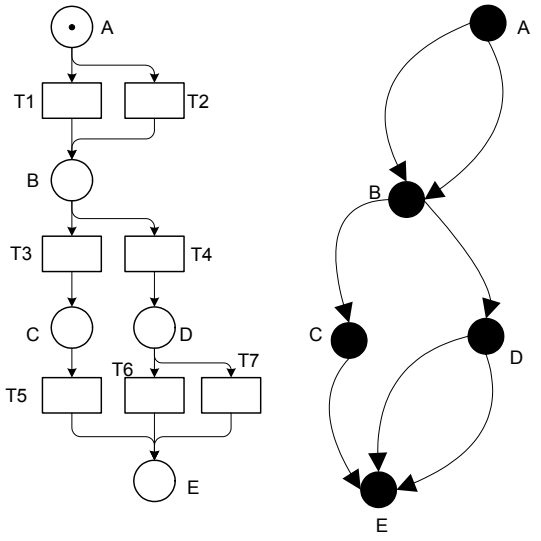


fig. 9 conversion example 3

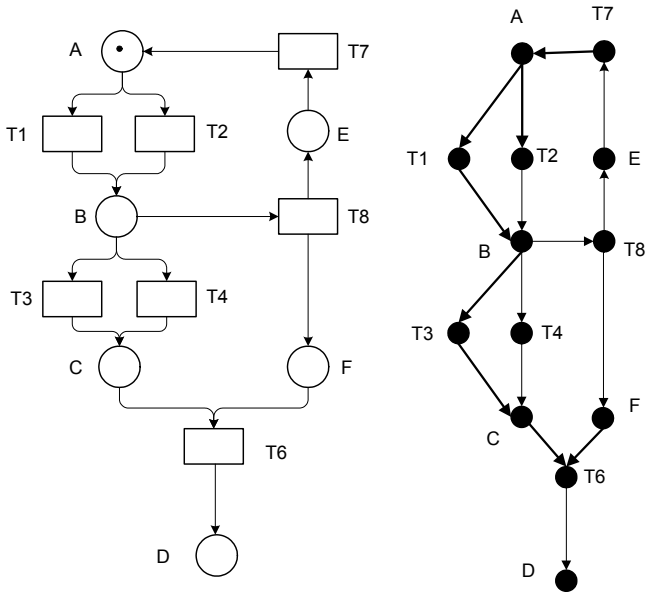


Fig. 10 conversion example 4

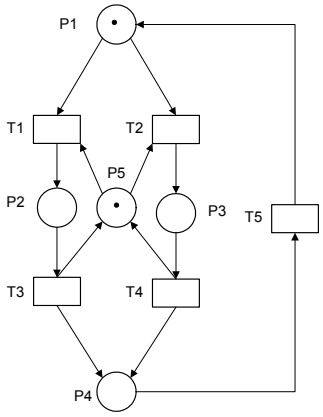


Fig. 11 Petri net example

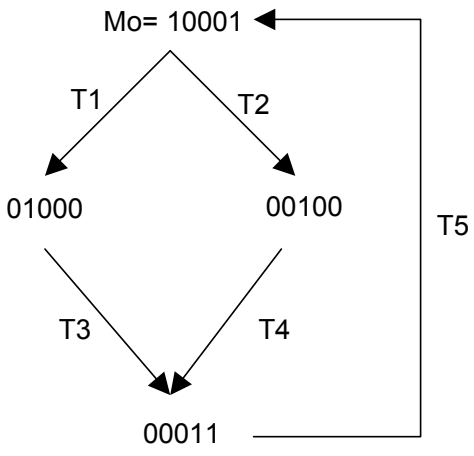


Fig. 12 marking graph for fig.10

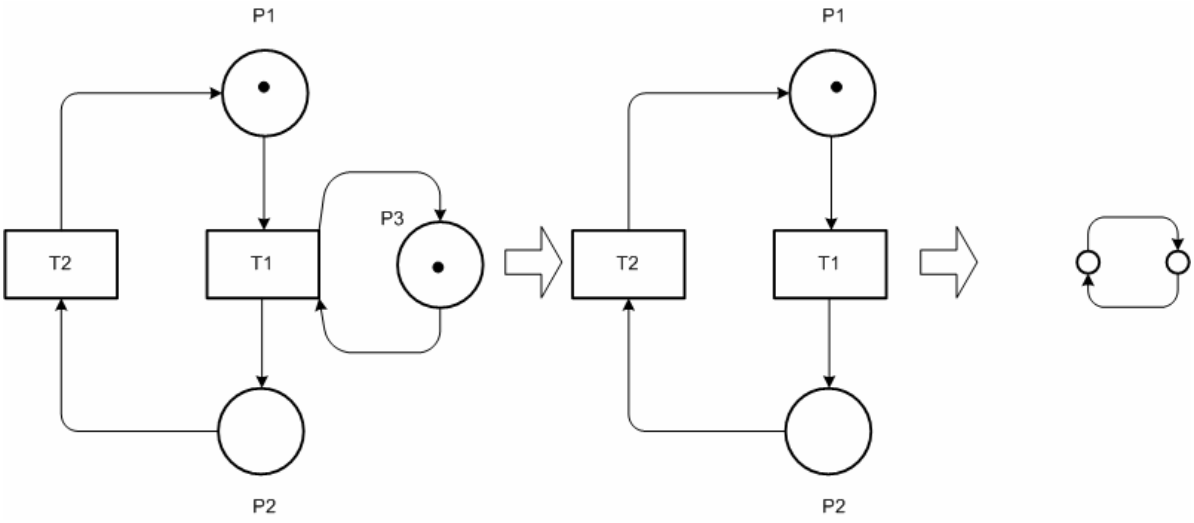


Fig. 13 reducing a Petri net, self loop place removal and generating its directed graph

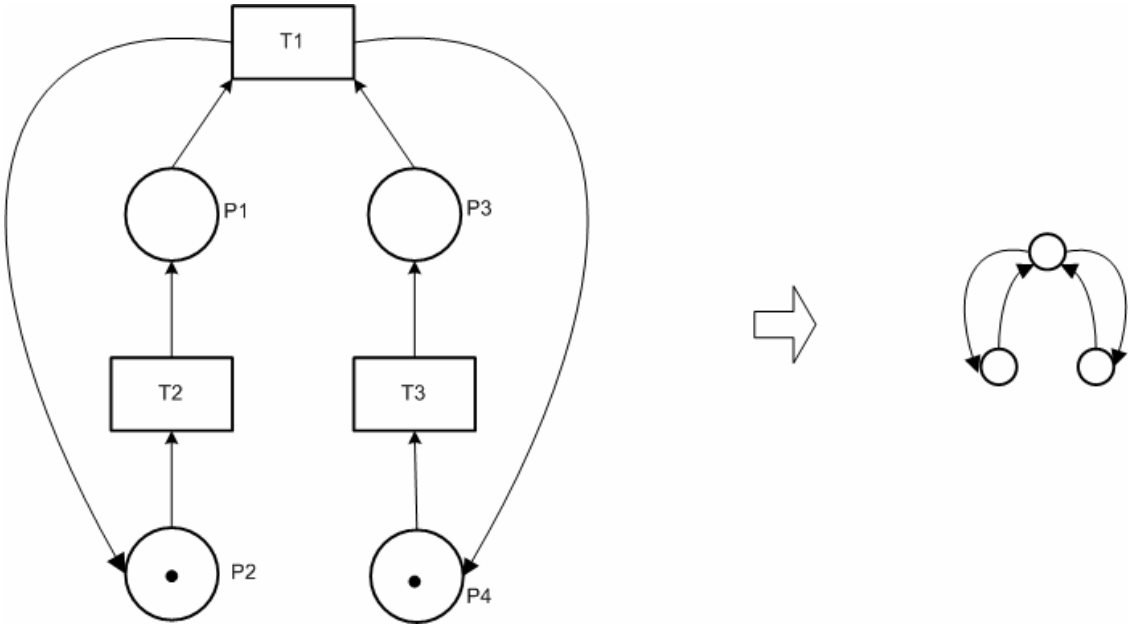


Fig. 14 transforming a Petri net into a directed graph

Fig. 11 shows a normal Petri net that is used for obtaining its marking graph. This Petri net contains a loop and is quite simple to obtain a small marking graph.

Fig. 12 shows the marking graph for fig. 11.

Fig. 13 shows a Petri net having a self loop place, the reduced version of this net and finally the directed graph obtained as follows. Transitions are replaced with nodes. Places, input/output arcs as replaced with edges. This shown on the right hand side of the diagram.

Fig. 14 shows another example where a Petri net is transformed into a directed graph. The method of replacing transitions with nodes and places, input/output connections as edges is used.

VIII. Results

For the marking graph in fig. 6 the adjacency matrix can be constructed. It is assumed that edges represent transitions. The adjacency matrix is shown in fig 6. The given directed graph can be analyzed using the usual graph analysis techniques.

Fig. 7 and 8 show that the graphs are reduced versions of the actual Petri net and are actually simpler.

The graph for the transformation (see fig. 9), where places are represented as nodes and transitions with connecting input/output arcs as represented edges, can be classified as a multi digraph. I.e. it is a directed graph that has multiple edges/arcs having the same source and target nodes. This is because some places are shared by more than one transition. If places are connected to single transitions the resulting graph will be different. Actually this graph is quite interesting because it represents the possible individual states of the system. The resultant graph in fig. 9 is similar to a state transition diagram, where nodes represent states and the edges represent transitions. Again the graph is actually simpler than the Petri net.

Fig. 10 shows a graph which is similar to the Petri net. This approach can be used for Petri nets that are structurally more complex.

Fig. 11 shows a normal Petri net and Fig. 12 shows its marking graph. The marking graph is a directed graph. The marking graph can be considered to be a directed graph of the possible system states when transitions or events take place.

Fig. 13 shows a Petri net having a self loop place. The reduced version of this net is more compact than the original one. Obviously reducing the net, implies that some information is lost. The final directed graph is quite simple having just two nodes and edges.

Fig. 14 again confirms that using the method where transitions are replaced as nodes gives a more compact structure.

The graphs can be further transformed, interpreted using basic graph theory concepts like loops, cycles, etc. It may be possible to construct adjacency matrices for the graphs. The graphs could be used for visualization purposes. In short, from the graphs it is possible to get many new interpretations, analysis and ideas.

IX. Conclusion and Further Work

The limitations of this work are that here only basic or simple Petri nets have been considered for conversion into graphs. In reality only Petri nets that have a limited number of states or limited in structure can be easily converted.

The Petri nets have to be structurally limited or bounded to make them convertible. Reducing complex Petri nets can be carried out, but information and detail will be lost. The marking graph option has less restrictions for the conversion process. Unfortunately the bigger the Petri net the more possible states which can lead to a state explosion problem. It is possible to postulate various theorems in relation to the directed or marked graph and even a diluted circuit. The marking graph can be used to check for invariants, cyclical behavior, safeness, connectedness, etc.

The approach presented here shows only one way of looking at the Petri nets. The approach is applicable to certain classes of Petri nets like elementary nets, cause event nets (CE nets), state machine nets, augmented marked graph Petri nets and others, where the net has limited structure and deterministic behavior. It is possible to find that some structures are isomorphic to other structures derived from different conversion methods of the Petri net to directed graphs.

If we relax our deterministic or net behavioral properties, then this conversion process can be extended to other general classes of Petri nets.

The approach presented here can be inverted. E.g. instead of transforming a Petri net into a directed graph, a given directed graph is transformed into a Petri net.

This work opens up the possibility to find other ways to describe and analyze Petri net structures. This is from a graphical perspective. The rules for reducing the Petri net structures can be useful for reducing complex Petri nets a priori to obtaining graph structures from them. Other forms of graphical representation can be used.

References

[1] E. Kindler, R. Wagner, Triple Graph Grammers: Concepts, Extensions, Implementations and Application Scenarios, Technical Report TR-RI-284, University of Paderborn, Paderborn, 2007, Available: http://www2.cs.uni-paderborn.de/cs/ag-schaefer/Veroeffentlichungen/Quellen/Papers/2007/tr-ri-07-284.pdf

[2] A. Sathaye, B. Krogh, "Supervisor Synthesis for Real-Time Discrete Event Systems", Discrete Event Dynamic Systems, vol. 8 , issue 1, *Springer*, 1998, pp. 5 – 35.

[3] A. Abellard, P. Abellard, *Systolic Petri Nets*, *Petri Nets Applications*, Pawel Pawlewski (Ed.), ISBN: 978-953-307-047-6, INTECH(2010), http://www.intechopen.com/articles/show/title/systolic-petri-nets

[4] K.S. Cheung, K.O. Chow, "Compositional Synthesis of Augmented Marked Graphs", Control and Automation, *ICCA, IEEE*, 2007, pp. 2810-2814.

[5] M.B. Dwyer, L.A. Clarke, "A compact Petri net representation and its implications for analysis", *IEEE Transactions on Software Engineering*, vol. 22, issue 11, 1996, pp. 794 – 811.

[6] T. Murata, "Petri nets: Properties, analysis and applications", *Proc. of IEEE*, vol.: 77, issue:4, 1989, pp. 541-580.

[7] M. Zhou, K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems A Petri Net Approach*, Series in Intelligent Control and Intelligent Automation vol. 6 World Scientific, MA USA, 1999.

[8] T. Gehrke, U.Goltz, H. Wehrheim, The Dynamic Models of UML: Towards a Semantics and its Application in the Development Process, TR. 11/98, University of Hildesheim, Germany, 1998.

[9] G. Stemersch, R.K. Boel, "Structuring acyclical Petri Nets for Reachability Analysis and Control", *International Journal of Intelligent Control and Systems*, vol. 10, no. 2, 2005, pp. 175-187.

[10] X. Xiaoxi, L. Cheng-Chew Lim, *Transfer-Resource Graph and Petri-net for System-on-Chip Verification, Petri Nets Applications*, Pawel Pawlewski (Ed.), ISBN: 978-953-307-047-6, InTech, Available from: http://www.intechopen.com/articles/show/title/transfer-resource-graph-and-petri-net-for-system-on-chip-verification

[11] A. Spiteri Staines, "Supporting Requirements Engineering with Different Petri Net Classes",*International Journal of Computers*, *NAUN*, issue 4 vol 4, 2010, pp. 215-222.

[12] V.Vlad, C.Ciufudean, A.Graur, C.Filote, "An example of modeling manufacturing systems using Petri nets and the IEC 61499 standard", *13th WSEAS Int. Conf. on Systems*, Greece, 2009, pp. 357-363.

[13] H.A. Ozkan, A. Aybar, "A Reversibility Enforcement Approach for Petri Nets Using Invariants, *WSEAS Transactions on Systems*, vol. 7, issue 6, 2008, pp.672-681.

[14] K. Mun Ng, Z. Alam Haron, "Visual Microcontroller Programming Using Extended S-System Petri Nets", *WSEAS Transactions on Computers*, issue 6, vol. 9, 2010, pp. 573-582.

[15] P. Strbac, M. Tuba, D. Simian, "Hierarchical model of a systolic array for solving differential equations implemented as an upgraded Petri net", *WSEAS Transactions on Systems*, vol. 8, issue 1, 2009, pp. 13-21.