Representing Petri Net Structures as Directed Graphs

Anthony Spiteri Staines
Department of Computer Information Systems
University of Malta
MSIDA MSD 2080
Malta
toni_staines@yahoo.com

Abstract: - This paper attempts to explain some basic properties about Petri nets and their relation to directed graphs that can be formal or informal. The related works justifies the reasoning how and why Petri nets are obtained or supported using graphs. 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.

Key-Words: - Petri nets, directed graphs, transformation, reduction

1 Introduction
Petri nets are expressive graphical formalisms that serve to model discrete event behavior that takes place in different systems. 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) reversibility, 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. As Petri nets are classified as directed bi-partite graph, they definitely share 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, whilst 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.

2 Related Works and Motivation

A normal Petri net is basically defined as directed bipartite graph or bipartite digraph that can be basically represented as a five tuple \((P, T, I, O, M_0)\) where \(P\) is a finite set of places, \(T\) is a finite set of transitions, \(I \subseteq (P \times T)\) Input arcs, and \(O \subseteq (T \times P)\) Output arcs, \(P \cup T \neq \phi \) and \(P \cap T = \phi \), \(M_0\) 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 presented. 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].

3 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. There are issues sometimes 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.

4 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.

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

4.1 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.

### 4.1.1 Class Reduction

Class reduction implies transforming or simplifying the net into a more basic class type. This normally 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. Normally the basic structural features of the Petri net should be retained.

### 4.1.2 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 reduction rules obtained in part from [5] are shown in fig. 1-5.

1) Serial Place Fusion/Reduction:

\[ \exists p_1, p_2 \in P, \text{out}(p_1) = \text{in}(p_2) \cap (|\text{in}(p_2)| = 1) \]

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

2) Serial Transition Fusion/Reduction

\[ \exists t_1, t_2 \in T, \text{out}(t_1) = \text{in}(t_2) \cap (|\text{out}(t_1)| = 1 \) \cap (|\text{in}(t_2)| = 1 ) \]

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

3) Parallel Place Fusion/Reduction

\[ \exists t_1, t_2 \subseteq T, \text{in}(t_1) = \text{out}(t_2) \cap \forall p \in \text{in}(t_1), \text{in}(p) = |\text{in}(p)| \cap |\text{out}(p)| = |\]

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.

4) Parallel Transition Fusion/Reduction

\[ \exists p_1, p_2 \subseteq P, \forall a, b \in p_1, \text{out}(a) = \text{out}(b) \cap \forall a, b \in p_2, \text{in}(a) = \text{in}(b) \]

\[ \cap \exists a \in p_1, b \in p_2, \text{out}(a) = \text{in}(b) \]

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.

5) Self Loop Place Removal

\[ \exists p_i \in P, \text{out}(p_i) = \text{in}(p_i) \]

remove place with self-loop connections.

6) Self Loop Transition Removal

\[ \exists t_i \in T, \text{out}(t_i) = \text{in}(t_i) \]

remove transition with self-loop connections.

### 4.1.3 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.

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

4.2.1 Transitions as nodes. Places and 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. 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.

4.2.2 Places as nodes. Transitions and 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.

4.2.3 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.

5 Some Examples
A simple example of the four conversion methods is given using a specific Petri net and a corresponding directed graph. 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 and 7 show two examples where transitions are treated as nodes and places and input/output arcs are treated as edges.

Fig. 8 shows an example where places are treated as nodes and transitions and their connecting input and output arcs are treated as edges.

Fig. 9 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.

Fig. 10 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. 11 shows the marking graph for fig. 10.
Recent Researches in Software Engineering, Parallel and Distributed Systems

Fig. 6 Conversion Example 1

Fig. 7 Conversion Example 2

Fig. 8 Conversion Example 3

Fig. 9 Conversion Example 4

Fig. 10 Petri Net Example

Fig. 11 Marking Graph for fig.10
6 Results

Fig. 6 and 7 show that the graphs are reduced versions of the actual Petri net and are actually simpler.

The graph for the transformation (see fig. 8), 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. The resultant graph in fig. 8 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. 9 shows a graph which is similar to the Petri net. This approach can be used for Petri nets that are structurally more complex.

Fig. 10 shows a normal Petri net and Fig. 11 shows its marking graph. The marking graph is a directed graph.

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

7 Conclusion

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 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 conversion. Unfortunately the bigger the Petri net the more possible states which can lead to a state explosion problem.

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: