An Introduction to Bi-Directional Transition Network Modeling

Anthony (tony) Spiteri Staines
Department of Computer Information Systems
University of Malta
Msida MSD 2080
MALTA
toni_staines@yahoo.com

Abstract: Ordinary Petri nets are classifiable as forward transition systems. This implies that by default, once a transition has fired it cannot be reversed. System representation and modeling is an interesting area used for modeling modern computer systems. Because of changes in technology systems contain diverse forms of behavior. In requirements engineering and requirements elicitation, system modelers can benefit from new approaches and new forms of system representation that build upon previous work. This work presents a new approach and view where it is possible to actually reverse a transition that has taken place. This idea is presented in a Bi-Directional or Bi-Directed transition net. The bi-directional net introduces the concept of reversing the normal firing order. In normal Petri nets once the firing occurs this is irreversible. In the bi-directional net it is possible to reverse the transition. In this work the motivation and ideas behind the Bi-Directional transition are explained and compared with normal Petri Nets. Some toy examples are included to support the ideas of the bi-directional net. From the findings and examples it is clearly indicated that the Bi-Directional Net can be used to create models that can invert the transition firing order and reverse their behavior. Even though the Bi-Directional Net might look simpler in reality it is more complex. The results discuss some of the main properties and issues behind the bi-directional net. This paper is divided into the following sections: i) introduction to the area, ii) related work about transition systems and Petri Nets, iii) motivation and problem definition, iv) proposed solution, v) implementation, vi) examples, vii) results and findings and viii) conclusions.

Key-Words: Bi-directionality, Symbolic Representation, Systems Modeling, Network Representation, Petri Nets, Transition Systems

1 Introduction

System representation is an interesting and intriguing area that requires considerable techniques and modeling disciplines. In the last decades substantial work and innovative changes have taken place in technologies and architectures of computer systems and their respective networking. Computer technologies have evolved and developed many new processes and formal or informal systems. Modern systems are large and complicated [1]-[6]. They contain many different forms of behavior. It is obvious that systems require to be modelled using new tools. E.g. a large system can be composed of several subnets because otherwise it would be impossible to distinguish different activities in different places. Many modern systems in reality have parts or states that can be reversed. Using bi-directional transitions can be useful for static representation and execution of structures representing these systems. Formal and semi-formal representations are important for requirements engineering and quality control.

Different notations in literature have been used for representing and comprehending system behavior [7]-[10]. Bi-directional flows/arcs give a better representation of system behavior [6]. Different types of structures and models have been brought along to represent these systems. Some models have representation that is good for visualization [8]-[10].

An interesting area of system modeling is that of symbolic modeling used for system representation. Symbolic modeling is how to computationally represent knowledge [7],[8]. Symbolic architectures offer advantages for human understanding of a system. In this work by symbolic modeling it is intended how to symbolically represent computer processes and architectures in a way that they are better understood by humans. As an idea, symbolic modeling is the use of visual constructs or notations that supply information to different stakeholders

[9],[10]. Symbolic modeling here implies that large systems are simplified and reduced. I.e. the state space is compacted through abstraction rendering the model more compact [13]-[16].

Transition systems are simple but powerful formalisms that explain the operational workings of concurrency models. They create a framework for presenting underlying relationships and interactions between different approaches for the study of distributed systems [9],[10].

One problem is that it is difficult to classify which type of transition system corresponds to which model. Transition systems seem to cover a vast range of different types of networks and Petri net types seem to be part of transition systems. Normal Petri nets can be considered to be a special type of forward directed transition systems.

A Petri net is a forward directed transition system [11]. In all classes of Petri nets the type of event ordering is normally forward based. Causal ordering or events can take place to a certain extent. However, reverse event ordering is usually disallowed from taking place.

Concurrency, causality, conflict and confusion are not so simple to represent and define [11]. For this reason authors have preferred to use reduced or restricted net structures like elementary nets. Transition systems have similarities to elementary nets. Some Petri net classes are restricted or reduced for the simple reason that they give more strict control. Operations and control are easily represented in restricted classes. Elaborate definitions of controlled structures can be extended by including elements like traces, non-sequential processes, etc. In the real world many modern systems do not exhibit controlled behavior.

In this work the normal view of Petri nets or similar types of networks [12] will be to go against the normal tradition of having Petri nets as being forward ordering transition systems. This is an oversimplification of real world transitions. This paper is structured as follows: 1 Introduction, 2 Some Related Works that describe transition systems and Petri nets, 3 Motivation and Problem Definition, 4 Proposed Solution, 5 Implementation, 6 Some Examples, 7 Results and Findings 8 Conclusions.

2 Some Related Work

2.1 Transition Systems

Transitions are the founding blocks of all system major processes. Transitions represent events that are taking place or are bound to take place in the future. Past events are also part of the scenario. Events can be composed of a number of transitions that occur in some causal ordering sequence. For modeling these types of behavior different notations have been created. The simplest form of diagrams for representing transitions is the state transition diagram or state transition graph. To this, different notations and elements have been added making it more complex and capable of representing more detail. In software modeling and software engineering other notations have been created like automata, UML activity diagrams, transition systems and Petri nets.

For simple representation of transitions their depiction can be normally done using visual graphical notations. Pictorially a transition system can be represented as an edge labelled digraph with a given root [9],[10]. Hence, this type of system can be called a forward directed transition system, implying that the future states of the system are the result of the current states. It transits into a forward state. This means that the states in the future are the result of the current state. Hence it transits into a forward state.

Formally a transition system is a pair (S,->) where S is a finite set of states, S= s1,s2,s3,…..,sn and -> is a finite set of transitions t1,t2,…..,tn. A transition from a state s1 to s2 ie. (s1,s2) is given as E ->, s1 ->s2. A labelled transition system (LTS) is a tuple (S,A,->) . The edge from s1 to s2 is written s1 –a-> s2. I.e. the edge is given a label [9],[10].

Transition systems (TS) are used to describe the potential behavior of discrete systems. TS consists of states and transitions between states. System labelling can be simple or complex. Normally, transition systems are representable using directed graphs. i) TS by default have direction or single direction. ii) TS refers to any type of behavior when a state and transitions are involved. Simulation pre-order represents the relationship between state transition systems where the moves of another system are intuitively matched.

2.2 Petri Nets

Petri nets are bi-partite graphs with special added features that can model concurrency, mutual exclusion, sequence and control in communicating systems that exhibit discrete behavior [11], [17]. Petri nets are based on automata principles and also on transition system properties. Petri nets can be considered to have two main types of structural representation. These are i) representational or pictorial structures and ii) executional structures that show the execution of the net. A third structure based on the mathematical representation of the net

can be included. In this work the focus is on the representational and executable structures. The executional structure of the net is a very important property of a Petri net.

Other more complex and abstract classes of Petri nets are possible. E.g. Colored Petri Nets [17], Object oriented nets, higher order nets. Normally these structures have a fixed direction. This is because transitions have entry points and exit points. From a logical and controlled perspective this makes sense because it is significantly easier to understand and control what is happening.

3 Motivation and Problem Definition

In this work the rules used in Petri nets are relaxed and not adhered to. Petri nets can be considered to be a special type of digraphs. The fact that digraphs are used implies that once a transition fires in a particular direction the firing cannot be reversed automatically. Transitions have deterministic and pre-ordered firing direction. The Petri net can be structured in such a way as to reverse its state, however this is normally done by adding more places, arcs and transitions. Reversing the state implies that deterministic behavior is still being used. A new network topology or a Petri net with undirected edges connected to nodes is created. This implies that transition firing can happen both ways i.e. bi-directionally. Modeling power is drastically increased and non-determinism principles are introduced. Behavior that is not allowed to take place in the definition of classic Petri nets is allowed here. Obviously, many new issues are created and some form of control principles or rules might have to be defined. The advantage of this approach is that the modeling power of Petri nets is extended and reflects in a more precise way what takes place in modern systems where normally it makes sense to be able to reverse transitions. As a simple example, in many modern autonomic or autonomous control systems it is possible to have state reversal as an automatic or reset process. Considerable research has been carried out in the direction of transition systems [7]-[9]. These structures are based on digraphs and are suitable for visualization.

Symbolic modeling is imperative for understanding and representing system structures. Visualization techniques are useful for representing system structures like networks. Bi-directional transitions serve to explain in better detail the real scenario. However, it must also be considered that having transitions that can fire bi-directionally definitely increases the undecidedness in the system. Some reasons for bi-directionality in transition firing are: i) it is more realistic, ii) shortcut modeling can be used, iii) weak association and weak controls can be introduced, iv) more realism of what happens in the real world is introduced.

4 Proposed Solution

The proposed solution here a prima facie seems to be very simple. Instead of using directed arcs as in normal Petri nets the direction of the arcs is simply removed creating undirected arcs.

However, this creates a number of issues that need to be addressed. The formal solution can be given in future works. A brief idea is given. By removing the direction of the arcs it is implied that tokens in the net can travel in both directions. This creates a structure that is more complex and abstract than a normal Petri net. The concept of undecidedness is introduced and the normal orderly or linear firing sequence that is found in Petri nets is no longer there. Even the normal firing routine in Petri nets will be changed.

This type of structure will closely imitate what happens in the real world, where behavior is not necessarily ordered. The bi-directional transition net is suitable for i) static and ii) dynamic modeling. The latter is more difficult to control and will require some specific rules according to the case being modelled.

5 Implementation

By definition the bi-directed transition net (BDTN) or bi-directional Petri net has the following properties. The bi-directed net is a five tuple structure N = (P,T,E,IF,OF) where P = {p1,p2,p3 ,…,pn} a finite non- empty and non- negative set of places. T ={t1,t2,t3,…,tn} a finite non-negative and non-empty set of transitions. Transitions and places are disjoint. i.e. no object can be both a place and a transition. $P \cup T = \phi$ and $P \cap T = \phi$. $E \subseteq \{(P \times T)\}$, $IF = P \times T \to c$ is an input function that defines the value of an arc from a place to a transition, $OF = T \times P \to c$ is an output function that defines the value of an arc from a transition to a place. C is a set of non-negative integer values. i.e. $c \geq 0$. For a normal Petri net, transition firing is controlled and regulated by the number of tokens available to a transition. A transition t where $t \in T$ is said to be enabled IFF every input place p of t where $p \in P$ contains at least the minimum number of tokens equal to the weight of the directed arc that connects p to t, i.e. M(p) ≥ I(t, p) where $p \in P$. By definition it has to have tokens for all the input places. For the BDTN this rule is relaxed. For firing to occur in the BDTN

$\forall p \in P$ input to t, where $t \in T$, the sum of all tokens in places connected to t $\geq$ firing value for the transition. I.e. for a transition to fire it has to have a certain value of tokens available directly.

The firing here is associated with the transition and not with input arcs as is done in Petri nets. Similarly, when firing a transition, the output values are placed in accordance with the output arc weights. Here this is not the case, as the output values are placed according to the transition values. It is possible to create input and output rules or functions for the transitions. This is partially explained in fig. 1. Other forms of control can be included. E.g. it is possible to have arcs with negative values to indicate that the arc removes tokens and arcs with positive values to indicate that they place tokens. The following general rules will serve to give a brief overview of the implementation of the BDTN. i) Places can serve as inputs or outputs. Alternatively they can be called channels or stores. ii) For transition firing there have to be a sufficient amount of tokens in the input places. However some input places can be empty because firing is controlled by the transition and not the arcs. iii) There are various possibilities for output once a transition fires. One place can be left empty and the tokens moved into another place. This is different from traditional Petri nets where normally the output places cannot be left empty after transition firing. iv) Additional rules might need to be added independently to a model for a particular scenario. This would be a scenario related rule. v) Some form of ordering in firing might be necessary.

6 Some Examples

6.1 A Simple Web Login Process

Fig. 1 clearly illustrates the concept of the BDTN in comparison to a typical Petri net. The idea of a web login process is introduced and compared. In fig. 1 a) the Petri net shows that at each stage of the process e.g. at the *pwd entered* stage it is possible to roll back to a previous state, this is achieved by the extra transitions that have been added. In fig. 1 b) the BDTN shows the same functionality of the Petri net but at each stage connecting arcs have no direction, implying that the tokens can flow bi-directionally. Hence automatically a state is reversible or a new state can be achieved. E.g. at the *signon pressed* state a token in this place can be removed and placed in the *pwd entered* state, thus reversing the state. It is obvious that the BDTN model is more compact and elaborate than the Petri net for modeling purposes.

6.2 Static Comparison of a Petri Net with a Bi-Directional Net

Fig. 2 shows how the BDTN compares with a normal Petri net. In a normal Petri net arcs are always directed and are either input or output arcs. On the other hand this paper is proposing a net structure where the arcs do not have a specific direction or are unidirectional. The overall structure still looks like a normal Petri net from the static point of view. However, the BDTN's dynamic functionality will be significantly different than that of the Petri net.

6.3 Dynamic Operations of an Elementary Bi-Directional Net

Fig. 3 shows how the dynamic operations for the BDTN given in fig. 2 ii). i.e. fig. 2 shows all the possibilities that are available when firing the main transition T1. Note that this is not a given sequence of how the outputs occur and all places have a restriction of one and the arcs can carry only one single token. The transition requires a minimum input of two tokens for firing. This is clearly indicated by the number two in the transition. There is no given temporal order how this firing will affect the output. I.e. the output and the input can happen in any order. The output can become the input for the next transition firing. Hence, the structure in fig. 2 ii) is a live structure that is continuously enabled. However fig. 3 shows all the possible six states of the system which are reachable. The structure is a live structure without deadlock.

6.4 A Simplified Bi-Directional Net with One Token

Fig. 4 shows a simpler version of fig. 3. This has been obtained by reducing the token to one and the transition requires only a single token for firing. Clearly the system has less states. These are just four states. It is shown again that the firing can occur in any order.

6.5 A More Complex Bi-Directional Net

Fig. 5 shows a more complex BDTN. In this example transition t1 and t2 require a minimum of two tokens for firing. The tokens can be present in any of the places connecting to t1 and t2. E.g. t1 is enabled and can fire as long as two tokens are available together from any given combination p1,p2,p3 and p4. In fig. 5 t1 is enabled because there is one token in p1 and one token in p2 which give the two tokens required by t1 for firing. If t1 fires then the possibilities listed in fig.3 are all available.

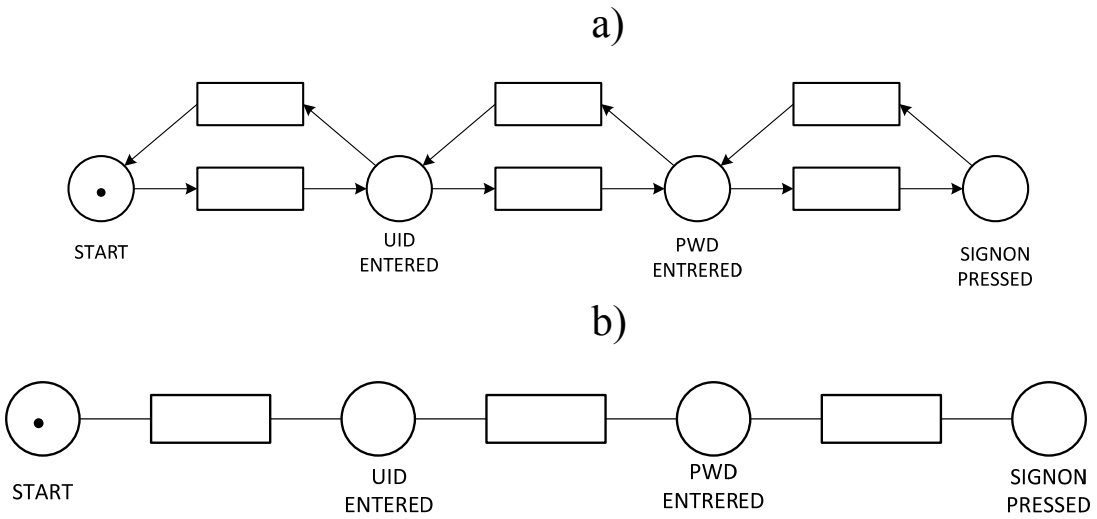

Figure 1: A Simple Web Login Process a) Petri Net Model and b) Bi-Directional Model

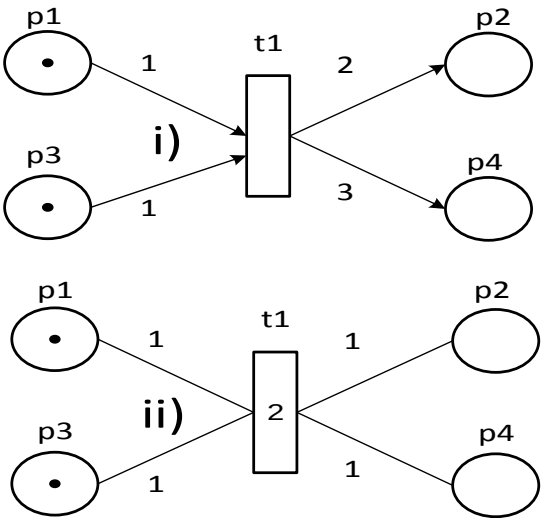

Figure 2: i) A Simple Petri Net vs ii) A Bi-Directional Net

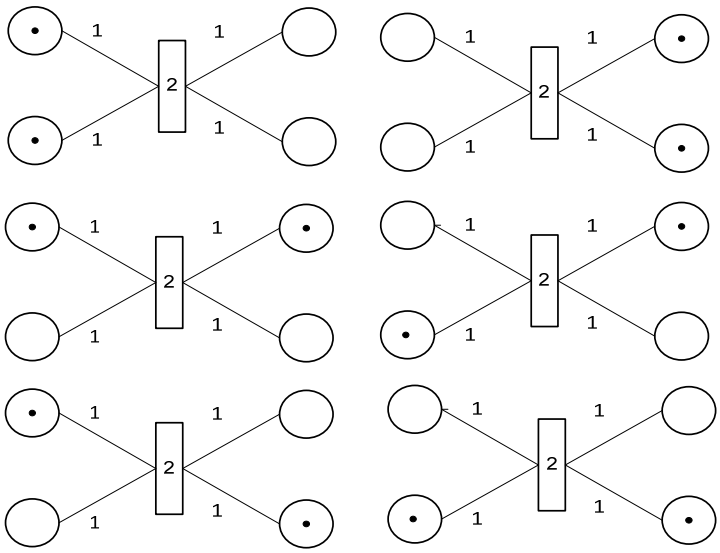

Figure 3: Possible Markings/States for an Elementary Bi-Directional Net

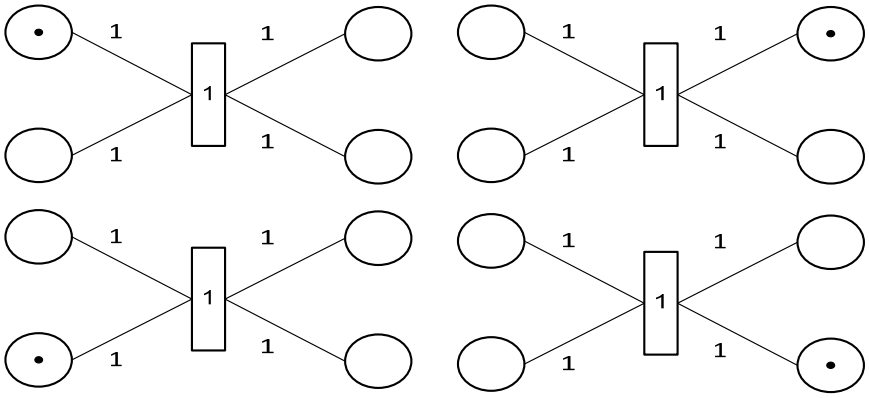

Figure 4: A Simplified BDTN with a Single Token

Transition t2 has been restricted. I.e. it can fire only if one token is present in p3 and one token in p4 or two tokens are present in p5. So t2 functions like a two way switch. This can be done because the place capacity of p5 is two and the connecting undirected arc from t2 to p5 and vice-versa has a weight of two implying that either two tokens are outputted or inputted in a single instance of firing t2. Fig. 6 indicates the presence of choice in this net. In fig. 6 having one token in p3 and one token in p4 and the concept of bi-directionality implies that both transition t1 and transition t2 are simultaneously enabled. But only one can fire because firing one automatically will disable the other. Again this net structure is completely reversible and live. I.e. the firing can go on indefinitely unless some restrictions or conditions are added. This shows that a plethora of new possibilities are opened up by connecting transitions, places and undirected arcs. The detailed workings of this network structure are not shown here.

6.6 A Circuit Representation using a Petri Net and a Bi-Directional Net

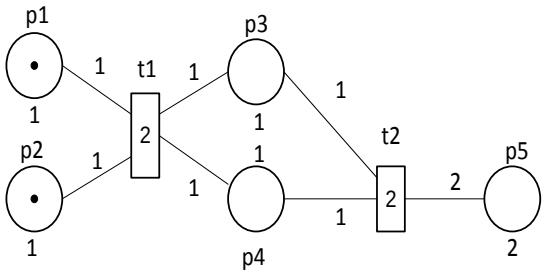

Figure 5: A Complex Bi-Directional Net

Fig. 7 shows a normal Petri net circuit and a BDTN. Even though structurally they look similar there are big differences in the operations that are possible. In the Petri net tokens can flow only in one direction. On the other hand in the BDTN the token flow can be in both directions. So the BDTN serves as a dual circuit.

7 Results and Findings

This paper shows how a new form of Petri Net called a BDTN can be used. This structure retains many of the main essential structural properties found in general Petri net classes. However, it is clearly indicated that the behavior and firing is not similar.

It is obvious that BDTNs offer more detail than Petri nets. This comes at the expense of having complex behavior that might not always be

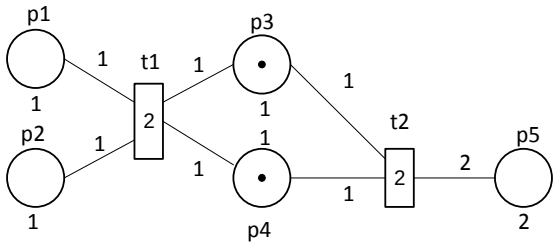

Figure 6: Free Choice in the BDTN

controllable. But if the BDTNs are properly constructed and restricted it is possible to achieve better control. This is indicated in fig. 6 where the possible firing of the net has been restricted by using place capacity restriction and arc weights. The simple web login process indicate the usefulness of the BDTN as a static and dynamic modeling notation and tool for very simple processes. These processes are serially linked together. This example shows how we can reverse the process from one stage to the previous state. Fig. 2 shows that even though the structure looks simple, just by increasing the number of tokens in the places it is possible to generate great complexity. This can be seen when fig. 3 is compared to fig. 4. Fig. 7 compares a normal Petri net with a BDTN. The bi-directional is reversible at every instance and hence it expresses more behavior than the Petri net. To represent the same behavior using a Petri net would require adding more arcs and transitions. Hence the result of

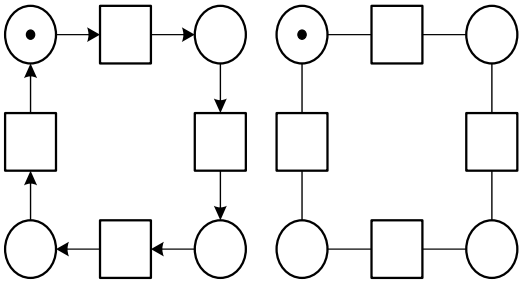

Figure 7: A Circuit in a i) Petri Net and ii) BDTN

this is that the BDTN is shown to be more compact. Fig. 5 indicates how even a simple structure becomes complex when the arc weights are greater than one. Fig. 6 shows how the net in fig. 5 contains

free choice. Either t1 or t2 can fire but not both. Fig. 8 shows the marking graph for the BDTN in fig. 4. The marking graph clearly indicates that it is possible to reverse a state to any other state directly. The marking graph in fig. 8 clearly shows the bi-directionality linking in the net. It is possible to construct a complete Petri net from this marking graph. The marking graph for the net in fig.3 is similar but more complex to construct and link and it will have six states. The more complex the BDTN, the more difficult is the construction of the marking graph. If a Petri net is compared to the BDTN in terms of the marking graph, the marking

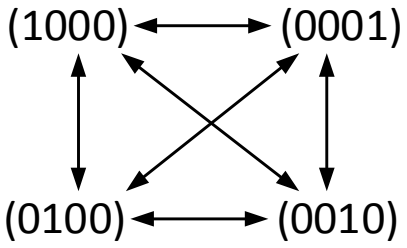

Figure 8: A Marking Graph for the Bi-Directional Net shown in Fig.4.

graph for the Petri net will be much simpler to construct and depict. The marking graph can be constructed symbolically to represent the structures.

The main results of this work have shown that i) it is possible to construct a BDTN, ii) the BDTN can capture more detail than a Petri net, iii) the BDTN will require some additional form of restriction or control.

When a BDTN structure has places that connect to multiple transitions, there is undecidedness or the issue of choice. This implies that it is unknown which transition fires. This analogy compares to the real world situation. E.g. if a person is in front of two or more shops he can enter whichever shop he wants. Petri nets can exhibit choice which is controlled to an extent. In the BDTN this choice is uncontrolled.

The BDTN can be transformed into a Petri net by adding more input arcs, output arcs and transitions. The resultant behavior should be identical. But obviously the resultant Petri net is a more complex structure.

An interesting feature of the BDTN is that this structure can be combined with normal Petri nets. This implies that there is a possibility of creating more dynamic and powerful structures.

A fundamental difference between the Petri net and the BDTN is that the Petri net structures allow for what can be called a memory effect, because the current state is the result of the previous states that are identifiable. The memory effect depends on the system transition timeline. This would be easily evidenced in a Petri net. On the other hand, this type of memory effect and timeline transition firing seems to be lost in the BDTN because the previous states cannot be simply recalled from the current state. This could cause problems or be of an advantage depending on what needs to be modelled. If the memory effect is needed for the BDTN some solution needs to be found. For strictly forward transition system behavior the BDTN is not suitable if the requirements need to be strictly adhered to.

A state explosion is when the number of states in the system increase exponentially. This situation becomes uncontrollable.

8 Conclusions

This work has introduced a new type of Petri net which has been called a BDTN. Although this structure behaves like a Petri net it can capture more detail. Even a simple looking BDTN can manifest complex behavior because if it has more than one transition and many places and two or more tokens.

Combining the BDTN with Petri nets will result in notations useful for more expressive modeling. The opposite also holds true.

There is the issue of the memory effect. The BDTN seems to cancel out memory and always remain in the current state. I.e. the fact that multiple previous states exist make it difficult to trace the original states and sequences.

This work is incomplete in the sense that just a very trivial idea of the BDTN has been presented along with some toy examples. Proper rules for firing have not yet been established. It has been left to the user to interpret and set the firing rules for these structure thus allowing a lot of room for experimenting and more possibilities.

The BDTN could prove to be useful for symbolic representation and modeling of some network and system classes in the real world.

For future work the BDTN can be constructed along with a Petri net and comparisons made as to which is the most useful for the problem approach. As previously mentioned combining the BDTN with Petri net structures could prove to be useful for requirements elicitation. From the representational point of view the BDTN is useful for system representation. It will help to stimulate thinking and visualization of system behavior. The BDTN can definitely benefit from the support of other notations. This work has shown the usefulness of the BDTN approach.

References:

[1] A. Spiteri Staines, A Colored Petri Net for the France-Paris Metro, *NAUN International Journal of Computers*, Issue 2,Vol. 6., 2012, pp. 111-118.

[2] T. Spiteri Staines and F. Neri, A Matrix Transition Oriented Net for Modeling Distributed Complex Computer and Communication Systems, *WSEAS Transactions on Systems*, Vol. 13, 2014, pp. 12-22.

[3] T. Spiteri Staines, Implementing a Matrix Vector Transition Net, *British Journal of Mathematics & Computer Science*, ISSN: 2231-0851, Vol.: 4, Issue.: 14, 2014, pp. 1921-1940.

[4] A. Spiteri Staines, Some Fundamental Properties of Petri Nets, *International Journal of Electronics Communication and Computer Engineering*, IJECCE, vol.4, Issue 3, 2013, pp. 1103-1109.

[5] K. van Hee, *Information Systems: A Formal Approach*, Cambridge Univ. Press, 2009.

[6] G.-C. Yang, Distributed System Modeling with Bidirectional Petri Nets, *Proc. Of the 'Computer Systems and Software Engineering conf. (CompEuro '92),* IEEE, 1992, pp.401-405.

[7] D. Kleyko, E. Osipov, On bidirectional transitions between localist and distributed representations: "The case of common substrings search using Vector Symbolic Architecture", *BICA 2104, 5th Annual Int. Conf. on Biologically Inspired Cognitive Architectures,* Procedia Comp. Science, Vol 41, 2014, pp 104-113.

[8] T.D. Kelly, Symbolic and Sub-Symbolic Representations in Computational Models of Human Cognition, *Theory& Psychology*, Sage Publications: Vol. 13, No. 6, 2003, pp. 847-860.

[9] F. van Ham, H. Van de Wetering And J.J. van Wijk, Interactive Visualization of State Transition Systems, *Transactions on Visualization and Computer Graphics*, Vol. 8, No.3, IEEE, 2002, pp. 1- 11.

[10] J. Osis, and E. Asnina, Topological Modeling for Model-Driven Domain Analysis and Software Development: Functions and Architectures, *Model-Driven Domain Analysis and Software Development: Architectures and Functions*, 2010, pp. 15-39.

[11] T. Murata, Petri Nets: Properties, Analysis and Applications, *Proc. Of the IEEE*, Vol 74 issue 4, IEEE, 1989, pp.541-89.

[12] M. Bouhalouane, S. Larbi And H. Haffaf, Combining Bond Graphs and Petri Nets Formalisms for Modeling Hybrid Dynamic Systems, *10th Int. Conf. on Future Networks and Communication*, Science Direct, Procedia Computer Science, Elsevier, 2015, pp. 252-259.

[13] H. Kaindl And J.M. Carroll, Symbolic Modeling in Practice, *Communications of the ACM*, vol. 42, No 1, 1999, pp. 28-37.

[14] D. Ross, Structured Analysis: A Language for Communicating Ideas, *IEEE Trans. Softw. Eng.* ,Vol.3, No 1, IEEE, 1977.

[15] R.A. Kremer And B.R. Gaines, "Embedded Interactive Concept Maps in Web Documents", *In Proc. Of WebNet96*, H. Maurer, Ed. Charlottesville, VA, 1996, pp. 273-280.

[16] M. Janakova, Software Development with Regards to Simulations: Are Interaction Features Needed for a Better Description of Actual Reality?, *WSEAS Transactions on Information Science and Applications,* Vol. 11, 2014,pp. 177-185.

[17] L. De Carvalho Vidal, L. E. De Souza, D. De Paula Santos Silva, R. Sebastiao Nadur, Petri Nets: an Analysis of its Properties through a Model of Titanium Injection System and Other Pulverized into Blast Furnaces by using the Software CPN Tools, WSEAS Transactions on Systems, Vol. 15, 2016, pp. 207-213.