Algebraic Representation for Ordinary Place Transition Petri Nets

A. Spiteri Staines

***Abstract*—** Ordinary place transition Petri nets are useful for modeling discrete systems at a low level. It can be shown that the behavior of these structures does not entirely depend on the static model but also on the resource distribution in the net. In this work the problem of representing Petri nets is presented. Some algebraic notations for modeling at the i) structural and ii) operational level are presented. Some simple examples are used to illustrate the usefulness of these notations and expose hidden concurrency issues in Petri nets. The results are discussed. The ideas presented here are just an outline of what can be done in this area.

***Keywords*—**Algebraic Representation, Concurrency, Petri Nets, System Modeling.

I. Introduction

Petri nets are elegant, well defined semi-formalisms used for modelling of discrete event systems [1]-[7]. They can naturally model sequential, concurrent and parallel behavior. These nets are based on important system properties: i) events, ii) conditions and iii) structure. Petri nets have a dual identity [7]. They can be represented i) graphically and ii) using mathematical terms like matrices and equations. The graphical depiction is normally more useful for visual modelling and communication, whilst the mathematical representation is useful for compactness, formal checking, verification and testing [1]-[4].

Comprehensive Petri net software simulators exist [2]. These are known for the token game simulation process. For ordinary place transition nets, the time dimension is not considered, however, this does not imply that it is unimportant.

The elementary and logical characteristics of Petri nets make them a versatile tool for representing and analysing behavior of certain system classes [2]-[4]. In literature, Petri nets are normally combined with other formalisms or notations for extended modelling power.

Due to their simplicity ordinary Petri nets can be modified to represent diagrammatically the salient characteristics of low-level behavior. Several formalisms like communicating sequential processes (CSPs), communicating system calculus (CCS), process algebras and various others share many common properties with ordinary Petri nets [3],[13], [11], [14]. These formal notations can be combined with ordinary Petri nets for more expressive analysis of different scenarios.

Petri nets might look similar to other graphical and formal notations. However, several properties of Petri nets single them out [1],[2],[10]. Originally Petri nets were intended to model concurrency and discrete system behavior. Some confusion exists as to whether Petri nets should be classified as formal or semi-formal constructs.

A. Spiteri Staines is an associate academic at the Department of Computer Information Systems, Faculty of ICT, University of Malta (e-mail: toni_staines@yahoo.com).

II. Some Background and Motivation

Several classes and extensions to ordinary place transition Petri nets exist. In principle the graphical representation of these structures will look similar because visually Petri nets are bi-partite digraphs always composed of i) places, ii) transitions, iii) input and iv) output arcs. Basically the places and transitions are the nodes and the connecting arcs are the edges. Unconnected edges and nodes are prevented through the formal definition and rules of Petri nets. Additionally other rules are added for places that can contain token values. Places are used for activation and firing of transitions. The visual representation of the net is what distinguishes them from other algebraic notations.

As new computer technologies are being created, it is important to have mathematically sound and precise models for representing systems. Hence there is a lot of focus on diverse modelling techniques and representation [7]-[10]. System correctness does not only depend on the architectural or static structure of a representation but also on the executional behavior of the system. Changes that occur in the dynamic behavior part are often difficult to predict even in ordinary Petri nets. There is a certain amount of unpredictability and non-determinism in Petri nets. Representing these issues is essential for the requirements elicitation process. Understanding the processing involved will eliminate the possibility of flaws that can emerge in later stages.

In previous work [12] it has been shown that even simple trivial place transition nets can hide some serious concurrency issues. When constructing concurrent systems, the correctness and validity of the model will save time and effort. Too much complexity in a system is undesirable thus making validation difficult to carry out.

Place transition Petri nets can behave non-deterministically. The output and behavior of the net depends on the token configuration which cannot always be known apriori.

All the major classes of Petri nets can be divided into two

parts. i) the static part and ii) behavior or dynamic part. The static part is all about structural representation. This is related to how the net will appear graphically but this is independent from the actual behavior.

An ordinary place transition petri net can be constructed accurately but actually none of the transitions might fire for various reasons. The behavior of the Petri net does not depend only on the structure but also on other factors like the token distribution (resources), the input and output arc values. To validate the net, behavior and firing of the net has to be completely checked to see if there are unreachable states, possibly infinite loops or undesirable behavior. Normally this is done by constructing the reachability graph or tree.

III. PROBLEM DEFINITION

For representation, the use of algebraic notations for

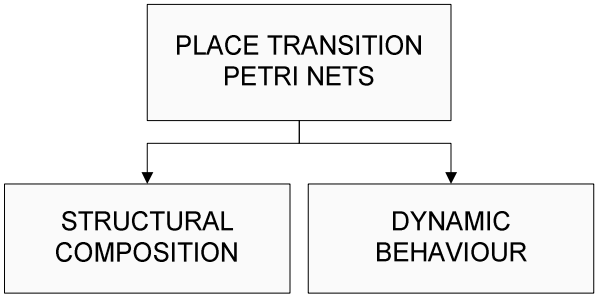

Fig. 1 Static vs Dynamic Composition

depicting the net would help to get better understanding and also check the executional properties [12]. It is important to find new ways to represent Petri nets because the visual structure of a net cannot describe all its properties and does not indicate if the net is error free or if there are other hidden configuration issues.

The following problems are related to proper concurrency and parallelism modeling in ordinary place transition Petri nets: i) The physical structure of an ordinary Petri net can support different processing behavior at different times depending on the configuration of the tokens or resources in the net, ii) Non-deterministic behavior can imply that a transition even though it is activated will never fire at all, iii) Once a transition fires it cannot be stopped. It is unstoppable, iv) The transition firing has to complete itself both in the input and output part, v) A transition can be blocked for several reasons. E.g. if there is no output space available the transition will never fire, vi) A Petri net can be composed of several subnets each with different behavior. Because of these issues concurrency and parallelism representation become a problem [12].

Mainstream literature seem to focus on the different classes of Petri nets and solutions to given problems. The fact that Petri nets often need the support of other tools or notations for

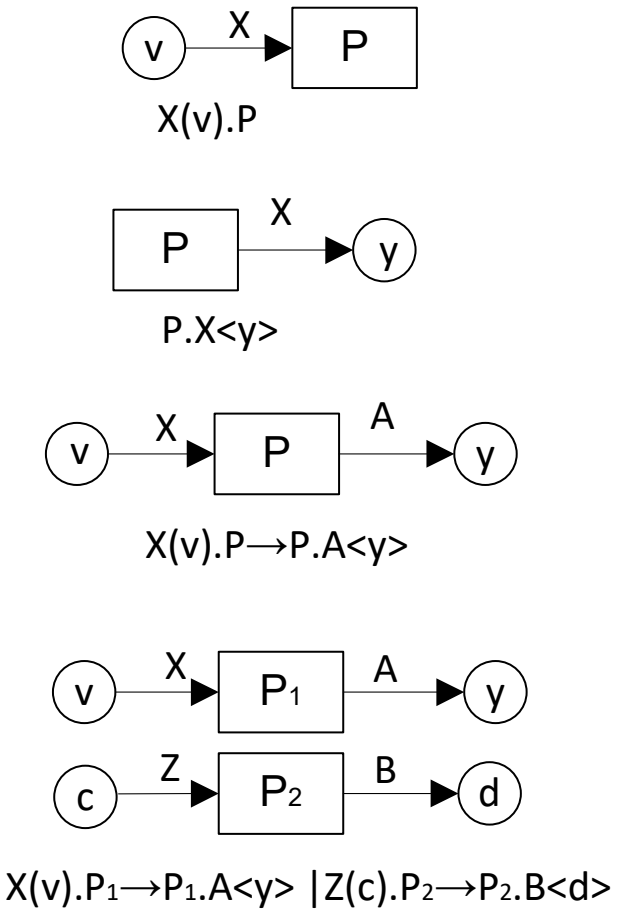

Fig. 2 Algebraic Notations for Static Representation

proper representation is often underestimated. The model needs to be verified and run to establish if it is correct or not and then the necessary modifications and corrections can be carried out.

A mathematical or algebraic approach for representing Petri

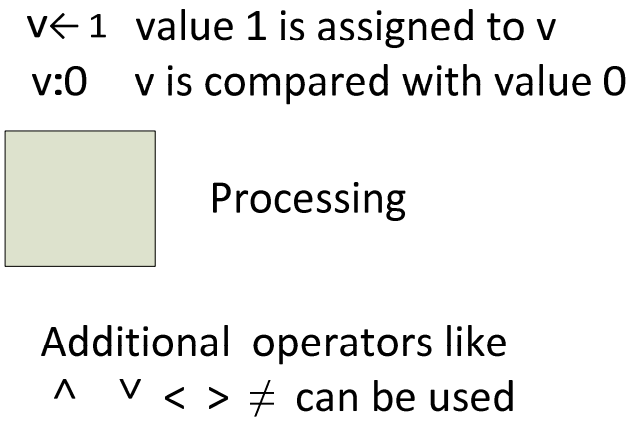

Fig. 3 Some of the Algebraic Notations for Dynamic Representation

nets is useful for system decomposition and composition. These would be useful for understanding concurrency and parallel behavior both at the structural and behavioral level. The equations or notations themselves could be put into a net or another system and neatly fit together.

IV. PROPOSED SOLUTION

This work proposes the use of algebraic notations for the static and notations for dynamic representation as in fig.1.

It is difficult to represent all the details of a system using modelling notations. However, an ordinary Petri net is a reduced form of a system so representing a Petri net will be simpler. The idea of combining visual notations with formal representation is not new. This has already been done using Z and the Vienna Development Method (VDM) [3]. The systematic representation of Petri nets even though simple looking requires suitable representation that should be concise, precise and consistent. The possibility to develop the syntax or algebraic notation further should exist. Existing formal languages are unsuitable because their principal goal was not geared towards this end. Some other forms of representation lack precision and conciseness.

A. Structural Representation

The Petri net structures are given as a set of operations. The input operation is given as X(v) where v is an input data value or a input store and X is an input arc. The output operation is given as X<v> where v is an output data value or an output store. Both X(v) and X<v> can contain token values. i.e. tokens in $v \geq 0$. Tokens are bound to input and output places respectively. Parallel operations can be represented using the | (parallel bar). Thus Q|P represents that P and Q are separate concurrent or parallel processes. The processes for the Petri net are composed of the transitions. Other additional operators are (and), (or), ; (sequence), . (binding) and (transition).

This notation is indicated in fig. 2. X(v).P implies that place or resource v is on input X that is bound to P. P.X<y> implies that process P outputs to a place or resource y via X. I.e. P is bound to X<y>. X(v).P→P.A<y> indicates that the input to transition P is X(v).P and the output is P.A<y>. X(v).P1→P1.A<y> |Z(c).P2→P2.B<d> indicates that there are two process P1 and P2 that can possibly execute in parallel or concurrently. The two processes P1 and P2 are completely isolated from each other in terms of communication.

B. Dynamic Representation

The dynamic representation of the Petri net deals with the programming or execution. Very simple symbols are used and the net is decomposed into several processes. The following symbols are used: (assignment). As an example X ←1 implies value of 1 is assigned to x. The : (colon) is used for comparison of two values or data types. X:0 compares the value of X to 0. Additionally mathematical and logical operators similar to < (less), <= (less than or equal), > (greater), => (equal or greater), (and), (or) and several others can be defined as required. Initializing the net, transition execution or processing is placed in an appropriate box for partitioning. This notation is briefly represented in fig. 3. Other elements and expressions could be combined with the notation to increase its expressivity.

V. CASE STUDIES AND DISCUSSION

Some trivial examples are used to illustrate the proposals of this paper. The algebraic notations that have been created are useful for diverse scenarios and can be used to represent even more complex structures than those that are presented here.

A. Conflict/Choice Representation

In this example there is a classical place transition Petri net where there are two transitions that require input from a common place. I.e. there is a shared place. The performance of the net depends on the resources in the shared place and in the other input places if there are any. This is indicated in fig.4. In this case the algebraic representation for the static structure of the net is given as follows:

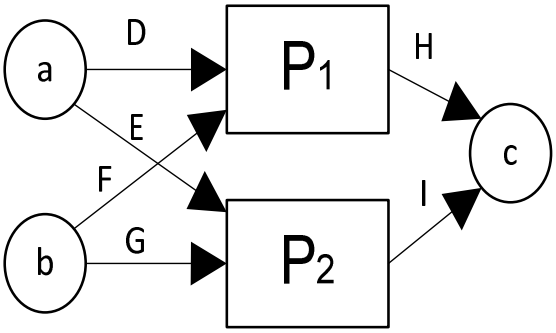

Fig. 4 Petri Net with Conflict or Choice

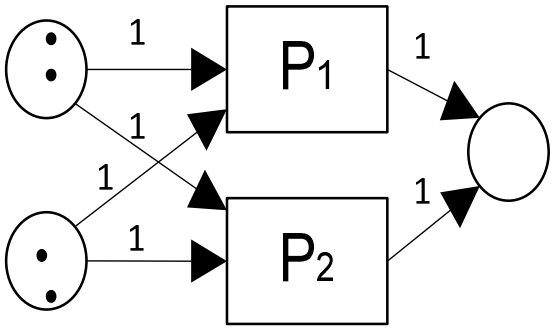

Fig. 5 Choice/Conflict Net with Resources

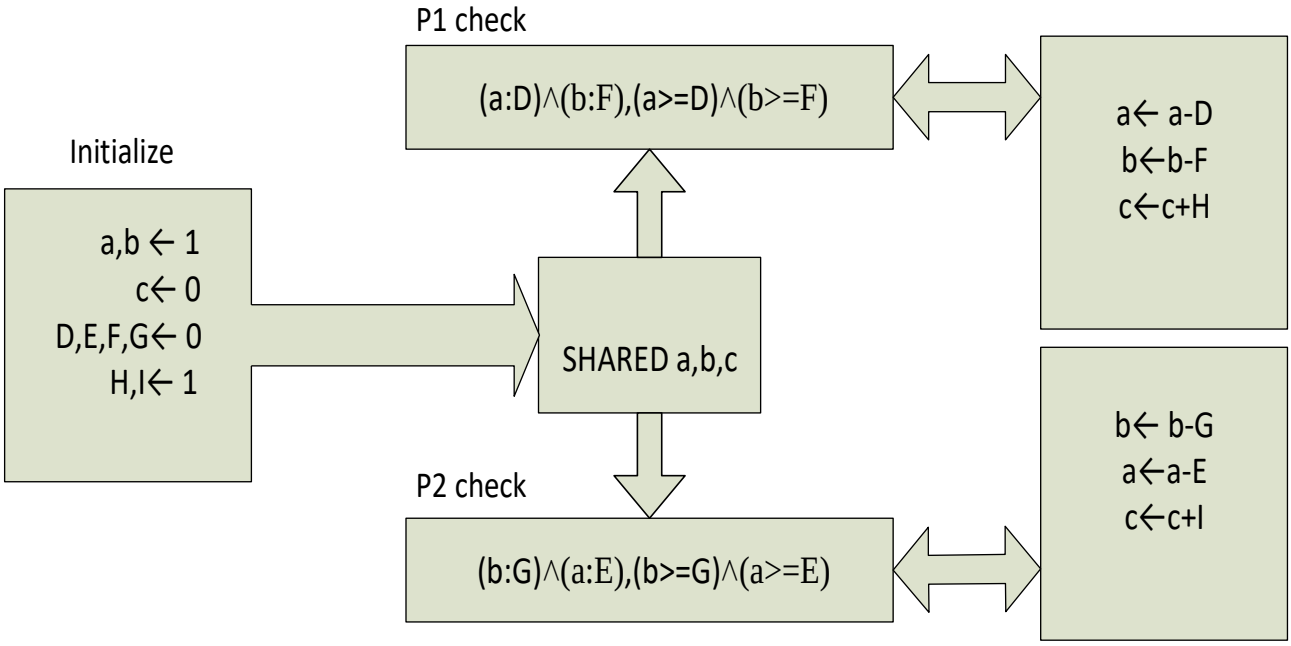

Fig. 6 Dynamic Representation for the Petri Net in fig. 4 and 5

((D(a).P1)∧(F(b).P1) →P1.H<c>|((G(b).P2)∧(F(a).P2)) →P2.I<c>

This equation has been created using the notations presented in fig. 2 and discussed in section iv). Basically the equation is stating that P1 and P2 are two separate processes. They do not communicate with each other and are independent of one another. Fig. 5 can be considered for more clarification. In this example both P1 and P2 are enabled simultaneously. This gives several possibilities. Some possibilities are: i) P1 can process or fire twice and so can ii) P2 or P1 and P2 can fire in parallel. Changing the resources in the input places will again change the number of possibilities. The equation previously presented just describes the static structure of the net.

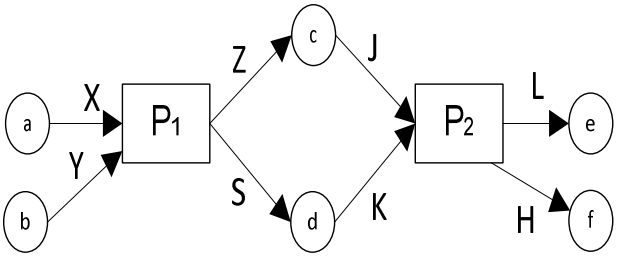

Fig. 7 Sequential Net with Possible Concurrency

The dynamic representation of the net given in fig.6 explains the possible behavior of the net. This diagram explains setting up the parameters of the net with initial values. The values given to the input/ output arcs do not change. The values given to places can change as explained by the parameters.

The places a,b,c are shared. This implies that they are global places and that these can be updated by either process P1 or process P2 independently. Any change to these shared places

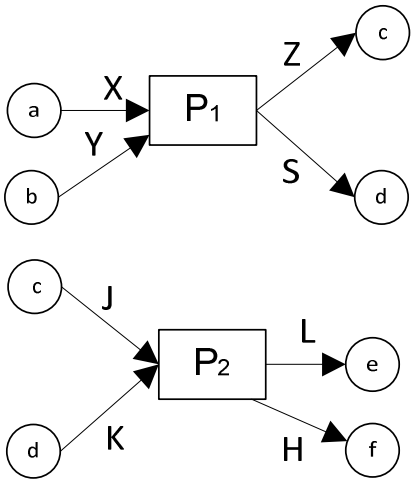

Fig. 8 Decomposed Net as per Equation

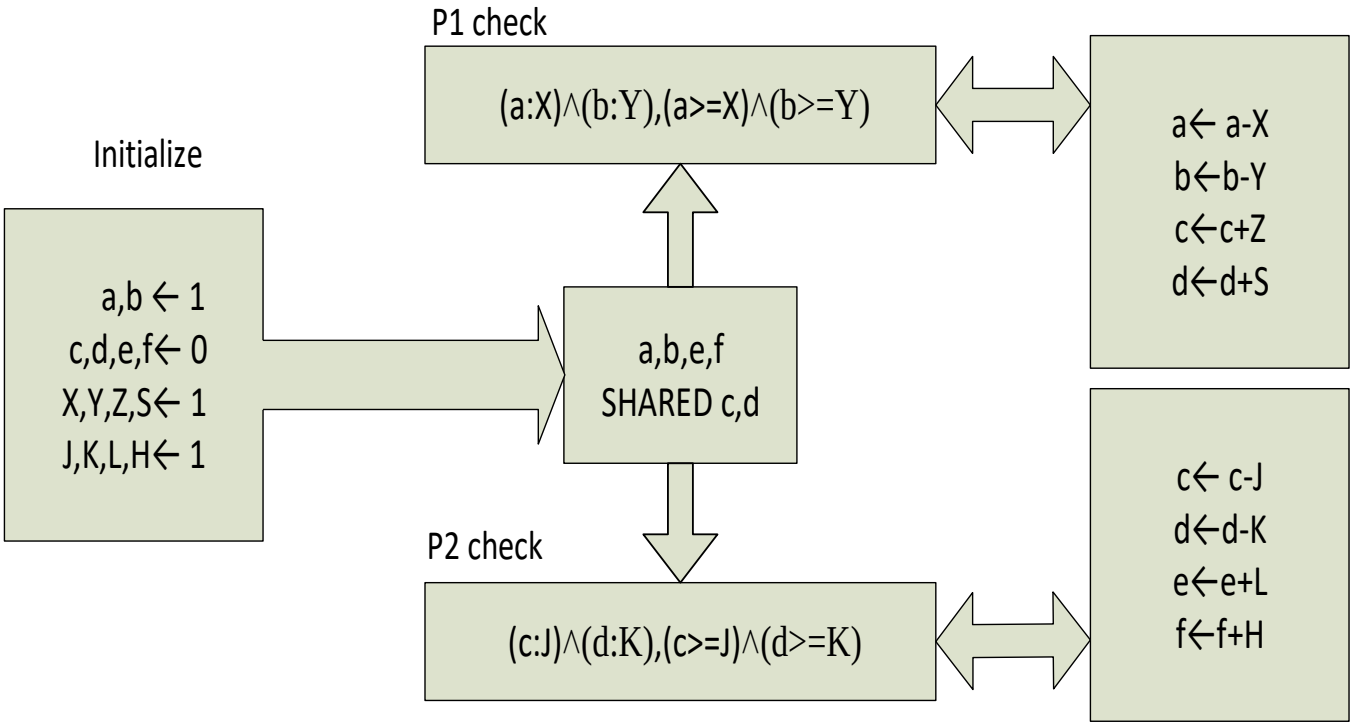

Fig. 9 Dynamic Representation for the Petri Net in fig. 7

i.e. the values of tokens in the places, can affect another process.

The equation (a:D)^(b:F),(a>=D)^(b>=F) implies that for P1 input places a,b are checked and that they both must have values greater than input arc D and input arc F respectively. If this is the case the next step can be executed. This is similar for process P2.

B. Possibly Concurrent Processes

A comprehensive example of possible concurrency is given using fig. 7. At first it might seem that this layout is sequential, however the distribution of resources will affect the functioning of the net. Process P1 and P2 can be simultaneously activated and operate concurrently. Normally this would not be the case if resources are placed into a,b only.

The equation that represents the static part can be given as: ((X(a))^(Y(b))).P1→P1.((Z<c>)^(S<d>))|((J(c))^(K(d))).P2 →P2.((L<e>)^(H<f>)). This equation shows that P1 and P2 are clearly separate processes. The | sign can be replaced by; if the processes were to definitely execute in sequential order. Thus the equation would be written as ((X(a))^(Y(b))).P1→P1.((Z<c>)^(S<d>));((J(c))^(K(d))).P2 →P2.((L<e>)^(H<f>)). However the use of the | sign indicates that the system is left open for concurrency or parallel behavior. The | could also indicate that undefined behavior is possible. The equation indicates that the structure can be represented graphically in a different decomposed form. This is given in fig. 8.

The dynamic part of this net is given in fig. 9 and is self-explanatory.

VI. RESULTS

Algebraic and similar notations have been widely used in computing. These notations form the very foundation of mathematical and formal approaches in this area. For Petri net representation algebraic expressions can represent important factors that often go unnoticed as has been indicated in the previous examples.

In the models, it is clearly seen that the static representation is insufficient to capture all the details of the net. In reality when Petri nets are executed using the token game the behavior is not necessarily clarified. Therefore a marking graph or tree could provide more details. The modelling approach presented can be used to validate or assess the validity of the Petri net. Essentially the behavior of the net can be classified into two types: i) sequential or ii) concurrent processing.

Other interesting results and findings can be presented. i) concurrency in ordinary place transition Petri nets does not really depend entirely on the structure of the net but rather on

the token or resource distribution inside the net ii) Petri net structures have non-deterministic behavior and are suitable for representing real world behavior or complex systems iii) even the most simple structures can offer surprising hidden complex behavior. This is evidenced in the model presented in fig. 4 and 7 iv) the static representation cannot predict the behavior of the net structure.

Additionally if the dynamic representation for fig.4 and fig. 7. as depicted in fig. 6 and 9 are considered they look quite similar. This is significant because there are two completely different Petri net structures which can give similar processing or behavior. This interesting finding indicates that from a fundamental point of view the behavior of the net does not really depend on the structure of the net but on the resource distribution.

The results clearly indicate that for system modeling the Petri net structures used should be validated and supported through the use of other structures or notations. This will yield better robust system design principles for requirements engineering.

Different authors have tried to formalize Petri net representation combining them with other formal methods which makes sense as in [3].

Even though at the specification level different forms of representation using many notations are offered, it is impossible to capture all the details in every scenario. Restriction of the design offers a greater amount of control and predictable outcomes.

This work identifies other possible areas of analysis that have not been checked here. These are the correct: i) initializing and ii) termination of Petri net execution. Even though they are simple in principle, they can involve different criteria complicating them.

The abstract Petri net structures can contain hidden concurrency issues. The concurrency problem does not only depend on the structure but on the layout of the net. Concurrency can become quite complex to manage in these structures.

If more complex classes of Petri nets like higher order nets are considered then more problems can arise specifically because of the language an enhanced structures that are represented.

The Petri net structures presented in this work are simple ones. If more complex models are used then the i) static and ii) dynamic representations will become rather complex and their creation will be time consuming. In this case there is the possibility to apply reduction principles in the net.

VII. Conclusion

The legacy and tradition of ordinary place transition nets and their importance is once again presented in the context of representing them algebraically. These were selected because they are simple and yet still expressive and valid today. Construction of models can be carried out very easily. Even though other classes of Petri nets exist, it can be indicated that no one class is better than another and each class is significant for solving problems addressed at a particular domain.

This work has briefly outlined possible algebraic notation for modeling ordinary place transition Petri nets. More work can be done to enhance and improve these notations, thus creating more robust modeling concepts. This work is extendable to other areas of Petri nets.

References

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

[2] M. Zhou, K. Ventkatesh, “Modelling Simulation, and Control of Flexible Manufacturing Systems, A Petri Net Approach”, World Scientific, 1999.

[3] K. van Hee, “Information Systems: A Formal Approach”, Cambridge Univ. Press, 2009.

[4] A. Knopfel et al., “Fundamental Modeling Concepts”, Wiley; 2005.

[5] A. Abellard, P. Abellard Ch. 5 Systolic Petri Nets. *In: Petri Nets Applications*, InTech; 2010.

[6] A. Spiteri Staines, “An Introduction to Bi-Directional Transition Network Modeling”, *International Journal of Computers*, IARAS, Vol. 2, 2017.

[7] A. Spiteri Staines, “Matrix Representations for Ordinary Restricted Place Transition Nets”, WSEAS Transactions on Computers, Vol 16, 2017.

[8] 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.

[9] A. Spiteri Staines, Modelling Simple Network Graphs Using the Matrix Vector Transition Net, CSSCC 2016, INASE, Vienna, 2016.

[10] S.K. Chang, “Principles of Pictorial Information Systems Design”, Pretence Hall, 1989.

[11] D. Lightfoot, “Formal Specification using Z”, Palgrave 2001, ISBN 0-333-76327-0, Ch 6, pp. 37 – 97.

[12] T. Spiteri Staines, “Concurrency Issues in Ordinary Place Transition Petri Nets”, CPA 2017.

[13] C.A.R. Hoare, “Communicating Sequential Processes”, Pretence Hall, 1985.

[14] R. Milner, “A Calculus of Communicating Systems” Springer Verlag, 1980. ISBN 0-387-10235-3.