RESEARCH ARTICLE OPEN ACCESS

# Methodology to Describe and Verify Systems Behaviour by Integrating Several Formal Languages

PhD. Bassem KOSAYBA [1], Waseem Ahmad [2]
Dept. of Software Engineering and Information Systems
University of Damascus, Damascus
Syria

**ABSTRACT**
Early stages of critical systems development include ensuring that implementation is valid across all operational scenarios that may be encountered. We examine systems models to find errors before going to the implementation. That allows to improve the software quality. Because we are ensured that the systems is in compliance with the imposed specifications and free of the rejected conditions. Petri nets is an important formal method for modeling the systems behavior, and this is confirmed by the increasing number of researches and studies to find conversions from different systems models to Petri nets. But Petri nets suffer from a low level of abstraction in both the system behavior verification as well as the modeling of systems behavior. That limits the Petri nets usage. In this paper, we have transformed the CSP models of systems behavior to Petri nets. CSP is a formal language to describe systems behavior as a combination of processes. The process is a set of events. Our proposed solution allows verification of the systems behavior using sophisticated query system rather than directly accessing the reachability graph of Petri net. That Allows to increase the level of abstraction of the Petri nets in both the systems verification and modeling.

***Keywords :***— Formal Method, CSP, Petri Nets, Temporal Logic, Reachability Graph, Verification Model.

## I. INTRODUCTION

The use of formal methods is an important factor in avoiding errors at an early stage of system design and verification of the required properties. Where formal methods tend to describe systems at a fairly simple level to facilitate modeling and analysis. Petri nets are a graphical method used to analyze complex distributed processing systems in terms of performance and reliability. Petri nets allow designers to verify the behavior of these systems by representing them using the concepts of petri nets (places, arcs, markers) and then get the reachability Graph equivalent to the Petri nets. Then, we can read this reachability Graph to verify that the states for the conditions that are rejected in the functioning of the system do not exist and that the conditions imposed are always fulfilled. The equivalent reachability Graph of the Petri net has the necessary information to predict all operating scenarios that a system represented by the Petri net can pass. In contrast, Petri networks suffer from the following points:

• Difficulty reading the reachability Graph because it is often too huge.

• The difficulty of modeling systems in Petri nets.

In this research we seek to take advantage of the power of the Petri nets to verify the behavior of critical systems. At the same time, we want to facilitate the use of Petri nets to verify the behavior of systems. In this paper, we have devised a new formal methodology to investigate the behavior of systems by characterizing systems using CSP and then converting them into a Petri net. Thus, our contribution is through modeling systems using high-level PETRI nets as written in CSP text, and from analyzing the tree of coverage for the resulting Petri network in order to inquire about the existence of the rejected and assigned conditions as query operations similar to temporal logic processes.

The chapter II, III, IV describes the methodology used to describe and verify the behavior of the systems: CSP, Petri networks, linear temporal logic (LTL). While in Chapter VI we explain our algorithms for applying transformations from the text description to the Petri nets and the use of temporal logic log processes in the tree trees of the Petri networks. In chapter VII we present the Conclusions and Future work.

## II. COMMUNICATING SEQUENTIAL PROCESSES(CSP)

Is a mathematical notation to describe the interaction and its use in the simultaneous behavior analysis of applications, CSP uses a set of processes to characterize behavioral characteristics [6]. Figure (1) [7] shows the use of CSP parameters to describe the effectiveness of the system,CSP processes use a number of transactions, such as choice, sequence, and parallelism. The names of processes are used in capital letters , and the lowercase letters are used to refer to events, the left side of each definition is the process name defined in the right side by expression A call to another process [6].

The sequence (Prefixing) is denoted by a → P ie that event a must occur before operation P. The internal choice is the P (⊓Q), where the non-deterministic system chooses to execute one of the processes P or Q. The external choice is

marked P (P Q Q) and is identical to the internal option, but the choice comes from outside the system (eg: user).

M , N … ∈ Names (process names)
P , Q … ∈ Process (processes)
a , b … ∈ Σ (Event)
P ::= M (Process call)
| a⟶ P (Prefixing)
| P⊓ Q (Internal choice)
| P□Q (External choice)
| $P\|_{X\subset\Sigma}$ Q (Synchronized parallelism)
| P□Q (External choice)
| (μx •P) (Recursion)
| STOP (Stop)

Figure 1: CSP syntax

Synchronized parallelism is symbolized by b (P || (X⊆Σ) Q) where both operations are performed in parallel with the X set of concurrent events, and there is a special case of parallel execution that is (interleaving) and symbolized by (|||) where there is no synchronization (X = ∅) So both processes can be performed in any order. The stop is symbolized by STOP, which is synonymous with deadlock, ie, the current operation is completed. Recursion, denoted by μx (P), is a recursive term in which each occurrence of X in P represents an instant recursion.Operations are synchronized by sending and receiving messages [13] which are done using input (?) And output (!). Where the expression Pi? X is the procedure that receives the Pi process of the value sent through the x event, while the expression Pj! x The procedure for sending the Pj operation describes a value through the x event.

## III. PETRI NETS (PN)

The Petri Network is a visual mathematical tool for modeling and verifying system behavior and is one of the most important tools for verifying the behavior of systems with concurrent tasks (competing for system resources) as well as distributed network protocols and systems that we can not easily verify because of critical sectors ,resource sharing and the need to synchronize and coordinate the various functions of the system. [1].
The Petri network consists of places, transitions and arcs. The arc reaches a place into a transition or vice versa. Places in the Petri nets may contain a separate number of tags called tokens. The mathematical expression of the Petri network is expressed through the pentagram [8] (P; T; I; O; M) where:
P = {p1; p2 ….$p_{np}$ } is the set of np places (drawn as circles in the graphical representation);
T = {t1; t2 ….$t_{np}$} tn} is the set of nt transitions (drawn as bars);
I is the transition input relation and is represented by means of arcs directed from places to transitions;
O is the transition output relation and is represented by means of arcs directed from transitions to places;
M = {m1;m2…..$m_{np}$} is the marking. The generic entry mi is the number of tokens (drawn as black dots) in place pi in marking M. The graphical structure of a PN is a bipartite directed graph: the nodes belong to two different classes (places and transitions) and the edges (arcs) are allowed to connect only nodes of different classes (multiple arcs are possible in the definition of the I and O relations [1]).
The dynamics of a PN is obtained by moving the tokens in the places by means of the following execution rules:
- A transition is enabled in a marking M if all its input places carry at least one token.
- an enabled transition fires by removing one token per arc from each input place and adding one token per arc to each output place.
The reachability graph G = (V,E, v0) of a net N = (P, T,W,M0) is defined inductively as follows:
– v0 = M0 ∈ V .
– If M ∈ V and M $\xrightarrow{t}$ M0 then M0 ∈ V and (M, t,M0) ∈ E.
– V and E contain no other elements.
The reachability graph of a net describes the dynamic behaviour of the net.

## IV. LINEAR TEMPORAL LOGIC

Linear Temporal logic is a type of time logic in which logical formulas are defined for a set of implementation paths of the system, ie, each path in this group represents an infinite path in the reachability tree that starts from the root, so in each case in this way There is only one possible future situation.

The syntax of Linear Time Logic can be defined as follow : Let AP be a set of atomic propositions, and p ∈AP, then LTL syntax can be recursively defined as follows

$$\emptyset := P \quad | \neg \emptyset \quad | \psi \wedge \emptyset \quad | \psi \vee \emptyset | \quad F\psi \quad | G\psi \quad | X\psi \quad | \psi U \emptyset | \quad \psi R \emptyset$$

where X, U,V G, and F are temporal operators that denote “next”, “until”, “weak until (a variant of until)”, “all (states)”, and “there exists (a future state)”, respectively, for a path.

The semantics of LTL formulas can be recursively defined as follows:

$\pi$ |= p , iff p ∈ L($\pi_0$).

$$\pi \vDash \neg\psi \; iff \; \pi \nvDash \psi$$
$$\pi \vDash \psi_1 \wedge \psi_2 \; iff \; \pi \vDash \psi_1 \; and \; \pi \vDash \psi_2$$
$$\pi \vDash X\psi \quad iff \; \pi^1 \vDash \psi$$
$$\pi \vDash G\psi \; iff \; \forall i \geq 0, \pi^i \vDash \psi$$
$$\pi \vDash G\psi \; iff \; \exists i \geq 0, \pi^i \vDash \psi$$
$$\pi \vDash \psi_1 \, U \, \psi_2 \; iff \; \exists j \geq 0 \; \pi^j \vDash \psi_2 \; and \; \forall 0 \leq i < j, \quad \pi^i \vDash \psi_1$$

## V. PIPE TOOL

Platform-Independent Petri Net Editor (PIPE), an open-source tool that supports the design and analysis of Generalized Stochastic Petri-Net (GSPN) models [4].

## VI. PROPOSED MODEL

In chapter II we noted that the CSP suggests high-level processes for characterizing systems behavior. We also noted in chapter IV that the liner temporal logic suggests high-level processes to verify system behavior. While the Petri net suffers from a low level of abstraction and no high level processes to verify system behavior despite the strength of the reachability tree in predicting all possible system scenarios. In this section, we propose a solution figure 2 that process of converting a characterization from CSP to a Petri net that achieves the correct characterization of the system and its integration with the query model based on LTL, which also allows roaming in the reachability tree to verify system behavior.

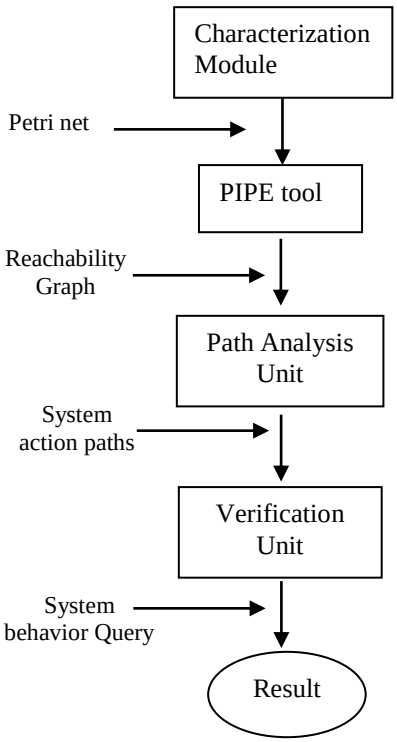


Figure 2: A diagram showing the parts of the proposed solution and the sequence of necessary processes

**a-Text conversion unit to Petri net :**
In this module, we propose a formal language as shown in Figure 3, explaining the CSP language rules through a full set of standard CSP translations that are corresponding to our proposed language.

**PROCESS** ProcessID :: ProcessList
**ProcessList** :: ParallelDef | SequenceDef | RecursionDef | DeterministicChoiceDef | NonDeterministicChoiceDef | StopDef
**ParallelDef** :: PAR { ProcessID , ProcessID }
**SequenceDef** :: SEQ { ProcessID , ProcessID }
**RecursionDef** :: REC { ProcessID , ProcessID }
**DeterministicChoiceDef** ::DC { ProcessID , ProcessID }
**NonDeterministicChoiceDef**::NDC { ProcessID , ProcessID }
**SEND** { ProcessID , ProcessID }
**RECIVE**{ ProcessID , ProcessID }
**ProcessID** :: [A-Z] [A-Z_a-z]* | ProcessList

Figure 3: syntax of Suggested language

We proposed the Petri nets corresponding after studying the structural characteristics of the Petri nets to achieve compatibility between the proposed model and the description as shown in Figure 4. Through these standard translations of the smallest possible conversion process, we can now switch from our proposed language to the Petri net by building a compiler that compiles operations according to the proposed translations in Figure 4.

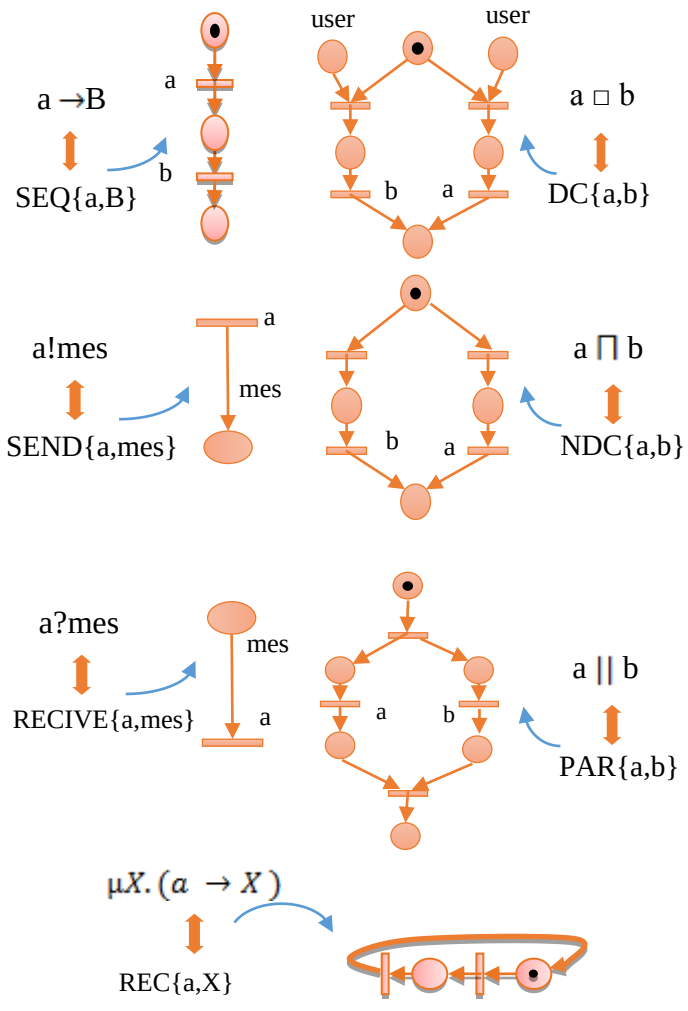


Figure 4: Proposed model for converting between CSP and Petri-nets

**b- System Path Analysis Unit:**
This unit relies on PIPE data structures for the reachability tree. Where we proceed from the initial marking and use a recursion algorithm to detect all the paths of the system, provided that we do not go over the marking twice in the same path in the same order. The algorithm begins its work by experimenting with all the paths associated with the initial marking S0 node and combining the marking that lead to it through the different paths from initial marking S0. This work is repeated for each marking M we get to. The path ends when trying to cross a marking that were previously crossed or for lack of subsequent marking.

**c- Verification Unit:**
In this unit we have built an interpreter to handle operations similar to the linear temporal logic operations for work on the reachability tree in Petri network. The interpreter analyzes these processes, the form in which we express the required conditions in the behavior of the systems, by applying them to the paths from the path analysis unit, path by path, until the discovery of a correspondence between the query and the marking in the path.

## VII. CONCLUSIONS AND FUTURE WORK

In this paper, we provide PIPE with two new modules, the first able to convert the description written according to our proposed language inspired by CSP to the Petri nets and the second module to provide petri nets with a mechanism to formulate queries about the behavior of systems represented by these networks, instead of manual simulation of all system operating scenarios Is not exhaustive and instead of the tedious manual roaming within the equal reachability tree of the Petri nets represented by the studied system. In addition to, allowing the abstraction of the characterization and verification of systems behavior using petri nets, especially for large-scale systems with competing tasks on common and highly synchronized resources.
The proposed method of characterization allows to increase abstraction in characterizing the behavior of the system and to compensate for the decrease in the low level of abstraction of the Petri network and to increase the efficiency of the query in the Petri network by applying the temporal logic on the reachability tree of petri nets .
The method of characterization and investigation of queries allows the addition of new processes suitable for different types of petri networks. By expanding the identification and verification units. Where we can extend the process of analyzing queries to include additional symbols and processing them. We are currently working to add new parameters to our tools, with the aim of working on Time Petri networks and CSP time.

## ACKNOWLEDGMENT

The research that led to this work was supported in the first part by the Damascus University Foundation.
The authors wish to thank the University administration and the College of Informatics Engineering for their assistance in providing technical support for this work.

## REFERENCES

[1] J.L. Peterson. Petri net theory and the modeling of systems. Prentice Hall, Englewood Cliffs, 1981.
[2] Llorens, M., Oliver, J., Silva, J., & Tamarit, S. (2012). Generating a Petri net from a CSP specification: A semantics-based method. Advances in Engineering Software, 50, 110-130.
[3] Leuschel, M., Currie, A., & Massart, T. (2001, March). How to make FDR spin LTL model checking of CSP by refinement. In International Symposium of Formal Methods Europe (pp. 99-118). Springer, Berlin, Heidelberg.
[4] Platform independent petri net editor download on sourceforge. https://sourceforge.net/projects/ pipe2/. Accessed: 15 August 2016.
[5] Nattira MANEERAT and Wiwat VATANAWOOD, 2016 - Translation UML activity diagram into colored Petri net with inscription. In the proceedings of 13th International Joint Conference on Computer Science and Software Engineering (JCSSE) pp 1-6.
[6] Boyle, R. (1985). Communicating sequential processes: CAR Hoare. Published by Prentice-Hall. 256pp.
[7] Jones, A. E. A. C. B., & Sanders, J. W. (2005). Communicating sequential processes. Published by Springer, Berlin, Heidelberg.335pp.
[8] Halder, A., & Venkateswarlu, A. (2006). A study of petri nets modeling analysis and simulation.Department of Aerospace Engineering Indian Institute of Technology Kharagpur, India.
[9] Tan, L., Sokolsky, O., & Lee, I. (2004, November). Specification-based testing with linear temporal logic. In Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, 2004. IRI 2004. (pp. 493-498). IEEE.
[10] V. RYBAKOV, 2008 -Linear temporal logic with until and next, logical consecutions. Annals of Pure and Applied Logic, ISSN: 0168-0072, Vol: 155, Issue: 1, Page: 32-45.