International Journal of Pure and Applied Mathematics
Volume 115 No. 3 2017, 607-619
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: 10.12732/ijpam.v115i3.14

A MATHEMATICAL LOGIC VALIDATION PROCEDURE FOR PETRI NETS MODELS

Zvi Retchkiman Königsberg

Instituto Politécnico Nacional, CIC
Mineria 17-2, Col. Escandon, Mexico D.F 11800, MEXICO

Abstract: A discrete event system, is a dynamical system whose state evolves in time by the occurrence of events at possibly irregular time intervals.. Place-transitions Petri nets, commonly called Petri nets, are a graphical and mathematical modeling tool applicable to discrete event systems in order to represent its states evolution.

This paper proposes a formal modeling and validation mathematical analysis methodology, which consists in representing the Petri net model of a discrete event system, by means of a formula in the propositional calculus logic. Then, using the concept of logic implication, and transforming this logical implication relation into a set of clauses, qualitative methods for validation are addressed.

AMS Subject Classification: 03B70, 93A30, 93D99, 93C10
Key Words: Petri net models, discrete event systems, propositional logic, validation, refutation methods

1. Introduction

A discrete event system, is a dynamical system whose state evolves in time by the occurrence of events at possibly irregular time intervals. Some examples

Received: May 29, 2017
Revised: July 19, 2017
Published: July 27, 2017

include: Manufacturing systems, Computer networks, Queuing systems, Communication systems, Business processes. Place-transitions Petri nets, commonly called Petri nets, are a graphical and mathematical modeling tool applicable to discrete event systems in order to represent its states evolution. Petri nets are known to be useful for analysing the systems properties in addition of being a paradigm for describing and studying information processing systems. Most often such Petri net models are constructed and implemented without any formal validation analysis or by studying its state space. This paper proposes a formal modeling and validation mathematical analysis methodology, which consists in representing the Petri net model of a discrete event system, by means of a formula in the propositional calculus logic. Then, using the concept of logic implication, and transforming this logical implication relation into a set of clauses, qualitative methods for validation, are addressed. The method of Putnam-Davis based for testing the unsatisfiability of a set of clauses as well as the resolution principle due to Robinson, are invoked. The paper is organized as follows. In Section 2, Petri nets are defined and a propositional calculus background summary which supports the procedure presented in the paper is given. In Section 3, the Putnam-Davis rules and the resolution principle for unsatisfiability, are recalled. In Section 3, we illustrate how the methodology is applied. Finally, the paper ends with some conclusions.

2. Petri Nets and Propositional Calculus

Notation. $N = \{0, 1, 2, ...\}$, $N_{n_0}^{+} = \{n_0, n_0 + 1, ..., n_0 + k, ...\}$, $n_0 \geq 0$.

A Petri net (PN) is a 5-tuple, $PN = \{P, T, F, W, M_0\}$ where: $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places, $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions (represented respectively by circles and bars), $F \subset (P \times T) \cup (T \times P)$ is a set of arcs, $W : F \to N_1^+$ is a weight function, M_0: $P \to N$ is the initial marking, $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$. Notice that if $W(p, t) = \alpha$ or $W(t, p) = \beta$, then, this is often represented graphically by α or β arcs from p to t or t to p (through arrowheads) each with no numeric label. Let $M(p_i)$ denote the marking (i.e., the number of tokens) at place $p_i \in P$ and let $M = [M(p_1), ..., M(p_m)]^T$ denote the marking (state) of PN. A transition $t_j \in T$ is said to be enabled if $M(p_i) \geq W(p_i, t_j)$ for all $p_i \in P$ such that $(p_i, t_j) \in F$. If a transition is enabled then, it can fire. If an enabled transition $t_j \in T$ fires then, the next marking for $p_i \in P$ is given by

$$M^{'}(p_i) = M(p_i) + W(t_j, p_i) - W(p_i, t_j). \tag{1}$$

2.1. Propositional Calculus

This sub-section presents a summary of the propositional logic theory. The reader is referred to [2] and [3] for details.

Definition 1. *(The alphabet for propositional formula) This alphabet consists of:*

1. *A countable set PS of proposition symbols called atoms: $P_0, P_1, P_2, \ldots$.*
2. *The logical connectives: $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implication), $\sim$ (not), and $\leftrightarrow$ (equivalence).*
3. *Auxiliary symbols: ((left parenthesis),) (right parenthesis).*

Definition 2. *The set of proposition formula is the smallest set of strings over the alphabet of definition 1, such that:*

1. *Every proposition symbol P_i is a proposition formula*
2. *Whenever A is a proposition formula , $\sim A$ is also*
3. *Whenever A, B are proposition formula, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$ and $(A \leftrightarrow B)$ are also*
4. *A string is a proposition formula only if it is formed by applying the rules (1),(2),(3).*

Definition 3. *By an assignment on a given set of atoms we mean a function v which maps each atom into the set $FALSE, TRUE = \{0, 1\}$, where we are identifying $FALSE$ with 0 and $TRUE$ with 1. Thus for each atom P_i we will have $v(P_i) = 0$ or $v(P_i) = 1$.*

Definition 4. *Given an assignment v on a set of atoms, we define recursively a value $\gamma^v \in \{0, 1\}$ for each formula A :*

1. *if A is an atom, then $A^v = v(A)$*
2. *If $\gamma = \sim A$ then $\gamma^v = \begin{cases} 1 & \text{if } A^v = 0 \\ 0 & \text{if } A^v = 1 \end{cases}$*
3. $(A \vee B)^v = \begin{cases} 0 & \text{if } A^v = B^v = 0 \\ 1 & \text{otherwise} \end{cases}$
4. $(A \wedge B)^v = \begin{cases} 1 & \text{if } A^v = B^v = 1 \\ 1 & \text{otherwise} \end{cases}$

5. $(A \to B)^v = \begin{cases} 0 & \text{if} \quad A^v = 1, B^v = 0 \\ 1 & \text{otherwise} \end{cases}$

6. $(A \leftrightarrow B)^v = \begin{cases} 1 & \text{if} \quad A^v = B^v \\ 0 & \text{otherwise} \end{cases}$

Definition 5. *A formula A is satisfiable if and only if there is an assignment v such that $v(A) = 1$ otherwise is said to be unsatisfiable.*

Definition 6. *A formula A is called a tautology if and only if every assignment of A satisfies it.*

Definition 7. *A formula A is a logical implication of formulas $F_1, F_2, \ldots, F_n$ if and only if for every assignment v, if $F_1, F_2, \ldots, F_n$ is true in v, A is also true in v.*

The following characterization of logical implication plays a very important role as will be shown in the rest of the paper.

Theorem 8. *Given formulas $F_1, F_2, \ldots, F_n$ and a formula A, A is a logical implication of $F_1, F_2, \ldots, F_n$ if and only if the formula $((F_1 \wedge F_2 \wedge \ldots, \wedge F_n) \to A)$ is a tautology if and only if the formula $(F_1 \wedge F_2 \wedge \ldots \wedge F_n \wedge \sim (A))$ is unsatisfiable.*

Next, Given a formula A, the following procedure transforms A into its conjunctive normal form (CNF).

(1) Eliminate $\to$ and $\leftrightarrow$, (2) Move $\sim$ inward, (3) Rename variables and (4) Pull quantifiers (details are provided in [2]).

Theorem 9. *There is an algorithm which transforms any given A formula into a formula B in CNF such that $A = B$.*

Definition 10. *A clause is a finite disjunction of zero or more literals (atoms or negation of atoms).*

We shall regard a set of literals as synonymous with a clause. A clause consisting of r literals is called an r-literal clause. A one-literal clause is called a unit clause. When a clause contains no literal, we call it the empty clause, denoted by $\square$. Since the empty clause has no literal that can be satisfied by an assignment, the empty clause is always false. The importance of transforming a formula A into its CNF results evident, thanks to the next result.

Remark 11. *Notice that given a formula A in CNF, the set of clauses S of A is the representation in terms of sets of its CNF. A set S of clauses is unsatisfiable if and only if it is false under all assignments which implies, thanks to the next result, that A the formula that it represents is unsatisfiable.*

Theorem 12. *Let S be a set of clauses that represents a formula A. Then a formula A in its CNF form is a tautology if and only if S is a tautology.*

We have seen that the problem of logical implication is reducible to the problem of satisfiability, which in turn is reducible to the problem of satisfiability of its CNF, or equivalently to the satisfiability of S. This can be used together with algorithms for unsatisfiability, Davis Putnam rules and the resolution principle, (discussed in the next section), to develop procedures for this purpose.

3. Unsatisfiability Methods

3.1. Davis Putnam Rules [2]

Davis and Putnam introduced a method for testing the unsatisfiability of a set of clauses. Their method consists of the following rules: (1) Delete all clauses from S that are tautologies. The remaining set $S^{'}$ is unsatisfiable if and only if S is, (2) If there is a unit clause L in S, obtain $S^{'}$ from S by deleting those clauses in S containing L. If $S^{'}$ is empty then, S is satisfiable, otherwise obtain a set $S^{''}$ by deleting $\sim (L)$ from $S^{'}$. $S^{''}$ is unsatisfiable if and only if S is, (3) A literal L in a clause of S is said to be pure in S if and only if the literal $\sim (L)$ does not appear in any clause in S. If a literal L is pure in S, delete all the clauses containing L. The remaining set $S^{'}$ is unsatisfiable if and only if S is, (4) If the set S can be written as: $(A_1 \vee L) \wedge (A_2 \vee L) \ldots (A_m \vee L) \wedge (B_1 \vee \sim L) \wedge (B_2 \vee \sim L) \ldots (B_m \vee \sim L) \wedge R$ where A_i, B_i and R are free of L and $\sim L$ then, obtain the sets $S_1 = A_1 \wedge A_2 \ldots A_m \wedge R$ and $S_2 = B_1 \wedge B_2 \ldots B_m \wedge R$. S is unsatisfiable if and only if both, $S_1 \cup S_2$ are.

3.2. The Resolution Principle [1]

We shall next introduce the resolution principle due to Robinson, which is also called the Davis and Putnam procedure in the literature [2]. It can be applied directly to any set S of clauses to test the unsatisfiability of S. Resolution is a sound and complete algorithm i.e., a formula in clausal form is unsatisfiable if and only if the algorithm reports that it is unsatisfiable. Therefore it provides a consistent methodology free of contradictions.

Definition 13. *Let C_1 and C_2 be two clauses (called parent clauses) with no variables in common. Let L_1 and L_2 be two literals in C_1 and C_2, respectively and consider L_1 and $\sim (L_2)$, then the clause $(C_1 - L_1) \cup (C_2 - L_2)$*

is called a binary resolvent of C_1 and C_2. The literals L_1 and L_2 are called the literals resolved upon.

Definition 14. *A resolvent of (parent) clauses C_1 and C_2 is one of the following binary resolvents: (1) a binary resolvent of C_1 and C_2, (2) a binary resolvent of C_1 and a factor of C_2, (3) a binary resolvent of a factor of C_1 and C_2, (4) a binary resolvent of a factor of C_1 and a factor of C_2.*

Definition 15. *Given a set S of clauses, a deduction of C from S is a finite sequence of clauses $C_1, C_2, \ldots, C_n$ such that each C_i, either is a clause in S or a resolvent of clauses preceding C_i,, and $C_k = C$. A deduction of $\square$ from S is called a refutation, or a proof of S.*

The main result of this sub-section, the soundness and completeness theorem for the resolution procedure, is next presented.

Theorem 16. *A set S of clauses is unsatisfiable if and only if there is a deduction of the empty clause $\square$ from S.*

4. Validation of Petri Net Models

In this section two examples are provided, the machine shop and a more computational challenging example given by the multirobotic system.

4.1. The Machine Shop System Modeling Problem

As an example, consider a simple machine shop whose Petri net model is depicted in fig.1.

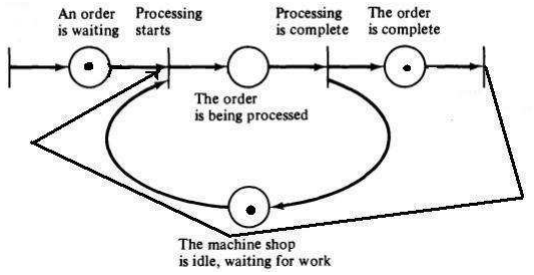

Figure 1. Machine Shop System

The machine shop waits until an order appears and then machines the ordered part and sends it out for delivery. The machine shop behavior as described

by the Petri net model is as follows: (1) States: $OWBP$: An order is waiting to being processed, OBP: An order is being processed, MI: The machine is idle, OC: the order is complete; (2) Rules of Inference: (a) if $(OWBP)$ and (MI) and (OC) then not(OBP), (b) if (OBP) then not(OC) and not(MI), (c) if $(OWBP)$ and not(MI) then (OBP), (d) if not(OBP) then (MI) and (OC) and $(OWBP)$.

Remark 17. *The main idea consists of: the Petri net model description is expressed by a propositional formula, some validation question is expressed as an additional formula. The question is assumed to be a logical implication of the formula (see theorem 13). Then, transforming this logical implication relation into a set of clauses by using the techniques given in Section 2, its validity can be checked.*

The formula that models the machine shop turns out to be:

$$[(OWBP) \wedge (MI) \wedge (OC) \rightarrow \sim (OBP)] \wedge [(OBP) \rightarrow \sim (OC) \wedge \sim (MI)] \wedge [(OWBP) \wedge \sim (MI) \rightarrow (OBP)] \wedge [\sim (OBP) \rightarrow (MI) \wedge (OC) \wedge (OWBP)]. \quad (2)$$

We are interested in the following validations:

(S1) Claim: $OBP \rightarrow \sim (MI)$.
The set of clauses is given by:

$$S = \{(\sim (OWBP) \vee \sim (MI) \vee \sim (OC) \vee \sim (OBP)), (\sim (OBP) \vee \sim (OC)), (\sim (OBP) \vee \sim (MI)), (\sim (OWBP) \vee (MI) \vee (OBP)), ((OBP) \vee (OC)), ((OBP) \vee (MI)), ((OBP) \vee (OWBP)), (OBP), (MI)\}.$$

Then a resolution refutation proof is as follows:
(a) $(OBP)(\sim (OBP) \vee \sim (MI)) \rightarrow \sim (MI), (\sim MI)(MI) \rightarrow \square$.

(S2) Claim: $\sim (MI) \rightarrow OBP$.
The set of clauses is given by:

$$S = \{(\sim (OWBP) \vee \sim (MI) \vee \sim (OC) \vee \sim (OBP)), (\sim (OBP) \vee \sim (OC)), (\sim (OBP) \vee \sim (MI)), (\sim (OWBP) \vee (MI) \vee (OBP)), ((OBP) \vee (OC)), ((OBP) \vee (MI)), ((OBP) \vee (OWBP)), (\sim OBP), (\sim (MI))\}.$$

Then a resolution refutation proof is as follows:

(a) $((\sim (MI))((OBP) \vee (MI)) \rightarrow OBP, (OBP)(\sim OBP) \rightarrow \square$.

(S3) Claim: $OC \rightarrow MI$.
The set of clauses is given by:

$$S = \{(\sim (OWBP) \vee \sim (MI) \vee \sim (OC) \vee \sim (OBP)), (\sim (OBP) \vee \sim (OC)), (\sim (OBP) \vee \sim (MI)), (\sim (OWBP) \vee (MI) \vee (OBP)), ((OBP) \vee (OC)), ((OBP) \vee (MI)), ((OBP) \vee (OWBP)), (OC), (\sim (MI))\}.$$

Then a resolution refutation proof is as follows:
(a) $(OC)(\sim (OBP) \vee \sim (OC)) \rightarrow \sim (OBP)$.
(b) $(\sim (MI))((OBP) \vee (MI)) \rightarrow OBP$.
Then, from (a) and (b) we conclude $\square$.

(S4) Claim: $\sim (OC) \rightarrow (OBP \wedge \sim (MI))$
The set of clauses is given by:

$$S = \{(\sim (OWBP) \vee \sim (MI) \vee \sim (OC) \vee \sim (OBP)), (\sim (OBP) \vee \sim (OC)), (\sim (OBP) \vee \sim (MI)), (\sim (OWBP) \vee (MI) \vee (OBP)), ((OBP) \vee (OC)), ((OBP) \vee (MI)), ((OBP) \vee (OWBP)), (\sim OC), (MI \vee \sim (OBP))\}.$$

Then a resolution refutation proof is as follows:
(a) $(MI \vee \sim (OBP))(\sim (OBP) \vee \sim (MI)) \rightarrow \sim (OBP)$, $(\sim (OBP))((OBP) \vee (OC)) \rightarrow OC, (OC)(\sim OC) \rightarrow \square$.

(S4) Finally, we claim that : $OBP \rightarrow OWBP$ and $OBP \rightarrow (\sim OWBP)$, are both satisfiable.

Setting $OBP = OWBP = 1$ and $MI = OC = 0$ the first implication as well as the formula that models the machine shop are satisfiable. The second implication as well as the formula that models the machine shop are also satisfiable assigning: $OBP = 1$ and $OWBP = MI = OC = 0$. Both conditions are consistent with the Petri net model.

4.2. The Multirobotic Modeling System Problem

A flexible manufacturing system is an efficient production line with versatile machines, an automatic transport system and a sophisticated decision making

system. Flexible manufacturing systems can be formed by subsystems that work concurrently and, therefore are suitable to be modeled by Petri nets.

In this Subsection, the philosophy presented in remark 17 is applied to a multirobotic system, (a particular example of a flexible manufacturing system), in order to validate its Petri net model. The multirobotic system consists of two robot arms which perform pick and place operations accessing a common workspace at times to obtain or transfer parts. In order to avoid collision, it is assumed that only one robot can access the workspace at a time. In addition, it is assumed that the common workspace has a buffer with a limited space for products. the process represents the operation of the two robots serving two different machining tools, with one robot arm transferring products from one machining tool to the buffer, and the other robot arm transferring semiproducts from the buffer to the other machining tool. The Petri net which represents the system is shown in fig. 2.

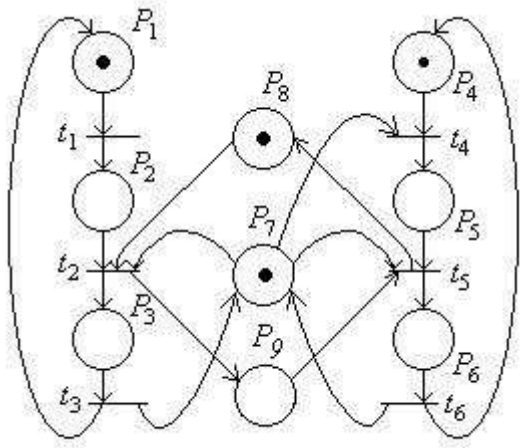

Figure 2. Multirobotic System

Where the interpretation of places and transitions is as follows:

Place	Interpretation
$P_1(P_4)$	Robot $\mathrm{R}_1(\mathrm{R}_2)$ performs tasks outside the common workspace
$P_2(P_5)$	Robot $\mathrm{R}_1(\mathrm{R}_2)$ waits for access to the common workspace
$P_3(P_6)$	Robot $\mathrm{R}_1(\mathrm{R}_2)$ performs in the common workspace
P_7	mutual exclusion
$P_8(P_9)$	number of empty (full) positions in buffer
Transition	Interpretation
$t_1(t_4)$	Robot $\mathrm{R}_1(\mathrm{R}_2)$ requests access to the common workspace
$t_2(t_5)$	Robot $\mathrm{R}_1(\mathrm{R}_2)$ enters the common workspace
$t_3(t_6)$	Robot $\mathrm{R}_1(\mathrm{R}_2)$ leaves the common workspace

The multirobotic system behavior as described by the Petri net model is as follows: (1) States ($i = 1, 2$): R_iOWP Robot i performs tasks outside the common workspace, R_iWAWP Robot i waits for access to the common workspace, R_iPWP Robot i performs in the working place, ME mutual exclusion, PB_i buffers; (2) Rules of Inference: (a) and (b) ($i = 1, 2$) $R_iOWP \to R_iWAWP$, (c) $R_1WAWP \wedge PB_1 \wedge ME \to R_1PWP \wedge PB_2$, (d) $R_2WAWP \wedge PB_2 \wedge ME \to R_2PWP \wedge PB_1$, (e) and (f) ($i = 1, 2$) $R_iPWP \to ME \wedge R_iOWP$, (g) and (h) $PB_1 \leftrightarrow \sim (PB_2)$.

The formula that models the multirobotic system turns out to be:

$$\begin{aligned}&[(R_1OWP) \to (R_1WAWP)] \wedge [(R_2OWP) \to (R_2WAWP)] \\ &\wedge [(R_1WAWP) \wedge (PB_1) \wedge (ME) \to (R_1PWP) \wedge (PB_2)] \wedge [(R_2WAWP) \\ &\qquad \wedge (PB_2) \wedge (ME) \to (R_2PWP) \wedge (PB_1)] \wedge [(R_1PWP) \\ &\qquad \to (ME) \wedge (R_1OWP)] \wedge [(R_2PWP) \to (ME) \wedge (R_2OWP)] \\ &\qquad\qquad \wedge [PB_1 \to \sim (PB_2)] \wedge [\sim (PB_2) \to PB_1]. \quad (3)\end{aligned}$$

We are interested in the following validations:

(S1) Claim: $R_1OWP \wedge PB_1 \wedge ME \to R_1PWP$.

The set of clauses is given by:

$$\begin{aligned}&S = \\ &\{(\sim (R_1OWP) \vee (R_1WAWP)), (\sim (R_2OWP) \vee (R_2WAWP)), (\sim (R_1WAWP) \\ &\qquad \vee \sim (PB_1) \vee \sim (ME) \vee (R_1PWP)), (\sim (R_1WAWP) \vee \sim (PB_1) \\ &\qquad \vee \sim (ME) \vee (PB_2)), (\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \\ &\qquad \vee (R_2PWP)), (\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \vee (PB_1)), \\ &(\sim (R_1PWP) \vee (ME)), (\sim (R_1PWP) \vee (R_1OWP)), (\sim (R_2PWP) \vee (ME)), \\ &\qquad (\sim (R_2PWP) \vee (R_2OWP)), (PB_1 \vee PB_2), (\sim (PB_1) \vee \sim (PB_2)),\end{aligned}$$

$$(R_1OWP), (PB_1), (ME), (\sim (R_1PWP))\}.$$

Then a resolution refutation proof is as follows:

(a) $(R_1OWP)(\sim (R_1OWP) \vee (R_1WAWP)) \rightarrow ((R_1WAWP))$;

(b) $(\sim (R_1WAWP) \vee \sim (PB_1) \vee \sim (ME) \vee (R_1PWP))(((R_1WAWP))(PB_1)(ME) \rightarrow (R_1PWP)$;

(c) $(R_1PWP)(\sim (R_1PWP)) \rightarrow \square$.

(S2) Claim: $R_1WAWP \wedge PB_1 \wedge ME \rightarrow PB_2$.
The set of clauses is given by:

$$\begin{aligned} S = \{&(\sim (R_1OWP) \vee (R_1WAWP)), (\sim (R_2OWP) \vee (R_2WAWP)), \\ &(\sim (R_1WAWP) \vee \sim (PB_1) \vee \sim (ME) \vee (R_1PWP)), (\sim (R_1WAWP) \\ &\vee \sim (PB_1) \vee \sim (ME) \vee (PB_2)), (\sim (R_2WAWP) \vee \sim (PB_2) \\ &\vee \sim (ME) \vee (R_2PWP)), (\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \\ &\vee (PB_1)), (\sim (R_1PWP) \vee (ME)), (\sim (R_1PWP) \vee (R_1OWP)), \\ &(\sim (R_2PWP) \vee (ME)), (\sim (R_2PWP) \vee (R_2OWP)), (PB_1 \vee PB_2), \\ &(\sim (PB_1) \vee \sim (PB_2)), (R_1WAWP), (PB_1), (ME), (\sim (PB_2))\}. \end{aligned}$$

Then a resolution refutation proof is as follows:

$$\begin{aligned} &(\sim (R_1WAWP) \vee \sim (PB_1) \vee \sim (ME) \vee (PB_2))((R_1WAWP)(PB_1)(ME) \\ &\qquad \rightarrow (PB_2)), (PB_2))(\sim (PB_2)) \rightarrow \square. \end{aligned}$$

(S3) Next, we validate the mutual exclusive property of the Petri net model.

Claim. $R_1WAWP \wedge R_2WAWP \wedge ME \rightarrow (R_1PWP \vee R_2PWP) \wedge \sim (R_1PWP \wedge R_2PWP)$.

The set of clauses is given by:

$$\begin{aligned} S = \{&(\sim (R_1OWP) \vee (R_1WAWP)), (\sim (R_2OWP) \vee (R_2WAWP)), \\ &(\sim (R_1WAWP) \vee \sim (PB_1) \vee \sim (ME) \vee (R_1PWP)), (\sim (R_1WAWP) \\ &\vee \sim (PB_1) \vee \sim (ME) \vee (PB_2)), (\sim (R_2WAWP) \vee \sim (PB_2) \\ &\vee \sim (ME) \vee (R_2PWP)), (\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \\ &\vee (PB_1)), (\sim (R_1PWP) \vee (ME)), (\sim (R_1PWP) \vee (R_1OWP)), \\ &(\sim (R_2PWP) \vee (ME)), (\sim (R_2PWP) \vee (R_2OWP)), (PB_1 \end{aligned}$$

$$\vee PB_2), (\sim (PB_1) \vee \sim (PB_2)), (R_1WAWP), (R_2WAWP), (ME), \\ (\sim (R_1PWP)), (\sim (R_2PWP))\}.$$

Then a resolution refutation proof is as follows:

(a) $(\sim (R_1WAWP) \vee \sim (PB_1) \vee \sim (ME) \vee (R_1PWP))(R_1WAWP)(ME)(\sim (R_1PWP)) \rightarrow (\sim (PB_1))$;

(b) $(\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \vee (R_2PWP))(R_2WAWP)(ME)(\sim (R_2PWP)) \rightarrow (\sim (PB_2))$;

(c) $(\sim (PB_1))(PB_1) \vee (PB_2)) \rightarrow (PB_2)$;

(d) Then, from (b) and (c) we get $\square$.

(S4) Claim: $R_1PWP \wedge PB_2 \wedge R_2WAWP \rightarrow R_2PWP \wedge R_1OWP \wedge ME \wedge PB_1$.

The set of clauses is given by:

$$S = \{(\sim (R_1OWP) \vee (R_1WAWP)), (\sim (R_2OWP) \vee (R_2WAWP)), \\ (\sim (R_1WAWP) \vee \sim (PB_1) \vee \sim (ME) \vee (R_1PWP)), \\ (\sim (R_1WAWP) \vee \sim (PB_1) \vee \sim (ME) \vee (PB_2)), \\ (\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \vee (R_2PWP)), \\ (\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \vee (PB_1)), \\ (\sim (R_1PWP) \vee (ME)), (\sim (R_1PWP) \vee (R_1OWP)), (\sim (R_2PWP) \vee (ME)), \\ (\sim (R_2PWP) \vee (R_2OWP)), (PB_1 \vee PB_2), (\sim (PB_1) \vee \sim (PB_2)), \\ (R_1PWP), (PB_2), (R_2WAWP), (\sim (R_2PWP) \vee \sim (R_1OWP) \\ \vee \sim (ME) \vee \sim (PB_1))\}.$$

Then a resolution refutation proof is as follows:

(a) $(R_1PWP)(\sim (R_1PWP) \vee (ME)) \rightarrow (ME)$;

(b) $(R_1PWP)(\sim (R_1PWP) \vee (R_1OWP)) \rightarrow (R_1OWP)$;

(c) $(\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \vee (PB_1))(R_2WAWP)(PB_2)(ME) \rightarrow (PB_1)$;

(d) $(\sim (R_2PWP) \vee \sim (R_1OWP) \vee \sim (ME) \vee \sim (PB_1))(ME)(R_1OWP)(PB_1) \rightarrow (\sim (R_2PWP))$;

(e) $(\sim (R_2WAWP) \vee \sim (PB_2) \vee \sim (ME) \vee (R_2PWP))(R_2WAWP)(PB_2)(\sim (R_2PWP)) \rightarrow (\sim (ME))$;

Then, from (a) and (e) we get $\square$.

5. Conclusions

The main contribution of the paper consists in proposing a formal reasoning deductive methodology based on the propositional calculus logic theory for validation of Petri net models of discrete event systems without having to construct its state space.

References

[1] J.A. Robinson, A machine-oriented logic based on the resolution principle, *Journal of the ACM*, **12**, No. 1 (1965), 23-41.

[2] M. Davis, R. Sigal and E. Weyuker, *Computability, Complexity, and Languages, Fundamentals of Theoretical Computer Science*, Academic Press, 1983.

[3] J. H. Gallier, *Logic For Computer Science Automatic Theorem Proving*, Dover, 2015.