International Research Journal of Computer Science (IRJCS)
Issue 3, Volume 2 (March 2015)
ISSN: 2393-9842
www.irjcs.com

Comparative Study between Higher-Order Petri Nets and Higher-Order Nets

Ali M. Meligy Hani M. Ibrahim Amal M. Aqlan
Department of Mathematics, Faculty of Science, Menoufia University

Abstract— This paper differentiates between two classes of Petri nets which are: Higher-Order Petri Nets and Higher-Order Nets. Each class offers different features from another class. First, we use the example of vending machine in this study, and then modeled using HOPN. Finally, two classes are compared using a set of concepts (Definition, Structure, Firing Rule, Example and First Paper) for illustration.

Keywords— Petri Nets, Higher-Order Petri Nets, Higher-Order Nets, Vending Machine.

I. INTRODUCTION

Petri nets were introduced in 1962 by Dr. Carl Adam Petri. Petri nets are a powerful modeling formalism in computer science, system engineering and many other disciplines. Petri nets combine a well-defined mathematical theory with a graphical representation of the dynamic behavior of systems. The theoretic aspect of Petri nets allow precise modeling and analysis of system behavior, while the graphical representation of Petri nets enable visualization of the modeled system state changes. This combination is the main reason for the great success of Petri nets. Consequently, Petri nets have been used to model various kinds of dynamic event-driven systems like computers networks, communication systems, manufacturing plants, command and control systems, real-time computing systems, logistic networks, and workflows to mention only a few important examples. This wide spectrum of applications is accompanied by wide spectrum different aspects which have been considered in the research on Petri nets [1].

The major restriction of the PN model described so far is that large nets are usually needed to describe systems of a medium complexity. Therefore, it is hard to compute net invariants and the Coverability tree usually suffers from state explosion. In the PN model there is no formal treatment or clear identification of individuals (i.e., distinguishable system entities) and their properties and relations. Explicit structuring mechanisms (e.g., composition operators) are not included in the formalism, while conditions and operations are modeled only by the enabling and firing rule. In addition, there is a lack of a formal and general transformation procedure to a programming language [2].

For all these reasons, more general approaches have been proposed. These are known as Higher-Order Nets (HONs) (or high-level Petri nets such as colored Petri net, timed Petri net, stochastic Petri net, etc.). Such models often provide more concise and manageable system representation and can be used to explicitly model data/control flow, specific firing conditions/actions, various system resources etc.

The basic idea of Higher-Order Net is the usage of individual tokens instead of elementary ones. In Colored Petri Nets each token is of a certain type, and each arc of the net is inscribed with a type. A transition can be enabled in a marking only if the tokens in the precondition set are of the types the arcs require [3]. Higher-Order Nets allow for a formal treatment of individual (distinguishable) system entities. In addition, they provide manageable and concise system representation, as well as explicit modeling of specific system conditions and actions.

The heuristic introduction of higher-order synaptic weights in neural networks has enabled us to extend the concept of an arc in PN to a general sense that there exist higher-order arcs in *PN*. This new class of Petri nets called Higher-Order Petri nets (*HOPN*). Concurrency in classical Petri net can occur between transitions in the same graph. In the *HOPN* the definition of firing differs from the corresponding one in *PN*. The enabled transitions in *HOPN* can have more enabled arcs. This means that there is no firing for two or more enabled arcs of one transition simultaneously. In this case, the more enabled arcs of the same transition are said to fire concurrently, but not in arcs phase (since there is no concurrency between two arcs of the same transition). This can be done by decomposing all transitions.

The rest of this paper is organized as follows. Section 2, 3 and 4 introduce review of Petri nets, Higher-Order Nets and Higher-Order Petri Nets, respectively. Section 5 discusses modeling of a vending machine using PN, HPN and HOPN and comparing tow classes (HPN and HOPN) according to a set of concepts (Definition, Structure, Firing Rule, Example and First Paper). Finally, the paper concludes with a short summary.

II. PETRI NETS

In Petri nets we can distinguish two basic elements, one that embodies the *passive nature* of the structure and the *active part* of the system to be modeled. *Passive elements* are represented by circles or ellipses which model normally conditions, states, resources or objects, while the *active elements* known as transitions, denoted by rectangles or boxes, models events, actions or activities that changes the values of the supposed conditions or objects. Petri nets also comprise *tokens*, which represent the value of a condition or object, drawn by black dots inside places. *Arcs* are used to describe interactions between the *active* and *passive* elements. It is only possible to connect nodes (transition/ place) of different type, the so called bipartite graphs. An *arc* may has attached a natural number written near the corresponding *arc*, the *arc* weight, which specifies that a *transition* is enabled only when the input *places* has at least as many *tokens* as given by the *arc* weight. When an enabled *transition* fires, an equal number of *tokens* given by the *arc* weight in the input *place* is destroyed. Firing a *transition* will also create an amount of *tokens* specified by the associated *arc* weight in the output *places* [4].

The *token* distribution, called Petri net *marking* changes with the firing of enabled transitions in a net. A firing sequence will determine a sequence of new marking distribution. If a Petri net has *m places*, their *marking* is represented by a (mx1) vector M, where each element corresponds to the number of the *tokens* in each corresponding *place*.

To describe the initial state of a model, a set of *tokens* is associated to the *places*, known as initial *marking* M_0.

Definition 1 (*Classical Petri Net*): A Petri net is defined as a 5-tuple, $PN = (P, T, F, W, M_0)$ [5], where:

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, with $P \cup T \neq 0$ and $P \cap T = \phi$;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs;
- $W: F \rightarrow N$ is a weight function, where M represents the set of non-negative integers;
- $M_0: P \rightarrow N$ is the initial marking.

Definition 2 (*Enabled Transition*): a transition t is said to be enabled if each input place p of t contains not less than n, number of tokens, where n, is equal to the weight of the directed arc connecting p to t.

Definition 3 (*Firing Rule*): (*i*) whether an enabled transition t would fire or not is dependent upon an additional condition. (*ii*) The firing of an enabled transition t removes n, number of tokens from each input

III. HIGHER-ORDER NETS (HONs)

A HON is a Petri net in which places are associated with different types of tokens and the tokens in each place are associated with a data structure. The transitions of a HON are associated with relationships that determine the enabling and firing of the transitions and determine the tokens produced by the transition firing [3].

In Higher-Order nets, an arc may be annotated by a variable or constant (or combination) of the same type as the arc's place. The evaluation of the arc expression is thus an element of the place's type. By convention, if an arc expression evaluates to an element of the place's type, this is interpreted as the corresponding singleton multiset over the place's type. This is necessary as an arc annotation when evaluated must be a multiset over the associated place's type. A similar convention applies when an arc expression evaluates to a set, which may be a subset of the associated place's type. This is interpreted as a multiset over the place's type, with corresponding multiplicities of zero and one [6].

Definition 4 (Higher-Order nets): A HON is a structure HON = (P, T, D, Type, Pre, Post, M_0) where**:**

- P is a finite set of elements called Places.
- T is a finite set of elements called Transitions disjoint from P ($P \cap T = \phi$).
- D is a non-empty finite set of non-empty domains where each element of D is called a type.
- $Type: P \cup T \rightarrow D$ is a function used to assign types to places and to determine transition modes.
- $Pre, Post: TRANS \rightarrow \mu PLACE$ are the *pre* and *post* mappings with
  $$TRANS = \{(t, m) \mid t \in T, m \in Type(t)\}$$
  $$PLACE = \{(p, g) \mid p \in P, g \in Type(p)\}$$
- $M_0 \in \mu PLACE$ is a multiset called the initial marking of the net.

Definition 5 (Marking of HON): A Marking of the *HON* is a multiset, $M \in \mu PLACE$.

Definition 6 (Enabling of Transition Modes):

- A transition mode, $t_r \in TRANS$, is enabled at a marking M iff
  $$Pre(t_r) \leq M$$
- A finite multiset of transition modes, $T_\mu \in \mu TRANS$, is enabled at a marking M iff
  $$Pre(T_\mu) \leq M$$

where the linear extension of *Pre* is given by

$$Pre(T_\mu) = \sum_{t_r \in TRANS} T(t_r)\, pre(t_r)$$

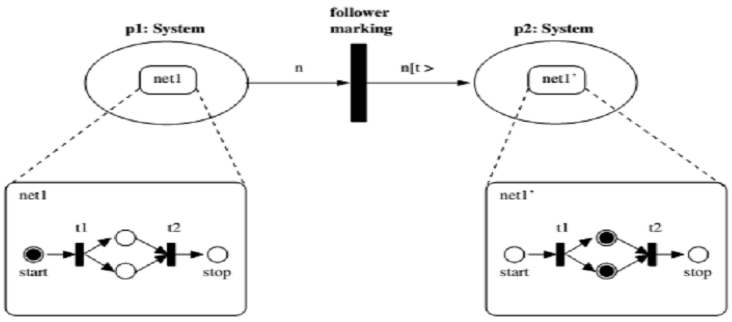

Fig.1: A Petri Net as a token [7]

All transition modes in T are said to be concurrently enabled if T_μ is enabled, i.e. there are enough tokens on the input places to satisfy the linear combination of the pre maps for each transition mode in T_μ.

Definition 7 (Transition Rule): Given that a finite multiset of transition modes, T_μ, is enabled at a marking M, then a step may occur resulting in a new marking M' given by

$$M_0 = M + Pre(T\mu) + Post(T\mu)$$

where the linear extension of *Post* is used.

A step is denoted by $M\,[T\mu\rangle\, M'$ or $M \xrightarrow{\mu} M'$.

For example 1: In Fig.1, we use the specification of Petri nets instead that of graphs leading to the notion of Higher-Order nets with Petri nets as tokens. We assume that the initial marking of the Higher-Order net in Fig.1 consists of the Petri net net_1. In net_1 the transition t_1 is enabled since the current marking of net_1 contains the token start. By the firing of the transition follower marking, on the one hand, the Petri net net_1 is consumed from place p_1. On the other hand, the marking of the Petri net net_1 is changed in the sense that the follower marking is computed and the resulting Petri net net_1' is added to the place p_2 [7].

IV. HIGHER-ORDER PETRI NETS (HOPNs)

Definition 8 (Higher-Order Petri Net) A Higher-Order Petri Net is defined as a 5-tuple, $HOPN = (P, T, F, W, M_0)$, where [5]:

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, $P \cup T = 0$ and $P \cap T = \phi$;
- $F \subseteq (P \times T) \cup (P^2 \times T) \cup \ldots \cup (P^m \times T) \cup (T \cup P)$ is a set of arcs. If $f \in (P^i \times T)$, then f is called the *ith-order* arc;
- W: $F \rightarrow N$ is a weight function, where N represents the set of non-negative integers;
- M_0: $P \rightarrow N$ denotes the initial token distribution, called the initial marking.

The major differences between an HOPN and the conventional Petri nets are the definitions of the arc and the weight. In HOPN, we denote $f^{(j)}_{\{r(i)\}_k}$, *where* $f^{(j)}_{\{r(i)\}_k} \in (P^k \times T)$, $k = 1, 2, \cdots, m$, as the kth-order input arc of transition j from places $p_{r(1)}, p_{r(2)}, \cdots, p_{r(k)}$, and its corresponding weight as $w^{(j)}_{\{r(i)\}_k}$, where $\{r(i)\}_k \equiv \{r(1), r(2), \cdots, r(k)\} \subseteq \{1, 2, \cdots, m\}$ *and* $r(1) < r(2) < \cdots < r(k)$. We also denote arc $f_{i,j}$, $f_{i,j} \in (P \bigcup T)$, as the output arc from transition i to place j and its corresponding weight as v_{ij}.

Definition 9 (Firing Rule of HOPN): An enabled transition t_j may or may not fire. When t_j fires then one of its enabled arcs fires. If the arc $f^{(j)}_{\{r(i)\}_k}$ fires then

$Q'(p_{\{r(i)\}_k}) = Q(p_{\{r(i)\}_k}) - W^{(j)}_{\{r(i)\}_k}, \forall p_{\{r(i)\}_k} \in p_{IN}(t_j) \wedge Q'(p_{out\,1}(t_j)) = Q(p_{out\,1}(t_j)) + W(p_{out\,1}, t_j),$
$\forall p_{out\,1}(t_j) \in p_{OUT}(t_j), \forall p_{\{r(i)\}_k} \in p_{IN}(t_j)$

where $Q'(p_{\{r(i)\}_k})$ is the new number of tokens in input places $p_{r(1)}, p_{r(2)}, \cdots, p_{r(k)}$ connected on the *kth-order* input arc $f^{(j)}_{\{r(i)\}_k}$ of transition t_j.

This means that when the transition t_j fires then the number of tokens in its input places and its output places are changed. The number of tokens in each of the input places $p_{\{r(i)\}_k}$ related to the fired arc is reduced by the number that is equal to the weight assigned to the fired arc (the arc that fires the transition) from $p_{\{r(i)\}_k}$ to t_j. The number of tokens in each of the output places is increased by the number that is equal to the weight of the corresponding output arcs from the transition t_j.

The number of tokens in each of the input places related to the fired arc, *p*, is reduced by the number that is equal to the weights assigned to the fired arc from p *to t*. And the number of tokens in each of its output places increases by the number that is equal to the weights of the outgoing arc from the transition t_j.

Definition 10 (Conversion Procedure): This procedure consists of two phases [8]:

First phase: we must ensure the validity of all of the following conditions:

- ***First:*** If there exist at least two of transitions that have the same output place(s).
- ***Second***: If there exist one of the previous transitions that have at least two of the incoming arcs.
- ***Third***: If all previous transitions have the same type (timed or immediate transitions).

Second phase: we apply the following steps:

- For each transition contain at least two of incoming arcs. An arcs become a single arc, as in the follow figure:

Fig.2: The process of integrating arcs [8]

- Merge all transitions to a single transition, as in the follow figure:

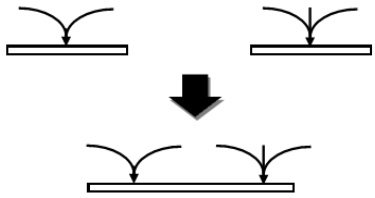

Fig.3: The process of integrating transitions [8]

An example 2: Fig.4, illustrates the definitions and the firing rule of *HOPN*. In this example p_1, p_2 and p_3 are the input places of transition t_1, and p_4, p_5 are the output places, and $w^{(1)}_{2,3}$ represents the weight of the *second-order* arc $f^{(1)}_{2,3}$ from places p_2, p_3 to transition t_1. The remaining arcs are conventional. Except $w^{(1)}_2 = 2$, all other weights are equal to 1 [5].

Fig.4 (b) shows the token distribution after the transition t_1 fires (which, is equivalent to having the arc fired). The arc is not enabled before firing, but and are enabled.

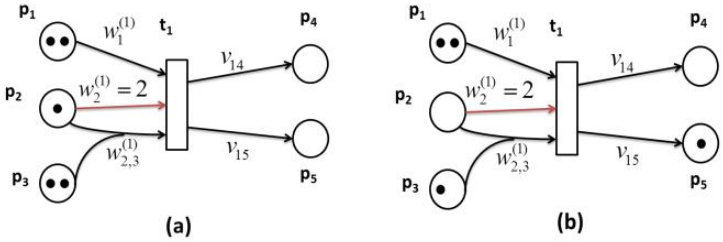

Fig.4: An example of illustration of HOPN. (a) The initial distribution of tokens. (b) The distribution state of tokens after t1, fired [5]

V. COMPARATIVE PRESENTATION

A Vending Machine problem is considered for the comparative study of HOPN and HON through modeling. A simple vending machine that operates by coin insertion is considered. The vending machine's main steps summarized are: i) coin insertion, ii) select and dispense item and iii) refill item [9]. The initial net is the elementary net. This is shown in Fig.5. This structure is used as the starting point for constructing the basic Petri net, Higher-Order nets and Higher-Order Petri net. Also Fig.5 presents the restricted behavior of the vending machine. The Petri net is very simple. Places can hold only 1 token and arcs remove only one token. It is a 1-safe Petri net. This structure depicts simple behavior and is easily constructed and understood.

A. Petri Net modeling

A Petri net is built from the elementary net as in Fig.6. Places can contain more than 1 token. Petri net contains more reasoning about the system. The execution logic is more complex than the EN system. PN structure allows for the possible selection of items using conflicting transitions (t_4, t_5, t_6, t_7). The refilling of products has been separated from the main structure (t_8, t_9, t_{10}, t_{11}). A coin counter p_{12} that allows maximum insertion of 4 coins is introduced. When a coin is inserted it is possible to accept it or reject it. Arc weights can be used to decide which items to dispense. In this example, for product A, 3 coins are necessary, so dispense product A is enabled if there are 3 tokens in coin accepted place. When an item is dispensed an item is removed and a refill takes place automatically so there are always 4 items buffered to be dispensed for each product available. Tables I, II list the various places, transitions and the meaning associated with each of them.

TABLE I
LIST OF TRANSITIONS OF PN

TRANSITION	MEANING
t_1	Insert Coin
t_2	Accept Coin
t_3	Reject Coin
t_4	Dispense Product A
t_5	Dispense Product B
t_6	Dispense Product C
t_7	Dispense Product D
t_8	Refill Product A
t_9	Refill Product B
t_{10}	Refill Product C
t_{11}	Refill Product D

TABLE II
LIST OF PLACES OF PN

PLACE	MEANING
p_1	Machine Ready
p_2	Coin Inserted
p_3	Coin Accepted
p_4	Item Dispensed A
p_5	Items A
p_6	Item Dispensed B
p_7	Items B
p_8	Item Dispensed C
p_9	Items C
p_{10}	Item Dispensed D
p_{11}	Items D
p_{12}	Coin Counter

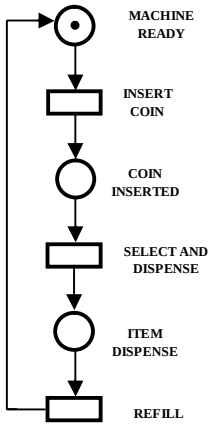

Fig.5: A Vending Machine Elementary Net [9]

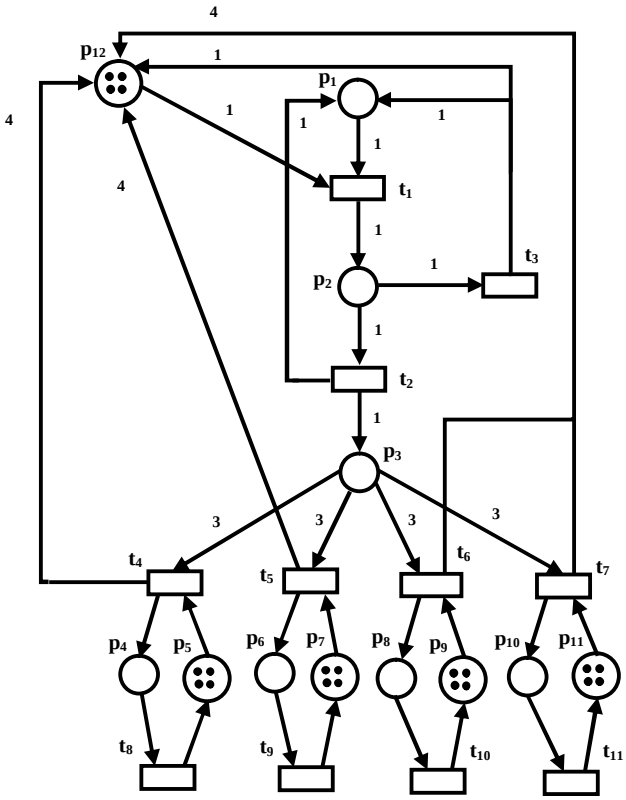

Fig.6: Modeling a Vending Machine using Petri Net [9]

B. Higher-Order Nets modeling

The Higher-Order net (Colored Petri Net) is constructed from the Petri net as in Fig.7. Token types are based on sets. Some token types are coin, change, amount and items. Coins are inserted into the vending machine using a random function to obtain the value. Product selection is done using a function called check. Once a product is selected the change is computed and given to the customer. Items have an actual name like "A" for product "A" and a quantity value. Thus a value (“A”, 4) means that there are 4 items of product type "A". One place can be used to manage all items [9]. Tables III, IV list the various places, transitions and the meaning associated with each of them.

TABLE III
LIST OF TRANSITIONS OF CPN

TRANSITION	MEANING
t_1	Insert Coin
t_2	Accept Coin
t_3	Reject Coin
t_4	Dispense Product A
t_5	Dispense Product B
t_6	Dispense Product C
t_7	Dispense Product D
t_8	Refill Product
t_9	Take Change

TABLE IV
LIST OF PLACES OF CPN

PLACE	MEANING
p_1	Machine Ready
p_2	Coin Inserted
p_3	Coin Accepted
p_4	Items A, B, C, D
p_5	Given Change

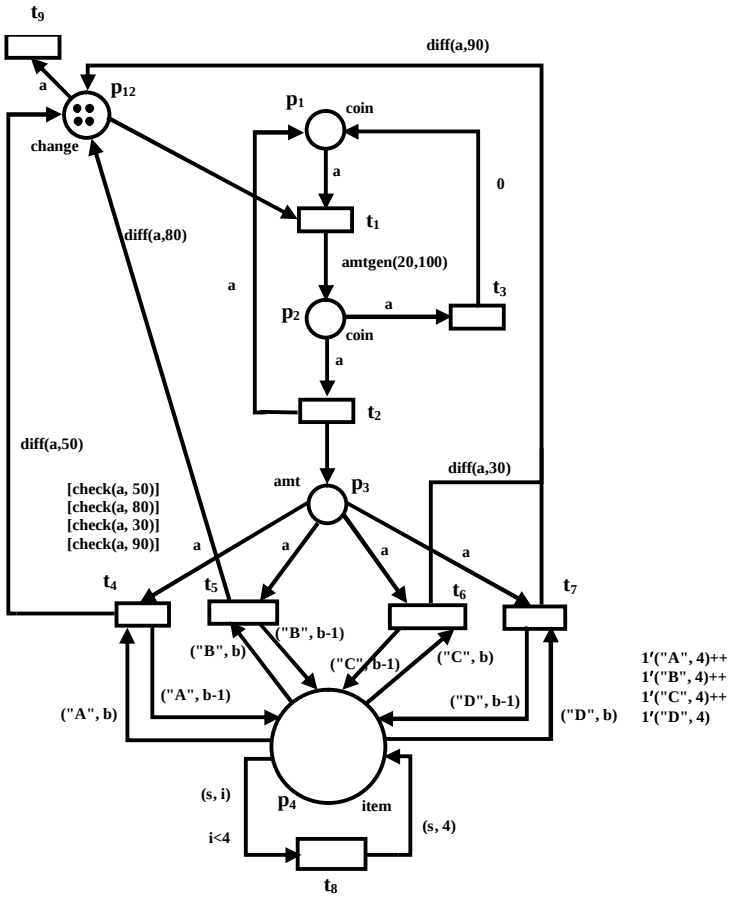

Fig.7: Modeling a Vending Machine using Colored Petri Net [9]

C. Higher-Order Petri Nets modeling

The Higher-Order Petri Net can actually be constructed from the Petri net as in Fig.8. For constructing the HOPN it is not necessary to alter the underlying net structure but apply the Conversion Procedure (def.10) of HOPN. The changes are: incoming arcs from places p_3, p_5 of transition t_4 becomes second-order input are of transition t_4. Similarly, incoming arcs from places p_3, p_7 of transition t_5, incoming arcs from places p_3, p_9 of transition t_6 and incoming arcs from places p_3, p_{11} of transition t_7 become second-order input arc of transition of t_4. Integrate of transitions t_4, t_5, t_6, t_7 (Dispense Product) to become one transition t_4, integrate of transitions t_8, t_9, t_{10}, t_{11} (Refill Product) to become one transition t_5 and integrate of places p_4, p_6, p_8, p_{10} (Items A, B, C, D) to become one p_8.

This structure allows for the possible selection of items using conflicting arcs ($f_{3,4}$, $f_{3,5}$, $f_{3,6}$, $f_{3,7}$) instead of conflicting transitions. Coins are inserted into the vending machine, it is possible to accept it or reject it. Product selection depends on value of second-order input arc fi,j (f3,4= f3,5= f3,6= f3,7= 3). Second-order input arc fi,j is enabled if there are 3 tokens in coin accepted place p_3. When an item is dispensed an item is removed and a refill takes place automatically so there are always 4 items buffered to be dispensed for each product available. Tables V, VI list the various places, transitions and the meaning associated with each of them.

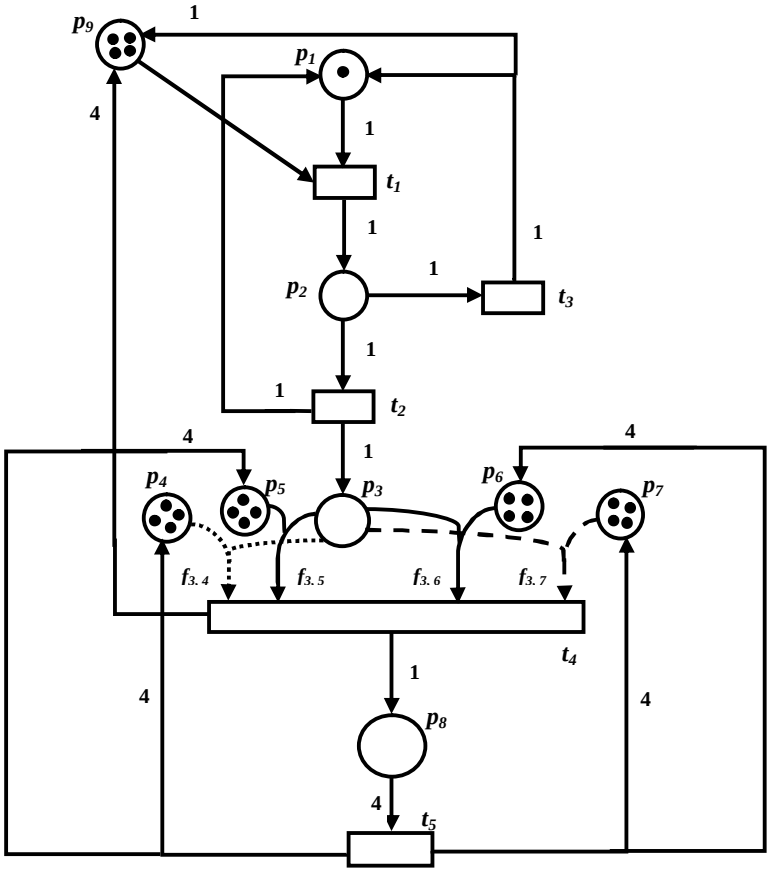

Fig.8: Modeling a Vending Machine using Higher-Order Petri Net

TABLE V - LIST OF TRANSITIONS OF HOPN

TRANSITION	MEANING
t_1	Insert Coin
t_2	Accept Coin
t_3	Reject Coin
t_4	Dispense Product
t_5	Refill Product

TABLE VI - LIST OF PLACES OF HOPN

PLACE	MEANING
p_1	Machine Ready
p_2	Coin Inserted
p_3	Coin Accepted

p_4	Items A
p_5	Items B
p_6	Items C
p_7	Items D
p_8	Item Dispensed
p_9	Coin Counter

Table VII describes the difference between Higher-Order Petri nets and Higher-Order nets using set of concepts (Definition, structure, firing rule, example and first Paper).

TABLE VII - COMPARISON BETWEEN HIGHER-ORDER PETRI NET AND HIGHER-ORDER NET

CONCEPTS	HIGHER-ORDER PETRI NET	HIGHER-ORDER NET
Definition	A new class of Petri net that there exist higher-order arcs	A general Petri net or elementary net by adding token types, firing rules and arc inscriptions.
Structures	The major difference is the definition of the arc and the weight	Characterized mainly by token types that can represent anything like an object, record, data sets, complex data types, etc.
Firing rule	Simple transition firing rule, were changed according of value of higher-order input arc (weights)	complex transition firing rules, were changed according of token type
Example	Adaptive Fuzzy Higher Order Time Petri Nets, Higher-Order Stochastic Reword Petri Net, Higher-Order Open Workflow Net.	Algebraic Petri nets, Predicate Transition nets (Prt), Product nets, environmental nets, object oriented Petri nets, colored Petri nets, etc.
First Paper	In 1997	In 1992

VI. CONCLUSIONS

In this paper, we discuss the differences between the two classes of Petri nets, namely: Higher-Order Nets and Higher-Order Petri Nets. To disperse between them we use the application of vending machine. In prior research has been modeling this application using Higher-Order Net [9]. A Higher-Order Net describes as following: 1) Highly structured places representing records, sets, objects, 2) Well formed, 3) Complex data types, 4) Complex transition firing rules, 5) Arc structures, 6) Complex structures overall.

In this paper, we present the modeling of the application of vending machine using Higher-Order Petri Nets. A model of higher-Order Petri Net produces from Petri net and depending on Conversion procedure. A Higher-Order Petri Net describes as following: 1) Merge transitions that correspond to the conversion procedure, 2) Reduction in the number of places, 3) Modification of the firing rule to fit the proposed Model, 4) the firing rule depends on value of higher-order input arc.

REFERENCES

[1] J. Wang, *Petri Nets for Dynamic Event-Driven System Modeling*, in: *Handbook of Dynamic System Modeling*, Ed: Paul Fishwick, CRC Press, 2007.
[2] V. C. Gerogiannis, A. D. Kameas, and P. E. Pintelas, "Comparative study and categorization of high-level petri nets", Journal of Systems and Software, vol. 43, pp. 133-160, 1998.
[3] G. Hassapis, and D. Ananidou, "Modeling and verification of a class of real-time systems by the use of High Level Petri Nets", Journal of Systems and Software, vol. 68, pp. 153–165, 2003.
[4] J.-I. Rocha, L. Gomes and O. Dias, "Dataflow Model Property Verification Using Petri net Translation Techniques", Industrial Informatics (INDIN), 9th IEEE International Conference, pp. 783–788, 2011.
[5] T. Chow and J.-Y. Li, "Higher-Order Petri Net Models Based on Artificial Neural Networks", Artificial Intelligence, vol. 92, pp. 289-300, 1997.
[6] *High-level Petri Nets - Concepts, Definitions and Graphical Notation*, Final Draft International Standard ISO/IEC 15909, Version 4.7.1, October 28, 2000.
[7] K. Hoffmann, "Formal Approach and Applications of Algebraic Higher-Order Nets", PhD thesis, University Berlin, 2006.
[8] A. Meligy, H. Ibrahim, A. Aqlan, "Modeling and Verification of 802.16 MAC Protocol using Higher-Order Petri Nets", International Journal of Computer Network and Information Security, vol. 6, No. 4, pp.21-28, 2014.
[9] A. Staines, "Supporting Requirements Engineering with Different Petri Net Classes", International Journal of Computers, vol. 4, pp. 215-222, 2010.