ELSEVIER

Artificial Intelligence 92 (1997) 289–300

Artificial Intelligence

Research Note

Higher-order Petri net models based on artificial neural networks

Tommy W.S. Chow [*], Jin-Yan Li

Department of Electronic Engineering, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong

Received May 1996; revised November 1996

Abstract

In this paper, the properties of higher-order neural networks are exploited in a new class of Petri nets, called higher-order Petri nets (HOPN). Using the similarities between neural networks and Petri nets this paper demonstrates how the McCullock–Pitts models and the higher-order neural networks can be represented by Petri nets. A 5-tuple HOPN is defined, a theorem on the relationship between the potential firability of the goal transition and the T-invariant (HOPN) is proved and discussed. The proposed HOPN can be applied to the polynomial clause subset of first-order predicate logic. A five-clause polynomial logic program example is also included to illustrate the theoretical results. © 1997 Elsevier Science B.V.

Keywords: Neural networks; Higher-order Petri nets; Polynomial clause program

1. Introduction

Petri nets (PN), when used as graphical and mathematical tools, have found considerable applications in a number of different areas [8]. On one hand, the places and transitions of Petri nets are interconnected in various ways to provide the properties of parallelism, and asynchronies. On the other hand, artificial neural networks (ANN) based on massively parallel distributed processors which have natural properties for storing experiential knowledge are being made available for different applications [5]. More importantly, ANN exhibits many characteristics similar to PN, for example, the activation function in ANN is similar to the firing rule in PN. Owing to the similarities

[*] Corresponding author. E-mail: eetchow@cityu.edu.hk.

between ANN and PN, many recent works on combining the characteristics of PN and ANN to form various types of new models have been reported [1,4,10]. As there has been an increasing need to model diverse and complex systems, conventional Petri nets have become inadequate for evaluating these situations. This prompted the development in new classes of nets, for example, high-level Petri nets, which have played an important role in automatic predicate logic programming [8]. While almost in the same period of time, there has been much exciting and promising progress in the area of ANN. Many new architectures, algorithms and theories have emerged in the area of ANN. Especially, the introduction of higher-order synaptic weights in first-order ANN has provided a marked contribution on both ANN theory and applications [2,3,6]. 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. In this paper we call this new class of Petri nets, higher-order Petri nets (HOPN).

In computer science the introduction of logic programming has also sparked a new era because it provides a uniform formalism for diverse aspects in computer science, especially for artificial intelligence [9]. Murata [8] and Peterka and Murata [9] have proposed a Petri net called predicate/transition net for a subset of Horn clause logic programs. It turns out that the goal transition of Horn clause logic programs is potentially firable if and only if there exists a non-negative T-invariant which includes the goal transition in its support.

In this paper, we apply the higher-order Petri nets to polynomial logic programs, which is a broad subset of logic programs and contains Horn clause logic programs. For such a subset of logic programs, the goal transition is potentially firable (in higher-order Petri nets) if and only if there exists a non-negative T-invariant (in higher-order Petri nets) which includes the goal transition in its support.

The rest of this paper is organised as follows. In Section 2, the concepts and the structures of conventional PN are briefly reviewed, and the McCulloch–Pitts model [5] and its Petri net model are also described. In Section 3, the higher-order neuron is described. The higher-order Petri net is formally defined and its related properties are introduced. Furthermore, the equivalent Petri net model of a higher-order neuron is also presented in this section. The main results of HOPN together with the theory of potentially firing goal transition is described in Section 4. Section 5 details the application of HOPN to polynomial logic programs, this includes the transformation procedure that translates a polynomial clause program into its net model. Such a transformation between the specification of the original problem and the model preserves a logical equivalence. A conclusion is drawn in Section 6.

2. Transformation of the McCullock–Pitts model into the PN model

2.1. McCullock–Pitts model

The McCullock–Pitts model [5] shown in Fig. 1 is the simplest form of neural networks. It is the fundamental building block of many multilayer feedforward networks

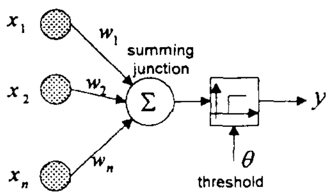


Fig. 1. The neural McCullock–Pitts model.

and recurrent networks. In this model the well-known adaptability comes from representing the synaptic action by a variable weight which determines whether the neuron should or should not fire. As there are many synapses, the summation node of the model computes a linear combination of these inputs applied to its synapses and compares them to a threshold. The resulting sum is then applied to a step function. If the sum exceeds the threshold, the neuron fires and the output y is obtained.

In mathematical terms, the output y of the neuron is described as $y = f(\sum_{i=1}^{n} w_i x_i - \theta)$, where $x_1, x_2, \ldots, x_n$ are inputs, $w_1, w_2, \ldots, w_n$ are the synaptic weights; θ is the threshold; and $f(\cdot)$ is the step function.

2.2. *Brief review of conventional PN*

Firstly, we briefly describe the concepts of conventional Petri nets [8].

The formal definition of conventional Petri net is described as a 5-tuple, $PN = (P, T, F, W, M_0)$, 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*, $P \cup T \neq 0$, and $P \cap T = \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*; $W : F \to \mathbb{N}$ is a weight function, where $\mathbb{N}$ represents the set of non-negative integers; $M_0 : P \to \mathbb{N}$ is the *initial marking*.

Enabled transition: a transition t is said to be enabled if each input place p of t contains not less than n_t number of tokens, where n_t is equal to the weight of the directed arc connecting p to t.

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_r number of tokens from each input place p, where n_r is equal to the weight of the directed arc connecting p to t. It also deposits n_d number of tokens in each output place p, where n_d is equal to the weight of the directed arc connecting t to p.

The *indegree* (*outdegree*) of a transition t is an integer equal to the sum of the weights of all incoming (outgoing) arcs of t. A transition is called a *source* (*goal*) if the indegree (outdegree) equals 0. It is also defined that a source transition is unconditionally enabled. The firing of a goal transition only consumes tokens but does not produce tokens. The firing rule is illustrated in Fig. 2.

Incidence matrix: for a Petri net with n transitions and m places, the incidence matrix $A = [A_{ij}]$ is an $n \times m$ matrix of integers, where a_{ij} is the weight of the arc from transition i to its output place j minus the weight of the arc to transition i from its input place j.

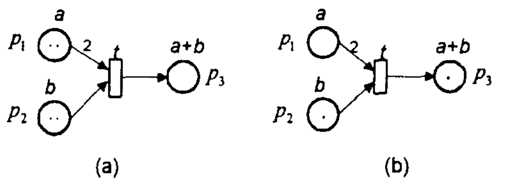


Fig. 2. An illustration of the firing rules. (a) The marking before firing the enabled transition t. (b) The marking after firing t, where t is disabled.

2.3. McCullock–Pitts model of the Petri net form

Using the similarities between neural networks and Petri nets, we are able to transform the McCullock–Pitts model into the form of a Petri net as shown in Fig. 3, where $t_{11}, t_{12}, \ldots, t_{1n}$ and t_0 are source transitions that create inputs to this model. The spatial integrating behaviour of the neural form is modelled by transition t. Transitions t_1 and t_2 mimic the step function, which is the activation function of the neural form.

3. Higher-order Petri net

The higher-order Petri net will be formally defined in this section. In this paper, we extend the conventional Petri nets to a general sense by the introduction of higher-order arcs. This process is similar to the heuristic introduction of higher-order synaptic weights in neural network systems. After the necessary syntactical definitions are given, this new type of HOPN is capable of evaluating more general and more complex systems such as higher-order neural networks, polynomial predicate logic programs, etc.

3.1. Higher-order neuron

The higher-order neuron [2,3,6] is the basic block of higher-order neural networks, and contains higher-order synaptic weights from its inputs. The total input of such a higher-order neuron consists of a linear combination of its inputs and combinations of its input products. For a fully-connected higher-order neuron with N-dimensional input neurons, $x = (x_1, x_2, \ldots, x_N)$, its output $f(x)$ is described as

$$f(x) = \sum_{j=0}^{N} \sum_{\{n(k)\}_j} w_{\{n(k)\}_j} x_{n(1)} x_{n(2)} \cdots x_{n(j)},$$

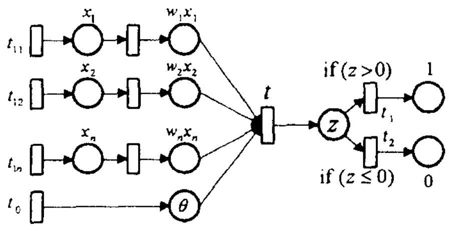


Fig. 3. The McCullock–Pitts model of the Petri net form, where $z = \sum_{i=1}^{n} x_i w_i - \theta$.

where $w_{\{n(k)\}_j}$ is the jth-order connection weights from the input neurons, $n(1)$, $n(2), \ldots, n(j)$, to the higher-order neuron. The inner summation runs over all situations. The symbol of $\{n(k)\}_j$ is a set of positive integers and is defined as $\{n(k)\}_j \equiv \{n(1), n(2), \ldots, n(j)\} \subseteq \{1, 2, \ldots, N\}$ and satisfies $n(1) < n(2) < \cdots < n(j)$. Especially, $\{n(k)\}_j \equiv \emptyset$, the empty set, if and only if $j = 0$. In this situation, $w_{\emptyset}^{(i)}$ stands for the threshold of the higher-order neuron.

3.2. Higher-order Petri net

Definition 1. A higher-order Petri net is defined as a 5-tuple, $HOPN = (P, T, F, W, M_0)$, 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*, $P \cup T \neq 0$, and $P \cap T = \emptyset$; $F \subseteq (P \times T) \cup (P^2 \times T) \cup \cdots \cup (P^m \times T) \cup (T \times P)$ is a set of *arcs*. If $f \in (P^i \times T)$, then f is called the ith-order arc; $W: F \to \mathbb{N}$ is a weight function, where $\mathbb{N}$ represents the set of non-negative integers; $M_0: P \to \mathbb{N}$ denotes the initial token distribution, called the *initial marking*.

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

Definition 2 (*Firing rule*). A transition t is said to be enabled or firable if there exist at least one of its kth-order input arcs such that each of this arc's places have at least as many tokens as the weight of this kth-order arc. Such an arc is defined as an *enabled arc*.

An enabled transition may or may not fire. When an enabled transition t fires, one of its enabled arcs fires. 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.

An example shown in Fig. 4 illustrates the definitions and the firing rule of HOPN. In Fig. 4(a), p_1, p_2 and p_3 are the input places of transition t_1, and p_4, p_5 are the output

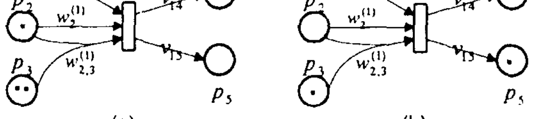


Fig. 4. An example of illustration of HOPN. (a) The initial distribution of tokens. (b) The distribution state of tokens after t_1 fired.

places, and $w_{2,3}^{(1)}$ represents the weight of the second-order arc f from places p_2, p_3 to transition t_1. The remaining arcs are conventional. Except $w_2^{(1)} = 2$, all other weights are equal to 1. Fig. 4(b) shows the token distribution after the transition t_1 fired, which is equivalent to having the arc $f_{2,3}^{(1)}$ fired. The arc $f_2^{(1)}$ is not enabled before firing but $f_1^{(1)}$ and $f_{2,3}^{(1)}$ are enabled.

Following Peterka and Murata [9], we know that a firing sequence $\sigma = \langle t_1, t_2, \ldots, t_n \rangle$ is said to be able to transform a marking M_0 into a marking M_n, where $M_0[t_1, M_1, t_2, M_2, \ldots, t_n] > M_n$, and M_i represents the marking state after the ith transition in the firing sequence fires. A firing sequence $\sigma = \langle t_1, t_2, \ldots, t_n \rangle$ is said to be *executable* from M_0 if t_1 is firable from M_0, and t_2 is firable from M_1, and so on for all transitions in σ. A transition t is said to be *potentially firable* if it can be made firable through a certain firing sequence.

Definition 3 (*Relation matrix and incidence matrix*). For an HOPN, if there are m places and n transitions, there are m *relation matrices*. The first one, A_1, is an $n \times m$ matrix as shown in the following.

$$A_1 = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & \cdots & p_m \end{array} \\ \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1m}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2m}^{(1)} \\ \vdots & \vdots & & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nm}^{(1)} \end{bmatrix} \end{array},$$

where, $a_{ij}^{(1)}$ is the weight of the first-order arc from transition i to place j minus the weight of the first-order arc from place j to transition i.

The second one, A_2, and the last one, A_m, are $n \times s$ and $n \times 1$ matrices respectively, where $s = C_m^2$.

$$A_2 = \begin{array}{c|cccccccccc} & (p_1, p_2) & (p_1, p_3) & \cdots & (p_1, p_m) & (p_2, p_3) & \cdots & (p_2, p_m) & \cdots & (p_{m-1}, p_m) \\ t_1 & a_{11}^{(2)} & a_{12}^{(2)} & & \cdots & \cdots & & \cdots & & a_{1s}^{(2)} \\ t_2 & a_{21}^{(2)} & a_{22}^{(2)} & & \cdots & \cdots & & \cdots & & a_{2s}^{(2)} \\ \vdots & \vdots & \vdots & & & & & & & \vdots \\ t_n & a_{n1}^{(2)} & a_{n2}^{(2)} & & \cdots & \cdots & & \cdots & & a_{ns}^{(2)} \end{array},$$

$$A_m = \begin{array}{c|c} & (p_1, p_2, \ldots, p_m) \\ t_1 & a_{11}^{(m)} \\ t_2 & a_{21}^{(m)} \\ \vdots & \vdots \\ t_n & a_{n1}^{(m)} \end{array},$$

$a_{ij}^{(2)}$ is the negative of the weight of the second-order arc from two places corresponding to j to transition i, and $a_{i1}^{(m)}$ is the negative of the weight of the mth-order arc from m places to transition i.

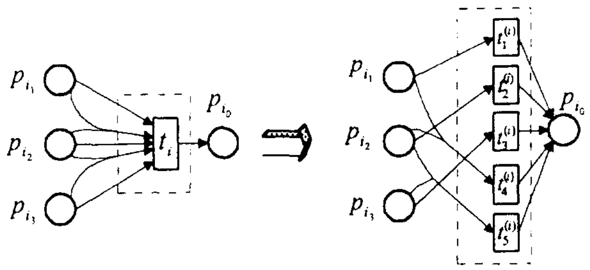


Fig. 5. An informal illustration of a decomposing procedure for HOPN.

An HOPN, that has all transitions with outdegree $\leqslant 1$, can be decomposed. Given a transition t_i, the decomposing procedure is shown in Fig. 5. If t_i has k non-zero input arcs, transition t_i could be decomposed into k transitions, and each of them connects to one arc. It is also noted that a higher-order arc can be regarded as a combination of several conventional arcs. Therefore, t_i creates a set of conventional transitions, denoted as $T_i = (t_1^{(i)}, t_2^{(i)}, \ldots, t_{k_i}^{(i)})$. After the transformation of all transitions, the definitions of the higher-order firing rule and the enabled transition are identical to the conventional one. The transformed HOPN incidence matrix is called the incidence matrix of this HOPN, and is denoted as A.

$$A^{\mathrm{T}} = \begin{array}{c|c} & \overbrace{t_1^{(1)} \; t_2^{(1)} \; \cdots \; t_{k_1}^{(1)}}^{t_1} \; \cdots \; \overbrace{t_1^{(n)} \; t_2^{(n)} \; \cdots \; t_{k_n}^{(n)}}^{t_n} \\ \begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_m \end{array} & \left[\begin{array}{c} \\ a_{ij} \\ \\ \end{array} \right] \end{array},$$

where A^{T} represents the transition of the matrix A. It is noticed that the elements of the incidence matrix depend on the relation matrices A_i $(i = 1, 2, \ldots, m)$.

Definition 4 (*T-invariant*). A vector of integers, X', is called a *pre-T-invariant* if $A^{\mathrm{T}}X' = 0$. The ith entry of the vector X' is denoted by $X'(i)$. An n-vector of non-negative integers, X, is called a *T-invariant* of HOPN only if $X(i) = \sum_j X'(j)$, where $X'(j)$ is an entry of X' whose corresponding transition belongs to T_i.

Following Peterka and Murata [9], it is known that a subset of transitions corresponding to non-zero entries of an n-vector $X \geqslant 0$ is called its support and is denoted by $\| X \|$. A T-invariant $X \geqslant 0$ is said to be *executable* from M_0, if there exists a firing sequence σ executable from marking M_0 such that its count vector $\bar{\sigma} = X$ [9].

3.3. Equivalent Petri net of a higher-order neural network

Fig. 6(a) shows the architecture of a second-order neuron. Exploiting the similarities between the Petri nets and the neural networks, Fig. 6(b) shows the informal transformation between higher-order Petri nets and higher-order neurons.

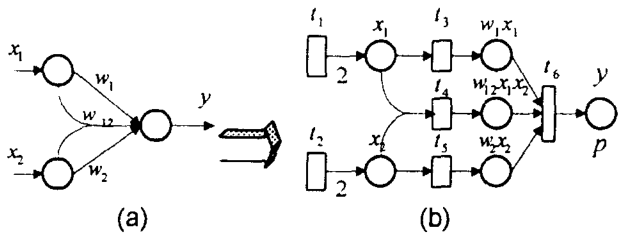


Fig. 6. (a) The architecture of a second-order neuron, where w_{12} is a second-order synaptic weight, $y = x_1w_1 + x_2w_2 + x_1x_2w_{12}$. (b) The equivalent Petri net model. t_1, t_2 are source transitions. If the firing sequence $\langle t_1, t_2, t_3, t_4, t_5, t_6\rangle$ is running, place p can get one token of form y.

4. Main results

Peterka and Murata [9] have identified the relationship between the T-invariant and the potential firability of a goal transition. Based on this method and the techniques for finding the T-invariant in high-level nets, Petri nets are very useful in solving practical problems, such as Horn clause logic programs. In this paper, we prove that such a relationship also exists in the HOPN. Thus, Petri nets can solve a broad class of logic programs, i.e., polynomial logic programs (the definition will be given in the next section).

Lemma [9]. *Let $PN = (P, T, F, W, M_0)$ be a Petri net (conventional) that has all transitions with outdegree $\leqslant 1$. Let t_g be a goal transition in T. There exists a firing sequence to reproduce the empty marking and to fire the goal transition t_g in PN iff PN has a T-invariant (conventional) X such that $X \geqslant 0$ and $X(t_g) \neq 0$.*

Theorem. *Let $HOPN = (P, T, F, W, M_0)$ be a higher-order Petri net that has all transitions with outdegree $\leqslant 1$. Let t_g be a goal transition in T. There exists a firing sequence to reproduce the empty marking and to fire the goal transition t_g in HOPN iff HOPN has a T-invariant X such that $X \geqslant 0$ and $X(t_g) \neq 0$.*

Proof. The necessity can be proved as follows. Firstly, if there exists such a firing sequence, then $X(t_g) \neq 0$. Secondly, after all transitions of this HOPN are transformed into a new Petri net according to the way described in Section 3, conventional firing rules can similarly be applied to the new Petri net and the original firing sequence changes into a new firing sequence to reproduce the empty marking. From the Lemma, the new PN has a T-invariant X' such that $X' \geqslant 0$. In fact, X' is the pre-T-invariant of HOPN. Therefore, the HOPN has a T-invariant X such that $X \geqslant 0$ and $X(t_g) \neq 0$.

The sufficiency can be proved as follows. Based on the definition of the T-invariant of HOPN, if the HOPN has a T-invariant X such that $X \geqslant 0$ and $X(t_g) \neq 0$, there exists a T-invariant of the new PN based on X after the transformation of the HOPN. From the Lemma, there exists a firing sequence of HOPN to reproduce the empty marking and to fire the goal transition t_g only if the entry $t_k^{(i)}$ of the PN firing sequence is considered as t_i for all i and k. □

5. Logic application of HOPN

The HOPN discussed in the above sections can be applied to polynomial clause logic programming which is the generalised Horn clause. In the first-order predicate logic, there is a special class of clauses of the form $B \leftarrow (\sum_{i=1}^{m} A_{n(i)})^m$, where B and $A_{n(i)}$ are atomic formulas [7], $\leftarrow$ is the implication symbol, and $m \geqslant 0$. An atomic formula is of the form $p(t_1, \ldots, t_k)$, where p is a k-place predicate symbol and $t_1, \ldots, t_k$ are terms. $(\sum_{i=1}^{m} A_{n(i)})^m$ can be expanded into the form $\sum(\prod A_{n(i)})$ only if we consider $A_{n(i)}^k$ as $A_{n(i)}$. It can be shown that $A_{n(i)}$ is equivalent to $A_{n(i)}^k$ in a logic sense. When $m = 0$, namely, $B \leftarrow$, which stands for the assertion of a fact, corresponds to a source transition without input places. Another special form, $\leftarrow \sum(\prod A_{n(i)})$, is a goal statement and corresponds to a goal transition.

Consider a simple polynomial logic program that consists of five clauses:

(1) Head(Lussy, Mike) $\leftarrow$;

(2) Tutor(Mike, John) $\leftarrow$;

(3) Head(a, b) $\leftarrow$ Tutor(a, b);

(4) Employer(x, y) $\leftarrow$ Tutor(x, y) + Head(x, y) + Head(x, z)Employer(z, y);

(5) $\leftarrow$ Employer(x, John).

Clause (1) states "Lussy is the Head of Mike" and is an assertion of a fact. Clause (4) states "x is an Employer of y if x is a Head of z and z is an Employer of y, or x is a Tutor of y, or x is a Head of y. Clause (5) is a goal statement saying "who is Employer of Tom?".

In the following section, we elaborate the conversion of a polynomial logic program into a higher-order graphic Petri net. Given a logic program that consists of n clauses and m distinct predicate symbols, the corresponding graphical HOPN has m places and n transitions. The above logic program can, therefore, be represented by Fig. 7, which has five transitions and three places. In this figure, the letters a, b, x, y and z are denoted as variables and the letters L, M and J are denoted as constants. For each transition, a variable of the same symbol appearing on the incoming and the outgoing arcs denotes the same variable. It can be noticed that the arcs in Fig. 7 are labelled with $\langle$M, J$\rangle$, $\langle$L, M$\rangle$, $\langle x, y\rangle$, $\langle a, b\rangle$, $\langle\langle x, z\rangle, \langle z, y\rangle\rangle$ and $\langle x$, J$\rangle$. According to the definition of HOPN, these labels are considered as the weights (in a broad sense) which

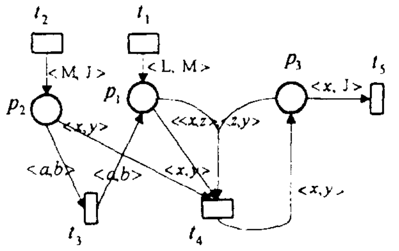
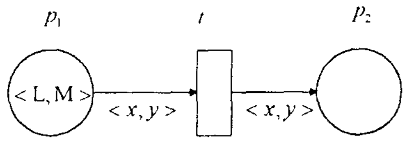
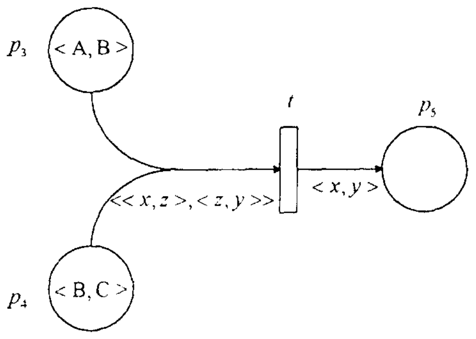


Fig. 7. A higher-order PN representation of a logic program.



(a)



(b)

Fig. 8. Two Petri-nets with "coloured" tokens [8], including a HOPN, for the illustration of the transition firing rule.

determine the number of tokens and the type of "coloured" tokens [8] to be removed from or added to the places. For example, when the transition t in Fig. 8 fires, the following things occur.

- p_1 loses one token of the colour $\langle x, y\rangle$. With the substitution $\{L \mid x, M \mid y\}$, p_2 gets one token of $\langle L, M\rangle$.
- p_3 and p_4 lose one token of the colour $\langle x, z\rangle$ and one token of the colour $\langle z, y\rangle$ respectively. With the substitution $\{A \mid x, B \mid z, C \mid y\}$, p_5 gets one token of $\langle A, C\rangle$.

It is also noted that the symbol $\langle\langle x, z\rangle, \langle z, y\rangle\rangle$ represents a higher-order weight. The incidence matrix, A, can be obtained from Fig. 7.

$$A = \begin{array}{ll|ccc|}
 & & \text{Head}(p_1) & \text{Tutor}(p_2) & \text{Employer}(p_3) \\
t_1 & & \langle L, M\rangle & 0 & 0 \\
t_2 & & 0 & \langle M, J\rangle & 0 \\
t_3 & & \langle a, b\rangle & -\langle a, b\rangle & 0 \\
 & t_1^{(4)} & -\langle x, y\rangle & 0 & \langle x, y\rangle \\
t_4 & t_2^{(4)} & 0 & -\langle x, y\rangle & \langle x, y\rangle \\
 & t_3^{(4)} & -\langle x, z\rangle & 0 & \langle x, y\rangle - \langle z, y\rangle \\
t_5 & & 0 & 0 & \langle x, J\rangle
\end{array}.$$

There are three firing sequences, σ_1, σ_2 and σ_3, to reproduce empty marking and fire goal transition, $\sigma_1 = \langle t_2, t_4, t_5 \rangle$, $\sigma_2 = \langle t_2, t_3, t_4, t_5 \rangle$, and $\sigma_3 = \langle t_1, t_2, t_3, t_4, t_4, t_5 \rangle$. These three firing sequences have the following substitution vectors X_1, X_2 and X_3.

$$X_1 = \begin{array}{r} t_1 \\ t_2 \\ t_3 \\ t_4 \begin{cases} t_1^{(4)} \\ t_2^{(4)} \\ t_3^{(4)} \end{cases} \\ t_5 \end{array} \begin{bmatrix} \emptyset \\ \{\} \\ \emptyset \\ \emptyset \\ \{\mathrm{M} \mid x, \mathrm{J} \mid y\} \\ \emptyset \\ \{\mathrm{J} \mid x\} \end{bmatrix}, \quad X_2 = \begin{array}{r} t_1 \\ t_2 \\ t_3 \\ t_4 \begin{cases} t_1^{(4)} \\ t_2^{(4)} \\ t_3^{(4)} \end{cases} \\ t_5 \end{array} \begin{bmatrix} \emptyset \\ \{\} \\ \{\mathrm{M} \mid a, \mathrm{J} \mid b\} \\ \{\mathrm{M} \mid x, \mathrm{J} \mid y\} \\ \emptyset \\ \emptyset \\ \{\mathrm{M} \mid x\} \end{bmatrix},$$

$$X_3 = \begin{array}{r} t_1 \\ t_2 \\ t_3 \\ t_4 \begin{cases} t_1^{(4)} \\ t_2^{(4)} \\ t_3^{(4)} \end{cases} \\ t_5 \end{array} \begin{bmatrix} \{\} \\ \{\} \\ \{\mathrm{M} \mid a, \mathrm{J} \mid b\} \\ \{\mathrm{L} \mid x, \mathrm{M} \mid y\} \\ \emptyset \\ \{\mathrm{L} \mid x, \mathrm{M} \mid z, \mathrm{J} \mid y\} \\ \{\mathrm{L} \mid x\} \end{bmatrix},$$

where $\emptyset$ denotes no firings and $\{\}$ denotes a firing with no substitutions. X_1, X_2 and X_3 can be interpreted as "pre-T-invariant" as they satisfy $A^{\mathrm{T}} \circ X_i = 0$, $i = 1, 2, 3$, where $\circ$ denotes "matrix product with substitution" [8,9].

6. Conclusions

Exploiting the properties of higher-order neural networks, the arcs and the weights of the conventional Petri nets have been extended to a general sense in this paper. The enabling conditions and the firing rules for HOPN are defined and do not contradict the conventional Petri nets. The proposed HOPN covers a broader range of cases compared to the conventional Petri nets. Therefore it can be regarded as a generalisation of the conventional Petri nets. As there is an inherent relationship between Petri nets and neural networks, this paper also demonstrates that the McCullock–Pitts model and higher-order networks can be represented by Petri nets. We have demonstrated that the application of an HOPN in the class of polynomial clause subset of the first-order predicate logic is straightforward. The relationship between the potential firability of the goal transition and the T-invariant (HOPN) has also been discussed. It is shown that the goal transition of an HOPN is potentially firable if and only if there exists a non-negative T-invariant which includes the goal transition in its support. Finally, practical results on the T-invariant have been obtained, and the theorem presented in this paper is believed to be practically useful.

References

[1] S.I. Ahson, Petri net models of fuzzy neural networks, *IEEE Trans. Syst. Man Cybern.* **25** (1995) 926–932.

[2] S.I. Amari, Dualistic geometry of the manifold of higher-order neurons, *Neural Networks* **4** (1991) 443–451.

[3] C.L. Giles and T. Maxwell, Learning, invariance, and generalization in high-order neural networks, *Appl. Optics* **26** (1987) 4972–4978.

[4] M.K. Habib and R.W. Newcomb, Neuron type processor modelling using a timed Petri net, *IEEE Trans. Neural Networks* **1** (1990) 282–289.

[5] S. Haykin, *Neural Networks: A Comprehensive Foundation* (Macmillan, New York, 1994).

[6] J.Y. Li and T.W.S. Chow, Functional approximation of higher-order neural networks, *J. Intell. Syst.* (to appear).

[7] J.W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 1984).

[8] T. Murata, Petri nets: properties, analyses and applications, *Proc. IEEE* **77** (1989) 541–580.

[9] G. Peterka and T. Murata, Proof procedure and answer extraction in Petri net model of logic programs, *IEEE Trans. Softw. Eng.* **15** (1989) 209–217.

[10] K. Venkatesh, O. Masory and A. Pandya, A high level Petri net model of olfactory bulb, in: *Proceedings International Conference on Neural Networks*, San Francisco, CA (1993) 766–771.