Analysis of Approximate Petri Nets by Means of Occurrence Graphs

Z. Suraj\textsuperscript{1}, B. Fryc\textsuperscript{2}, G. Owsiany\textsuperscript{2}

\textsuperscript{1}Chair of Computer Science
University of Rzeszów, Poland
\textsuperscript{2}Chair of Computer Science Foundations
University of Information Technology and Management
Rzeszów, Poland
e-mail: zsuraj@univ.rzeszow.pl, \{bfryc, gowsiany\}@wsiz.rzeszow.pl

\textbf{Abstract.} Approximate Petri nets (\textit{AP}-nets) can be used for the knowledge representation and approximate reasoning. The \textit{AP}-net model is defined on the basis of the rough set approach, fuzzy Petri nets and coloured Petri nets. One of the main advantages of \textit{AP}-net model is a possibility to present the reachability set of a given \textit{AP}-net by means of an occurrence graph. Such graphs can serve, among others, for analyzing and evaluating an approximate reasoning realized by using \textit{AP}-net model. The main contribution of the paper is to present the algorithms for construction and analysis of occurrence graphs for the \textit{AP}-nets, especially in the context of searching for the best decision and finding the shortest distance in order to compute such decision. This approach can be applied to the design and analysis of the formal models for expert systems, control systems, communication systems, etc.

\textbf{Keywords:} approximate Petri nets, approximate reasoning, knowledge representation, occurrence graphs

1 Introduction

In the last few years, we have assisted in a growing interest in using Petri net models as the basis for construction of knowledge representation and approximate reasoning frameworks [6], [8], [1], [3]. Approximate Petri nets (\textit{AP}-nets) introduced by the authors [3] are modified coloured Petri nets (\textit{CP}-nets) [4]. The \textit{AP}-nets allow rule-based decision making systems to be represented and executed. They are constructed on the basis of production rules automatically extracted by using rough set methods [7] from the given experimental decision tables. The decision tables in Pawlak’s sense include the values of parameters (condition attributes) measured by sensors together with an expert decision. These sensors produce outputs after unknown but finite number of time units. The constructed \textit{AP}-net allows us to identify objects in decision tables to an extent which makes appropriate decision possible. The outputs from sensors are propagated through the net with maximal speed. This is done by appropriate implementation of all rules (decision rules and conditional rules) in a given decision table. The decision rules describe the relationship between the values of
parameters and the decision. However, the condition rules represent the relationship between the values of parameters. The rules may be adapted by changing the threshold values for the transitions, what could be done, e.g., in training sessions in which thresholds are adjusted to minimize the decision error (the usual type of learning for classifier systems). The rough set approach seems to be very suitable for real-time decision making from incomplete or uncertain information. In the paper we assume that the knowledge encoded in a given decision table is uncertain in the sense that unseen objects can have parameter values partially consistent with rules extracted from a given decision table. The AP-nets use also fuzzy logic [9], and they reason efficiently without the need for human interaction. They are easily specified, designed, and modified by using self-implemented computer tools. This paper describes how AP-nets can be analyzed by means of the occurrence graphs constructed for the AP-nets. The occurrence graph contains a node for each reachable state (marking) and an arc for each possible state changes (by firing an enabled transition with a binding element). The presented approach seems to have some significance for real-time applications in such areas as real-time decision making by groups of intelligent agents, error detecting in distributed systems, navigation of intelligent mobile robots and in general in real-time knowledge-based control systems.

The paper is organized as follows. In Section 2 we fix some notions and notation used in the paper. Section 3 deals with the definition of coloured Petri nets. Section 4 presents the definition of approximate Petri nets. Section 5 includes an algorithm for constructing an occurrence graph for a given AP-net. An algorithm for analysis of the occurrence graph is presented in Section 6. Concluding remarks are made in the last section.

2 Preliminaries

This section just fixes some notions and notations concerning rough sets [7] used in the paper. We assume that the reader is acquainted with the basic notions of fuzzy sets [9].

2.1 Rough Sets

Let $S = (U, A)$ be an information system, where $U$ denotes a set of objects and $A$ - a set of attributes. The set $V = \bigcup_{a \in A} V_a$ is the domain of $A$, where $V_a$ is a set of values of $a$.

By a decision system we mean any information system of the form $S = (U, A \cup D)$, where $A$ is a set of conditional attributes (conditions), $D$ is a set of decision attributes (decisions), and $A \cap D = \emptyset$ (the empty set).

Example 1. Let us consider an example of a decision system $S = (U, A \cup D)$ as shown in Table 1.

In the system $S$, the set of objects $U = \{u_1, u_2, ..., u_6\}$. Each object is described by three conditions $H$, $M$ and $T$. The decision is represented by an attribute $F$. 

59
Table 1. An example of a decision system $S$.

<table>
<thead>
<tr>
<th>$U$</th>
<th>$A$</th>
<th>$D$</th>
<th>$H$</th>
<th>$M$</th>
<th>$T$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_2$</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_3$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_4$</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_5$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_6$</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The sets of possible values of attributes from $S$ are equal as follows: $V_H = V_F = \{1, 2\}$ and $V_M = V_T = \{1, 2, 3\}$.

Let $S = (U, A')$ be a decision system, where $A' = A \cup D$. Let $V'$ be the domain of $A'$. Any expression $R$ of the form: IF $\phi$ THEN $\psi$, where $\phi$ and $\psi$ are terms over $A'$ and $V'$ denotes a rule in $S$. The terms $\phi$ and $\psi$ are called the predecessor and successor of $R$, and denoted by $\text{Pred}(R)$ and $\text{Succ}(R)$, respectively.

In the sequel, we will distinguish two main kinds of rules: (1) The rules expressing some relationships between values of conditions, called conditional rules. (2) The rules expressing certain relationships between the values of conditions and the decision, called decision rules.

Several numerical factors can be associated with a given rule. In the paper we need the so called certainty factor.

Let $S = (U, A')$ be a decision system, where $A' = A \cup D$ and IF $\phi$ THEN $\psi$ be a rule in $S$.

By the certainty factor ($CF$) of a given rule we mean the number equal to the ratio of the number of objects satisfying simultaneously the predecessor and the successor of the given rule, and the number of objects satisfying only the predecessor of the rule. It is easy to see that $CF \in [0, 1]$. If $CF = 1$ then we say that a given rule is deterministic, otherwise - non-deterministic.

Let $S = (U, A')$ be a decision system. In the following we accept the following notation for $S$: $\text{RULE}(S)$ - a set of all conditional rules, $\text{RULE}_d(S)$ - a set of all deterministic conditional rules, $\text{RULE}_n(S)$ - a set of all non-deterministic conditional rules, $\text{RULE}(S)$ - a set of all decision rules, $\text{RULE}_d(S)$ - a set of all deterministic decision rules, and $\text{RULE}_n(S)$ - a set of all non-deterministic decision rules, $\text{RULE}(S)$ - a set of all rules in $S$.

Example 2. A set of all rules for the decision system $S$ consists of: 13 deterministic conditional rules, 44 non-deterministic conditional rules, 7 deterministic decision rules, and 12 non-deterministic decision rules. For simplicity of the description, we present here a sample of rules for this decision system including only 7 rules together with the appropriate certainty factors ($CF$): $R1$: IF $M = 3$ THEN $F = 2$ ($CF=1$, deterministic decision rule), $R2$: IF $T = 1$ THEN $F = 1$ ($CF=1$, deterministic decision rule), $R3$: IF $M = 2$ AND $H = 2$ THEN $F = 1$ ($CF=1$, deterministic decision rule), $R4$: IF $M = 1$ AND $H = 2$ THEN $F = 1$ ($CF=0.66$, non-deterministic decision rule), $R5$: IF $M = 1$ AND $H = 2$ THEN...
\( F = 2 \) (\( CF=0.33 \), non-deterministic decision rule), \( R6: IF \ T = 1 \ THEN \ H = 2 \) (\( CF=1 \), deterministic condition rule), \( R7: IF \ M = 1 \ THEN \ H = 2 \) (\( CF=1 \), deterministic condition rule).

3 Coloured Petri Nets

Coloured Petri Nets (the \( CP \)-nets, in short) are high-level nets proposed by K. Jensen [4]. The structure in \( CP \)-net is a directed graph with two kinds of nodes: places (drawn as ellipses) and transitions (drawn as rectangles), interconnected by arcs - in such a way that each arc connects two different kinds of nodes (i.e., a place and a transition). The places and their tokens represent states, while the transitions represent state changes. The data value which is attached to a given token is referred to as the token colour. The declaration of the net tells us about colour sets and variables. Each place has a colour set attached to it and this means that each token residing on that place must have a colour which is a member of the colour set. Each net inscription is attached to the place, transition or arc. Places have four different kinds of inscriptions: names, colour sets, initialization expressions and current markings. Transitions have two kinds of inscriptions: names and guards, while arcs only have one kind of inscription: arc expressions. The initialization expression of a place must evaluate to a multiset over the corresponding colour set. Multisets are analogous to sets except that they may contain multiple appearances of the same element. The guard of a transition is a Boolean expression which must be fulfilled before the transition can occur. The arc expression (as the guard) may contain variables, constants, functions and operations that are defined in the declarations. When the variables of an arc expression are bound, then the arc expression must evaluate to a colour that belongs to the colour set attached to the place of the arc. A distribution of tokens (on the places) is called a marking and denotes \( M \). The initial marking \( M_0 \) is the marking determined by evaluating the initialization expressions. A pair, where the first element is a transition and the second element is a binding of that transition, is called an occurrence element. If an occurrence element is enabled in a given marking then we can speak about the next marking which is reached by the occurrence of the occurrence element in the given marking.

The formal definition of \( CP \)-nets will be given below.

A coloured Petri net is a tuple:

\[ CPN = (\Sigma, P, T, A, N, C, G, E, I) \]

satisfying the following requirements:

- \( \Sigma \) is a nonempty, finite set of types which are called colour sets,
- \( P \) is a finite set of places,
- \( T \) is a finite set of transitions, \( A \) is a finite set of arcs,
- \( N : A \to (P \times T) \cup (T \times P) \) is a node function,
- \( C : P \to \Sigma \) is a colour function,
- \( G \) is a guard function,
E is an arc expression function, 
I is an initialization function.

A behaviour of CP-nets is described in [4].

4 Approximate Petri Nets

In this section approximate Petri nets (AP-nets) presented in [3] will be recalled. The main idea of approximate Petri nets derives from coloured Petri nets and fuzzy Petri nets [1]. The formal definition of AP-nets and their dynamic are presented in [3]. We can understand an approximate Petri net as a pair:

\[ APN = (CPN, f), \]

where:

- CPN is a modified coloured Petri net, where a marking of places is a fuzzy set instead of a multiset which is used in CP-net. Next difference concerns arc expression function. In APN, arc expression function associated with output arcs must yield a fuzzy set in a universe of discourse of the colour set that is attached to the corresponding place.
- \( f \) is a certainty factor function \( f: T \rightarrow [0, 1] \), which maps each transition to a real value between zero and one (called a certainty factor value).

Now, we present only a scheme of APN construction on the basis of a given decision system \( S \) and a set of all rules \( RUL(S) \). The exact algorithm is given in [3].

Each place \( p_{a_i} \) of APN corresponds to one attribute (conditional or decision) of \( S \). For each place \( p_{a_i} \), its colour set consists of colours corresponding to individual values of the attribute \( a_i \). Each transition \( t_{r_j} \) of \( APN_S \) represents one rule (conditional or decision) extracted from the decision system \( S \).

The set of all places APN will be denoted by \( P(APN) \) (in short \( P \)). This set includes the set of places corresponding to conditional attributes called conditional places and denoted by \( P^c(APN) \) (in short \( P^c \)) and the set of places corresponding to decision attributes called decision (goal) places and denoted by \( P^d(APN) \) (in short \( P^d \)).

**Example 3.** Let us consider a decision system \( S \) together with a sample of rules presented in Example 2. The AP-net model corresponding to these rules is presented in Figure 1. In the AP-net from Figure 1, places \( p_H, p_M, p_T \) represent the conditional attributes \( H, M, T \) from \( S \), respectively. However, the place \( p_F \) represents the decision \( F \). In the example, the set \( P^c = \{p_H, p_M, p_T\} \) and the set \( P^d = \{p_F\} \). The transitions \( t_1, \ldots, t_5 \) represent the rules \( R1, \ldots, R5 \), respectively. Transitions \( t_6, t_7 \) represent the rules \( R6, R7 \). The colour sets (types) are the following: \( H = \{H_1, H_2\}, M = \{M_1, M_2, M_3\}, T = \{T_1, T_2, T_3\}, F = \{F_1, F_2\} \). For example, transition \( t_4 \) represents the decision rule: IF \( M = 1 \) AND \( H = 2 \) THEN \( F = 1 \) (\( CF=0.66 \)). The input arc expressions are the following: \( e_7 = \{x_H\}, \)
$e_6 = \{x_M\}$, where $x_H$, $x_M$ are variables of the type $H$ and $M$, respectively. The output arc expression has the form $e_8 = CF \ast \max(\mu(x_H), \mu(x_M))/y_F$, where $y_F$ is a variable of the type $F$. It is easy to see that putting values of variables $x_H$, $x_M$, $y_F$ we obtain a fuzzy set over $F$. The guard expression for $t_4$ is the following:

$g_4 = [x_M = M_1 \land x_H = H_2 \land x_F = F_1]$. Moreover, $CF_3 = 0.66$. Analogously we can describe other transitions and arcs.

In APN the marking $M$ consists of $M^c$ (i.e., the marking for $P^c$) and (i.e., $M^d$, the marking for $P^d$). In the AP-net from Figure 1 $M_0 = \{M(p_H), M(p_M), M(p_T)\}$, $M_0^d = \{M(p_F)\}$.

Let us consider the following an initial (start) marking $M_0$ for the given net model from Fig. 1: $M(p_H) = \{0/H_1 + 0/H_2\}$, $M(p_M) = \{1/M_1 + 0/M_2 + 0/M_3\}$, $M(p_T) = \{0/T_1 + 0/T_2 + 0/T_3\}$, $M(p_F) = \{0/F_1 + 0/F_2\}$.

We say the marking $M'$ is directly reachable from the marking $M$ by the firing of the enabled transition $t$, with binding $b$, which modifies a marking at least in one place. This fact is written $M[be > M'$, or $M \xrightarrow{be} M'$. A finite occurrence sequence is a sequence of markings and binding elements $be \in BE$: $M_0[be > M_1[be > M_2[be > M_3[be > ...M_n[be > M_{n+1}$ such that $n \in N$, $M_0$ is a start marking and $M_{n+1}$ is an end marking. A marking $M'$ is reachable from a marking $M_0$ if and only if there exists a finite occurrence sequence starting in $M_0$ and ending in $M'$. The set of all markings reachable from $M_0$ is called the reachability set and denoted by $[M_0 >$.

Example 4. Let us consider AP-net from Figure 1. The marking $M_1$ directly
reachable from the start marking $M_0$ after firing transition $t_7$ is shown in the Table 2.

5 Occurrence Graphs

The basic idea behind occurrence graphs is to construct a graph which contains a node for each reachable state (marking) and an arc for each possible change of state. For AP-nets such a graph is finite if each place of a net is bounded. A finite occurrence graph may be used to verify an approximate reasoning process realized by a given AP-net.

In this section, we present basic definitions and an algorithm related to the occurrence graphs. In the paper, we consider only bounded AP-nets, i.e., the nets in which all variables have finite types [5]. An occurrence graph $OG$ of an AP-net is the directed graph $OG = (V, A, N)$ satisfying the following requirements: (1) The set of nodes $V = [M]$, where $[M]$ is a set of all markings reachable from a start marking $M_0$, (2) The set of arcs $A = \{(M_1, be, M_2) \in V \times BE \times V : M_1 [be > M_2] \}$, and (3) $N$ denotes a node function such that: $\forall a = (M_1, be, M_2) \in A : N(a) = (M_1, M_2)$.

Below, we propose an algorithm to construct the occurrence graph for a given AP-net.

Let $V$ be a set of nodes. In the beginning, it contains the nodes for which we have not found the successors, yet i.e., the nodes that wait to be processed. In the algorithm presented below, $Node(M)$ denotes a procedure which creates a

![Fig. 2. An occurrence graph for AP-net from Figure 1.](image-url)
new node $M$ and adds $M$ to $V$. If the node $M$ already belongs to the set $V$, then
the procedure has no effect. Analogously, $Arc(M_1, be, M_2)$ denotes a procedure
which creates a new arc $(M_1, be, M_2)$ with a source $M_1$ and a destination $M_2$.
By $Next(M_1)$ we denote a set of all possible "next moves" for a given marking
$M_1 \in [M >]$: $Next(M_1) = \{(be, M_2) \in BE \times [M_0 >] : M_1[be > M_2]\}$.

**ALGORITHM 1** for constructing an occurrence graph $OG$ for $AP$-net.

**INPUT:** A given $AP$-net and its start marking $M_0$.

**OUTPUT:** An occurrence graph $OG$ of $AP$-net.

begin
$V = \emptyset$
$Node(M_0)$
repeat
select a node $M_0 \in V$
for all $(t, M_1) \in Next(M_0)$ do
begin
$Node(M_1)$
$Arc(M_0, be, M_1)$
remove $M_0$ from $V$
end;
until $V = \emptyset$
end.

The complexity of building an occurrence graph is exponential. An occurrence
graph grows very fast, when number of transitions and colour sets increase.

**Example 5.** Figure 2 presents the occurrence graph for the $AP$-net from Exam-
ple 1. All markings (represented by nodes in $OG$) reachable from a start marking
are shown in Table 2. Arcs are labeled with names of fired transitions and bind-
ings. For example, the first arc is labeled with $(t_7, < x_M = M_1, y_H = H_2 >)$,
where $t_7$ is transition, $< x_M = M_1, y_H = H_2 >$ is a binding.

Table 2. Markings for the $AP$-net presented in Figure 1.

<table>
<thead>
<tr>
<th>Node in $OG$</th>
<th>$M(p_H)$</th>
<th>Marking of places</th>
<th>$M(p_M)$</th>
<th>$M(p_T)$</th>
<th>$M(p_F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>$0/H_1 + 0/H_2$</td>
<td>$1/M_1 + 0/M_2 + 0/M_3$</td>
<td>$0/T_1 + 0/T_2 + 0/T_3$</td>
<td>$0/F_1 + 0/F_2$</td>
<td></td>
</tr>
<tr>
<td>$M_1$</td>
<td>$0/H_1 + 1/H_2$</td>
<td>$1/M_1 + 0/M_2 + 0/M_3$</td>
<td>$0/T_1 + 0/T_2 + 0/T_3$</td>
<td>$0/F_1 + 0/F_2$</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>$0/H_1 + 1/H_2$</td>
<td>$1/M_1 + 0/M_2 + 0/M_3$</td>
<td>$0/T_1 + 0/T_2 + 0/T_3$</td>
<td>$0/F_1 + 0.66/F_2$</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>$0/H_1 + 1/H_2$</td>
<td>$1/M_1 + 0/M_2 + 0/M_3$</td>
<td>$0/T_1 + 0/T_2 + 0/T_3$</td>
<td>$0.33/F_1 + 0/F_2$</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>$0/H_1 + 1/H_2$</td>
<td>$1/M_1 + 0/M_2 + 0/M_3$</td>
<td>$0/T_1 + 0/T_2 + 0/T_3$</td>
<td>$0.33/F_1 + 0.66/F_2$</td>
<td></td>
</tr>
</tbody>
</table>
6 Analysis of AP-nets by Means of Occurrence Graphs

Let AP-net be an approximate Petri net, $|M_0|$ is the reachability set, $C$ - the cardinality of the $|M_0|$, and $OG$ - an occurrence graph of AP-net. We use the occurrence graph in order to search for the *best decision*, i.e., the element of fuzzy set in marking of decision place, which value of membership function is the highest. Moreover, it is also very important for us to find the shortest distance from the start marking to the final marking with the best decision (measured by the number of firable transitions).

Below, we present an algorithm to analyze the occurrence graph with respect to given criteria. In the algorithm we use the following auxiliary notations: $V$ denotes a set of all nodes in $OG$, $Table(C)$ - a procedure to create a new table $Tab$ of size $(C)$, $GetMax(M^d_c)$ - a function returning the height of fuzzy set associated with marking in decision place $M^d_c$, $SearchMax()$ - a function returning a maximum in a table $Tab$, $Max$ - a search maximum value (a value returned by a function $SearchMax()$), $BFS(Max)$ - a function finding all nodes in $OG$ including searched element $Max$ in marking for a decision in all possible states of AP-net, $Nod$ denotes a vector of nodes, $ShortDist()$ - a function finding the shortest distance between nodes $M_0$ and all nodes in $Nod$ and returning the best decision in a vector $BD$. By a distance we understand the sequence of firing transition to get the best decision.

**ALGORITHM 2** for search of the best marking for all decisions in the occurrence graph $OG$.

**INPUT:** An occurrence graph $OG$.

**OUTPUT:** The shortest distance between nodes $M_0$ and all nodes which are associated with markings having the best decision.

**begin**

$Nod = 0$

$Table(C)$

for $c=1$ to $C$

\[ Tab[c] = GetMax(M^d_c) \]

$Max = SearchMax()$

$Nod = BFS(Max)$

$BD = ShortDist()$

**end.**

In this algorithm we use *Breath First Search method* (see e.g. [2]). This algorithm traverses a connected component of a given graph. It starts at a given node, which is at the level 0 (it is represented by a start marking $M_0$). In the first stage, we visit all nodes at the level 1. In the second stage, we visit all nodes at the level 2. The new nodes, which are adjacent to the level 1, and so on. This algorithm terminates when every node has been visited. It is also possible to use *Depth First Search method* (see e.g. [2]) in this algorithm.
Example 6. From the $AP$-net from Figure 1 the best decision is equal to 0.66 for the element $F_2$ in the decision place $pF$. It is contained in the node $M_2$. We have to fire two transitions $t_7$, $t_5$ in order to compute this value for $F_2$. The same value we can also get by firing three transitions $t_7$, $t_4$, $t_5$.

7 Concluding Remarks

In the paper a method of analysis of occurrence graphs for $AP$-nets has been presented. Such graphs may be used for verifying and evaluating an approximate reasoning realized by a given $AP$-net constructed automatically on the base of the decision system. The presented approach seems to have some significance for real-time knowledge-based control systems. This aspect of the net model will be considered in the next paper. Giving time to the net model broadens a region of applications of $AP$-nets.

Acknowledgment. The research has been partially supported by grant 3 T11C 005 28 from the Ministry of Scientific Research of Information Technology of the Republic of Poland.

We would like to thank Krzysztof Pancerz for his useful suggestions to improve the presentation of the paper.

References