Abstract

Petri net theory has been a successful tool for the study of systems because it allows their detailed and precise mathematical representation. Analysis of Petri nets can reveal important information about the structural and dynamic behavior of the modeled system, and this information can then be used for evaluation and to suggest improvements or changes. This paper presents a new algorithm for calculating deadlocks and traps in Ordinary Petri nets. The algorithm is independent of the form of the Petri net being analyzed, and does not impose any structural restrictions as is the case with the rest of the reported algorithms. In an illustration of the usefulness of this type of analysis for C2 systems, the deadlocks and traps in a Petri net representing the decision process of an organization are shown to reveal certain logical errors in the set of decision rules.

1 INTRODUCTION

Several C2 systems have been modeled, tested, and analyzed by modeling and analytical tools available in Petri net theory (Gaylord et al. 1991; Zhuo and Levis, 1991; Shah and Levis, 1993, Prabhakar and Levis, 1993). The analysis tools of Petri nets have been used for evaluating performance, timeliness, accuracy, and other structural and behavioral properties of the C2 systems. On a parallel track, researchers have been modeling organizational decision procedures with the help of Petri nets. These models, by virtue of being executable, help determine the behavior of the organizations that implement these sets of rules. The analysis of decision making rules by S-invariants and Occurrence graphs of the Petri net models representing these rules has been shown to reveal logical inconsistencies, redundancies, and incompleteness in the rule bases (Zaidi and Levis, 1995).

A less extensively used analysis technique in Petri net theory is the deadlock and trap analysis. The identification of deadlocks and traps in a Petri net structure representing a physical system might help identify certain structural and dynamic characteristics of the system prior to implementing it. Research on deadlock and trap analysis started in the early 70’s, as indicated in Silva and Esparza (1990). Basic deadlock and trap problems in Petri nets have been resolved to a certain extent, but computation of deadlocks and traps has been limited to strongly connected Free Choice (FC) nets, which are a subset of Ordinary Petri nets. The classical methods use Boolean equations, sometimes translated into linear inequalities (Silva and Esparza, 1990). An alternative approach was studied by Lautenbach (1987), in which deadlocks and traps were related to special P-semiflows of an associated net, thus opening up the possibility of applying the S-invariant method to the calculation of strongly connected deadlocks and traps in strongly connected FC nets. This algorithm requires transformation of the net into an associated marked graph, which may become a very tedious procedure for very large Petri net structures.

The practical application of the deadlock and trap theory requires efficient algorithms for more general forms of Petri nets. Because of the complexity of possible net transformations and the limited types of deadlocks and traps that can be calculated by the currently available algorithms, a more general and easy to implement algorithm is needed to find deadlocks and traps in Ordinary Petri nets.

The next section, Section 2, presents the terminology used throughout this paper. Section 3 describes a new algorithm for calculating deadlocks and traps in Ordinary Petri nets that imposes no structural restrictions on their form. The algorithm generalizes the classical solution to the deadlock and trap problem without complex transformations of the original net. In an illustration of the usefulness of the approach to C2 systems, the algorithm is applied to a Petri net representing a rule base. The results of the analysis are shown to identify certain patterns of Petri net structures that correspond to cycles and inconsistencies in the rule base.

2 DEFINITIONS

Deadlocks are sets of places in a Petri net structure which remain unmarked once they lose their markings. Traps, on the contrary, are sets of places which remain marked once they have gained at least one token. A formal definition of traps and deadlocks is given as:
Definition  Deadlock and trap
Let \( N = (P, T, I, O) \) be a Petri net.
- A nonempty set \( D \subseteq P \) is called a deadlock if and only if \( *D \subseteq D^* \).
- A nonempty set \( Q \subseteq P \) is called a trap if and only if \( Q^* \subseteq *Q \).

Definition  Minimal Deadlocks and traps
- Let \( D \) be a deadlock; \( D \) is called minimal if there is no deadlock contained in \( D \) as a proper subset.
- Let \( Q \) be a trap. \( Q \) is called minimal if there is no trap contained in \( Q \) as a proper subset.

Figure 1 presents an example of a Petri net and identifies the minimal traps and minimal deadlocks in the structure.

As is apparent from the definitions, the deadlocks of a Petri net \( PN \) are the traps of a Petri Net \( PN' \) which is obtained by reversing the orientation of the arcs in \( PN \). Due to this relationship between traps and deadlocks of a Petri net structure, the algorithm in this paper is presented with the help of deadlocks only. The algorithm, if applied to \( PN' \) will yield traps in the net \( PN \).

Definition  Deadlock Support
Given a net \( N = (P, T, I, O) \), a set of places \( X \), where \( X \subseteq P \), is a deadlock support if:
- \( X \) is a deadlock: \( *X \subseteq X^* \), and
- These places in \( X \) are in the same directed path.
The trap support is denoted as \( d<X> \).

Deadlock Support Properties
- A deadlock support \( d<W> \) is included in a deadlock support \( d<U> \) if and only if \( d<W> \subseteq d<U> \) \hspace{1cm} (1)
- The union of deadlock supports may not be a deadlock support.
- A deadlock can either be a deadlock support or the union of deadlock supports.
- A deadlock support is minimal if it does not include any other deadlock supports but itself.

Example

![Figure 2 Trap Support Example]

In the PN of Figure 2, all the deadlock are:
\{p1\}, \{p2\}, \{p1, p3\}, \{p1, p4\}, \{p2, p3\}, \{p2, p4\},
\{p1, p2\}, \{p1, p3, p4\}, \{p2, p3, p4\}, \{p1, p2, p3\},
\{p1, p2, p4\}, \{p1, p2, p3, p4\}

The deadlock supports are:
\{p1\}, \{p2\}, \{p1, p3\}, \{p1, p4\}, \{p2, p3\}, \{p2, p4\}

The rest of the deadlocks are not deadlock supports.

Definition  S-component of a Deadlock
The S-component associated with a deadlock \( D \) is the subnet whose places are the places of \( D \) and whose transitions are the transitions in \( *D \).

Definition  S-component of a Trap
The S-component associated with a trap \( Q \) is the subnet whose places are the places of \( Q \) and whose transitions are the transitions in \( Q^* \).

Definition  Extended Incidence Matrix
An Ordinary Petri net with \( n \) places and \( m \) transitions can be represented by a \( n \times m \) matrix \( C \), the Extended Incidence Matrix. The rows correspond to places, the columns correspond to transitions.

\- \( C_{ij} = -2 \) if there is a directed arc from the j-th transition to the i-th place and a directed arc from i-th place back to j-th transition.
\- Otherwise,
\- \( C_{ij} = 1 \) if there is a directed arc from the j-th transition to the i-th place.
\- \( C_{ij} = -1 \) if there is a directed arc from the i-th place to the j-th transition.
\- \( C_{ij} = 0 \) if there is no arc from the j-th transition to the i-th place.

Note that \( C \) allows a representation of self loops present in a PN structure in the matrix form which is absent in the conventional Incidence matrix definition.

Example
The extended incidence matrix \( C \) of the net in Figure 3 is given as:
Definition Preset-Postset Relation (PPR) Matrix

Given a Petri net $N = (P, T, I, O)$, $n = |P|$, $m = |T|$. A place set $PS_i$ is a subset of $P$. Suppose $n_r$ such place sets are chosen. The Preset-Postset relation is represented by a $n_r \times m$ matrix denoted as $R$ (PPR matrix) where the rows represent the place sets and the columns the transitions. $R_{ij}$ denotes the element in the $i$-th row and the $j$-th column in the matrix; it can take one of the following values:

- $R_{ij} = -1$, if place set $PS_i$ has the property that transition $t_j \in PS_i^*$.
- $R_{ij} = 1$, if place set $PS_i$ has the property that transition $t_j \in ^*PS_i$, and $t_j \notin PS_i^*$.
- $R_{ij} = 0$, if place set $PS_i$ has the property that transition $t_j \notin PS_i^*$ and $t_j \notin ^*PS_i$.
- $R_{ij} = -2$, if a place set $PS_i$ has the property that a transition $t_j$ has a self loop, where $t_j \in ^*PS_i$ and $t_j \in PS_i^*$.

Example

Given the net in Figure 3, the PPR matrix is:

$P = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & -1 & 0
\end{bmatrix}$

$C = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 1 & -1
\end{bmatrix}$

$R = \begin{bmatrix}
-1 & 0 & 0 & -1 & 0 & 0 \\
1 & -1 & 0 & 1 & -1 & 0 \\
0 & 1 & -1 & 0 & 1 & -1 \\
0 & 0 & -2 & 0 & 0 & -2 \\
-1 & -1 & 0 & 1 & -1 & 0 \\
-1 & -1 & 0 & 1 & -1 & 0
\end{bmatrix}$

From the illustration in Table 1, it can be seen that $\{p1, p2\}$'s PPR element in column $t1$ can be obtained from $\{p1\}$ and $\{p2\}$'s PPR elements in column $t1$. Rules of operation are, therefore, set up to derive the additional place set rows in the PPR matrix. If place sets that are related to deadlocks and traps are derived in the PPR matrix, deadlocks and traps can be found. The commutative operation rules are defined on the PPR matrix elements, as follows:

1. $1 + 1 = 1$; 6. $(-1) + 0 = -1$;
2. $0 + 0 = 0$; 7. $(-2) + 1 = -1$;
3. $1 + 0 = 1$; 8. $(-2) + (-1) = -1$;
4. $(-1) + (-1) = -1$; 9. $(-2) + 0 = -1$;
5. $(-1) + 1 = -1$; 10. $(-2) + (-2) = -1$.

In the PPR matrix the union of place sets, represented as rows, is therefore calculated by adding two rows with the operation applied to corresponding elements in both rows to form a new row. The new row represents the union of the two place sets. For the rationale behind these operation rules, see Jin (1994).

From the PPR matrix the union of place sets, as rows, is therefore calculated by adding two rows with the operation applied to corresponding elements in both rows to form a new row. The new row represents the union of the two place sets. For the rationale behind these operation rules, see Jin (1994). Contrary to the PPR union operations, the usual set-theoretic union operation is carried out by the following commutative rules:

1. $1 + 1 = 1$; 3. $1 + 0 = 1$;
2. $0 + 0 = 0$; 4. $1 + 1 = 1$;
3. $1 + 0 = 1$.

In matrix $R$, if a row $j$ has all its elements be either -1 or 0, then its corresponding place set satisfies the deadlock definition and therefore is a deadlock. In the illustration, row 6 presents such a situation. So $\{p1, p2\}$ is a deadlock, while place set $\{p2, p3\}$ in row 5 is not.

The preset and postset relations of place set $\{p1, p2\}$ in row 6 can be derived from row 1 and row 2. Similarly, that of place set $\{p2, p3\}$ in row 5 can be derived from row 2 and row 3. Table 1 presents this derivation.

Table 1 Derivation of Rows in PPR Matrix

<table>
<thead>
<tr>
<th>place set</th>
<th>PPR element</th>
<th>PP relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t1 column)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${p1}$</td>
<td>-1</td>
<td>$t1 \in (p1)^*$</td>
</tr>
<tr>
<td>${p2}$</td>
<td>1</td>
<td>$t1 \in ^*(p2)$,</td>
</tr>
<tr>
<td>${p1, p2}$</td>
<td>(-1) + 1 = (-1)</td>
<td>$t1 \in (p1, p2)^*$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>place set</th>
<th>PPR element</th>
<th>PP relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t2 column)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${p1}$</td>
<td>0</td>
<td>$t2 \in (p1)^*$</td>
</tr>
<tr>
<td>${p2}$</td>
<td>-1</td>
<td>$t2 \in ^*(p2)$,</td>
</tr>
<tr>
<td>${p1, p2}$</td>
<td>0</td>
<td>$t2 \in (p1, p2)^*$</td>
</tr>
<tr>
<td>${p1, p2}$</td>
<td>-1</td>
<td>$t2 \in ^*(p1, p2)$,</td>
</tr>
</tbody>
</table>

3 THE ALGORITHM

Given a net $N = (P, T, I, O)$, $n = |P|$, $m = |T|$, the extended incidence matrix of the net $N$ is $C$. $I_n$ is an identity matrix with dimension equal to the number of places in the net $N$. The algorithm starts by constructing the matrix $[I_n, C]$ and evolves to $[D_i, R_i]$. $[I_n, C]$ will be modified by linear combinations of its rows. Matrix $D_i$ includes place sets of deadlock supports of net $N_i$, where $N_i$ denotes the net obtained by taking transitions $t_1, \ldots, t_i$ into account in the PPR matrix. Matrix $R_i$ represents the PPR matrix that originated from the extended incidence matrix. In short, $[D_i, R_i]$ is the matrix obtained after the $i$-th iteration in the following algorithm. After the $m$-th iteration, $D_m$ includes the deadlock support for the net $N_m$, which is the net $N$, and $R_m$ shows the PPR matrix of certain place sets in net $N$. 

-3-
Some redundancy in the deadlock supports may occur, and can be eliminated in the end.

**Step 1 Initialization**

Construct the matrix \([ I_n, C ]\).

Let \(D_0 = I_n; R_0 = C\); \(i = 0\).

**Initial Checking:**

Check all rows in \([D_0, R_0]\) to determine if there is a row \(r\) that has all its \(R_0\) matrix elements \(R_{rj}\) (\(j = 1\) to \(m\)) be negative or zero; if so, append this whole row \(r\) to the matrix \([D_0, R_0]\).

**Step 2 Iterations**

*Repeat* for \(i = 1\) to \(m\) (\(m\) is the number of transitions in net \(N\))

- Determine \(J = \{ j \mid R_{ji} > 0 \}\) and \(K = \{ k \mid R_{ki} < 0 \}\).
- For each \((j, k)\) of \(J \times K\), add row \(j\) and row \(k\) in \([D_0, R_0]\) element by element. The adding operation of \(D_i\) matrix and \(R_i\) matrix follows set-theoretic union rules and PPR union operation rules; then append the resulting rows to the matrix \([D_i, R_i]\).
- Suppress all the rows in \(J\) in which elements \(R_{ji} > 0\) (\(j \in J\)).
- Eliminate from \([D_i, R_i]\) the identical rows. Two rows are identical if and only if every element in one row in matrix \(D_{i-1}\) has the same value as the corresponding element in another row in \(D_{i-1}\).
- Now we have a new matrix \([D_i, R_i]\).
- Go to the *Repeat* statement.

**Step 3 Check**

Check rows in matrix \([D_m, R_m]\). Find out those rows in \(R_m\) in which every element in the row is either negative or zero. The corresponding rows in \(D_m\) represent the place sets of deadlock supports. If there are no such rows in \(R_m\), then this net does not have any deadlock supports, which means there are no deadlocks in this net.

**Step 4 End of the Algorithm.**

Those place sets that have subsets are eliminated in each iteration in Step 2, and, finally, all minimal deadlocks are determined at the end of the algorithm.

The traps of a PN can be calculated by applying the presented algorithm to the PN' (net obtained by reversing the arcs of PN). The deadlocks of PN' are traps of PN.

### 3.1 Example

The algorithm is applied to the net \(N\) shown in Figure 4. The following steps are performed:

**Step 1: Initialization**

- Construct matrix \([I_n, C]\). \(C\) is the extended incidence matrix of net \(N\). \(I_n\) is the identity matrix, \(n = |P|\). This is presented in Figure 5.
- Let \(D_0 = I_n; R_0 = C\).

**Step 2: Iterations**

**Iteration 1:** Add transition \(t_1\) to net \(N_0\), the deadlock supports are shown in Figure 6.

- \(J = \{ j \mid R_{j1} > 0 \} = \{2\}\),
- \(K = \{ k \mid R_{k1} < 0 \} = \{5\}\),
- \(J \times K = \{(2, 5)\}\),
- Add rows in \(J \times K\) according to \(D_i\) matrix union rules and \(R_i\) PPR union operation rules, append the result row 7 to the matrix \([D_0, R_0]\).
- Suppress rows in \(J\), which in this case is row 2.
- There are no identical rows in \([D_1, R_1]\), so there are no rows to be eliminated.

Matrix \(D_1\) contains the deadlock supports of net \(N_1\). They are \(\{p1\}, \{p3\},\) and \(\{p1, p2\}\). Iteration 1 is shown as follows:

**Figure 6 Iteration 1– Example**

**Iteration 2:** Transition \(t_2\) is added to net \(N_1\), now we have \(N_2\) as shown in Figure 7.
\[ J = \{ j \mid R_{j2} > 0 \} = \{ 4 \}, \]
\[ K = \{ k \mid R_{k2} < 0 \} = \{ 6, 7 \}, \]
\[ J \times K = \{(4, 6), (4, 7)\}; \]
add rows in \( J \times K \) respectively (according to \( D \) matrix union rules, and \( R \) \( PPR \) union operation rules); append the results row 8 and row 9 to the matrix \([D_1, R_1]\).

• Suppress rows in \( J \), which in this case is row 4.
• No two rows are identical, so no row is eliminated.

The result of iteration 3 is the same as iteration 2.

\[ \begin{align*}
\{ 1000 \} & \Rightarrow \{p1\} \\
\{ 0010 \} & \Rightarrow \{p3\} \\
\{ 1100 \} & \Rightarrow \{p1, p2\} \\
\{ 1010 \} & \Rightarrow \{p1, p3\} \\
\{ 1101 \} & \Rightarrow \{p1, p2, p4\}
\end{align*} \]

Figure 9 Result -- Example

The traps of the net are also calculated and are given as follows:

\{p2, p3\}, \{p3\}, and \{p1, p2, p3\}

4 APPLICATION

This section presents an illustration of the usefulness of the analysis for organization design problem. The illustrative example contains a set of decision rules that are to be assigned to decision makers in an organization. This set of decision rules are transformed to an equivalent Petri net (PN) representation by the methodology proposed by Zaidi and Levis (1995). The traps of this PN are shown to identify certain patterns of PN structure that correspond to cycles and inconsistencies in the rule base.

The example set of organizational decision rules are taken from the illustration of Zaidi and Levis (1995) which was motivated by the "Message Puzzle Task (MPT)" of Wesson and Hayes-Roth (1980). The MPT involved a game-like environment in which words and phrases move about in a two-dimensional grid that resembles a puzzle board. A group of players, each of whom can see a portion of the grid, must communicate among themselves to identify the moving items and eventually solve the puzzle. In the illustration presented in this section, a 4x3 grid (consisting of 12 sectors) representing the puzzle board is considered (Figure 10). The messages on the grid appear on certain sectors in the grid. The messages consist of letters from an input alphabet of integers, where each integer represents an event. Based on the appearance of these messages in certain sectors, the set of rules infers a sequence of events out of a possible three sequences in which these events can occur.
The set of all events is given as: \( E = \{1, 2, 3, 4, 5, 6\} \), where each integer represents the occurrence of an event. The appearance of an integer (from set \( E \)) in one of the sectors of the grid is considered as the basic input to the set of decision rules. The following 12 basic inputs are identified:

\[
U = \{P_1, P_2, P_3, ..., P_{12}\} \tag{1}
\]

where the proposition symbols \( P_1-P_{12} \) represent the following information from the grid:

- \( P_1: \) Integer 1 in Sector 1
- \( P_2: \) Integer 2 in Sector 2
- \( P_3: \) Integer 3 in Sector 3
- \( P_4: \) Integer 1 in Sector 4
- \( P_5: \) Integer 2 in Sector 5
- \( P_6: \) Integer 3 in Sector 6
- \( P_7: \) Integer 4 in Sector 7
- \( P_8: \) Integer 4 in Sector 8
- \( P_9: \) Integer 5 in Sector 9
- \( P_{10}: \) Integer 6 in Sector 10
- \( P_{11}: \) Integer 5 in Sector 11
- \( P_{12}: \) Integer 6 in Sector 12

Based on these inputs, a decision process tries to interpret the inputs in terms of three possible outcomes (sequence of events), which are characterized as the main concepts of the rule base:

\[
\Psi = \{A, B, C\} \tag{4}
\]

where

- \( A: \) Sequence of Events is 4, 5, 6, 3, 2, 1
- \( B: \) Sequence of Events is 1, 2, 3, 4, 5, 6
- \( C: \) Sequence of Events is 6, 5, 4, 3, 2, 1

The objective of the organization design problem is to decompose and partition the decision process into several decision making rules so that the rules can be assigned to five decision makers (DM), shown in the organizational hierarchy of Figure 11. The physical interactional structure (shown by arrows in the figure) in Figure 11 imposes certain restriction on how and what rules are decomposed and partitioned. A hypothetical decomposed rule base \( RB_1 \) is taken for the illustration. In the decomposed rule base \( RB_1 \) rules of the form "\( \text{R}_i \land \text{R}_j \land ... \land \)" represent the rules assigned to DM1. The \( \text{R}_i \)'s represent the responses of the lower level decision makers communicated to DM1. Similarly, the set of rules defined at the intermediate layer is assigned to DM2, where the rules are of the form "\( \text{Q}_i \land \text{Q}_j \land ... \land \)". Finally, the set of rules at the lowest layer of the rule base is further decomposed horizontally (Mesarovic, 1970) into three sets and is assigned to three decision makers, DM3, DM4, DM5, where the rules are of the form "\( \text{P}_i \land \text{P}_j \land ... \land \)". The decomposition of the original rule base is done by taking into account the fact that the set of basic concepts \( U \) can be divided into the following three subsets (not necessarily disjoint), where each set represents information from a different sector (area of awareness) assigned to a decision maker (DM3, DM5, and DM4 respectively).

\[
U_1 = \{P_1, P_2, P_3, P_4, P_5\} \quad U_2 = \{P_6, P_7, P_8, P_9, P_{10}\} \quad U_3 = \{P_8, P_9, P_{10}, P_{11}, P_{12}\} \tag{5}
\]

**Rule Base, \( RB_1 \)**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>( P_1 \land P_2 \land P_3 \rightarrow Q_1 )</td>
</tr>
<tr>
<td>Rule 2</td>
<td>( P_1 \land P_2 \rightarrow Q_1 )</td>
</tr>
<tr>
<td>Rule 3</td>
<td>( P_6 \land P_7 \rightarrow Q_4 )</td>
</tr>
<tr>
<td>Rule 4</td>
<td>( \neg Q_1 \land Q_4 \rightarrow R_2 )</td>
</tr>
<tr>
<td>Rule 5</td>
<td>( P_4 \land P_5 \rightarrow Q_3 )</td>
</tr>
<tr>
<td>Rule 6</td>
<td>( P_3 \rightarrow Q_{10} )</td>
</tr>
<tr>
<td>Rule 7</td>
<td>( Q_{10} \land Q_3 \rightarrow R_3 )</td>
</tr>
<tr>
<td>Rule 8</td>
<td>( P_4 \land P_5 \rightarrow Q_3 )</td>
</tr>
<tr>
<td>Rule 9</td>
<td>( P_9 \land P_{12} \rightarrow Q_5 )</td>
</tr>
<tr>
<td>Rule 10</td>
<td>( P_9 \rightarrow Q_6 )</td>
</tr>
<tr>
<td>Rule 11</td>
<td>( Q_3 \land Q_5 \rightarrow R_3 )</td>
</tr>
<tr>
<td>Rule 12</td>
<td>( R_2 \land R_4 \rightarrow C )</td>
</tr>
<tr>
<td>Rule 13</td>
<td>( P_7 \land P_9 \land P_{10} \rightarrow Q_2 )</td>
</tr>
<tr>
<td>Rule 14</td>
<td>( P_8 \land P_9 \rightarrow Q_5 )</td>
</tr>
<tr>
<td>Rule 15</td>
<td>( R_1 \land R_3 \rightarrow Q_3 )</td>
</tr>
<tr>
<td>Rule 16</td>
<td>( Q_3 \land Q_5 \rightarrow R_5 )</td>
</tr>
<tr>
<td>Rule 17</td>
<td>( R_5 \rightarrow A )</td>
</tr>
<tr>
<td>Rule 18</td>
<td>( P_{11} \land P_{12} \rightarrow Q_2 )</td>
</tr>
<tr>
<td>Rule 19</td>
<td>( P_8 \rightarrow Q_8 )</td>
</tr>
<tr>
<td>Rule 20</td>
<td>( Q_2 \land Q_8 \rightarrow R_4 )</td>
</tr>
<tr>
<td>Rule 21</td>
<td>( Q_1 \land Q_2 \rightarrow R_6 )</td>
</tr>
<tr>
<td>Rule 22</td>
<td>( R_6 \rightarrow B )</td>
</tr>
<tr>
<td>Rule 23</td>
<td>( P_6 \land P_7 \rightarrow R_1 )</td>
</tr>
</tbody>
</table>

where

- \( Q_1: \) Partial Sequence of Events is 1, 2, 3
- \( Q_2: \) Partial Sequence of Events is 4, 5, 6
- \( Q_3: \) Partial Sequence of Events is 2, 1
- \( Q_4: \) Partial Sequence of Events is 4, 3
- \( Q_5: \) Partial Sequence of Events is 5, 6
- \( Q_6: \) \( P_9 \) Observed
- \( Q_8: \) \( P_8 \) Observed
- \( Q_{10}: \) \( P_3 \) Observed
- \( R_1: \) Partial Sequence of Events is 4, 3
- \( R_2: \) Partial Sequence of Events is 4, 3, 2, 1
- \( R_3: \) Partial Sequence of Events is 3, 2, 1
- \( R_4: \) Partial Sequence of Events is 6, 5
- \( R_5: \) Sequence of Events is 4, 5, 6, 3, 2, 1
- \( R_6: \) Sequence of Events is 1, 2, 3, 4, 5, 6

For illustration purposes, the following two sets of mutually exclusive propositions are also defined:
\[ \mu_1 = \{P8, P10\} \]
\[ \mu_2 = \{R1, R5\} \]  

As pointed out in (Zaidi and Levis, 1995), the way a rule base is decomposed might introduce errors in the resulting decomposed rule set. The effect of the new rules, added as a result of decomposition, on the rest of the rule base cannot be determined unless they are checked against the entire set of rules. The rest of this section presents an approach in which a decomposed rule set is first transformed into an equivalent representation and then minimal traps are calculated for it. The calculated traps of the net are shown to identify certain logical errors in the rules.

4.1 Petri Net Representation Of Rules

An individual rule of the form \(P_1 \land P_2 \ldots \land P_n \rightarrow Q\) is transformed to a PN with a single transition with \(n\) input places, each representing a single input proposition \(P_i\), and an output place representing the assertion \(Q\) (For a general and more detailed description of this technique, see Zaidi, 1994). The labels of the places and transitions correspond to the propositions and rules they represent. The entire set of decision rules is then obtained by unifying all the individual rules. The process represents the causal relationship among the rules and the facts of the knowledge base. This method unifies the rules by merging all the places with identical labels (proposition symbols) into a single place. The PN obtained as a result of applying this technique to RB1 is shown in Figure 12. In Figure 12, the basic inputs and main concepts defined for the rule base are shown aggregated into virtual inputs and outputs, \(P_{\text{in}}\) and \(P_{\text{out}}\), with transitions \(T_{\text{in}}\) and \(T_{\text{out}}\), respectively. The parts of the net in the figure that are drawn by a broken line represent this aggregation and do not represent any rule in the rule base.

4.2 Trap Analysis

Definition **Conflict-free Net**

A Petri net \((P, T, I, O)\) is a conflict-free net if \(\forall p \in P, |p^*| = 1\).

The analysis calculates the minimal traps of the PN after it is transformed to an equivalent Conflict-free net. The algorithm to convert a PN into a Conflict-free net (CFN) is given as follows:

- Merge the virtual input and output places, \(P_{\text{in}}\) and \(P_{\text{out}}\), into a single external place \(P_e\).
- \(\forall p \text{ so that } |p^*| = n (>1)\), create \(n\) copies of \(p\), denoted by \(p^i\), where \(i = 1, 2, \ldots, n\). Connect these newly created places with all the input transitions of the place \(p\) so that \(\forall i, |p^i| = |p|\). Now, connect each of these \(n\) places, \(p_i\), with only one of the \(n\) output transitions of the original place \(p\) (Figure 13 illustrates the process.)
- Delete all the dangling nodes (transitions/places)—nodes with only inputs or outputs—in the net.
The following Theorem provides a useful interpretation to the traps calculated for a CFN representing a rule base.

Theorem 1 (Zaidi, 1994)
The minimal traps of CFN are its directed elementary circuits.

The net in Figure 12 is converted to an equivalent CFN representation, and traps are calculated for it. (Table 2 lists the calculated minimal traps.)

Based on the results from Theorem 1, the following Proposition characterizes two different types of circuits in the CFN.

Proposition 1
Let Xi be the minimal trap;
• If Xi contains the external place Pe, then the S-component associated with Xi represents a directed path from a basic input to one of the main concepts.
• If Xi does not contain the external place Pe, then the S-component associated with Xi represents a loop inside the Petri net structure.

The following Proposition interprets the minimal traps for CFN in terms of circular and possible inconsistent rules. The notation p(.), q(.) used in the proposition indicates the fact that the place p might get replicated during the process of converting PN to CFN. The notation illustrates that the algorithm only identifies the predicate p with no regard to the superscript assigned to it during the conversion process.

Proposition 2
• If there exists a trap Xi which contains places p(.) and q(.), where p and q ∈ μ (set of mutually exclusive concepts), then Xi indicates the presence of inconsistent rules in the rule base. The S-components associated with Xi identify the set of inconsistent rules
• If there exist Xi and Xj, where p(i) ∈ Xi and q(j) ∈ Xj, p and q ∈ μ, and Xi ∩ Xj ≠ ∅, then Xi and Xj indicate the presence of possible inconsistent rules in the rule base. The S-components associated with Xi and Xj identify the set of inconsistent rules.
• If there exists a Xi which does not contain the external place Pe, then Xi indicates the presence of circular rules in the rule base. The transitions in Xi* identify the set of circular rules.

The first case presented in Proposition 2 corresponds to rules where it is possible to reach from a predicate "p" to its negation or to a mutually exclusive concept "q", which is not permissible in formal logic. The second inconsistent case presented corresponds to a rule which requires conflicting assertions to be simultaneously true in order to execute.

The application of this analysis performed on PN representing RB1 resulted in the following problematic cases:

• Circular Rules
  X1 and X2 indicate the presence of the following circular rules:
  Rule15 and Rule7;
  Rule15 and Rule11

Figure 14 shows the reported circular cases in the PN. In one of the reported circular cases (Rule15 and Rule7) the system infers R3 (Partial Sequence of Events in 3, 2, 1) through Q10 (P3 Observed) and Q3 (Partial Sequence of Events in 2, 1) as illustrated by Rule7. On the other hand, Rule15 requires Q3 to infer R3. Similarly in the second case (Rule15 and Rule11), Rule11 infers R3 through Q3, where Q3 is required to be inferred through R3 in Rule15.

<table>
<thead>
<tr>
<th>i</th>
<th>Minimal Traps, Xi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Q31, R3}</td>
</tr>
<tr>
<td>2</td>
<td>{Q32, R3}</td>
</tr>
<tr>
<td>3</td>
<td>{Pe, P1, Q1, R6, B}</td>
</tr>
<tr>
<td>4</td>
<td>{Pe, P1, Q1, R6, C}</td>
</tr>
<tr>
<td>5</td>
<td>{Pe, P2, Q1, R6, B}</td>
</tr>
<tr>
<td>6</td>
<td>{Pe, P2, Q1, R6, C}</td>
</tr>
<tr>
<td>7</td>
<td>{Pe, P3, Q1, R6, B}</td>
</tr>
<tr>
<td>8</td>
<td>{Pe, P3, Q1, R6, B}</td>
</tr>
<tr>
<td>9</td>
<td>{Pe, P3, Q1, R6, B}</td>
</tr>
<tr>
<td>10</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>11</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>12</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>13</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>14</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>15</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>16</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>17</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>18</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>19</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>20</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>21</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>22</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>23</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>24</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>25</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>26</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>27</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>28</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>29</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>30</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>31</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>32</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>33</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>34</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
<tr>
<td>35</td>
<td>{Pe, P4, Q1, R6, B}</td>
</tr>
</tbody>
</table>

• Inconsistent Rules
  Traps X14, and X16 indicate the presence of an inconsistent case, since both have R1 (Partial Sequence of Events is 4, 3) and R5 (Sequence of Events is 4, 5, 6, 3, 2, 1) present in them, where R1 and R5 are defined as mutually exclusive concepts. The following rules correspond to this inconsistency.
Rule15(R1 ∧ R3→Q3) and Rule16 (Q3 ∧ Q5→R5).

Traps X29, and X21 indicate the presence of another inconsistent case. The S-components corresponding to these traps reveal the fact that Rule20 requires both P10 and P8 to be present simultaneously in order to fire. P10 and P8 are defined as mutually exclusive concepts, and can not be true at the same time. Therefore, the following inconsistent rules are found:

Rule13, Rule19, and Rule20

Figure 14  Circular Cases

5 CONCLUSION

The paper presented a new algorithm for calculating deadlocks and traps in Ordinary Petri nets. The algorithm generalizes the classical solution to the deadlock and trap problem without complex transformations of the original nets. The illustrative example applied the deadlock and trap analysis to a Petri net representing a rule base. The results of the analysis were shown to identify certain patterns of Petri net structures that correspond to circular rules and inconsistency in the rule base. The application of deadlock and trap analysis on a set of decision rules is an extension of an earlier work by Zaidi and Levis (1995). The identification of problematic and erroneous rules in a rule-based (expert) system will not only help improve the performance of such systems, but will deter any chances of system-wide failures caused by these errors.

REFERENCES


Jin, Z. (1994) "Deadlock and Trap Analysis in Petri Nets," MS Thesis, Computer Science Department, George Mason University, Fairfax, VA.


