Petri Nets Modeling and Analysis Using Extended Bag-Theoretic Relational Algebra

Young Chan Kim and Tag Gon Kim

Abstract—Petri nets are a powerful modeling tool for studying reactive, concurrent systems. Analysis of the nets can reveal important information concerning the behavior of a modeled system. While various means for the analysis of the nets has been developed, a major limitation in the analysis, is explosion of large states space in simulation. An efficient method to manage large states space would overcome such a limitation. This paper proposes a framework for the modeling and analysis of Petri nets using relational database technologies. Formalism of the framework is based on a bag-theoretic relational algebra extended from the conventional. Within the framework, Petri nets are formalized by bag relations, and analysis algorithms are developed based on such formal relations. Properties associated with the nets are formalized by queries described in terms of the bag-theoretic relational algebra. The framework has been realized in a commercial relational database system using a standard SQL.

I. INTRODUCTION

Petri nets are a powerful tool for describing and studying systems that are characterized as being concurrent, asynchronous, distributed, parallel, and/or nondeterministic. A strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems. But, a major weakness is that modeled Petri nets tend to become too large for analysis even for a modest-size system. For example, Petri nets, which have been designed for the purpose of communications between finite-state machines, seem a quite natural tool for modeling the protocols [1]–[3]. Reachability analysis can be used in the protocol design phase to explore the global states of the system in order to detect undesirable behaviors such as unboundedness, deadlocks and unreachable states. While reachability analysis has been used for formal verification of protocols of low complexity, the practical use of reachability analysis for more complex interactions has been constrained by the problem of state space explosion. The size of states space feasible for full search is known as approximately $10^5$ states [4]. This problem can be alleviated by the relational approach, because relational database management systems can handle large amounts of data efficiently. This approach has been used in [5], [6]. However, formal models used in both were finite state machines, not Petri nets. Furthermore, we use a bag-theoretic relational algebra rather than relational algebra as a formal basis. In fact, a relational database management system (DBMS) was used to realize a Petri net interpreter in [7]. However, their works dealt only with algorithms designed for Petri nets simulation.

This paper proposes a framework for the modeling and analysis of Petri nets using relational database technologies extended with bag theory. Formalism of the framework is based on a bag-theoretic relational algebra. In fact, some existing database systems support bags in their data model [8]–[10]. There has also been some previous research in extending the relational model to the bag-theoretic relational model

As recognized in [12], motivations of such previous research are either to support the desired semantic modeling capability of the data model or to save the cost of duplicate elimination that is required to implement some set operations. Our motivation is similar to the former one. Since Petri nets are based on bag theory, it is natural to use bag-theoretic relational algebra. However, no previous research supports to express the number of occurrences of tuples in bag relations by queries. Therefore, above all, we extend the conventional, bag-theoretic relational algebra to provide a much better expressive power. We then develop a formal means for representing a Petri net using mathematical relations defined in the bag-theoretic relational algebra. Next we develop algorithms for generating a reachability tree from such formal relations. Finally, we develop a set of queries on the relation of the reachability tree to analyze the Petri nets.

This paper is organized as follows. In Section II, we introduce the definition and properties of Petri nets. In Section III, we describe the notation and concept of the bag-theoretic relational algebra and extend it. In Section IV, we formalize Petri nets and its properties in the bag-theoretic relational algebra. In Section V, we realize our framework in relational database management systems and explain its experiments. Conclusions are drawn in the last section.

II. PETRI NETS AND ITS PROPERTIES

In this section, we present basic definitions of Petri nets and explain their behavioral properties. For a more detailed description, refer to [13]. A formal definition of Petri nets follows.

Definition II.1: A Petri net, $PN$, is a five-tuple structure, $PN = (P, T, I, O, M_0)$ where

1) $P$ is a finite set of places.
2) $T$ is a finite set of transitions.
3) $I: P \times T \rightarrow N$ is the input function, a mapping from Cartesian product of the set of places and the set of transitions to nonnegative integers.
4) $O: T \times P \rightarrow N$ is the output function, a mapping from Cartesian product of the set of transitions and the set of places to nonnegative integers.
5) $M_0: P \rightarrow N$ is the initial marking function, a mapping from the set of places $P$ to the nonnegative integers subject to the constraint:

$$P \cup T = \emptyset.$$  

Note that the inputs and outputs of a transition are bags of places.

Terminology for Petri nets is defined below.

Definition II.2: Let $PN = (P, T, I, O, M_0)$ be a Petri net.

1) A function $M_k: P \rightarrow N$, where $k \in N$ is called a marking of $PN$. $M_k(p)$ represents the number of tokens in the place $p$.
2) A transition $t \in T$ is enabled at marking $M_k$ iff $I(p, t) \leq M_k(p), \forall p \in P$. An enabled transition may or may not fire.
3) If $t \in T$ is a transition which is enabled at $M_k$ then $t$ may fire, yielding a new marking $M_{k^t}$ given by the equation:

$$M_{k^t}(p) = M_k(p) - I(p, t) + O(t, p), \forall p \in P$$

4) Firing $t$ changes the marking $M_k$ into the new marking $M_{k^t}$; we denote this fact by $M_k \xrightarrow{\text{t}} M_{k^t}$.
5) The set of all possible markings reachable from $M_k$ in $PN$, denoted $R(M_k)$, is the smallest set of markings of $PN$ such that:

a) $M_k \in R(M_k)$

b) if $M_{k'} \in R(M_k)$ and $M_{k'} \rightarrow M_{k''}$ for some $t \in T$ then $M_{k''} \in R(M_k)$.

Many behavioral properties have been studied in Petri nets theory. Among them, we consider five properties such as **boundedness, conservation, liveness, reachability, and coverability**.

**Definition II.3:**

1) **Boundedness:**

a) $p \in P$ is n-bounded iff $\forall M_k \in R(M_0), M_k(p) \leq n$;

b) $PN$ is n-bounded iff $\forall p \in P, p$ is n-bounded;

c) $PN$ is safe iff $PN$ is 1-bounded.

d) $PN$ is bounded iff $\exists n \in N, PN$ is $n$-bounded.

2) **Conservation:**

a) $PN$ is strictly conservative iff $\forall M_k \in R(M_0), \sum_{p \in P} M_k(p) = \sum_{p \in P} M_0(p)$;

b) $PN$ is conservative with respect to a weighting function $W: P \rightarrow N$, iff $\forall M_k \in R(M_0), \sum_{p \in P} W(p) \cdot M_k(p) = \sum_{p \in P} W(p) \cdot M_0(p)$.

3) **Liveness:**

a) $t \in T$ is live iff $\forall M_k \in R(M_0), \exists M_{k'} \in R(M_k), \exists t$ is enabled at $M_{k'}$;

b) $PN$ is live iff $\forall t \in T, t$ is live.

c) $PN$ is deadlock free iff $\forall M_k \in R(M_0), \exists t \in T, t$ is enabled at $M_k$.

4) **Reachability Problem:** Given a Petri net $PN$ and a marking $M_k$, is $M_k \in R(M_0)$?

5) **Coverability Problem:** Given a Petri net $PN$ and a marking $M_k$, is there a reachable marking $M_{k'} \in R(M_0)$ such that $M_{k'} \geq M_k$?

### III. BAG-THEORETIC RELATIONAL ALGEBRA

**A. Bag-Theoretical Relational Model**

In the bag-theoretic relational model, a **database schema** is a set of **relation schemes**, each of which is a set of **attributes**. Associated with each attribute $A$ is its **domain**, $dom(A)$. There is a special attribute # such that $dom(#)$ are nonnegative integers. A **bag relation** $R$ over the relation scheme $R = \{A_1, A_2, \cdots, A_n\}$ is a finite set of pairs $(t, i)$ where $t$ is a tuple over $R$ and $i$ is a value over #. Let $r$ be a tuple in $R$. The component of $r$ corresponding to an attribute $A_i$ is denoted by $r[A_i]$. When conceived as a table, $R$ over $R$ has the following properties:

- each row $r = (t, i)$ represents a tuple of $R$ where $t = r[R]$ and $i = r[#];$
- the ordering of rows is immaterial;
- all rows are distinct from one another in content.

Fig. 1 shows examples of bag relations.

**B. Bag Relational Algebra (BRA)**

There has been some previous research in extending the relational model to the bag-theoretic relational model [11], as well as in developing techniques for algebraic query optimization for data models supporting bags [12]. These research results are briefly summarized below.

![Fig. 1. Examples of bag relations.](image)

Let $R$, $S$ be two bag relations over sets of attributes $R$, $S$, respectively, and let $X \subseteq R$, and $a \in dom(A)$.

1) **Selection:**

$$\sigma_{A=a} R = \{(t, i) | (t, i) \in R \land t[A] = a\}$$

2) **Projection:**

$$\pi_X R = \{(x, k) | x \text{ is a map on } X \text{ and } k = \sum_i i \text{ for } (t, i) \in R \text{ such that } x = t[X]\}$$

3) **Union:**

$$R \cup S = \{(t, k) | \exists i, j[(t, i) \in R \land (t, j) \in S \land k = \max(i, j)]\}$$

4) **Intersection:**

$$R \cap S = \{(t, k) | \exists i, j[(t, i) \in R \land (t, j) \in S \land k = \min(i, j)]\}$$

5) **Difference:**

$$R - S = \{(t, k) | \exists i, j[(t, i) \in R \land (t, j) \in S \land k = i - j]\}$$

6) **Addition:**

$$R + S = \{(t, k) | \exists i, j[(t, i) \in R \land (t, j) \in S \land k = i + j]\}$$

7) **Cartesian Product:**

$$R \times S = \{(t, k) | \exists r, i, s, j[(r, i) \in R \land (s, j) \in S \land t = (r, s) \land k = i \times j]\}$$

8) **Theta-Join:**

$$R \bowtie S = \{(t, k) | \exists r, i, s, j[(r, i) \in R \bowtie(s, j) \in S \land \theta \land t = (r, s) \land k = i \times j]\}$$

9) **Natural Join:**

$$R \bowtie S = \{(t, k) | \exists r, i, s, j[(r, i) \in R \bowtie(s, j) \in S \land r = t[R] \land s = t[S] \land k = i \times j]\}$$

The above definitions have a few deficiencies in that queries could not be expressed about the occurrence attribute #. Examples of such deficiencies are of the following; First, the **select** operation cannot express queries such as "list each tuple of which the occurrence number is greater than three." Second, the **natural join** cannot express queries such as "natural-join two relations left, right but use the occurrence numbers of left as the occurrence numbers of resultant tuples.

Note that tabular representation of a bag relation over $R$ is exactly same to tabular representation of a set-theoretical relation.
over \( R \cup \{ \# \} \). It is natural that \( \# \) is treated as a general attribute. So, we will extend the select, Cartesian product, theta-join, and natural join operations with such notion.

Let \( F \) be a predicate involving constants or attributes including \# as operands. Let \( R, S \) be two bag relations over sets of attributes \( R, S \), respectively, and let \( \vartheta \) be an arithmetic expression involving \( R, S, \# \), or constants as operands.

1) **Extended Selection**: The extended selection of \( R \) on \( F \) is

\[
\sigma_F R = \{(t, i) \mid \{t, i\} \in R \land F\}
\]

The following bag relation is obtained by applying \( \sigma_{\# > 1} \) to \( R \) of Fig. 1

\[
\sigma_{\# > 1} R = \begin{vmatrix}
A & B & \# \\
a_2 & b_1 & 2
\end{vmatrix}
\]

Observe that this query cannot be accepted in the previous bag-theoretic relational algebra.

2) **Extended Cartesian Product**: The extended Cartesian product of \( R \) and \( S \) on \( \vartheta \) is

\[
R \times^{\vartheta} S = \{(t, k) \mid \exists r, i, s, j, s(i, r) \in R \land \{s, j\} \in S \land t = \{r, s\} \land k = \vartheta\}
\]

The extended Cartesian product of \( R \) and \( S \) of Fig. 1 on \( S, \# - R, \# \) gives

<table>
<thead>
<tr>
<th>A</th>
<th>R.B</th>
<th>S.B</th>
<th>C</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>b_2</td>
<td>b_1</td>
<td>c_1</td>
<td>2</td>
</tr>
<tr>
<td>a_2</td>
<td>b_2</td>
<td>b_2</td>
<td>c_3</td>
<td>3</td>
</tr>
<tr>
<td>a_2</td>
<td>b_1</td>
<td>b_1</td>
<td>c_1</td>
<td>1</td>
</tr>
<tr>
<td>a_2</td>
<td>b_2</td>
<td>b_2</td>
<td>c_2</td>
<td>2</td>
</tr>
</tbody>
</table>

Note that \( R \times R, \# \times S, \# \equiv R \times S \).

3) **Extended Theta-Join**: The extended theta-join of \( R \) and \( S \) on \( \# \) and \( \vartheta \) is

\[
R \bowtie^{\vartheta}_{\#} S = \{(t, k) \mid \exists r, i, s, j, s(i, r) \in R \land \{s, j\} \in S \land t = \{r, s\} \land k = \vartheta\}
\]

The extended theta-join of \( R \) and \( S \) of Fig. 1 on \( R, \# + S, \# \) gives

<table>
<thead>
<tr>
<th>A</th>
<th>R.B</th>
<th>S.B</th>
<th>C</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>b_2</td>
<td>b_2</td>
<td>c_3</td>
<td>5</td>
</tr>
<tr>
<td>a_2</td>
<td>b_1</td>
<td>b_1</td>
<td>c_1</td>
<td>5</td>
</tr>
</tbody>
</table>

Note that \( R \bowtie^{\vartheta}_{\#} R, \# \times S, \# \equiv R \bowtie^{\vartheta} S \).

4) **Extended Natural Join**: The extended natural join of \( R \) and \( S \) on \( \# \) and \( \vartheta \) is

\[
R \bowtie^{\vartheta} S = \{(t, k) \mid \exists r, i, s, j, s(i, r) \in R \land \{s, j\} \in S \land t = \{r, s\} \land k = \vartheta\}
\]

The extended natural join of \( R \) and \( S \) of Fig. 1 on \( R, \# \) gives

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>b_2</td>
<td>c_3</td>
<td>1</td>
</tr>
<tr>
<td>a_2</td>
<td>b_1</td>
<td>c_1</td>
<td>2</td>
</tr>
</tbody>
</table>

Note that \( R \bowtie^{\vartheta} R, \# \times S, \# \equiv R \bowtie^{\vartheta} S \).

---

**IV. FORMALIZATION OF PETRI NETS ANALYSIS IN BRA**

In this section we formalize Petri nets and their analysis in the bag-theoretic relational algebra (BRA).

**A. Formalism of Petri Nets and Reachability Tree in BRA**

Petri nets are formalized in bag relations in the following:

**Definition IV.1**: A Petri net in BRA is defined as a five-tuple of bag relations

\[
P_{BRA} = (P, T, I, O, M)
\]

where

1) \( P \) is a bag relation on \( P = \{ pid \} \) where \( dom(pid) \) is the set of place identifiers.
2) \( T \) is a bag relation on \( T = \{ tid \} \) where \( dom(tid) \) is the set of transition identifiers.
3) \( I \) is a bag relation on \( I = \{ pid, tid \} \) where \( dom(pid) \), \( dom(tid) \) are the set of place identifiers, the set of transition identifiers, respectively.
4) \( O \) is a bag relation on \( O = \{ tid, pid \} \) where \( dom(pid) \), \( dom(tid) \) are the set of place identifiers, the set of transition identifiers, respectively.
5) \( M \) is a bag relation on \( M = \{ mid, pid \} \) where \( dom(mid) \), \( dom(pid) \) are the set of marking identifiers, the set of place identifiers, respectively.

with the following constraints:

1) \( dom(pid) \rightarrow dom(\#) \) in \( P \).
2) \( dom(tid) \rightarrow dom(\#) \) in \( T \).
3) \( dom(pid) \times dom(tid) \rightarrow dom(\#) \) in \( I \).
4) \( dom(tid) \times dom(pid) \rightarrow dom(\#) \) in \( O \).
5) \( dom(mid) \times dom(pid) \rightarrow dom(\#) \) in \( M \).
6) \( M \) has \( M_0 \) as an initial value.

---

1) In Database discipline, it is called a functional dependency that hold for relation scheme \( P, pid \rightarrow \# \). It means that the values of a tuple on the set of attributes, \( \{ pid \} \) uniquely determine the values on the set of attributes, \( \{ \# \} \).
On the other hand, the performance of operators of BRA heavily depends on the available access indexes (B-Tree or Hash) to the referred relations. Nevertheless, queries expressed in BRA cannot refer to index files and the details of physical storage of data. To fill this gap, the query optimizer of database systems should be used to select the best method of evaluating such queries. Thus, it is difficult to estimate the exact time complexity of BRA queries.

The special symbol, $\omega$, is necessary for the construction of the reachability tree. The rules for $\omega$ are:

$$n < \omega \text{ for all } n \in N;$$

$$\omega + \omega = \omega + n = \omega - n = \omega \text{ for all } n \in N;$$

$$\omega \leq \omega.$$  

Let $PN_{BRA}$ be a Petri net. The algorithm for generating the reachability tree are as follows:

1) **Enabled-Transition Operation $ET(M_k)$:** Given $M_k$, this operation, denoted $ET(M_k)$, yields a bag of enabled transitions in the marking $M_k$. It is defined by the following function:

\[
\begin{align*}
\text{begin} \\
\text{Marked\_Input} := \\
\pi_{\text{pid,tid}}[T \bowtie (I \bowtie (I \bowtie M_{\text{pid}}) \bowtie O) \bowtie M_{\#}]; \\
\text{Missing\_Input} := I - \text{Marked\_Input}; \\
\text{Not\_Enabled} := \pi_{\text{tid}}(\text{Missing\_Input}); \\
\text{return} T - \text{Not\_Enabled}; \\
\end{align*}
\]

2) **Next-State Operation $NS(M_k, t)$:** This operation $NS(M_k, t)$ yields the new marking (state) which results from firing the transition $t$ in the marking $M_k$. It is defined by the following function:

\[
\begin{align*}
\text{begin} \\
\text{Pre} := \pi_{\text{pid}}[\sigma_{\text{tid}}(I)]; \\
\text{Pos} := \pi_{\text{pid}}[\sigma_{\text{tid}}(O)]; \\
\text{return} M_k - \text{Pre} + \text{Pos}; \\
\end{align*}
\]

3) **Checking-Duplicated-Marking Operation $DUP(k)$:** This operation $DUP(k)$ returns true if $M_k$ had been already explored during the reachability tree generation. Otherwise, it returns false. It is defined by the following function:

\[
\begin{align*}
\text{begin} \\
\text{for each} \ (p_{\text{mid}}, t_{\text{id}}, c_{\text{mid}}, \text{type}, \text{cnt}) \in R \text{ do} \\
\text{if} \ M_k = M_{c_{\text{mid}}} \text{ then return "true"}; \\
\text{endif} \\
\text{done} \\
\text{return "false"}; \\
\end{align*}
\]

4) **Update-Marking Operation $UM(New_M, parent_{\text{mid}})$:** Given $New_M$, which is immediately reachable from $M_{parent_{\text{mid}}}$, if there exists a path from the root to $M_{parent_{\text{mid}}}$ containing a marking $M_k$ such that $New_M > M_k$ and for each place $p$ of $PN_{BRA}$, if $t_1[#] > t_2[#]$ where $t_1 = \sigma_{\text{pid}} = p New_M$ and $t_2 = \sigma_{\text{pid}} = p M_k$, then replace $t_1[#]$ by $\omega$. It is defined by the
following procedure:

```
begin
  k := parent.mid;
  while k ≠ NULL do
    if NewM > Mk
      then for each \( \langle \text{pid}_1, \text{cnt}_1 \rangle \in \text{NewM} \) do
        \( \langle \text{pid}_2, \text{cnt}_2 \rangle := \sigma_{\text{pid} = \text{pid}_1, \text{M}_k} \);
        if \( \text{cnt}_1 > \text{cnt}_2 \) then \( \text{cnt}_1 := \omega \);
      endif
    done
  endif
  done
  t := \sigma_{\text{mid} = k}(R);
  k := t[pmid];
end
```

5) **Reachability-Tree Operation RT**: Using the above defined operations, we could construct the reachability tree of a Petri net as follows.

```
begin
  k := 0; R := \emptyset;
  frontier_nodes := \{(NULL, NULL, k, NULL, 1)\};
  while frontier_nodes ≠ \emptyset do
    (* Without loss of generality we assume
     * frontier node is queue *)
    x := dequeue(frontier_nodes);
    (* Extract front element from queue *)
    if DUF(x[pmid])
      then x[type] := DUPLICATE;
          R := R \cup \{x\};
          continue;
    endif
    enabled_transitions := ET(x[cmdid]);
    if enabled_transitions ≠ \emptyset
      then x[type] := TERMINAL;
          R := R \cup \{x\};
          continue;
    endif
    x.type := INTERNAL;
    R := R \cup \{x\}
  for t ∈ enabled_transitions do
    M' := NS(x[cmdid], t);
    UM(M', x[cmdid]);
    k := k + 1;
    M := M \cup \{(k, 1)\} \times M';
    frontier_nodes := frontier_nodes \cup \{(x[cmdid], t, k, NULL, 1)\};
  done
end
```

For example, the reachability tree of Fig. 4 is obtained by applying RT operation to the Petri net \( PN_{BRA-1} \). The contents of \( M \) and \( R \) are shown in Fig. 5.

### C. Formalization of Petri Nets Properties in BRA

The reachability tree is a very powerful tool for analysis of behavioral properties. But, several properties such as reachability and liveness, in general, cannot be solved by using the reachability tree because of the existence of the \( \omega \) symbol [13]. However, if the modeled system satisfies boundedness, then all the behavioral properties can be determined by using the reachability tree because it contains all possible markings. Fortunately, most realistic problems

```
Fig. 4. The reachability tree of \( PN_{BRA-1} \).

```

```
Fig. 5. The marking table \( M \) and the reachability table \( R \) of \( PN_{BRA-1} \). should be bounded. Thus, we assume that the reachability tree is bounded for all properties but safeness and boundedness.

1) **Safeness**: Safeness is a special case of the more general boundedness property. This property requires that each place \( p \in P \) should not have more than one token for each marking \( M_k \in M \). To show safeness we first obtain the bag of unsafe places. It is easy to show that the bag of unsafe places is given by

\[ Unsafe.Place := \pi_{\text{pid}}[\geq 1(M)] \]

Next, we justify the safeness with \( Unsafe.Place \). That is, if \( Unsafe.Place \) is empty, then the modeled system is safe.

```
begin
  Unsafe.Place := \pi_{\text{pid}}[\geq 1(M)];
  if Unsafe.Place ≠ \emptyset
    then return Unsafe.Place;
    else return "safe";
  endif
end
```
In the reachability tree of Fig. 4, every place of $\text{PN}_{\text{BRA-1}}$ is safe.

2) **Boundedness:** This property requires that each place $p \in P$ should not have more than $k$ tokens for each marking $M_k \in M$. In other words, if the modeled system is bounded, then $k$ is the maximum number among the number of tokens of all places for all markings.

begin
UnboundedP $\triangleq \sigma_{\#}^{\leq \omega} (M)$;
if UnboundedP $\neq \emptyset$ then return UnboundedP; endif
for each $(mid, pid, cnt) \in M$ do
  if cnt $>$ k then $k \leftarrow$ cnt; endif
return "$k$-bounded$";
end

In the reachability tree of Fig. 4, $\text{PN}_{\text{BRA-1}}$ is 1-bounded.

3) **Conservation:** A petri net is strictly conservative if it does not lose or gain tokens but merely moves them around. This property can be easily tested by the following procedure. First, boundedness is checked, because the necessary condition of conservation is boundedness. Next, if the modeled system is bounded, then we check whether the number of tokens for each marking $M_i \in M$ is equal or not.

begin
$\text{AllMid} \triangleq \pi_{\text{mid}}(M)$;
$s_0 \triangleq \text{get markingsum}(0)$;
for each $(t, i) \in \text{AllMid}$ do
  $s_t \triangleq \text{get markingsum}(t)$;
  if $s_0 \neq s_t$ then return "unconservative"; endif
end
return "conservative";

where
function get markingsum(k)
begin
  $M_k \triangleq \pi_{\text{pid}}\sigma_{\text{mid}}^=k M$; $s \triangleq 0$;
  for each $(mid, pid, cnt) \in M_k$ do
    $s \triangleq s + cnt$; done
  return $s$;
end

We can also test conservation by considering the weighting factor given to each place. Because this is very similar to the above procedure, the procedure is not enumerated here. In the reachability tree of Fig. 4, $\text{PN}_{\text{BRA-1}}$ is not strictly conservative.

4) **Coverability:** The coverability problem is that given a Petri net $\text{PN}$ with initial marking $M_0$ and a marking $M_j$, is there a reachable marking $M_k \in M$ such that $M_k(p) \geq M_j(p)$ for each $p \in P$?

begin
$\text{AllMid} \triangleq \pi_{\text{mid}}(M)$;
for each $(t, i) \in \text{AllMid}$ do
  if $M_i \subseteq M_k$ then return $M_i$; endif done
return "uncoverable";
end

A marking $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)$ is coverable in the reachability tree of Fig. 2 because the marking is covered by $M_6$.

5) **Reachability:** The reachability problem is that given a Petri net $\text{PN}$ with initial marking $M_0$ and a marking $M_j$, is $M_j \in M$?

begin
$\text{AllMid} \triangleq \pi_{\text{mid}}(M)$;
for each $(t, i) \in \text{AllMid}$ do
  if $M_i = M_j$ then return "reachable"; endif done
return "unreachable";
end

A marking $(0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0)$ is reachable in the reachability tree of Fig. 2 because the marking is $M_{20}$.

6) **Liveness:** We say that a Petri net is in deadlock if no transition in the net is enabled. These deadlock markings correspond to the deadlock markings of the reachability tree. We assume that a Petri net is live if it doesn’t have any deadlock states.

begin
$\text{Deadlock} \triangleq \pi_{\text{mid}}\sigma_{\text{type}}^{=\text{Terminal}}(R)$;
if Deadlock $\neq \emptyset$ then return Deadlock; endif
else return "deadlock-free";
end

In the reachability tree of Fig. 4, there are two deadlock markings given below:

<table>
<thead>
<tr>
<th>deadlock</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadlock</td>
<td>21</td>
</tr>
</tbody>
</table>

V. REALIZATION IN RDBMS AND EXPERIMENTS

A. Realization in RDBMS

Our framework has been constructed in a hierarchy, BRA Operators Module built in Isqlper provides BRA operators for Petri Nets Analysis Module. Petri Nets Analysis Module provides several subroutines for BRA&SQL and Isqlper application programs. BRA&SQL is a user-interface program that accepts queries to verify Petri nets. For rapid prototyping, we have used an interpretive language, "Isqlper" [14]. Isqlper allow direct access to Informix databases from within a Perl script. Perl is a language for easily manipulating text, files and processes [15]. Currently, we have implemented eight BRA operators, which translated into sequences of SQL queries.

- SQL($from sql tbl, $to tbl, [$sql oncond])
  - implements to tbl $\triangleq \sigma_{\text{oncond}}$(from tbl).
- DJS($from tbl, $to tbl, $pjlist)
  - implements to tbl $\triangleq \pi_{\text{pjlist}}$(from tbl).
- DIP($left tbl, $right tbl, $to tbl)
  - implements to tbl $\triangleq left tbl \cup right tbl$.
- DUN($left tbl, $right tbl, $to tbl)
  - implements to tbl $\triangleq left tbl \cup right tbl$.
- AD($left tbl, $right tbl, $to tbl)
  - implements to tbl $\triangleq left tbl \cup right tbl$.
- DIN($left tbl, $right tbl, $to tbl, $oncond, $exprlist)
  - implements to tbl $\triangleq left tbl \cup right tbl$.
- INC($left tbl, $right tbl, $to tbl, $oncond, $exprlist)
  - implements to tbl $\triangleq left tbl \cup right tbl$.
- JXS($from tbl, $to tbl, $pjlist, $oncond, $exprlist)
  - implements to tbl $\triangleq \pi_{\text{pjlist}}\sigma_{\text{oncond}}$(from tbl).
- JJS($from tbl, $to tbl, $pjlist, $oncond, $exprlist)
  - implements to tbl $\triangleq \pi_{\text{pjlist}}$(left tbl $\triangleq$ right tbl)
On this basis, Petri Nets Analysis Module provides operations for reading text files of Petri nets in the framework, generating a reachability tree and analyzing behavioral properties:

- $RT()$
- $\text{saftiness}()$
- $\text{boundedness}()$
- $\text{conservation}()$
- $\text{reachability}()$
- $\text{liveness}()$
- $\text{read.petrinet}(\text{file.name})$
- $\text{write.reachabilitytree}(\text{file.name})$

Most operations except $\text{read.petrinet}(\text{file.name})$ and $\text{write.reachabilitytree}(\text{file.name})$ are mentioned in Section IV. $\text{read.petrinet}(\text{file.name})$ accepts an input file $\text{file.name}$ and initializes the graph relations such as $P$, $T$, $I$, $O$, and $M$. $\text{write.reachabilitytree}(\text{file.name})$ writes the reachability tree information stored in $R$ to a given output file $\text{file.name}$. This output file is used to display the reachability tree graphically. If a Petri net has been inputted and a reachability tree has been generated by calling $RT()$, behavioral properties can be verified by using operations of Petri Nets Analysis Module.

BRA&SQL Module is under development. Current implementation supports the following queries:

- $\text{read.petri.net from file.name}$
- $\text{generate.reachability.tree}$
- $\text{verify saftiness}$
- $\text{verify boundedness}$
- $\text{verify conservation}$
- $\text{verify coverability of marking}$
- $\text{verify reachability of marking}$
- $\text{verify liveness}$

in addition to standard SQL queries.

B. Experiments

We implemented our framework on a SUN SPARCstationII computer running Informix DBMS. Within our framework a Petri net is constructed by executing "read petri net from (file)." The reachability tree then will be produced by executing "generate reachability tree." Since the reachability tree is stored in the database, several properties of the Petri net will be analyzed by executing queries such as "verify saftiness," "verify liveness," and so on.

We have successfully applied our framework to Petri net modeling of the dining philosophers problem. The problem has $n \times 3$ places and $n \times 2$ transitions where $n$ is the number of dining philosophers. For the problem, we varied the number of dining philosophers from 10-13. Because the time for generating a reachability tree is much longer than the time for applying analysis algorithms, the former is only experimented in our framework. For each number of dining philosophers, the number of nodes of the reachability tree is 681, 1211, 2137, and 3745, respectively, and generating the reachability tree takes about 1396, 2604, 4943, and 9412 s (elapsed time), respectively.

Current realization of the proposed framework is in the very first implementation to verify our framework. To make analysis of realistic problems with more than $10^5$ states possible, many optimizations are needed to speed up the framework. For example, it is necessary to implement BRA operators efficiently and to replace interpretive ISqlperl implementations with C language implementations.

VI. CONCLUSIONS

We have extended the conventional bag-theoretic relational algebra (BRA) to give complete control over the occurrence number of tuples in bag relations. This extension allows one to use BRA in more realistic applications. Using such the extended BRA, we have proposed a framework for the modeling and analysis of Petri nets. Advantages of the proposed framework in the modeling and analysis of Petri nets are as follows. First, the modeling of Petri nets is based on a formal means, BRA, and various properties of Petri nets are also formalized in terms of BRA. Thus, there is no semantic gap between a modeling tool and its analysis environment since they are based on the same formalism. Second, once the framework is realized in a database system, the advantages of the database system in management of large amounts of data is exploited. The advantages include efficiency in managing large states space and searching specific states in the space during simulation of a Petri net. We have shown that the proposed framework can easily be realized even in relational database systems. As one possible approach, we have implemented BRA operators on top of a commercial relational database system, Informix, using a standard SQL. On such basis, algorithms for generating a reachability tree of a Petri net have been implemented in terms of BRA, and queries to verify properties of a Petri net have also been implemented in terms of BRA.

REFERENCES