관계 대수를 이용한 페트리 네트의 모델링

Modeling of Petri Nets Using Relational Algebra

김영찬*, 김탁곤*
Young-Chan Kim* and Tag-Gon Kim*

Abstract

This paper proposes an analysis method of Petri nets(PNs) using the relational algebra(RA). More specifically, we represent PNs in relations of the relational model. Based on such representation, we first develop an algorithm for generating reachability trees of PNs. We then develop algorithms for analyzing properties of PNs, such as boundedness, conservation, coverability, reachability, and liveness.

The advantage of this approach is as follows: First, the algorithms represented by RA can be easily converted to a query language such as SQL of the widely used, commerical relational database management systems(DBMSs). Second, we can alleviate the problem of state space explosion because relational DBMSs can handle large amounts of data efficiently. Finally, we can use the DBMS's query language to interpret the Petri nets and make simulation.

1. 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, in recent years considerable effort has been given to develop models for specification and validation of protocols. 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] [2]

* 한국과학기술원 전기 및 전자공학과 컴퓨터공학연구실

[3] A difficult practical and theoretical issue in the design of protocols is *protocol verification*. *Reachability analysis* can be used in the protocol design phase to explore the global states of the system to detect undesirable behaviors, e.g., unboundedness, dealocks 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. That is, as the network of communicating finite-state machines increases in complexity, the total number of possible global states grows rapidly. This problem can be alleviated by the relational approach, because relational DBMSs can handle large amounts of data efficiently. This approach has been used in. [4] [5]

Specifically, we describe an implementation of reachability analysis for systems(which can be protocols) modeled by Petri nets. In this approach, systems are represented as set of relations. Using these relations, the reachable global states can be determined by an interative sequence of operations of RA that eventually generates a reachability tree for the system. This final reachability tree can be examined by specific queries, again described in terms of RA to detect properties of Petri nets.

This paper is organized as follows. In the next section we introduce the notation and concept of the relational algebra(RA). In section 3 we apply RA to Petri net and formulate the several properties of Petri nets in terms of operators of RA. Conclusions are drawn in the last section.

2. The Relational Model and Algebra

In this section we briefly introduce the relational model and algebra.

2.1 The Relational Model

The mathematical concept underlying the relational model is the *set-theoretical relation*(s-relation), which is a subset of the Cartesian product of a list of sets. Given sets S_1, S_2, $\cdots$, S_n, a s-relation R is a subest of the Cartesian product $S_1 \times S_2 \times \cdots S_n$. S-relation R is said to be of of *degree* n. Each of the sets S_1, S_2, $\cdots$, S_n on which one or more s-relations are defined is called a *domain*.

A relation R in the relational model[6] [7] is very similar to its counterpart in mathematics. Relations can perceived as tables, where each row is a tuple and each column has a distinct name called an *attribute*. A relation R on $A=\{A_1, A_2 \cdots A_n\}$ will be denoted by $R(A)$. Let t be a tuple in $R(A)$. The components of t corresponding to the set of attributes $X \subseteq A$ is denoted by $t.X$. If a_1 is a constant from the domain of A_1, then$\langle a_1, a_2, \cdots, a_n \rangle$ is a constant tuple over $A_1 A_2 \cdots A_n$

2.2 Operations of Relational Algebra

There are five fundamental operations that serve to define relational algebra. These operations are: *union, difference, Cartesian product, project,* and *select.*

(1) *Union* $\cup$: The union of relations R and S, denoted $R \cup S$, is smallest set containing all tuples of R and all tuples of S.

$R \cup S = \{t \mid t \in R \vee t \in S\}$

(2) Difference $-$: The difference of relations R and S, denoted $R-S$, the set containing the tuples of R that are not in S.

$R-S = \{t \mid t \in R \wedge t \notin S\}$

(3) *Cartesian Product* $\times$: The Cartesian product of R and S is the set of all ordered pairs $\langle r, s \rangle$ such that the first element of ordered pair, r, is from R and second element of the ordered

pair, s, is from S.

$R \times S = \{(r, s) | r \in R \wedge s \in S\}$

(4) *Projection* π : Projection chooses a subset of the columns. Let R be a relation on a set of attributes $A = \{A_1, A_2, \cdots, A_n\}$ and X be subset of A. The projection $\pi_X(R)$ is obtained by dropping columns with attributes not in the set X and removing duplicate tuples in what remains.

$\pi_X(R) = \{t(X) \mid t \in R\}$

In general, we can extend X as a set of the airthmetic expression of elements of A.

(5) *Selection* σ : Let F be a formula which is one of the following types:

1. $F = \phi$.
2. $F = a\theta b$ where,
   (a) a(b) are constants, attribute names, or component numbers; component i is represented by $\$i$.
   (b) $\theta \in \{<, =, >, \leq, \neq, \geq\}$.
3. F_1 and F_2 are two formulas and $F = F_1 \wedge F_2$ or $F = F_1 \vee F_2$ or $F = -F_1$.

Then $\sigma_F(R)$ is the set of tuples t in R such that when we substitute the ith component of t for any occurrences of $\$i$ in the formula F for all i and substitute the corresponding components of t for attribute names in the formula F, the formula F becomes true.

$\sigma_F(R) = \{t \mid F(t) \wedge t \in R\}$

In addition to the five fundamental operations, there are some other useful operations on relations that can be defined in terms of the five fundamental operations above, namely, *intersection*, *theta join*, and *natural join*.

(6) *Intersection* $\cap$: The intersection of relations R and S, denoted $R \cap S$, is the smallest set containing all tuples that are members of both R and S.

$R \cap S = R - (R - S)$

(7) *Theta join* $\bowtie_\theta$: The theta join of R and S on columns i and j, written $R \underset{i\theta j}{\bowtie} S$ is shorthand for $\sigma_{\$i\theta\$(r+j)}(R \times S)$, if R is of degree r. θ is an arithmetic comparison operator(=, <, and so on).

$R \underset{i\theta}{\bowtie} S = \sigma_{\$i\theta\$(r+j)}(R \times S)$, if R is of degree r

(8) *Natural Join* $\bowtie$: Let R and S be relations on a set of attributes $A = \{A_1, A_2, \cdots, A_n\}$ and X, Y, and Z be a subset of A. The natural join combines two relations on all their common attributes. The natural join, written $R(XY) \bowtie S(YZ)$, is a relation $T(XYZ)$ of all tuples t over XYZ such that there are tuples $t(XY) \in R$ and $t(YZ) \in S$.

$R(XY) \bowtie S(YZ) = \{t \mid t(XY) \in R \wedge t(YZ) \in S\}$

3. Modeling of Petri Nets using RA

Petri nets are a promising 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.

This problem can be alleviated by the relational approach, because relational DBMSs can handle large amounts of data dficiently. That is , if we represent Petri nets as relations, their reachability tree can be obtained as relations by using RA operators. Furthermore, we can draw several properties of Petri nets from the reachability tree in forms of relations.

3.1 Definition and Properties of Petri Nets

In this section, we present basic definitions of Petri nets and explain behavioral properties of Petri nets. A formal definition of Petri net follows.

Definition 3.1 *A* **Petri net,** *PN, is a five-tuple structure, where* $PN=(P, T, I, O, M_0)$

1. *P is finite set of* **Place**
2. *T is a finite set of* **transitions.** *The set of places and the set of transitions are disjoint.* $P \cap T = \phi$.
3. $I: T \rightarrow P^{\infty}$ *is the* **input** *function, a mapping from transition to bags of places.*
4. $O: T \rightarrow P^{\infty}$ *is the* **output** *function, a mapping from transition to bags of places.*
5. $M_0 : P \rightarrow \mathbb{N}$ *is the* **initial marking** *function, a mapping from set of places P to the nonnegative integers* $\mathbb{N}$.

Note that the inputs and outputs of a transition are bags of places. The *multiplicity* of an input place p for a transition t is the number of occurrences of the place in input bag of a transition, $\#(p, T(t))$. Similarly, the *multiplicity* of an output place p for transition t is $\#(p, O(t))$.

definition 3.2 *Let* $PN=(P, T, I, O, M_0)$ *be a Petri net.*

1. *A function* $M_k : P \rightarrow N$, *where* $k \in \mathbb{N}$ *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* $\#(p, T(t)) \leq M_k(p)$, $\forall p \in P$. *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 *given by the equation:*

$M_k'(p) = M_k(p) - \#(p, I(t)) + \#(p, O(t))$, $\forall p \in P$

4. *Firing t changes the marking* M_k *into the new marking* M_k'; *we denote this fact by* $M_k \xrightarrow{t} M_k'$.
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* $M_k' \xrightarrow{t} M_k''$ *for some* $t \in T$ *then* $M_k'' \in R(M_k)$.

There are two types of properties studied with a Petri-net model: behavioral and structural properties. In this paper, we discuss only behavioral properties because structural properties depend on the topological structures of Petri nets and can be well characterized in terms of the incidence matrix and its associated homogeneous equations or inequalities.

Many behavioral properties have been studied in Petri nets[8] [9] [10] but we consider only boundedness, conservation, liveness, reachability, and coverability. Boundedness can be interpreted as a stable factor in the system. For example, if the modeled Petri net is unbounded, then this may indicate the occurrence of overflow of some buffers in the system. Conservation is an important property in that if a Petri net models resource allocation systems, then tokens, which represent resources, are neither created nor destroyed. Liveness has an important meaning for many systems and is closely related to the complete absence of deadlocks in concurrent systems. Reachability is a fundamental basis for studying the dynamic properties of any system. Coverability is closely related to $L1$-liveness. [8] Let M_k be the minimum marking needed a transition t. Then t is dead if and only if M_k is not coverable. That is, t is $L1$-live if and only if M_k is coverable.

Definition 3.3

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* $\exists n \in \mathbb{N}$, *PN is n-bounded.*

2. Conservation:
   (a) *PN is* **strictly conservative** *iff* $\forall M_k \in R(M_0)$,

$\Sigma_{p\in P} M_k(p) = \Sigma_{p\in P} M_0(p)$;

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

3. Liveness:

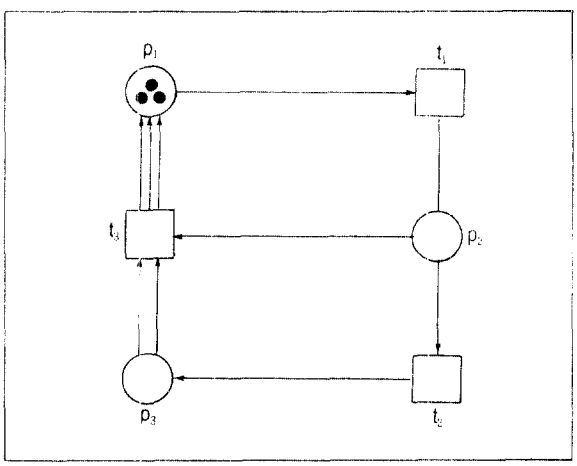
(a) $t \in T$ *is* **live** *iff* $\forall M_k \in R(M_0)\ \exists M_{k'} \in R(M_k)$, t *is enabled at* $M_{k'}$;

(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* 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* $M_{k'} \geq M_k$?



〈**Figure 1**〉 **A simple Petri net** PN_1

3.2 Tabular Representation of Petri nets

In this section we define relations to represent Petri nets.

Schema definition :

1. $P(pid)$ is a table to store place P of PN.
2. $T(tid)$ is a to store transitions T of PN.
3. $I(tid, pid, cnt)$ is a table where each tuple i corresponds to an input arc of $i(tid) \in T$.

$I(tid, pid, cnt) = \{i \mid i \in I \wedge i(tid) \in T \wedge i(pid) \in P \wedge cnt = \sharp(pid, I(tid))\}$

4. $O(tid, pid, cnt)$ is a table where each tuple o corresponds to an output arc of $o(tid) \in T$.
5. $M(mid, pid, cnt)$ is a table to store all marking $M_k \in R(M_0)$. A marking M_k can be retrieved by $\pi_{pid}(\sigma_{mid = k}(M))$.

$M_k = \pi_{pid}(\sigma_{mid = k}(M))$

6. $R(mind, pid, mid')$ *is a table to store the reachability tree of PN.*

The table defined above for Figure 1 are shown in Figure 2.

$P =$

pid
p_1
p_2
p_3

$T =$

tid
t_1
t_2
t_3

$I =$

tid	*pid*	*cnt*
t_1	p_1	1
t_2	p_2	1
t_3	p_2	1
t_3	p_3	2

$O =$

tid	*pid*	*cnt*
t_1	p_2	1
t_2	p_3	1
t_3	p_1	3

$M =$

mid	*pid*	*cnt*
0	p_1	3
0	p_2	0
0	p_3	0

〈**Figure 2**〉 **The relation representation of the simple Petri net** PN_1

3.3 Algorithm for Generating the Reachability Tree

The reachability tree represents the reachability set of a Petri net. It is a useful tool for solving behavioral properties. For the reachability tree to be finite, the special symbol, ω, are necessary for the construction of the reachability tree[8]. The rules for ω are:

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

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

$\omega\leq\omega$.

Now, we develop the algorithm for generating the reachability tree of a Petri net based on RA.

1. Enabled-transition operation ET(M_k) : Given PN and M_k, this operation, denoted ET(M_k), yields a set of enabled transitions in the marking M_k. It is defined by the following procedure:

step1. $MInput := \sigma_{tid,\,pid,\,cnt}(I_{\subset}(I_{pid}=M_{k\cdot pid})\wedge(I_{cnt}\leq M_{k\cdot cnt})\,M_k)$;

step2. $UMInput := I-MInput$;

step3. $NETrans := \pi_{tid}(UMInput)$;

step4. $ET(M_k) := T-NETrans$;

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. Since t can fire only if it is enabled, NS(M_k, t)=M_k of t is not enabled in marking M_k. If t is enabled, then NS(M_k,t)=M_k', where M_k' is the marking which results from removing tokens from the inputs of t and adding tokens to the outputs of t. It is defined by the following procedure:

step1. $Pre := \pi_{pid,\,cnt}(\sigma_{tid=t}(I))$.

step2. $Pos := \pi_{pid,\,cnt}(\sigma_{tid=t}(O))$;

step3. $Tmp1 := \pi_{pid=M_{k\cdot pid},\,cnt=M_{k\cdot cnt}}(M_k{}_{\subset}(M_{k\cdot pid}=P_{re\cdot pid}Pre)$;

step4. $Tmp2 := \pi_{pid=M_{k\cdot pid},\,cnt=M_{k\cdot cnt}-P_{re\cdot cnt}}(M_k{}_{\subset}(M_{k\cdot pid}=P_{re\cdot pid}Pre)$;

step5. $M_k' := (M_k-Tmp1)\cup Tmp2$;

step6. $Tmp1 := \pi_{pid=M_{k'\cdot pid},\,cnt=M_{k'\cdot cnt}}(M_{k'}{}_{\subset}(M_{k'\cdot pid}=P_{os\cdot pid}Pos)$;

step7. $Tmp2 := \pi_{pid=M_{k'\cdot pid},\,cnt=M_{k'\cdot cnt}+P_{os\cdot cnt}}(M_{k'}{}_{\subset}(M_{k'\cdot pid}=P_{os\cdot pid}Pos)$;

step8. **NS**(M_k,t) := $(M_k'-Tmp1)\cup Tmp2$;

3. Parent-marking operation PM(R, M_k): Given the reachability tree R and any marking M_k which is a node of R, this operation yields all the parent markings of M_k. It is defined by the following procedure:

step1. $Parent := \{k\}$;$New := \{k\}$;

step2. **while** $New\neq\phi$ **do**

(*Without loss of generality we assume New is queue *)

i:=DEQUEUE(New);(*Extract front element from queue *)

if $i\neq 0$ **then**

for each $j\in\pi_{mid}(\sigma_{mid'=i}(R))$**do**

if $j\notin Parent$ **then**

$Parent := Parent\cup j$;

$New := New\cup j$;

end if

end for

end if

end while

step3. PM(R, M_k):=$Parent$;

4. Update-marking operation UM(M_k, M_k'): Given M_k', which is immediately reachable from M_k, if there exists a path from the root to M_k containing a marking M_j such that

$\pi_{cnt}(\sigma_{pid=p}(M_j))\leq\pi_{cnt}(\sigma_{pid=p}(M_k'))$ for all $p\in P$,

then replace $m_k'[cnt]$(where $m_k'\in M_k'$ and $m_k'[pid]=p$) by ω wherever $\pi_{cnt}(\sigma_{pid=p}(M_j))<\pi_{cnt}(\sigma_{pid=p}(M_k'))$.

step1. $Tmp := \phi$;

step2. $Pmid := $PM($R$, M_k);

step3. for each $i \in Pmid$ **do**

if $\pi_{cnt}(\sigma_{pId}=p(M_j)) \leq \pi_{cnt}(\sigma_{pId}=p(M_k'))$ for all $p \in P$ **then**

for each $p \in P$ **do**

if $\pi_{cnt}(\sigma_{pId}=p(M_j)) < \pi_{cnt}(\sigma_{pId}=p(M_k'))$ **then**

$Tmp := Tmp \cup \pi_{pId,\,cnt}=\omega(M_k')$;

else

$Tmp := Tmp \cup \pi_{pId,\,cnt}(M_k')$;

end if

end for

$M_k' := Tmp$

end if

end for

5. Reachability-tree operation RT: Using the above defined operations, we could construct the reachability tree of a Petri net as follows. Note that the relation R is initially empty.

step1. $New := \{o\}$; $j := o$;

step2. while $New \neq \phi$ **do**

(* Without loss of generality we assume New is queue *)

$i := \text{DEQUEUE}(New)$; (*Extract front element from queue*)

if $\sigma_{mId}=i(\pi_{mId}(R)) = \phi$ **then**

for each $t \in ET(M_i)$ **do**

$M' := NS(M_i, t)$;

$UM(M_i, M')$;

if $M' = M_k$, where $k \in \pi_{mId}(M)$ **then**

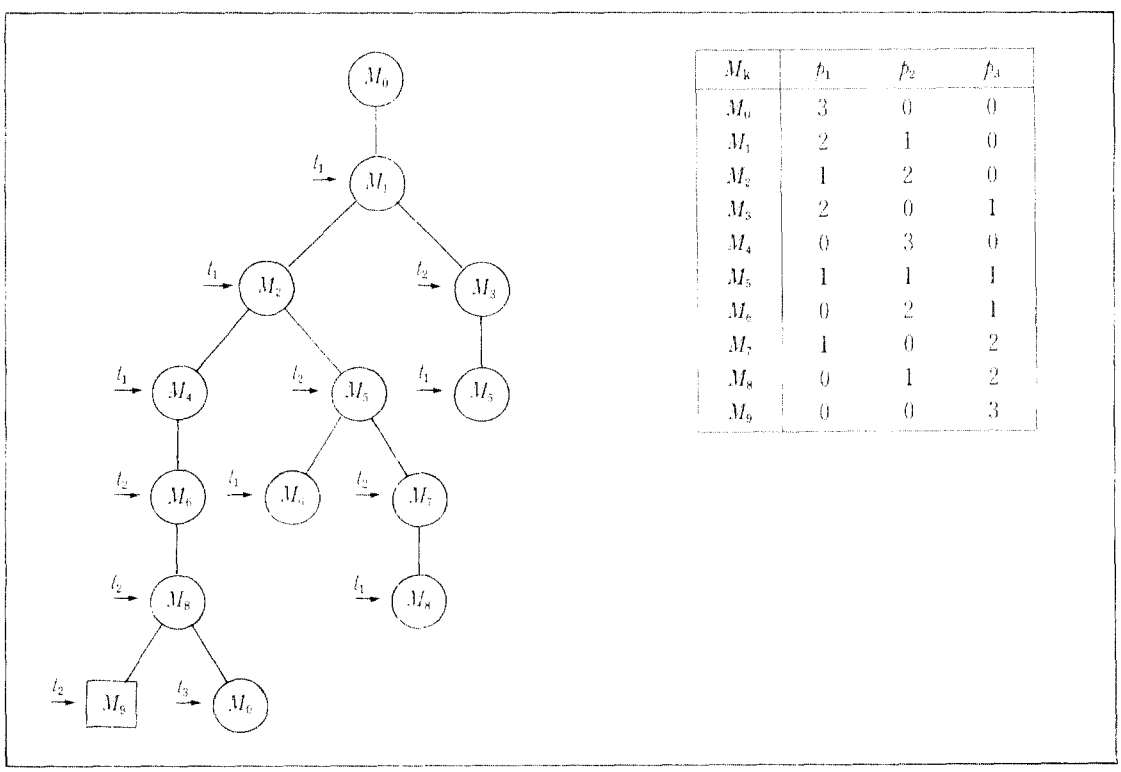
$R := R \cup \{\langle i, t, k \rangle\}$;

else

$j := j+1$;

$M := M \cup \{\langle j, m' \rangle | m' \in M'\}$;

$R := R \cup \{\langle i, t, j \rangle\}$;



M_k	p_1	p_2	p_3
M_0	3	0	0
M_1	2	1	0
M_2	1	2	0
M_3	2	0	1
M_4	0	3	0
M_5	1	1	1
M_6	0	2	1
M_7	1	0	2
M_8	0	1	2
M_9	0	0	3

〈**Figure 3**〉 **The reachability tree of** PN_1

M =

mid	pid	cnt
0	p_1	3
0	p_2	0
0	p_3	0
1	p_1	2
1	p_2	1
1	p_3	0
2	p_1	1
2	p_2	2
2	p_3	0
3	p_1	2
3	p_2	0
3	p_3	1
4	p_1	0
4	p_2	3
4	p_3	0
5	p_1	1
5	p_2	1
5	p_3	1
6	p_1	0
6	p_2	2
6	p_3	1
7	p_1	1
7	p_2	0
7	p_3	2
8	p_1	0
8	p_2	1
8	p_3	2
9	p_1	0
9	p_2	0
9	p_3	3

R =

mid	tid	mid'
0	t_1	1
1	t_1	2
1	t_2	3
2	t_1	4
2	t_2	5
3	t_1	5
4	t_2	6
5	t_1	6
5	t_2	7
6	t_2	8
7	t_1	8
8	t_2	9
8	t_3	0

〈Figure 4〉 The marking table M and reachability table R of PN_1

$New := New \cup \{j\};$

end if

end for

end if

end while

For example, we can obtain the reachability tree of Figure 3 by applying RT operation to the Petri net of Figure 1. The contents of M and R is shown Figure 4.

3.4 Algorithm for Analysis of Petri Nets

The reachability tree is a very powerful tool for analysis of behavioral properties. But, reachability and liveness, in general, cannot be solved by using the reachability tree because of the existence of the ω symbol. The reachability tree does not necessarily contain enough information to solve reachability or liveness. However, if the modeled system satisfies boundedness then all of behavioral properties can be determined by using the reachability tree because it contains all possible markings. Fortunately, most realistic problems should be bounded. Thus, we assume that the reachability tree is bounded for properties such as reachability and liveness.

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 set of unsafe places. It is easy to show that the set of unsafe places is given by

$$Unsafe\text{-}Place := \pi_{pid}(\sigma_{cnt}>1(M)).$$

Next, we justify the safeness with *Unsafe-Place*. That is, if *Unsafe-place* is empty, then the modeled system is safe.

step1. $Unsafe\text{-}Place := \pi_{pid}(\sigma_{cnt}>1(M))$;
step2. **if** $Unsafe\text{-}Place \neq \phi$ **then**
return unsafe;
end if
step3. **return** safe;

In the reachability tree of Figure 3, there are three unsafe places given below:

$Unsafe\text{-}Place =$

pid
p_1
p_2
p_3

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.

step1. $Unbounded\text{-}Place := \pi_{pid}(\sigma_{cnt}=\omega(M))$;
step2. **if** $Unbounded\text{-}Place \neq \phi$ **then**
return unbounded;
end if
step3. $k := max\{\pi_{cnt}(M)\}$;
step4. **return** k-bounded;

In the reachability tree of Figure 3, PN_1 is 3-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 easily be 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 sum of the number of tokens of each marking $M_i \in M$ is equal or not.

step1. $Unbounded\text{-}Place := \pi_{pid}(\sigma_{cnt}=\omega(M))$;
step2. **if** $Unbounded\text{-}Place \neq \phi$ **then**
return unconservative;
end if
step3. $Rmid := \pi_{mid}(M)$;
step4. $sum_0 := \Sigma\{\pi_{cnt}(\sigma_{mid}=i(M))\}$;
step5. **for each** $i \in Rmid$ **do**
$sum_1 := \Sigma\{\pi_{cnt}(\sigma_{mid}=i(M))\}$;
if $sum_0 \neq sum_1$ **them**
return unconservative;
end if
end for
step4. **return** conservative;

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 Figure 3, PN_1 is strictly conservative.

4. *Converability:* The coverability problem is that given a Petri net PN with initial marking M_0 and a marking M_j, is there a reachable marking $M'j \in M$ such that $\pi_{cnt}(\sigma_{pid=p}(M'j)) \geq \pi_{cnt}(\sigma_{pid=p}(M_j))$, for all $p \in P$?

step1. $Rmid := \pi_{mid}(M)$;
step2. for each $i \in Rmid$ do
if $\pi_{cnt}(\sigma_{pid=p}(M^1)) \leq \pi_{cnt}(\sigma_{pid=p}(M_j))$ for all $p \in P$ **then**
return coverable;
end if
end for
step4. return uncoverable;

We can know that a marking $M_j=(1, 1, 0)$ is coverable in the reachability tree Figure 1 because M_j is covered by M_1 or M_2.

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

step1. $Rmid := \pi_{mid}(M)$;
step2. for each $i \in Rmid$ do
if $M_j = M_1$ **then**
return reachable;
end if
end for
step4. return unreachable;

We can know that a marking $M_j=(0, 1, 2)$ is reachable in the reachability tree of Figure 1 because M_j is M_8.

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 terminal nodes of the reachability tree.

$Deadlock := \pi_{mid}(R) - \pi_{mid}(R)$

We assume that a Petri net is live if it doesn't have any deadlock states. In the reachability tree of Figure 3, there is one deadlock marking(state) given below:

$Deadlock =$

mid
9

4. Conclusions

We have shown the relational methodology for the analysis of Petri nets based on RA representation and manipulation. Specifically, we have described an implementation of reachability analysis for systems modeled by Petri nets. In this approach, systems are represented as a set of relations. Using these relations, the reachable global states are determined by an iterative sequence of operations of RA that eventually generate a reachability tree for the system. This final reachability tree can be examined by specific queries, again, described in terms of RA, to verify properties of Petri nets. Because the several properties of Petri nets have been formulated in terms of RA's operators, all procedure of analysis of Petri nets can easily transfer to commercial relational DBMSs. In conclusion, we believe that RA is a useful tool for mainpulating data.

4. References

[1] G. Berthelot and R. Terrat, "Petri nets theory for the correctness of protocols", *IEEE tran. on communications*, vol. COM-30, pp. 2497-2505, Dec. 1982.

[2] G. R. W. J. Billington and M. C. wilbur ham, "Protean: A high-level petri net tool for the specification an verfication of communication protocols", *IEEE tran. on Software Engineering*, vol. 14, pp. 301-316, Mar. 1988.

[3] Y. Zhang, K. Takahashi, N. Shiratori, and S.

Noguchi "An interactive protocol synthesis algorithm using a global state transition graph", *IEEE tran. on Software Engineering*, vol. 14, pp. 394-403, Mar. 1988.

[4] T. T. Lee and M. Lai, "A relational algebraic approach to protocol verification", *IEEE tran. on Software Engineering*, vol. 14, pp. 184-193, Feb. 1988.

[5] O. Frieder and G. E. Herman, "Protocol verification using deatabase technology", *IEEE Jounal on Selected Areas in Communications*, vol. 7, pp. 324-334, Apr. 1989.

[6] E. F. Codd, "A relational model of data for large shared data banks", *Comm. ACM*, vol. 13, pp. 377-387, Jun. 1970.

[7] E. F. Codd, "Extending the database relational model to capture more meaning", *ACM tran. on Database Systems*, vol. 4, pp. 397-434, Dec. 1979.

[8] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1981.

[9] W. Reisig, *Petri Nets*. Berlin: Springer-Verlag, 1985.

[10] T. Murata, "Petri nets: Properties, analysis and applications", *Proceeding of the IEEE*, vol. 77, pp. 541-580, Apr. 1989.

저자소개

김영찬

1985년 2월 아주대학교 공과대학 전자공학학사
1987년 2월 한국과학기술원 전자공학석사
1987 ~ 1990년 삼성전자종합연구소 ASIC개발실
1990년 9월 ~ 현재 한국과학기술원 전기 및 전자공학과 박사과정
관심분야: Petri Nets, Database Management Systems, Protocol Analysis

김탁곤

1975년 2월 부산대학교 전자공학과 졸업(공학사)
1980년 2월 경북대학교 전자공학과 졸업(공학석사)
1988년 5월 아리조나대학교 전기 및 전산공학과 졸업(공학박사)
1975년 2월 ~ 1977년 6월 육군 통신장교
1980년 9월 ~ 1983년 1월 부산수산대학교 전자통신공학과 전임강사
1987년 8월 ~ 1989년 7월 아리조나 환경연구소 연구엔지니어
1989년 8월 ~ 1991년 8월 캔사스대학교 전기 및 전산공학과 조교수
1991년 9월 ~ 현재 한국과학기술원 전기 및 전자공학과 조교수
관심분야: 모델링이론, 병렬/지능형 시뮬레이션 환경, 컴퓨터 시스템 등