페트리네트 Slice를 이용한 페트리네트 모델의 합성적 분석

(Compositional Analysis of Petri Net Models using Petri net Slices)

이 우 진 † 차 성 덕 †† 권 용 래 †† 김 흥 남 †††

(Woo Jin Lee) (Sung Deok Cha) (Yong Rae Kwon) (Heung-Nam Kim)

요 약 Place/Transition(P/T) nets은 병행성 기술이 용이하고 도달성 분석 등 다양한 분석 방법이 제공되므로 프로토콜, 병행적 시스템 검증 등에 많이 이용되어 왔으며 또한 실시간 시스템, 객체지향 시스템 등 다양한 분야에 이용되는 고급 Petri nets과 객체 지향 Petri nets 등의 기반 정형적 기법으로 이용되고 있다. 하지만 P/T nets과 확장된 Petri nets 등에서는 모델에 내재된 병행성을 다루지 않으면 복잡한 병행적 모델의 도달성 분석 시에 시스템 상태가 급증하는 상태 폭발(state explosion)이 발생할 수 있다. 이 연구에서는 다양한 확장된 고급 Petri nets의 근간이 되는 P/T net 모델에서 구조적 병행성을 정의하고 이를 기반으로 시스템을 병행적인 단위로 분할, 합성적인 도달성 그래프를 생성하는 방법을 제시하여 복잡한 P/T net 모델을 효율적으로 분석할 수 있는 방법을 제시한다. 그리고 합성적 도달성 그래프를 이용하여 플레이스의 유한성, 트랜지션의 수행가능성 등의 특성을 효율적으로 분석할 수 있도록 한다. 이러한 분할 분석 기법은 P/T net 모델 뿐만 아니라 P/T nets에 기반을 두고 있는 모든 고급 Petri net model 분석에도 이용될 수 있다.

Abstract Place/Transition(P/T) nets has been used in protocol verification and concurrent system verification since it is suitable for describing concurrency and provides several well-established verification techniques. And it has been used as a base formalism for such high-level Petri nets as colored Petri nets, object-oriented Petri nets and etc. However, when analyzing complex models using P/T nets and P/T nets-based high-level Petri nets, there may be state explosion in reachability analysis due to improper handling of concurrency. In this paper, we define a structural concurrency in P/T nets, propose a partitioning algorithm based on the detected structural concurrency, and provide analysis techniques for such properties as boundedness of places and liveness of transitions, which are performed on compositional reachability graphs. The analysis techniques based on Petri net slices can be used in efficiently analyzing P/T nets-based high-level Petri net models as well as P/T net models.

1. 서 론

Petri nets은 병행성 기술이 용이하고 도달성 분석 등 다양한 분석 방법이 제공되므로 프로토콜 검증, 실시간 시스템과 병행적 시스템 등의 모델링 및 분석에 널리 이용되고 있다[1, 2, 3, 4]. 기본 Petri nets인 Place/Transition(P/T) nets[5, 6]은 간단한 의미(semantics)를 지니고 풍부한 분석 방법을 제공하므로 비교적 소규모 시스템 모델링 및 분석에 많이 이용되어 왔다. 하지만 시스템이 복잡해질수록 모델의 이해 및 분석이 어려운 단점이 있다. 이러한 표현의 복잡성을 극복하기 위하여 P/T nets에 함수적 언어인 ML(Meta Language)을 가미한 Colored Petri nets(CPN)[7]과 Predicate/Transition nets[8] 등과 같은 고급 Petri nets, 상속성과 동적 바인딩을 지원하는 객체지향 Petri nets[9, 10], 시나리오를 기반으로 모델을 작성할 수 있는 Constraints-based Modular Petri nets(CMPN)[11] 등이 등장하였다. 이러한 확장된 Petri nets의 특징은 P/T nets의 분석력을 그대로 유지시키면서 간결성 및

† 정 회 원 : 한국전자통신연구원 실시간컴퓨팅연구부 연구원
woojin@etri.re.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
cha@salmosa.kaist.ac.kr
yrkwon@cs.kaist.ac.kr
††† 비 회 원 : 한국전자통신연구원 실시간컴퓨팅연구부 연구원
hnkim@etri.re.kr
논문접수 : 1999년 5월 10일
심사완료 : 1999년 12월 15일

편리성의 표현적 요소를 더욱 보강하여 쉽게 시스템을 작성하고 이해할 수 있도록 하고 있다.

P/T nets과 확장된 Petri nets 등에서는 모델에 내재된 병행성을 다룰 수 있는 방법이 제공되지 않아 복잡한 병행적 모델의 도달성 분석 시에 시스템 상태가 급증하는 상태 폭발(state explosion)이 발생할 수 있다. 반면, 모듈화 Petri nets[12, 13]에서는 모델 작성시 병행적인 단위로 시스템을 분할하여 작성할 수 있으므로 분석 시 합성적 분석 방법[4]을 응용하여 상태 폭발의 가능성을 줄일 수 있다. Notomi[14]는 모듈화 Petri net의 하나인 Ada-net에서 계층적인 도달성 그래프 생성에 관한 연구를 수행하였다. 하지만, 모듈화 Petri nets은 모델 작성시에 작성자의 임의의 기준에 의해 시스템이 분할되므로 병행성이 제대로 반영되지 않을 수도 있으며 객체지향 방법이나 고급 Petri nets에서 제공되는 다양한 모델링 편의와 개발 방법론을 제공하지 않으므로 모델 작성이 어렵다는 단점이 있다.

이 연구에서는 모델을 기술할 때에는 P/T nets을 기반으로 확장된 고급 Petri nets에서 제안된 다양한 표현 기법들을 이용하여 효율적으로 시스템 모델을 기술하고 분석 시에는 병행성을 바탕으로 모델을 분할하여 모듈화 Petri nets으로 변환한 후 합성적인 분석 기법을 이용할 수 있도록 한다. 대부분의 확장된 Petri nets은 P/T nets을 기반으로 표현력을 향상시킨 것이므로 P/T net 모델을 대상으로 모델에 내재된 구조적 병행성을 S-invariant 개념을 이용하여 파악하고 이들을 기반으로 모델을 병행적 단위인 slice 들로 분할하여 Petri net Slices를 생성하는 분할 알고리즘을 제시한다. 그리고 생성된 Petri net Slices들을 기반으로 합성적 분석 기법을 이용하여 시스템의 교착상태, 플레이스의 유한성, 트랜지션의 수행가능성 등을 검사할 수 있는 방법을 기술한다.

이 논문의 구성은 아래와 같다. 먼저 2절에서는 P/T nets과 Petri net 모델에 내재된 병행성에 대해 다루고 3절에서는 구조적 병행성을 파악하는 방법과 이를 이용하여 전체 시스템 모델을 분할하는 알고리즘을 설명한다. 4절에서는 Petri net Slices의 장점을 보이기 위해 P/T nets과 모듈화 Petri nets으로 기술된 식사하는 n명의 철학자 모델에서 생성되는 상태의 수를 비교하고 Petri net Slices를 기반으로 한 합성적 분석에 대해 기술한다. 그리고 5절에서는 결론을 맺고 향후 연구 방향에 대해 언급한다.

2. Place/Transition nets

고급 Petri nets, 객체지향 Petri nets, CMPNs과 같은 확장된 Petri nets의 기본 정형적 기법으로 널리 사용되는 Place/Transition nets은 아래와 같이 정의된다[5].

정의 2.1 Place/Transition nets

아래 조건을 만족하는 *5-tuple N = (P, T; F, W, M)*은 *Place/Transition(P/T) nets*이다.

- *(P, T; F)*는 유한 네트이다. 여기에서 *P*는 플레이스의 집합, *T*는 트랜지션의 집합, *F*는 호(arc)의 집합을 나타낸다.
- $W : P \rightarrow N \setminus \{0\}$, 호의 가중치를 나타낸다.
- $M : P \rightarrow N \cup \{\omega\}$, 초기 토큰의 마킹을 나타낸다.

예제로 식사하는 철학자 문제를 P/T nets으로 나타내면 그림 1과 같다. 철학자는 생각에 잠겨 있다가 식사를 하기 위해 양쪽에 있는 두 개의 포크를 잡고 식사를 한 후 다시 생각에 잠기게 된다. 이 모델에서 교착상태를 방지하기 위해 어느 한쪽의 포크를 이용할 수 없을 경우 이미 잡은 포크를 내려놓고 생각에 잠길 수 있게 되어 있다.

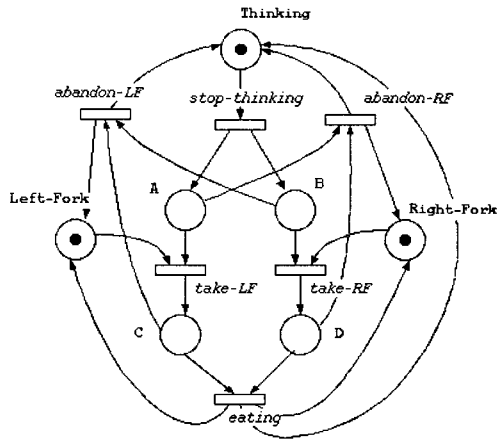


그림 1 식사하는 철학자 문제의 P/T net 모델

Petri nets 모델에는 시스템의 기능적 특성뿐만 아니라 병행적인 특성이 함께 나타나 있다. Petri net 모델에서 병행성이 나타나는 유형은 구조적인 병행성과 토큰에 의한 병행성으로 나누어 볼 수 있다. 구조적인 병행성은 모델의 구조적인 특성에 의해 병행성이 표현되는 경우이다. 그림 1에서 플레이스 A, B, Left-Fork와 Right-Fork에 토큰이 존재하면 항상 트랜지션

*take-LF*와 *take-RF*는 병행적으로 수행된다. 이러한 구조적인 병행성은 마킹의 상태를 고려하지 않고도 트랜지션들 간의 병행성을 검사할 수 있다. 토큰에 의한 병행성은 구조적인 특성과 무관하게 토큰에 의해 일어날 수 있는 병행성을 일컫는다. 그림 1에서 트랜지션 *stop-thinking*과 *take-RF*는 구조적으로는 순차적으로 수행되어야 하는 것이지만, 토큰이 Thinking, B와 Right-Fork에 동시에 존재할 경우에는 트랜지션 *stop-thinking*과 *take-RF*가 병행적으로 수행될 수 있다. 토큰에 의한 병행성은 모델의 구조와는 무관하게 토큰의 상태 변화에 따라 병행성의 존재 유무가 결정되므로 시스템이 가지는 특성이라 간주하기는 어렵다. 일반적인 P/T net 모델에서는 이 두 가지 병행성이 복합적으로 나타난다.

3. 구조적 병행성을 이용한 모델의 분할

3.1 병행적 기본 단위

병행적 기본 단위는 모델에 내재한 구조적인 병행성을 기반으로 독립적으로 수행될 수 있는 제어 흐름들을 말한다. Petri net 모델에서 제어 흐름은 토큰의 흐름으로 표현된다. 하나의 제어 흐름은 하나의 토큰이 머무는 영역을 나타나며 트랜지션에 의해 여러 개의 토큰으로 분할되는 경우는 제어 흐름 또한 여러 개로 나뉘어진다. 그리고 하나의 제어 흐름을 이루기 위해서는 제어 토큰이 항상 기본 단위를 이루는 부분 내에 머무를 수 있어야 한다

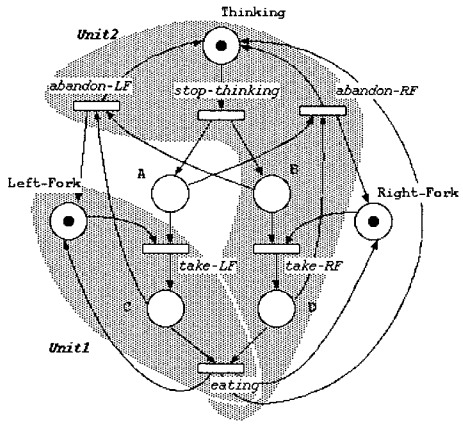


그림 2 병행적 기본단위의 예

위의 기본 단위의 조건을 만족하는 부분은 하나의 토큰이 항상 머무르는 부분을 나타내는 상태불변(S-invariant)의 특별한 형태로 나타난다. 상태불변의 직관적인 의미는 트랜지션의 수행과 상관없이 항상 토큰의 합이 일정하게 유지되는 플레이스들의 집합을 나타낸다. 즉, 병행적 기본 단위는 토큰 하나에 대한 상태불변이므로 제어는 하나만 존재하며 또한 제어의 흐름이 항상 상태불변내에 존재하게 된다. 그림 2에 나타난 모델은 식사하는 철학자의 병행적 기본 단위의 예를 보여준다. 그림에서 띠로 묶여진 부분이 상태불변의 조건을 만족한다. 첫 번째 기본 단위는 플레이스 Left-Fork에 있는 초기 토큰이 머무를 수 있는 하나의 패적을 나타내고 두 번째 기본 단위는 철학자가 오른쪽 포크를 잡는 행위를 나타내는 부분을 나타낸다.

위의 예는 호의 가중치가 1인 경우를 다루고 있지만 일반적인 호의 가중치가 양의 정수값인 경우의 상태 불변은 아래와 같이 정의된다[5].

정의 3.1 상태불변(S-invariant)

*N*을 *P/T net*이라 할 때, $N' \bullet i = 0$을 만족하는 플레이스 벡터 $i : P \rightarrow Z$ 를 상태불변이라 한다. 여기에서, N'은 *Petri net* 모델을 행렬(트랜지션 × 플레이스)로 나타낸 것이다.

Petri net 모델의 플레이스 수를 n이라 하면, 상태불변, 즉, 행렬식 $N' \bullet i = 0$의 해는 시간 복잡도 $O(n^3)$으로 구할 수 있다[5]. 일반적인 상태불변은 음의 값으로 표현되기도 하며 여러 상태불변들이 합해져 하나의 상태불변을 이루기도 한다. 병행적 기본 단위로 사용되는 상태불변은 아래와 같이 제약적인 형태로 표현되는 극소 상태불변이다.

정의 3.2 극소 상태불변(*minimal invariant*)

상태불변으로 양의 값으로만 표현되며 다른 상태불변을 포함하지 않는 극소 형태(*minimal*)를 말한다.

그림 2에서 극소 상태불변은 $Unit_1$과 $Unit_2$로 표시된 {*Left-Fork, C*}와 {*Thinking, B, D*}이외에도 {*Right-Fork, D*}와 {*Thinking, A, C*}가 있다.

3.2 Petri net 모델의 분할

극소 상태불변을 기본 단위로 하여 시스템을 나누고 기본 단위에 포함되지 않는 플레이스들의 정보는 인근 극소 상태불변에 추가하여 모든 플레이스들이 기본단위에 나타나도록 한다. 시스템 분할 알고리즘을 상세히 설명하면 아래와 같다. 먼저 상태불변을 구하여 극소 상태불변을 선택하여 병행적 기본 단위들을 모두 구한다(*find_minimal_invariants*). 병행적 기본 단위 집합(*SetofInvariant*)에서 *SliceSet*에 속하지 않는 플레이스를 포함하는 가장 작은 상태불변을 하나 선택하여

*SliceSet*에 추가한다. 이때 동일한 크기의 상태불변이 여러 개 존재할 경우에는 *SliceSet*에 속하지 않는 플레이스를 더 많이 가진 상태불변을 선택한다. 이와 같은 과정은 모델의 모든 플레이스들이 *SliceSet*에 추가되었을 때*(Place(N) == Place(SliceSet))* 또는 더 이상 새로운 플레이스를 가진 상태불변이 존재하지 않을 때 *(Place(SetofInvariant) ⊆Place(SliceSet))*까지 반복한다. 만약 *SliceSet*에 포함되지 않는 플레이스들이 존재하면, 인근 상태불변의 출력 또는 입력으로 연결하여 모든 플레이스들이 포함되게 한다. 아래 알고리즘은 시스템 분할 알고리즘을 나타내고 있다.

Petri net 모델 분할 알고리즘

```
SliceSet = ∅;
SetofInvariant = find_minimal_invariants(N);
do {
    small_invariant = find_small_uncovered_Invariant
                      (SetofInvariant);
    SliceSet = SliceSet ∪ {small_invariant};
    SetofInvariant = SetofInvariant - {small_invariant};
} while ( Place(SetofInvariant) ⊆ Place(SliceSet) OR
  Place(N) == Place(SliceSet) );
if ( Place(SliceSet) ≠ Place(N) ) {
    Uncovered_Place_Set = Place(N) - Place(SliceSet);
    for ∀p ∈ Uncovered_Place_Set do {
        slice = find_near_slice(SliceSet, p);
        slice = slice ∪ {p};
    }
}
```

분할 알고리즘에 의해 구해진 병행적 기본 단위에 의해 결정되는 Petri net Slices는 아래와 같이 정의된다.

정의 3.3 *Petri net Slices*

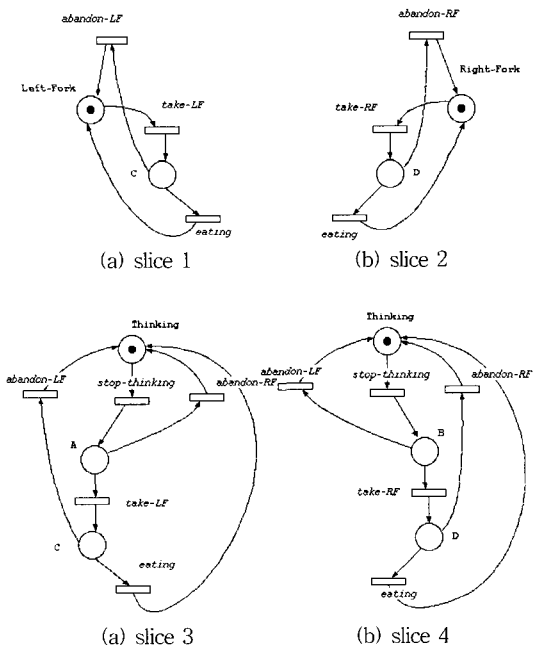
전체 모델 $N = (P, T, F, W, M)$에서 모델 분할 알고리즘에 의해 분할된 플레이스 집합을 $SliceSet = \{P\_Slice_i \mid i = 1 \cdots n\}$ 라 하면, *Petri net Slices*은 $\{Slice_i \mid i = 1 \cdots n\}$로 정의되고 각각의 *Petri net* $Slice_i = (P_i, T_i, F_i, L_i, W_i, M_i)$는 아래 조건을 만족한다.

- $P_i = P\_Slice_i$,
- $T_i = \{ t \in T \mid s \in Pi, (s, t) \in F$ 혹은 $(t, s) \in F\}$,
- $F_i = \{ (p, t) \in F, (t, p) \in F \mid p \in Pi, t \in Ti \}$,
- $L_i : T \rightarrow \Sigma^*$는 트랜지션에 문자열의 레벨을 정의한 함수,
- $\forall p \in P_i$, $W_i(p) = W(p)$이고 $\forall p \in M_i$, $M_i(p) = M(p)$.

Petri net 모델에서 트랜지션 t의 수행조건은 t에 연결된 이전 플레이스들($\bullet t$)에 토큰이 존재하여야 한다. Petri net Slices는 여러 subnet으로 분할되어 있으며 트랜지션의 레벨에 의해 subnet 간의 동기화가 이루어진다. Petri net Slices에서 트랜지션의 수행조건은 아래와 같이 정의할 수 있다.

- $L_i(t)$가 공유되지 않을 때 : 트랜지션 t의 수행조건만 만족하면 된다.
- $L_i(t)$가 여러 Slice에 공유될 때 : $L_i(t)$를 가지는 모든 트랜지션들이 수행조건을 만족하여야 한다.

그림 3은 그림 2의 모델에 대해 분할 알고리즘을 적용한 결과를 보여주고 있다.



(a) slice 1 (b) slice 2

(a) slice 3 (b) slice 4

그림 3 식사하는 철학자 모델의 Slice 집합

그림 3(a)에 나타난 slice는 Left-Fork를 포함하는 상태불변이 하나의 slice를 구성한 예이고 그림 3(b)는 Right-Fork를 포함하는 상태불변이 하나의 slice를 구성하고 있다. 그림 3(c)는 철학자가 왼쪽 포크를 잡는 행위를 나타내는 부분이 하나의 slice를 구성하고 있고 그림 3(d)는 철학자가 오른쪽 포크를 잡는 과정을 나타낸다. 식사하는 철학자 모델에서는 상태불변에 포함되지 않는 플레이스는 존재하지 않는다.

분할 알고리즘에 의해 분할되기 전의 원래 모델과 분

할된 Slices 모델이 동일한 행동을 보임을 보이기 위해 행동일치 개념을 정의한다. 행동일치는 내부 구조는 무시하고 외부적으로 나타나는 행동(observational behavior)만을 고려한다[14].

정의 3.4 Petri net 모델의 행동일치(Behavioral Equivalence)(P ~ Q) :

두 Petri Net 모델 P와 Q의 도달성 그래프들 간의 일대일 매핑이 가능할 때 P와 Q는 행동일치라 한다.

정리 3.1 Petri net 모델을 N이라 하고 분할 알고리즘에서 구해진 Petri net Slices 모델을 S라 하면 Slice 트랜지션 수행조건을 만족하는 합성시스템과 원래 모델 N은 행동일치이다(N ~ S).

(증명)

분할 알고리즘이 플레이스의 입출력 호를 그대로 유지하므로 N과 S의 해당 플레이스들은 동일한 행동을 가진다. 그리고 S는 N으로부터 해당되는 부분만을 추출한 부모델이며 N의 모든 플레이스들이 S에 포함되므로 S와 N의 트랜지션의 집합은 동일하다. 그러므로 두 Petri net 모델이 동일한 행동을 수행함을 보이기 위해서는 우선 두 모델의 대응되는 트랜지션들의 수행 전후 조건이 동일함을 보여야 한다. 즉, 임의의 트랜지션 t에 대해 N에서의 수행전 플레이스 집합($\bullet t$)과 S에서의 t와 연관된 $Slice_i$의 $\bullet t$를 모은 Slice 수행전 플레이스 집합($\circ t$)이 동일함을 보이고 수행후 플레이스 집합에 대해서도 동일함을 보여야 한다($t\bullet = t\circ$). $t\bullet = t\circ$의 증명은 $\bullet t = \circ t$ 와 유사하게 증명될 수 있으므로 여기서는 $\bullet t = \circ t$ 만 증명한다.

Petri net 모델을 $N = (P, T; F, W, M)$으로 정의하고 N으로부터 분할 알고리즘을 수행하여 얻은 Slice 모델을 $S = \{ Slice_i \mid i = 1 \ldots n \}$으로 정의하고 $Slice_i = (P_i, T_i, F_i, L_i, W_i, M_{0i})$로 정의한다.

(경우1) $\circ t \subseteq \bullet t$

$Slice_i$는 N으로부터 추출된 것이므로 t의 Slice 수행전 플레이스 집합($\circ t$)의 모든 원소는 $\bullet t$ 에 속한다.

(경우2) $\bullet t \subseteq \circ t$

모든 $p \in \bullet t$에 대해, p는 반드시 특정 $Slice_i$에 포함되어야 한다. 그리고 slice에서는 플레이스와 연관된 모든 정보가 포함되어야 하므로 p와 연관된 모든 트랜지션이 slice 내에 포함되어 있다. 즉 $p \in \circ t$가 만족된다.□

4. 합성적 분석

합성적 분석은 시스템 모델이 여러 부시스템으로 나뉘어 기술된 경우에 적용 가능하며 우선 각각의 부시스템에 대해 도달성 그래프를 생성하여 국부적인 특성에 대한 분석을 수행하고 각 도달성 그래프를 축약한 후 전체 시스템으로 합성하여 전역적인 특성에 대한 분석을 수행한다. 부시스템으로 나뉘어진 모델에서 합성적 도달성 그래프의 생성과 분석 방법은 Yeh의 논문[4]에 잘 기술되어 있다. 이 절에서는 합성적 도달성 그래프를 생성하는데 있어 Petri net Slices 방법과 모듈화 Petri nets 방법을 비교하고 Slices를 이용한 플레이스의 유한성 검사와 트랜지션의 수행가능성 검사에 대해 간략히 서술한다.

4.1 도달성 그래프 생성

합성적 분석에서는 합성적인 방법으로 계층적인 도달성 그래프를 생성한다. 효율적인 분석을 위해서는 플레이스들의 토큰의 마킹으로 표현되는 시스템 상태의 수와 시스템 상태들간의 상태전이를 최소로 갖는 도달성 그래프를 생성하여야 한다.

표 1 식사하는 철학자 모델에서 도달성 그래프의 상태와 상태전이의 수

	P/T nets	Modular Petri nets	Petri net Slices
철학자 1명	5 (8)	15 (24)	18 (29)
철학자 2명	18 (46)	20 (40)	38 (61)
철학자 3명	76 291)	55 (134)	58 (93)
철학자 4명	322 (1644)	77 (195)	78 (126)
철학자 5명	1364 (8705)	99 (256)	98 (157)
철학자 6명	5778 (44250)	121 (317)	118 (189)
철학자 7명	24476 (218687)	143 (378)	138 (221)
철학자 8명	103682 (1058712)	165 (439)	158 (253)
철학자 9명	-	187 (500)	178 (285)
철학자 10명	-	209 (561)	198 (317)
철학자 20명	-	429 (1171)	398 (637)
철학자 30명	-	649 (1781)	598 (957)
철학자 50명	-	1089 (3001)	998 (1597)

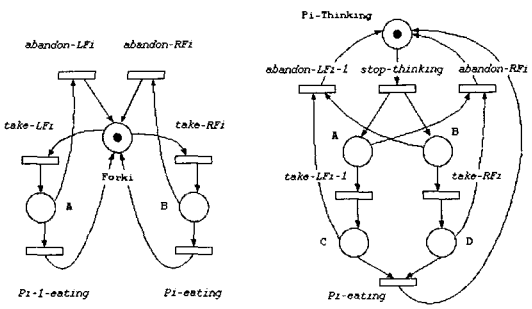
* 표에서 각 칸의 숫자는 도달성 그래프의 상태의 수(상태전이의 수)를 나타낸다.

표 1은 n 명의 식사하는 철학자들을 일반적인 P/T nets, Modular Petri nets과 Petri net Slices를 이용하여 분석할 경우 생성되는 시스템 상태와 상태전이의 수를 비교하여 보여주고 있다. P/T net 모델에서는 그림 1의 모델을 확장하여 n 명의 철학자와 n 개의 포크를 기술하며 Modular Petri net 모델에서는 그림 4와 같이 철학자와 포크를 각각 독립적인 부시스템으로 기술한다. Petri net Slices는 P/T net 모델을 바탕으로 분할 알

고리즘을 수행한 후 Slices를 구한 것으로 그림 3을 일반화하면 된다. 합성적 방법에서 합성하는 순서에 따라 중간과정에서 생성되는 상태의 수에는 많은 차이가 있다[4]. 공정한 비교를 위해 Modular Petri net 모델과 Petri net Slices의 합성 순서를 동일하게 하였다.

표 1에서 알 수 있는 바와 같이 P/T net 모델의 상태와 상태전이의 수는 기하급수적으로 증가하는데 반해 Modular Petri net 모델과 Petri net Slices는 합성적인 방법으로 점진적으로 상태를 생성하므로 산술적으로 증가한다. Petri net Slice는 Modular Petri nets 보다 더 많은 부시스템으로 이루어져 있으므로 철학자 수가 적을 때는 상태와 상태전이의 수가 상대적으로 많지만 철학자 수가 증가하면 상태의 수는 조금씩 차이를 보이게 되고 상태전이의 수는 현격하게 차이를 나타낸다.

위의 표에서와 같이 Petri net Slice 기법은 시스템 모델을 작성할 때 병행성을 고려하지 않고 다양한 모델링 기법을 이용하여 작성하고 분석할 때에 병행성을 고려하여 slice로 분할하여 분석하더라도 작성할 때에 병행성을 고려하여 기술한 Modular Petri net 모델과 유사한 결과를 얻거나 경우에 따라서 Modular Petri nets을 이용할 경우 개발자가 모델을 기술할 때에 간과한 병행성까지 반영하므로 좀더 나은 결과를 얻을 수 있다.



(a) ich fork (b) ith philosophy

그림 4 Modular PN을 이용한 n 명의 철학자 모델

4.2 유한성 검사

유한성 검사는 플레이스가 가질 수 있는 토큰의 수를 검사한다. 예를 들어, 어떤 플레이스는 하나의 토큰만을 가질 수 있어야 하며(safe 조건) 어떤 플레이스는 무한의 토큰을 가지지 않아야 한다는 제약조건을 검사할 수 있다. 모델의 분할 시 상태불변의 특성에 의해 플레이스에 관련된 모든 입출력 정보는 하나의 slice 내부에 존재하도록 되어 있고 Petri net Slices간에 트랜지션만으로 동기화가 이루어지므로 각 플레이스들은 slice 내부 요소에 의해서만 제약을 받게 된다. 즉, 플레이스의 유한성 문제는 각 slice 내부의 문제로 귀결된다. 그러므로 플레이스의 유한성 검사는 각각의 slice의 도달성 그래프를 생성하여 검사할 수 있다. 만약 slice의 도달성 그래프에 ω(즉, 토큰을 무한히 가질 수 있는 플레이스)가 존재하면 이에 대응되는 전체 시스템의 플레이스가 무한한 토큰을 가짐을 알 수 있다.

4.3 수행가능성 검사

플레이스의 유한성과 달리 트랜지션의 수행가능성은 하나의 slice 내부에 국한되지 않는다. Slice들간에 트랜지션이 공유되므로 트랜지션의 수행가능성은 전역적으로 검사되어야 한다. 즉, 하나의 slice 내부에서 수행된다고 해서 그 트랜지션이 전체 모델에서 수행된다고 말할 수 없다. 하지만, 하나의 slice 내부에서 수행되지 않는 트랜지션은 다른 slice와 무관하게 수행되지 않는다.

트랜지션의 수행가능성은 합성적 도달성 그래프를 이용하여 계층적으로 검사된다. 하나의 slice 내부의 모든 트랜지션들이 수행가능하다는 것은 slice에 있는 공유되는 트랜지션들이 모두 제대로 수행된다는 가정이 만족되면 slice 내부의 모든 트랜지션은 수행가능하다. 만약 공유되는 트랜지션이 제대로 수행되지 않으면 그 트랜지션이 포함된 slice 내부의 다른 트랜지션들도 영향을 받는다. 합성적 도달성 그래프를 이용한 계층적 수행가능성 검사는 우선 각각의 slice 내부의 트랜지션에 대해 검사를 수행하고 slice들간의 합성적 도달성 그래프 상에서 공유되는 트랜지션이 제대로 수행되는지 검사한다.

5. 결론 및 향후연구방향

P/T nets은 자체적으로 다양한 분야로 이용되고 있을 뿐만 아니라 Colored Petri nets, 객체지향 Petri nets, CMPNs 등의 확장된 고급 Petri nets의 기반 언어로 이용되고 있다. Petri net 모델 분석에 있어 가장 큰 단점은 모델에 내재된 병행성을 제대로 다루지 못하는데 있다. 이 연구에서는 Petri net 모델에 내재된 구조적 병행성을 파악하고 이를 기반으로 모델을 분할하여 합성적 분석을 수행할 수 있도록 하였다. 이러한 연구 결과는 기존의 Petri nets 모델뿐만 아니라 실시간 시스템, 객체 지향 소프트웨어, 병행 시스템의 모델링에 사용되는 다양한 고급 Petri net 모델들의 도달성 분석에 유용하게 이용할 수 있다. 모든 고급 Petri net 모델에 Slice 기법을 적용할 수 있는 것은 아니며 P/T nets에 기반을 두고 있어 자동적으로 P/T nets으로의 변환이 가능한 모델에서만 적용될 수 있다.

향후 연구 과제로서는 병행성 파악 및 분할 알고리즘을 효율적으로 Petri net 모델에 적용할 수 있도록 기존의 다양한 고급 Petri nets의 모델링 기법과 호환이 가능한 합성적 분석 도구의 개발을 들 수 있다.

참 고 문 헌

[1] T. Suzuki, S.M. Shatz, "A Protocol Modeling and Verification Approach Based on a Specification Language and Petri Nets," IEEE Trans. on Software Engineering, Vol. 16, No. 5, pp. 523-536, May 1990.

[2] C. Ghezzi, D. Mandrioli, S. Modasca, and M. Pezze, "A Unified High-Level Petri Net Formalism for Timed-Critical Systems," IEEE Trans. on Software Engineering, Vol. 17, No. 2, pp. 160-172, 1991.

[3] G. Bucci and E. Vicario, "Compositional Validation of Time-Critical Systems Using Communicating Time Petri Nets," IEEE Trans. on Software Engineering, Vol 21, No.12, pp. 969-992, Dec. 1995.

[4] Wei Jen Yeh, Controlling State Explosion in Reachability Analysis, PhD. Thesis, Purdue University, Dec. 1993.

[5] Wolfgang Reisig, Petri Nets: An Introduction, Springer-Verlag, 1985.

[6] James L. Peterson, Petri Net Theory and The Modeling of Systems, Prentice-Hall, 1981.

[7] K. Jensen, Coloured Petri Nets: Basic Concepts, Analysis methods and Practical Use, Volume 1, Springer-Verlag, 1992.

[8] H.J. Genrich, "Predicate/Transition Nets," Petri Nets: Applications and Relationships to other Models of Concurrency(ed. W. Brauer, W. Reisig, and G. Rozenberg), pp. 207-247, Springer-Verlag, 1987.

[9] E. Battiston and F.D. Cindio, "Class Orientation and Inheritance in Modular Algebraic Nets," Proceedings of IEEE Conference on System, Man, and Cybernetics, Vol. 2, pp. 712-723, Oct. 1993.

[10] A. Perkusich and J. Figueiredo, "G-Nets: a Petri Net Based Approach for Logical and Timing Analysis of Complex Software Systems," Journal of System and Software, Vol. 39, pp. 39-59, 1997.

[11] W.J. Lee, S.D. Cha, and Y.R. Kwon, "Integration and Analysis of Use Cases Using Modular Petri Nets in Requirements Engineering," IEEE Trans. On Software Engineering, Vol. 24. No. 12, pp. 1115-1130, Dec. 1998.

[12] W. Damm, G. Dohmen, V. Gerstner and B. Josko, "Modular Verification of Petri Nets," LNCS #430, 1989.

[13] S. Christensen and L. Petrucci, "Modular State Space Analysis of Coloured Petri Nets," Applications and Theory of Petri Nets '95(LNCS #935), 1995.

[14] R.Miler, Communication and Concurrency, Prentice-Hall, 1989

[15] Masato Notomi and Tadao Murata, "Hierarchical Reachability Graph of Bounded Petri Nets for Concurrent-Software Analysis," IEEE Trans. On Software Engineering, Vol. 20, No. 5, May 1994.

이 우 진

1992년 경북대학교 컴퓨터과학과 학사. 1994년 한국과학기술원 전산학과 석사. 1999년 한국과학기술원 전산학과 박사. 1999년 7월 ~ 현재 한국전자통신연구원 컴소연 실시간컴퓨팅연구부. 관심분야는 요구사항 분석, 실시간 소프트웨어 명세 및 분석, 실시간 소프트웨어 개발도구, 정형적 명세 기법, Petri nets

차 성 덕

1983년 University of California at Irvine 전산학 학사. 1986년 University of California at Irvine 전산학 석사. 1991년 University of California at Irvine 전산학 박사. 1990년 ~ 1991년 Hughes Aircraft Co. 연구원. 1991년 ~ 1994년 The Aerospace Corp. 연구원. 1994년 9월 ~ 현재 한국과학기술원 전산학과 조교수.

권 용 래

1969년 서울대학교 문리대학 이학사. 1971년 서울대학교 대학원 이학석사. 1971년 ~ 1974년 육군사관대학교 전임강사. 1978년 미국 피츠버그대학 이학박사. 1978년 ~ 1983년 미국 Computer Science Corporation 연구원. 1983년 ~ 현재 한국과학기술원 전산학과 교수. 관심분야는 실시간 병렬 소프트웨어 검증, 실시간 시스템의 객체지향 기술, 고신뢰도 소프트웨어의 품질 보증

김 홍 남

1980년 서울대학교 전자공학과 학사. 1989년 미국 Ball State University 전산학 석사. 1996년 미국 Pennsylvania State University 전산학 박사. 1983년 ~ 현재 한국전자통신연구원 컴소연 실시간컴퓨팅연구부 실시간시스템지원도구 연구팀장. 관심분야는 Video compression algorithm, Real-time System, Software Engineering, Distributed Multimedia System