상태 공간 추상화에 기반한 실시간 시스템의 분석을 위한 상태 감소

박지연*° 박주호* 조기환** 이문근**
*전북대학교 컴퓨터과학과 **전북대학교 전자정보공학부
{jypark, juhpark, ghcho, mklee}@cs.chonbuk.ac.kr

State Minimization for Analysis of Real-Time Systems Based on State Space Abstraction

Ji-yeon Park*° Ju-ho Park* Gi-hwan Cho** Moon-kun Lee**
*Dept. of Computer Science, Chonbuk National Univ.
**Division of Electronics and Information Engineering, Chonbuk National Univ.

요 약

본 논문에서는 실시간 상태 기계(*Real-time State Machine*, RSM)로 명세된 실시간 시스템의 행위의 쉽고 간결한 이해, 분석을 위한 새로운 상태 최소화 방법을 기술한다. 시스템의 행위를 보여주는 RSM 실행에 대한 상태는 제어 변수, 자료 변수, 시간 변수의 집합에 의해 정의된다. 상태 최소화는 4단계 추상화인 계산(*computation*), 제너릭(*generic*) 패턴, 한계 간격(*limit interval*), 동일 범위(*coordinate scope*) 추상화를 통해 이루어진다. 계산 추상화 단계에서는 연속적인 계산으로 연결된 다수의 상태를 하나의 상태로, 일반 패턴 추상화 단계에서는 상수 또는 함수 관계에 있는 동일 제어의 연속된 일련의 상태들의 집합을 하나의 제너릭 패턴으로 통합한다. 한계 간격 추상화 단계에서는 특정 값으로부터 음의 무한대나 양의 무한대 값으로 단조 증가, 단조 감소하는 값 사이에 있는 상태들을 하나의 상태로 통합한다. 마지막으로, 동일 범위 추상화 단계에서는 같은 범위에 존재하는 일련의 상태들을 하나의 상태로 통합한다. 각 추상화의 적용은 제어, 데이터, 시간의 무한한 상태 공간을 유한한 상태 공간으로 감소시킬 수 있으며 많은 상태 감소를 가능하게 한다. 따라서, 시스템 행위에 대한 이해와 분석이 복잡도가 적은 개념 단계에서 수행될 수 있다.

1. 서론

상태 기반 정형 기법에서 상태 폭발 문제[1, 2]는 상태 기계로 표현된 시스템의 행위에 대한 이해와 분석을 어렵게 한다. 이는 각 상태 기계의 수와 각 기계가 가진 상태의 수가 증가할수록 전체 상태의 수는 기하급수적으로 증가하기 때문이다. 특히, 실시간 시스템에서는 실시간 속성으로 인하여 비(非)실시간 시스템에 비해 상태 공간의 복잡도가 높으며 이에 따라 상태 기반에 따른 실시간 시스템의 이해와 분석도 더욱 어렵게 된다. 상태 폭발 문제를 해결하기 위해 많은 연구가 진행되고 있으며 이 연구들은 시스템의 실행 시 발생하는 상태 혹은 상태 공간의 크기를 줄이는 것을 주요 목적으로 한다[1, 2, 3, 4, 5, 6].

상태 폭발 문제를 해결하기 위해, 본 논문에서는 4단계의 추상화를 통하여 상태 공간을 최소화시킬 수 있는 방법론을 제안한다. 추상화의 주요 목적은 특정 조건을 만족하는 상태들을 하나의 상태로 통합하여 시스템의 분석 관점에서의 상태의 수를 감소시키는 것이다. 상태 최소화를 위한 4단계 추상화는 *계산 (computational)*, *제너릭 패턴(generic pattern)*, 한계 간격 (*limit interval*), *동일 범위 추상화(coordinate scope abstraction)*로 정의된다.

본 논문에서의 접근 방법은 4단계의 추상화를 연속적으로 적용하여 무한히 발생될 수 있는 상태를 유한 상태로 변환하며, 또한 유한한 상태도 보다 적은 수의 상태로 감소시킨다. 결과적으로, 실시간 시스템의 이해와 분석이 복잡도가 낮은 개념 단계에서 수행될 수 있다.

2. 관련 연구

실시간 시스템을 명세하고 상태 폭발 문제 해결에 초점을 두고 시스템의 속성을 검증하기 위해 다양한 연구가 있다.

[2, 3]의 연구는 발생 가능한 상태 폭발을 구조적 계층성을 이용하여 서브 시스템의 세부 사항을 숨김으로써 방지하지만 시스템 제어 결정에 영향을 미치는 데이터 값의 다양한 영향이 제어 결정이 데이터 값과 독립적으로 수행된다는 가정을 하고 있어 실제 시스템 분석에는 부적합하다.

CSM(*Communicating State Machine*)[4]은 데이터 도메인을 다수의 동치 클래스로 클러스터링하는 상태 최소화 방법을 제안한다. 이 방법은 실시간 시스템과 같이 많은 변수가 사용되는 시스템에는 적용이 어렵다. 변수가 많아지면 많아질수록 클러스트링을 위한 복잡도는 기하급수적으로 증가하기 때문이다. 또한, CSM은 시스템의 시간 속성 명세를 위한 정의를 제공하지 않으며 시스템의 모든 변수를 고려하기 때문에 불필요한 데이터 도메인을 생성할 수 있는 가능성이 존재한다.

[5]의 접근 방법은 시간 공간에 대해 무한한 도달성 그래프를 유한한 도달성 그래프로 변환한다. 그러나, 데이터와 제어 공간 또한 고려되어야 한다.

[6]은 상태 최소화를 위해 *history equivalence*와 *transition bisimulation* 개념을 사용한다. 이러한 방법은 결국 시간 개념이 없는 오토마타 도달성 그래프를 분석하는 것과 유사하다. 즉, 데이터 공간이나 이벤트에 대한 도달성 분석은 가능하나 시간 공간에 대한 분석은 고려할 수 없다.

3. 명세와 실행 모델

3.1 실시간 상태 기계

각 RSM은 병렬적으로 실행 가능한 실시간 시스템이나 분산 시스템 혹은 실시간 분산 시스템인 태스크나 프로세스를 나타낸다.

정의 3.1 : 상태 시간 기계 M은 $<\Sigma, N, n_0, F, T, V, C>$의 튜플로 정의된다. Σ는 다른 RSM과의 상호 작용을 위한 포트의 유한 집합을, N은 노드의 유한 집합을, $n_0 \subseteq N$은 시작 노드를 나타낸다. $F \subseteq N$는 최종 노드의 유한 집합이며, T는 전이 조건과 시간 제약을 가진 노드 간 전이의 유한 집합, V는 데이터 변수의 유한 집합을, C는 지역 시계(*local lock*)를 나타낸다. □

3.2 RSM 실행 모델

RSM 실행 모델에서 상태는 *상태 변수(state variable)*인 데이터 변수, 제어 변수, 시간 변수 값에 의해 표현된다. 데이터 변수의 집합은 RSM의 V이며 제어 변수는 노드 번호로써 RSM의 실행 시점을 나타낸다. 시간 변수는 지역 시계의 이산 실시간 값을 갖으며 RSM의 C이다. 한 상태는 RSM의 실행 상태인 특정 시점에서의 변수 값과 실행 위치, 시간 값을 보인다.

전이의 실행은 한 상태에서 다른 상태로의 전환 즉, 상태 변수 값의 변화를 의미한다. 만약, 한 노드에서의 전이가 자기 자신으로 발생한 경우에는 상태 변수 중 제어 변수의 값은 변하지 않는다. 또한, 즉각적인 실행 시간 요구를 가진 전이가 실행될 경우 시간 변수의 값은 변하지 않는다.

본 연구는 한국과학재단 특정기초연구(1999-2-203-003-3) 지원으로 수행되었음

정의 3.2 : RSM M의 실행은 튜플 $E=<S, T_e^t>$ 로 정의한다. S는 상태의 무한 혹은 유한 집합이며, T_e^t는 실행된 전이의 유한 또는 무한 집합이다. 노드 n_i에서의 상태는 제어, 데이터, 시간 변수의 값의 유한 집합, 즉, $<v_i^c, I_1(v_1^d), I_1(v_2^d), \cdots, I_j(v_n^d), v_i^t>$ 이다. v_i^c는 노드 번호를 가진 제어 변수의 값을 나타내며, 각 $I_j(v_j^d)$, $1 \le j \le n$은 각 데이터 변수 $v_j^d \in V$의 값을 나타낸다. v_i^t는 시간 변수의 값이다. M의 실행 전이 τ_e^t는 (τ_e, t_e^t), $\in T_e^t \tau_e \in T_e \subset T$로 정의된다. t_e^t는 전이에 소요되는 시간이다. □

3.3 시스템 모델

시스템 $\mathcal{S}$는 시스템을 구성하는 RSM들의 유한 집합, $\mathcal{S} = \{M_1, M_2, \cdots, M_m\}$이다.

정의 3.3 : 시스템 $\mathcal{S} = \{M_1, M_2, \cdots, M_m\}$의 실행은 튜플 $\mathcal{E} = <S^s, T_e^{ts}>$로 정의한다. 여기에서 S^s는 $M_j, 1 \le j \le m$의 실행에 관련한 시스템의 유한 혹은 무한 상태 집합이며, T_e^{ts}는 시스템 실행 전이의 무한 혹은 유한 집합이다. 시스템 상태는 각 RSM의 특정 상태의 집합 $s_i^s = <s_i^1, s_i^2, s_i^m> \in S^s \subseteq (S^1 \times S^2 \times, \cdots, \times S^m)$로 정의된다. 시스템 전이는 $\tau_e^{ts} = (\tau_e, t_e^t) \in T_e^{ts} \tau_e \in T_e^s \subseteq (T_e^1 \times T_e^2, \times \cdots \times T_e^m)$ 로 정의된다. t_e^t는 전이가 이루어진 시간이다. □

4 추상화를 통한 상태 최소화

4.1 계산 추상화

정의 4.1 : RSM 노드 x와 y사이에 계산 전이 t가 존재한다면 x와 y는 *계산 관계(computational relation)*를 가진다. □

정의 4.2 : RSM M의 계산 관계에 대한 공집합이 아닌 노드의 집합 N'이라 할 때 N'에 대한 파티션 Π^c은 다음 조건을 만족하는 노드 집합의 모임이다.

i) $N' = N - (\{n_0\} \cup F)$, N은 M의 노드 집합이다.

ii) 모든 $\pi_i^c \in \Pi^c$ 와 $\pi_j^c \in \Pi^c$ 에 대해

① $\pi_i^c = \pi_j^c$ 이거나 $\pi_i^c \cap \pi_j^c = \varnothing$,

② $n_m, n_n \in \pi_i^c, n_m \sim n_n$, $\sim$은 계산 관계,

③ $n_m \in \pi_i^c$, $n_n \in \pi_j^c, i \neq j$, $n_m \not\sim n_n$, 여기서, $\not\sim$는 비(非)계산 관계를 나타낸다.

iii) $N' = \cup_{\pi^c \in \Pi} \pi^c$. □

각각의 $\pi^c \in \Pi$는 *계산 모드(computational mode), c-mode*라 하며 c-mode로 추상화된 RSM을 cRSM으로 표기한다. c-mode는 계산 관계로 연결된 연속된 상태들이다. 만약 어떠한 계산 관계도 없거나 혹은 계산 전이 이외의 다른 전이가 존재하여 계산 모드로 추상화 될 수 없는 단일 노드가 있다면 이 노드는 *공(空)계산 모드(empty computational mode)*라 한다. RSM M의 노드들에 대해 c-mode를 생성하는 것을 *계산 추상화(computational abstraction)*라 한다. 계산 추상화 단계에서는 데이터 상태 공간의 감소를 유도하며 RSM의 이해와 분석을 보다 쉽게 할 수 있도록 한다.

정의 4.3 : cRSM M의 실행은 튜플 $E^c = <S_c, T_{ce}^t>$ 이다. S_c는 실행 상태의 유한 또는 무한 집합이며, T_{ce}^t는 실행된 전이의 유한 혹은 무한 집합이다. c-mode에서 상태, *c-state* 역시 제어변수와 데이터, 시간 변수의 내부 값의 유한 집합으로 표현된다. $((v_{i,l}^c, v_{i,u}^c), (I_{i,l}(v_1^d), I_{i,u}(v_1^d)), (I_{i,l}(v_2^d), I_{i,u}(v_2^d)), \cdots, (I_{i,l}(v_n^d), I_{i,u}(v_n^d)), (v_{i,l}^t, v_{i,u}^t))$ 의 형태로 각 변수는 대한 경계값(bound value)형식의 값을 갖는다. 실행 전이는 $\tau_{ce}^t = (\tau_{ce}, t_{ce}^t) \in T_{ce}^t$ 이며 $\tau_{ce} \in T_{ce}$이며 T_{ce}는 M의 전이의 집합 T에 대해 $T_{ce} \subseteq T$이다. 또한 t_{ce}^t는 전이가 수행될 때 걸리는 시간을 나타낸다. cRSM의 실행 E^c는 RSM의 실행 E에 대한 *계산 추상화 실행(computationally abstracted execution,* CAE)이다. □

정의 4.4 : cRSM의 조합인 시스템 $\mathcal{S}^c = \{M_1^c, M_2^c, \cdots, M_m^c\}$ 의 실행은 튜플 $\mathcal{E}^c = <S_c^s, T_{ce}^{ts}>$ 로 정의한다. S_c^s는 계산 추상화가 적용된 시스템 실행 상태의 유한 또는 무한 집합이다. 시스템 하나의 상태는 $s_i^s = <s_{i,n}^1, s_{i,n}^2, \cdots, s_{i,n}^m> \in S_c^s \subseteq (S_c^1 \times S_c^2 \times \cdots \times S_c^m)$ 이다. S_c^i는 각 cRSM의 상태이다. 시스템의 실행 전이는 $\tau_{ce}^{ts} = (\tau_{ce}, t_{ce}^t) \in T_{ce}^{ts}$ 로 정의된다. τ_{ce}는 $\tau_{ce} \in T_{ce}^s \subseteq (T_{ce}^1 \times T_{ce}^2 \times, \cdots \times T_{ce}^m)$이다. 시스템 $\mathcal{S}^c$의 실행 $\mathcal{E}^c$는 시스템 $\mathcal{S}$의 실행 $\mathcal{E}$의 CAE다. □

4.2 제너릭 패턴 추상화

정의 4.5 : $s_{i_k} \xrightarrow{t_{i_k}} s_{i_{k+1}} \xrightarrow{t_{i_{k+1}}} \cdots \xrightarrow{t_{i_{k+j-1}}} s_{i_{k+j}}$, $1 \le k \le n$와 같은 다수의 연속 σ_k에 대해 모든 상태 $s_{i_k,n}, 1 \le k \le n$, $0 \le l \le j$가 다음의 조건이 만족할 경우 *동치 패턴(equivalent pattern)* p^e라 한다:

① $I_{i_k+l}^{lb}(v^m) - I_{i_k+l}^{lb}(v^m) = 0$,

② $I_{i_k+l}^{ub}(v^m) - I_{i_k+l}^{lb}(v^m) = a$

③ $I_{i_k+l}^{ub}(v^m) - I_{i_k+l}^{lb}(v^m) = f((I_{i_k+l}^{lb}(v^m), I_{i_k+l}^{ub}(v^m)), (I_{i_k+l}^{lb}(v^m), I_{i_k+l}^{ub}(v^m)))$

v^m은 제어 변수를 제외한 모든 상태 변수이다.

이러한 조건은 M_c의 실행에서 모드의 연속 $\sigma_k, 1 \le k \le n$에 동치 패턴 p^e가 존재한다면 모든 $s_{i_k}, s_{i_k+j}, 1 \le j \le n-1$ 에 대해서도 만족하게 된다. □

모드의 연속 σ_k가 정의 4.5를 만족하는 경우 일정하거나 혹은 함수적 관계성을 가진 특정한 패턴으로 통합될 수 있다. 이와 같은 패턴을 *제너릭 패턴* p^g라하며, 제너릭 패턴으로의 통합을 *제너릭 패턴 추상화(generic pattern abstraction,* GPA)라 한다.

제너릭 패턴 추상화 또한 cRSM으로 구성된 시스템 $\mathcal{S}^c$에 적용되며 그 결과 무한 상태에서 유한 상태로 변환 가능하며 제어, 데이터, 시간 공간의 감소를 유도하게 된다.

4.3 한계 간격 추상화

정의 4.6 : 다음의 조건을 만족할 때 변수 v의 간격(interval) 값 (v_l, v_u)은 *한계 간격(limit interval,* v-LI)이라고 정의한다:

i)상한 값(upper bound value) v_u에서 하한 한계 값(lower bound limit value) $v_l = -\infty$ 또는 $v_l = c$ (c는 상수) 사이의 모든 상태는 하나의 간격이 될 수 있다.

ii)하한 값(lower bound value) v_l에서 상한 한계 값(upper bound limit value) $v_u = +\infty$ 또는 $v_u = c$ 사이의 모든 상태는 하나의 간격이 될 수 있다. □

정의 4.7 : cRSM M이 가진 c-mode $m_i, m_{i+1}, \cdots, m_n$ 의 특정 연속에 대해서 변수 v의 간격 값이 각각 (v_l^i, v_u^i), $(v_l^{i+1}, v_u^{i+1}), \cdots, (v_l^n, v_u^n)$ 이라고 할 때:

i) $j > i$일 때 변수 v에 대해 $v_u^j \le v_l^{j-1}$ 이면 *단조 증가 간격(monotonically increasing interval,* MII) 즉, v-MII이라고 하며,

ii) $j > i$일 때 변수 v에 대해 $v_u^{j-1} \le v_l^j$이면 *단조 감소 간격(monotonically decreasing interval,* MDI) 즉, v-MDI라고 한다. □

정의 4.8 : cRSM M이 가진 c-mode $m_i, m_{i+1}, \cdots, m_n$ 의 연속에 대해서 변수 v의 간격 값 $(v_l^i, v_u^i), (v_l^{i+1}, v_u^{i+1}), \cdots, (v_l^n, v_u^n)$ 에 대해 다음과 같이 정의할 수 있다:

i)간격이 MII이고 v_u^n가 $+\infty$ 또는 가장 큰 상수일 때 간격 값을 변수 v에 대한 *최소 경계 간격(lower limit bound interval,* LLBI) 즉, v-LLBI라고 하며,

ii)간격이 MDI이고 v_u^n가 $-\infty$ 또는 가장 작은 상수일 때 간격 값을 변수 v에 대한 *최대 경계 간격(upper limit bound interval,* ULBI) 즉, v-ULBI라고 정의한다. □

cRSM M이 가진 c-mode $m_i,m_{i+1},\cdots,m_n$ 의 연속에 υ-MII이고 LLBI가 존재하거나 υ-MDI이고 ULBI이 존재할 때 이 간격을 하나의 모드로 통합하는 과정을 변수 υ에 대한 *한계 간격 추상화(limit interval abstraction,* LIA) 즉, υ-LIA라 한다.

한계 간격 추상화를 통하여 cRSM 또는 시스템의 상태 변수인 데이터 공간 값과 시간 공간 값이 간격화되어 전체 시스템 측면에서 상태 변수의 미시적인 측면이 제거된 즉, 복잡도가 적은 측면에서의 시스템 분석을 수행할 수 있게 된다.

4.4 동일 범위 추상화

정의 4.10 : 다음은 *동일 범위 관계(coordinate scope relation)*이다.

i)c-mode x, y가 소스 코드 상에서 같은 범위에 존재하는 연산 혹은 동작을 나타낼 경우, x, y는 cRSM에서 같은 *물리적 범위(physical scope)* 내에 존재하며,

ii)c-mode x, y가 독립적 의미에서 구축하기 위해 서로 밀접하게 관련되어 있는 경우, x와 y는 cRSM에서 같은 *개념적 범위(conceptual scope)* 내에 존재한다. 또한,

iii)c-mode x, y가 조건문, 반복문과 같은 특정 구조에 존재한다면, x와 y는 cRSM에서 같은 *구조적 범위(structural scope)*에 존재한다. □

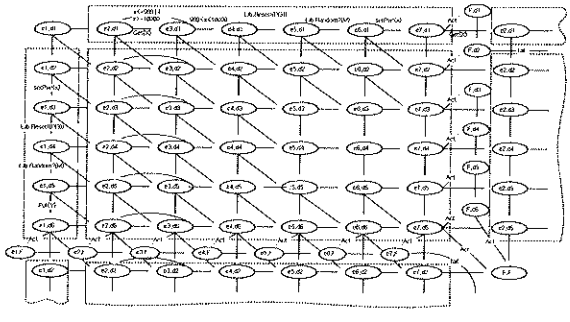
이와 같은 동일 범위 관계를 가진 파티션의 집합 Π^s를 생성하는 과정을 *동일 범위 추상화(coordinate scope abstraction,* CSA)라 한다.

각각의 $\pi^s \in \Pi^s$는 범위 모드(*scope mode*), *s-mode*이며 s-mode로 추상화된 RSM은 sRSM이다. s-mode로의 추상화는 각 상태 변수의 상태 공간을 줄이며, 시스템 행위의 이해와 분석을 용이하게 한다.

또한, 동일 범위 추상화가 시스템에 $\mathcal{S}^s$에 적용되어 시스템 실행의 상태를 감소시킨다.

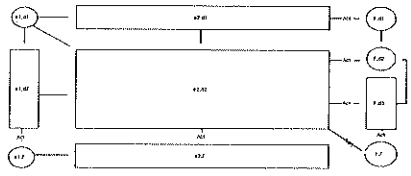
5. 예제

각 추상 단계에서의 상태 감소를 보이기 위해 Encoder-Decoder(ED) 예제를 사용한다. ED 예제는 두 개의 RSM으로 구성되어 있으며, 각 RSM은 병렬적으로 실행가능한 Ada 태스크를 나타낸다. <그림 1>은 Encoder, Decorder RSM이 병렬 실행되는 시스템 $S(DE)$={*Encoder,Decoder*} 의 상태와 전이 관계를 보여주는 그림이다. 이는 Encoder, Decorder RSM에 대해 각각 4단계 추상화를 적용한 뒤 생성한 그림이다.



<그림 1> 시스템 DE의 실행 $\mathcal{S}^0(DE)$

<그림 2>는 <그림 1>이 보여주는 실행 상태에 대해 4단계 추상화를 적용한 최종 결과이다.



<그림 2> 예제 DE에 대한 System DE의 실행

6 복잡도

상태의 수는 각 추상단계를 통하여 준다. 정형적인 상태 감소 분석을 위해, 시스템을 구성하는 RSM M^1과 M^2의 노드 개수를 각각 f와 g로 가정하자. 만약 $2^{h-1} \le f \le 2^h$ 이고 $2^{i-1} \le g \le 2^i$ 이면, 상태의 반복이 없는 경우, 시스템에 있는 RSM의 실행 복잡도는 $O(2^{h+1})$ 이다. 계산 추상화를 적용했을 때, M^1과 M^2에 대한 cRSM에서 만약 c-mode의 개수가 m과 n 이라면, $m \le f, n \le g$ 이다. 이때 $2^{s-1} \le m \le 2^s$, $2^{t-1} \le n \le 2^t$ 이라면, cRSM에서 실행 시스템의 복잡도는 상태의 반복이 없는 경우에 $O(2^{s+t})$이다. 최악의 경우, $O(2^{h+i})$ 와 $O(2^{s+t})$ 사이에 차이가 없으며 최적의 조건에서, 상태의 개수는 $(1/2)^k$로 줄어들게 된다. 즉 $O(2^{h+i}) \le O(2^{s+t})$ 이다. 제너릭 패턴과 한계 간격 추상화를 적용할 때, 복잡도는 $O(2^{s+t-x}) \le O(2^{s+t}) \le O(2^{h+i})$, $0 \le x \le s+t$ 의 관계를 만족하는 정도로 감소하게 된다.

7 결론 및 향후 연구

본 논문에서는 네 단계의 추상화를 적용한 새로운 상태 감소 방법을 기술하였다. 연속된 많은 상태의 수가 계산, 한계 간격, 제너릭 패턴, 동일 범위 추상화를 통해 급격히 줄어 하나의 모드로 통합하는 상태 감소 방법을 제안하였다. 시스템의 행위적 속성에 대한 이해와 분석이 매우 개념적인 단계에서 수행가능하므로, 시스템 실행에 대한 복잡도는 간단해진다.

실질적인 실시간 시스템의 응용 분야에서 접근을 위해 수행되어야 할 향후 연구 사항들이 아직 많이 남아있다. 이러한 것으로는 예외처리, 확률, 분산 속성 등과 같은 것이 있으며, 또한 자동화 도구로써 개발중인 SUVT(Specification Understanding and Verification Tool)[10]을 발전시켜 실질적인 다수의 RSM을 다루고, 분석할 수 있도록 해야한다.

참고문헌

[1]E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, April 1986, 8(2), pp. 244-263.

[2]W. J. Yeh and M. Young. Compositional Reachability Analysis using Process Algebra. In *Proceedings of Conference on Testing, Analysis and Verification*, August 1992, pp. 49-59.

[3]E. M. Clark, D. E. Long, and K. L. McMillan. Compositional Model Checking. In *Proceedings of Fourth Annual Symposium on Logic in Computer Science*, 1989.

[4]B. Josson and J. Parrow. Deciding Bisimulation Equivalences for a Class of Non-finite-state Programs. Technical Report SICS/R-89/8908, Swedish Institute of Computer Science, August 1989.

[5]S. Raju. An Automatic Verification Technique for Communicating Real-Time State Machines. Technical Report 93-04-08, Dept. of Computer Science and Engineering, Univ. of Washington, April, 1993.

[6]I. Kang and I. Lee. State Minimization for Concurrent System Analysis Based on State Space Exploration. In *Proceeding of Conference on Computer Assurance*, 1994, pp. 123-134.

[7]I. Kang, I. Lee and Y. Kim. An Efficient State Space Generation for the Analysis of Real-Time Systems. *IEEE Transaction on Software Engineering*, Vol. 26, No. 5, May 2000, pp. 453-477.

[8]Feldman and Koffman, *Ada95*, Addison-Wesley, 1996.

[9]노경주, 박지연, 이문근. 실시간 시스템의 순환공학을 위한 정형기법: 추상 시간 기계. 한국정보과학회 논문집(A), 제27권 1호, 2000, 4. pp. 558-560.

[10]노경주, 이문근. 추상시간기계를 기반으로한 실시간 시스템의 시간 명세와 분석. 한국정보과학회 소프트웨어공학회지, 제13권 제3호, 2000, 9. pp. 45-54.