JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 17, No. 10 : 2330~2336, Oct. 2013

합성 및 축소화 기법을 이용한 시스템의 해석 방법

이완복*

System Analysis Method Using Composition and Minimization

Wan Bok Lee*

Department of Game Design, Kongju National University, Kongju 314-701, Korea

요 약

사람이 만든 시스템들은 서로 상호 작용하는 여러 개의 요소 시스템들로 이루어지기 때문에, 시스템 오류로 인한 비정상적인 현상이 발생하였을 경우 그 논리적 오류를 분석하는 것이 매우 어렵다. 또한 높은 품질과 안정적인 시스템 개발을 위해서는 논리적 해석 방법이 반드시 적용되어질 필요가 있다. 본 논문에서는 시스템의 합성과 축소화 두 연산을 정의하고 이것을 기반으로 시스템을 해석하는 방법을 제안한다. 제안한 방법에서는 시스템의 합성과 축소화 과정을 반복적으로 수행하되 내부 이벤트들을 간소화 시킴으로써 주요 입출력 이벤트들의 응답이 같음을 확인하는 방법이다. 제안한 방법의 효용성을 보이기 위해 기존의 간단한 통신 프로토콜에 대하여 적용하여 보았으며, 전송매체에 전달된 메시지는 수신측에 반드시 전달된다는 것을 확인할 수 있었다.

ABSTRACT

Since many man-made systems consist of autonomous and interactive components, it is intrinsically difficult to analyze their abnormal behavior. The logical analysis of such a system is an indispensable process for high quality and reliable system development. In this paper, we propose an analysis method using two algebraic operations, named composition and minimization. Repetitive composition and minimization of component models with respect to a set of important events produces a new analysis model that has the same input output responses to an environment. An analysis example of the alternating bit protocol demonstrates the effectiveness of the proposed method showing that each message generated at the sender side eventually arrives to a receiver.

키워드 : 시스템 분석, 시스템 합성, 이산사건 시스템, 모델 검증

Key word : System analysis, composition, discrete event system, verification

접수일자 : 2013. 02. 08 심사완료일자 : 2013. 06. 25 게재확정일자 : 2013. 09. 04

* **Corresponding Author** Wan Bok Lee(E-mail:wblee@kongju.ac.kr, Tel:+82-41-850-0355)
Department of Game Design, Kongju National University, Kongju 314-701, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2013.17.10.2330 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

사람들이 만든 대부분의 시스템은 상호 작용을 하는 여러 요소 시스템들로 구성되어 있다. 이러한 시스템들은 많은 수의 상태수를 가지기 때문에, 설계단계에서 예측하지 못했던 오류 상황이 발생할 수 있다. 이러한 상황의 예로는 교착 상태나 상호 배제 위반(Violation of Mutual Exclusive access)등이 있는데, 그 원인을 사후에 파악하는 것은 쉽지 않은 일이 된다. 또한 시스템 오류는 각종 사건 사고로 이어질 수 있다[1]. 그렇기 때문에, 항공 조정 시스템이나 핵발전소의 플랜트 제어 등과 같이 고수준의 안정성이 요구되는 분야에서는 다양한 방법을 이용하여 시스템의 안정성을 검증하려는 노력을 하고 있다.

시스템의 올바른 동작성을 확인하기 위해서 시뮬레이션 기법도 많이 사용되어져 왔다. 특히 시뮬레이션을 활용하면 대상 시스템이 올바르게 동작하는지 실질적으로 확인할 수 있으며, 그 성능까지 통계적 실험과정을 통해 확인 가능한 장점이 있다. 그러나 시뮬레이션으로는 시스템이 실행되면서 도달할 수 있는 모든 경우에 대한 검사가 이루어지는 것이 아니기 때문에 시스템의 안정성을 완전히 하기 어려운 단점이 있다. [2][3]

Zeigler에 의해 제안된 DEVS 모델[4]은 시스템의 행위를 수학적 표현에 기반을 두고 있기 때문에, 논리적인 오류 분석 및 시뮬레이션 과정을 통한 성능 분석을 할 수 있다. DEVS 모델은 일반적인 이산사건 시스템의 특성을 완전하게 표현할 수 있으며 그 표현력은 입출력 시스템 수준으로 CFSM, Automata 등의 표현력과 동등한 수준이다. 하지만 DEVS 모델은 다른 이산사건 시스템 모델과 달리 입력과 출력이 명백히 구분되고 내, 외부상태천이함수가 구분되어 자발적 천이를 표현하기 쉽다는 특징을 가지고 있다. 또한 모듈러하고 계층적인 표현 방법을 가지고 있어서 실제 시스템을 직관적으로 표현할 수 있는 표현력을 제공한다. 이는 계층적인 시스템 설계 시 매우 유용한 특성이다.

현재까지 DEVS 모델은 주로 시뮬레이션 영역에서 많이 사용되어 왔지만, 본 연구에서는 모델 복합과 축소 과정을 효과적으로 적용하여 시스템의 논리적인 오류를 분석하는 방법을 제안하려 한다.

Ⅱ. 시스템의 논리 해석 방법

시스템이 논리적으로 오류가 없다는 것은 시스템 설계에서 명시된 각종 특성들이 구현된 시스템에서 만족된다는 것을 입증하면 된다. 일반적으로 시스템 검증과정에서는 특성 명세, 행위 모델 명세, 일치성 판단 기준, 체크 알고리즘이 언급되어진다.

특성 명세는 시스템이 만족해야 할 속성들에 대한 명세이다. 보통 시스템이 절대로 멈추지 않는다든지, 주어진 입력에 대해 출력이 반드시 만들어진다든지 하는 것이 이러한 특성 명세의 예가 될 수 있다. 기존의 Temporal Logic[5], CCS[6] 명세 같은 것들이 이러한 명세에 해당한다.

행위모델 명세는 시스템의 동작 특성을 기술하는 모델이다. 이러한 모델 명세의 예로는 이벤트 기반이나 상태 기반으로 시스템을 명세하는 오토마타, 페트리 네트, StateChart, DEVS 등과 같은 모델이 있다.

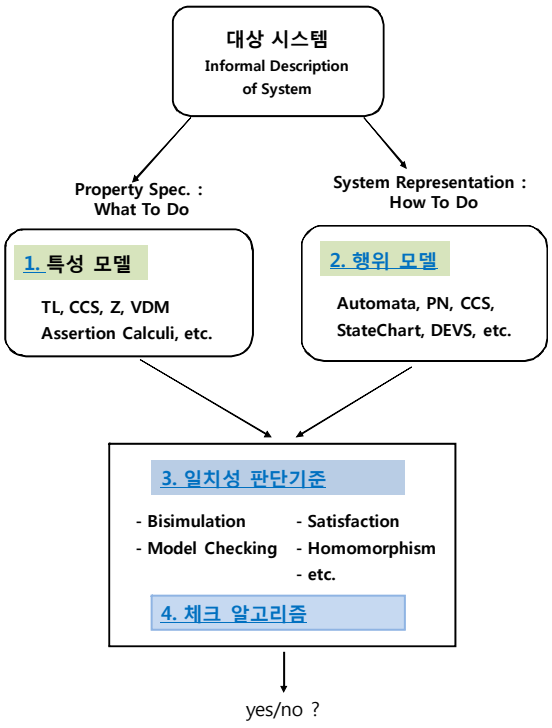

그림 1. 시스템의 논리 해석 과정
Fig. 1 System Analysis Flow

행위모델이 특성 명세를 만족하는지 그렇지 않은지를 판단하기 위해서는 일치성 판단기준이 수학적으로 잘 정의되어질 필요가 있다. Satisfiability[5]나 model checking[6] 기법은 상태 머신(State Machine)으로 표현된 시스템 행위 모델이 로직으로 표현된 특성 명세들을 만족하는지 아니지 그 여부를 판단하는 기준의 일종이다. 일치성 판단 과정에서는 시스템의 모든 상태를 탐색해 나가면서 주어진 특성을 판단하여야 하기 때문에, 효과적인 탐색 알고리즘이 요구된다. 특히 시스템을 구성하는 요소 시스템의 개수가 늘어갈수록 전체 시스템의 상태 수는 기하급수적으로 늘어나기 때문에 상태폭주 현상을 피해갈 수 있는지가 매우 중요한 문제점이 된다.

그림 1은 시스템 해석 과정을 보여준다. 대상 시스템에 대해 특성 모델과 행위모델이 만들어지면 일관성 판단기준에 의해 행위 모델이 스펙에 해당하는 특성 모델을 따르는지 그 여부를 판단할 수 있게 된다.

2.1. 행위 모델의 명세

행위모델의 명세를 위해서 DEVS 모델을 변형하여 사용하였다. DEVS모델은 이벤트들이 발생하는 시간을 정확히 지정하는 원자 모델들의 결합들로써 구성된 모델이다. 해석과정에서는 이러한 원자 모델들이 도달하는 모든 상태들을 복합한 모델이 필요하게 되며, 분석을 위해서 이벤트들의 선수관계는 유지시키되 발생시간은 추상화한 후, 해석 과정에 사용하는 것이 일반적이다[7]. CDEVS는 이와같은 용도를 충족하기 위해 기존 DEVS 모델을 변형하여 제작된 모델이다. CDEVS 모델에서는 시간전진 함수를 생략하기 때문에 non-determinism[8]이 발생한다. 구체적으로 내부 상태 천이 함수와 출력 함수에서 발생하기 때문에 내부 상태 천이 함수와 출력함수를 함수에서 관계 표현으로 변형하게 되었다.

CDEVS 모델은 다음과 같이 정의된다.

$$CDEVS = < X, Y, Q, \delta_{ext}, T_{int} > \quad (1)$$

$,where$

- X: $a\ set\ of\ input\ events$;
- Y: $a\ set\ of\ output\ events$;
- Q: $a\ set\ of\ states$;
- $\delta_{ext} : Q \times X \rightarrow Q$, $an\ external\ transition\ function$;
- T_{int}: $Q \times (Y \cup \{\tau\} \times Q$, $an\ internal\ transiton\ relations$;

2.2. 해석을 위한 논리 연산

행위모델의 명세를 위해서는 여러 요소 모델들로서 구성된 전체 시스템을 하나의 모델로 나타낼 필요가 있다. 이러한 과정은 연결 관계를 나타내는 CS (Coupling Scheme)이 주어질 때 합성 과정을 통해 만들어질 수 있다. CS는 다음과 같이 정의 된다.

$$CS = < X, Y, D, \{M_i\}, EIC, EOC, IC > \quad (2)$$

$,where$

- X, Y, D, EIC, EOC, IC: $the\ same\ as\ DEVS\ model$;
- M_i : $an\ atomic\ CDEVS\ model$.

두 개의 CDEVS 모델 M1, M2와 연결관계를 나타내는 CS가 주어졌을 때, 그 두 모델과 대등한 CDEVS 모델은 복합 (||) 연산을 통하여 다음과 같이 구해진다.

$$M1 \| M2 = < X, Y, Q, \delta_{ext}, T_{int} > \quad (3)$$

$,where$

- X: $X1 \cup X2$;
- Y: $Y1 \cup Y2$;
- Q: $Q1 \cup Q2$;
- $\delta_{ext} : Q \times X \rightarrow Q$;
- $T_{int} : Q \times (Y \cup \{\tau\} \times Q$;

여기서 Xi, Yi, Qi 는 모델 Mi의 입력 이벤트 집합, 출력 이벤트 집합, 상태 집합에 해당하며, 외부 상태 천이 함수 δ_{ext} 는 다음과 같이 생성된다. 이때 “?”로 시작하는 이벤트는 입력 이벤트를 의미하며, “!”로 시작하는 이벤트는 출력 이벤트를 의미하며, “τ”는 히든 이벤트를 의미한다.

$$(s_1, s_2) \xrightarrow{?x} (s_1', s_2') \quad (4)$$

$$\text{if } (x, x_1) \in EIC \wedge (x, x_2) \in EIC \wedge s_1 \xrightarrow{?x_1} s_1' \wedge s_2 \xrightarrow{?x_2} s_2'$$

$$(s_1, s_2) \xrightarrow{?x} (s_1', s_2) \quad (5)$$

$$\text{if } \left[(x, x_1) \in EIC \wedge (x, x_2) \in EIC \wedge s_1 \xrightarrow{?x_1} s_1' \wedge s_2 \xrightarrow{?x_2} \right] \vee \left[(x, x_1) \in EIC \wedge \forall x_2 (x, x_2) \notin EIC \wedge s_1 \xrightarrow{?x_1} s_1' \right]$$

$$(s_1,s_2)\xrightarrow{?x}(s_1,s_2')$$
$$\text{if}\left[(\mathrm{x},\mathrm{x}_1)\in \mathrm{EIC}\wedge(\mathrm{x},\mathrm{x}_2)\in \mathrm{EIC}\wedge \mathrm{s}_1\xrightarrow{?\mathrm{x}_1}\diagdown\right]\vee$$
$$\left[\wedge\,\mathrm{s}_2\xrightarrow{?\mathrm{x}_2}\mathrm{s}_2'\right]$$
$$\left[(x,x_2)\in EIC\wedge\forall x_1(x,x_1)\notin EIC\wedge s_2\xrightarrow{?x_2}s_2'\right] \quad (6)$$

T_{int}는 다음과 같이 생성된다. 이때 적용한 동기화의 조건은 Heyman이 제안한 동기화 모델 중 broadcast 동기화에 속한다[9]. Broadcast 동기화 모델은 각 콤포넌트들의 출력이 그것과 연결되어 있는 입력들에 의해 블록 되어지지 않는다는 특징을 보장하는데, 이것은 자율적으로 상호작용하는 실세계의 소프트웨어의 특성과 잘 부합한다.

$$((s_1,s_2),y,(s_{1i}',s_2'))\in T_{int}\quad if\ \exists\, s_{1i}'\ such\ that\ (y_1,y)\in EOC\wedge(y_1,x_2)\in IC\wedge s_1\xrightarrow{!y_1}s_{1i}'\wedge s_2\xrightarrow{?x_2}s_2' \quad (7)$$

$$((s_1,s_2),y,(s_{1i}',s_2))\in T_{int}\quad if\ \exists\, s_{1i}'\ such\ that \quad (8)$$
$$\left[(y_1,y)\in EOC\wedge(y_1,x_2)\in IC\wedge s_1\xrightarrow{!y_1}s_{1i}'\wedge s_2\xrightarrow{?x_2}\diagdown\right]\vee$$
$$\left[(y_1,y)\in EOC\wedge\forall x_2(y_1,x_2)\notin IC\wedge s_1\xrightarrow{!y_1}s_{1i}'\right]$$

$$((s_1,s_2),\tau,(s_{1i}',s_2'))\in T_{int}\quad if\ \exists\, s_{1i}'\ such\ that\ (y_1,y)\notin EOC\wedge(y_1,x_2)\in IC\wedge s_1\xrightarrow{!y_1}s_{1i}'\wedge s_2\xrightarrow{?x_2}s_2' \quad (9)$$

$$((s_1,s_2),\tau,(s_{1i}',s_2))\in T_{int}\quad if\ \exists\, s_{1i}'\ such\ that \quad (10)$$
$$\left[(y_1,y)\notin EOC\wedge(y_1,x_2)\in IC\wedge s_1\xrightarrow{!y_1}s_{1i}'\wedge s_2\xrightarrow{?x_2}\diagdown\right]\vee$$
$$\left[(\mathrm{y}_1,\mathrm{y})\notin \mathrm{EOC}\wedge\forall \mathrm{x}_2(\mathrm{y}_1,\mathrm{x}_2)\notin \mathrm{IC}\wedge \mathrm{s}_1\xrightarrow{!\mathrm{y}_1}\mathrm{s}_{1\mathrm{i}}'\right]$$

$$((s_1,s_2),\tau,(s_{1i}',s_2'))\in T_{int}\quad if\ \exists\, s_{1i}'\ such\ that\ s_1\xrightarrow{!y_1}s_{1i}' \quad (11)$$

$$((s_1,s_2),y,(s_1',s_{2i}'))\in T_{int}\quad if\ \exists\, s_{2i}'\ such\ that\ (y_2,y)\in EOC\wedge(y_2,x_1)\in IC\wedge s_2\xrightarrow{!y_2}s_{2i}'\wedge s_1\xrightarrow{?x_1}s_1' \quad (12)$$

$$((s_1,s_2),y,(s_1,s_{2i}'))\in T_{int}\quad if\ \exists\, s_{2i}'\ such\ that \quad (13)$$
$$\left[(y_2,y)\in EOC\wedge(y_2,x_1)\in IC\wedge s_2\xrightarrow{!y_2}s_{2i}'\wedge s_1\xrightarrow{?x_1}\diagdown\right]\vee$$
$$\left[(y_2,y)\in EOC\wedge\forall x_1(y_2,x_1)\notin IC\wedge s_2\xrightarrow{!y_2}s_{2i}'\right]$$

$$((s_1,s_2),\tau,(s_1',s_{2i}'))\in T_{int}\quad if\ \exists\, s_{2i}'\ such\ that\ (y_2,y)\notin EOC\wedge(y_2,x_1)\in IC\wedge s_2\xrightarrow{!y_2}s_{2i}'\wedge s_1\xrightarrow{?x_1}s_1' \quad (14)$$

$$((s_1,s_2),\tau,(s_1',s_{2i}'))\in T_{int}\quad if\ \exists\, s_{2i}'\ such\ that \quad (15)$$
$$\left[(y_2,y)\notin EOC\wedge(y_2,x_1)\in IC\wedge s_2\xrightarrow{!y_2}s_{2i}'\wedge s_1\xrightarrow{?x_1}\diagdown\right]\vee$$
$$\left[(y_2,y)\notin EOC\wedge\forall x_1(y_2,x_1)\notin IC\wedge s_2\xrightarrow{!y_2}s_{2i}'\right]$$

$$((s_1,s_2),\tau,(s_1',s_{2i}'))\in T_{int}\quad if\ \exists\, s_{2i}'\ such\ that\ s_1\xrightarrow{\tau}s_{2i}' \quad (16)$$

모델 복합(||)연산을 거치면, 두 콤포넌트 모델과 동일한 행위를 가지는 단일 모델을 구할 수 있게 된다. 이 과정에서 구해진 모델은 일반적으로 상태 수가 매우 많아질 수 있기 때문에, 동일한 행위를 하지만 상태 수가 축소된 모델을 구하는 축소(Min) 연산을 정의할 수 있다. 축소 연산은 입출력 측면에서 고려해 볼 때, 같은 상태로 취급할 수 있는 상태들을 통합하는 연산이다.

축소 연산은 두 단계의 과정으로 이루어진다. 첫 번째 단계는 내부화(internalization) 과정이다. 상호작용하는 이벤트들 중에서 외부와 관련 없이 내부적으로만 연결되는 이벤트들을 히든 이벤트 τ로 변경하는 과정이다. 두 번째 단계는 대등한 상태들(equivalent states)을 합치는 과정이다. 비교하고자 하는 두 상태가 대등한지 그렇지 않은지의 여부는 bisimulation 관계를 적용하여 판단할 수 있다[10].

예를 들어, 그림 2의 머쉰 M에서 상태 (1,1)과 (2,2)

는 발생시킬 수 있는 모든 이벤트 궤적이 동일하기 때문에 대등한 상태로 볼 수 있다. 이러한 상태들을 대표 상태 A로 합쳐서 표현한 것이 Min(M) 머쉰이다. Min(M)의 상태 A와 M의 (1.1) 또는 (2,2)는 모두 같은 이벤트 궤적을 발생시킬 수 있으나, Min(M)은 상태 수가 더 적은 단순한 모델이 된다. 이와 같이 축소 연산을 이용하면 똑 같은 행위를 하지만, 상태 수가 적은 단순화된 모델로 변환할 수 있다.

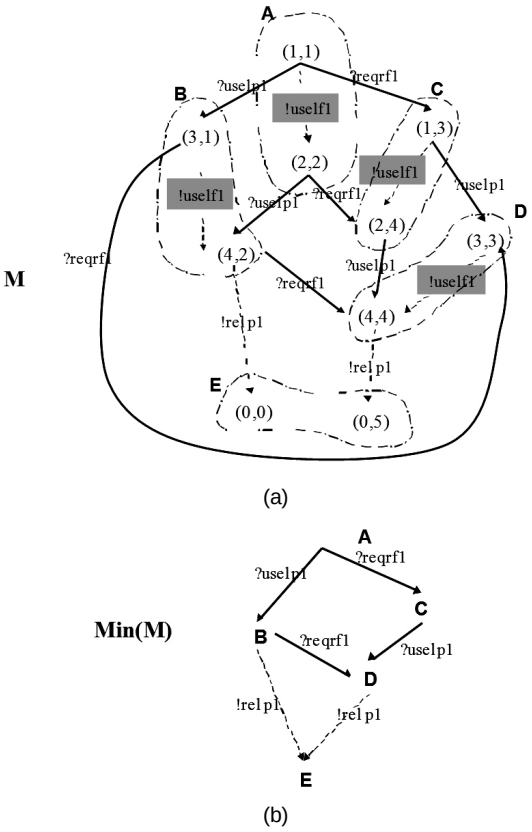

그림 2. (a) 상태 머신 M; (b) 축소된 머신 - Min(M)
Fig. 2 (a) Machine M; (b) Reduced Machine - Min(M)

Ⅲ. 논리 해석 방법 및 예제

앞에서 정의한 연산들을 이용하면 시스템의 논리적 해석을 체계적으로 수행할 수 있다. ABP(Alternating Bit Protocol) 프로토콜은 채널 손실이 발생할 수 있는 미디어에 대해 재전송 메카니즘을 적용하여 데이터를 안전하게 전달할 수 있는 통신 프로토콜이다[11]. 여기서는 ABP 포로토콜에 대해 송신자측에서 생성된 메시지가 언젠가는 수신자에가 전달될 수 있다는 특성을 위 연산들을 이용하여 살펴보려고 한다.

ABP 포로토콜을 모델링하기 위해 그림 3에서 보이는 바와 같이 전체 시스템을 8개의 요소 시스템으로 구성하였다. Sender 모델은 전송 채널을 의미하는 Trans 모델에 메시지를 송부하는데, msg 이벤트가 이를 의미한다. Trand 에서는 메시지 손실이 발생하면 메시지 전달을 Receiver 모델에 하지 못 할 수도 있다. 정상적인 경우에는 trans 이벤트를 통하여 메시지를 Receiver에게 전달하게 된다. 또한 전달시마다 0과 1 비트를 순차적으로 태그하여 보냄으로써, 재전송 메카니즘이 동작할 수 있도록 한다. Ack 모델은 수신자 측에서 송신자 측으로 메시지를 전송할 때 이용하는 전송 채널을 의미하는 모델이다. 요소 모델은 2장에서 설명한 CDEVS 모델로 구성된다. Trans 모델의 경우 그림 4와 같이 표현되어 지며, 나머지 7개의 모델도 이와 유사하게 표현될 수 있다. ABP 프로토콜에 대한 자세한 설명은 [11][12]에서 참조할 수 있다.

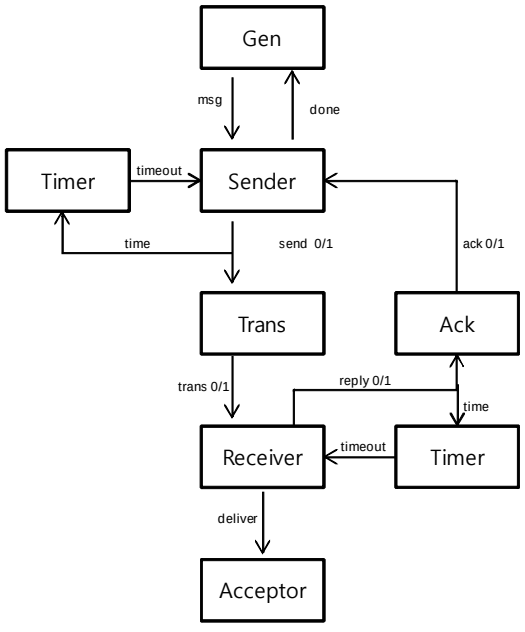

그림 3. ABP 모델
Fig. 3 ABP Model

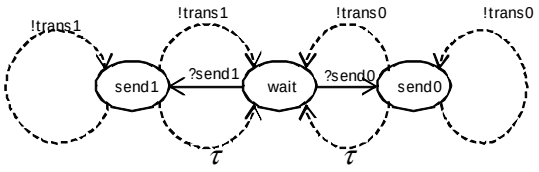

그림 4. Trans 모델
Fig. 4 Trans Model

송신자 측에서 보낸 메시지가 수신자 측에서 전달 받을 수 있다는 특성을 살펴보기 위해서는 이벤트 msg와 deliver에만 관심을 두고 나머지 이벤트 들은 내부 이벤트로 간주하면서 축소시켜 보면 된다.

중간 단계에서 발생하는 총 상태 수는 합성 순서에 따라 조금씩 달라질 수 있다. 여기서는 다음 그림 5와 같은 순서를 기준으로 모델을 복합시키고 축소시켰다. 이때 발생하는 총 상태 수는 다음 표 1과 같았다.

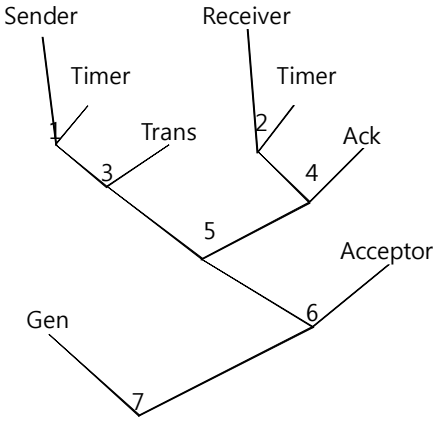

그림 5. ABP 모델
Fig. 5 ABP Model

표 1. 복합 및 축소 연산 후의 상태 수
Table. 1 Number of States after Composition, Minimization

복합 단계	복합 연산 후	축소 연산 후
1	14	8
2	11	7
3	24	22
4	19	17
5	100	3
6	3	3
7	3	2

모델 복합 연산과 축소 연산을 병행하여 진행하면, 최종적으로 다음 그림 6와 같이 축소되는 것을 확인할 수 있다.

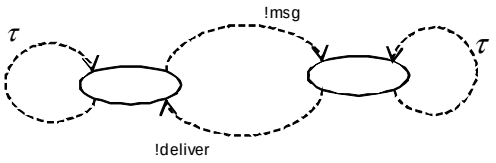

그림 6. 축소된 ABP 모델
Fig. 6 Reduced ABP Model

축소된 모델에서는 메시지 출력 이벤트 !msg가 발생하면 !deliver가 언젠가는 틀림없이 발생하여 전송된 메시지가 수신자 측에 도착할 수 있음이 보장됨을 보여준다. !msg 와 !deliver 이벤트 사이에 히든 이벤트 τ 가 여러 번 발생 할 수 있는 것은 전송 미디어를 모델링하고 있는 Transd 나 Ack에서 메시지 손실이 발생할 수 있기 때문이며, 이러한 경우에 내부적으로는 timeout이 발생하여 재전송 메카니즘이 동작하게 된다.

Ⅳ. 결 론

소프트웨어 품질을 높이려 하거나, 안정성이 보장되어야 하는 시스템을 디자인 할 경우에 시스템의 오류 해석 방법은 반드시 적용되어질 필요가 있다. 본 연구에서는 이산사건 시스템의 복합 연산과 축소 연산을 이용하여 시스템의 행위를 중요한 이벤트 위주로 간략화 시키는 기법을 적용한 시스템의 논리적 해석 방법을 보였다. 특히, 본 연구에서 사용한 시스템 모델은 시뮬레이션 분야에서 널리 사용되어지고 있는 DEVS 모델이기 때문에, 제안된 방법으로 시스템의 오류 해석이 완료되면 DEVS 시뮬레이션 방법론을 이용하여 시스템의 성능평가를 곧바로 이어서 진행할 수 있는 잇점이 있다.

현재 사용하고 있는 시스템 모델에서는 그 복잡성으로 인해 시간 표현이 되지 않고, 발생하는 이벤트들의 순서 관계만을 고려하고 있다. 향후 시간 표현을 효과적으로 할 수 있는 방법으로 연구가 확대된다면 매우 많은 범주의 실시간 시스템에 본 기법을 적용할 수 있을 것으로 기대된다.

REFERENCES

[1] M. H. Hwang and B. P. Zeigler, “Reachability Graph of Finite and Deterministic DEVS Networks”, *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 3, pp. 454 – 467, 2009.

[2] G. P. Hong, and T. G. Kim, “A Framework for Verifying Discrete Event Models Within a DEVS-Based System Development Methodology,” *Transactions of the Society for Computer Simulation*, vol. 13, no. 1, pp. 19 – 34, 1996.

[3] Md. Sipon Miah, and Insoo Koo, "Performance Analysis of ILEACH and LEACH Protocols for Wireless Sensor Networks", *Journal of Information and Communication Convergence Engineering (JICCE)*, vol. 10, no. 4, pp. 384-389, 2012.

[4] Bernard Zeigler, *Theory of Modeling and Simulation (first ed.)*. Wiley Interscience, New York. ISBN 0-12-778455-1.

[5] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*, Springer-Verlag, N.Y., 1992.

[6] R. Milner, *A Calculus of Communicating Systems*, LNCS, vol. 92, Springer-Verlag, 1980.

[7] R. Alur, “Timed Automata”, *Theroretical Computer Science*, vol. 126, pp. 183-235, 1999.

[8] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing, Reading Massachusetts, 1979. ISBN 0-201-02988-X.

[9] M. Heymann, “Concurrency and discrete event control”, *IEEE Control Systems Magazine*, vol. 10, no. 4, pp. 103-112, Jun. 1990.

[10] R. Paige and R. E. Tarjan, “Three Partition Refinement Algorithms”, *SIAM Journal of Computing*, vol. 16, no. 6, pp. 973-989, 1987.

[11] R. Milner, *A Calculus of Communicating Systems*, LNCS, vol. 92, Springer-Verlag, 1980.

[12] G.J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall, 1991.

이완복(Wan Bok Lee)

전자통신공학과 공학박사
2004.2 KAIST 전자전산학과 전기 및 전자공학 전공(공학박사)
2007.3 ~ 현재 공주대학교 게임디자인학과 교수
※관심분야 : 게임엔진, 시뮬레이션, 이산사건시스템